Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Soft-HGRNs: soft hierarchical graph recurrent networks for multi-agent partially observable environments*#

Yixiang REN†§1, Zhenhui YE†§1,2, Yining CHEN†1, Xiaohong JIANG2, Guanghua SONG†‡1

[1]*School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China*

[2]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*

†E-mail: yixiangren@zju.edu.cn; zhenhuiye@zju.edu.cn; ch19930611@zju.edu.cn; ghsong@zju.edu.cn

Received Feb. 25, 2022; Revision accepted Aug. 11, 2022; Crosschecked Jan. 4, 2023

Abstract: The recent progress in multi-agent deep reinforcement learning (MADRL) makes it more practical in real-world tasks, but its relatively poor scalability and the partially observable constraint raise more challenges for its performance and deployment. Based on our intuitive observation that human society could be regarded as a large-scale partially observable environment, where everyone has the functions of communicating with neighbors and remembering his/her own experience, we propose a novel network structure called the hierarchical graph recurrent network (HGRN) for multi-agent cooperation under partial observability. Specifically, we construct the multi-agent system as a graph, use a novel graph convolution structure to achieve communication between heterogeneous neighboring agents, and adopt a recurrent unit to enable agents to record historical information. To encourage exploration and improve robustness, we design a maximum-entropy learning method that can learn stochastic policies of a configurable target action entropy. Based on the above technologies, we propose a value-based MADRL algorithm called Soft-HGRN and its actor-critic variant called SAC-HGRN. Experimental results based on three homogeneous tasks and one heterogeneous environment not only show that our approach achieves clear improvements compared with four MADRL baselines, but also demonstrate the interpretability, scalability, and transferability of the proposed model.

Key words: Deep reinforcement learning; Graph-based communication; Maximum-entropy learning; Partial observability; Heterogeneous settings

https://doi.org/10.1631/FITEE.2200073 **CLC number:** TP181

1 Introduction

Human society achieves efficient communication and collaboration, and can be regarded as a large-scale partially observable multi-agent system. In recent years, multi-agent deep reinforcement learning (MADRL) has been facilitated to solve real-life problems such as package routing (Adler et al., 2005; Ye DY et al., 2015) and unmanned aerial vehicle (UAV) control (Zhang Y et al., 2020), which are typically large-scale and partially observable tasks. To solve the environmental instability in the multi-agent system training process, Lowe et al. (2017) proposed multi-agent deep deterministic policy gradient (MADDPG), in which the centralized training and decentralized execution (CTDE) framework was introduced, leading to many variants (Iqbal and Sha, 2019; Ye ZH et al., 2022a). However, centralized training (CT) brings high computational complexity and poor scalability, and decentralized execution

§ These two authors contributed equally to this work

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2018AAA010230)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2200073) contains supplementary materials, which are available to authorized users

ORCID: Yixiang REN, https://orcid.org/0000-0001-7460-4438; Zhenhui YE, https://orcid.org/0000-0002-7105-014X; Guanghua SONG, https://orcid.org/0000-0003-3330-4978

(DE) limits the ability of agents to obtain the necessary information for collaboration under partial observability. To handle the information insufficiency in partially observable environments, Foerster et al. (2016) proposed differentiable inter-agent learning (DIAL), pioneering the paradigm of communication learning and performing communication among all agents. An alternative is to allow agents to communicate within a certain range. Wang et al. (2020) proposed recurrent-MADDPG (R-MADDPG), which allows agents to communicate with a fixed number of nodes, and thus could restrain the range of cooperation. In this case, to achieve efficient neighboring communication, a scalable and flexible information aggregation mechanism is needed. Moreover, the recent progress in deep learning and graph learning provides a new idea for multi-agent reinforcement learning (MARL), i.e., to regard the multi-agent system as a graph. Based on intuition, DGN (Jiang et al., 2020) and HAMA (Ryu et al., 2020) adopt a graph convolutional network with a self-attention kernel (Veličković et al., 2018) as the communication structure and achieve better performance and scalability. Based on these works, this study further explores a highly scalable MADRL algorithm for a large-scale partially observable Markov decision process (POMDP).

Inspired by the observation that everyone in human society obtains necessary information for collaboration by communicating with his/her colleagues and recalling the experience, we propose a network structure called the hierarchical graph recurrent network (HGRN). HGRN regards the multi-agent system as a graph and each agent as a node. Each node encodes its local observation as the node embedding. Prior knowledge, such as distance and network connectivity, is included when connecting the nodes. To achieve communication between heterogeneous agents, we modify the hierarchical graph attention (HGAT) layer (Ryu et al., 2020) to better process the graph data with heterogeneous nodes. Specifically, we employ self-attention as the convolutional kernel and extract valuable information from different groups of neighboring agents separately. With the HGAT-based communication channel, each agent can aggregate information from its heterogeneous neighbors to alleviate the information loss in POMDP. In addition, recalling valuable information in temporal histories is helpful for solving partially observable problems. Thus, we use the gated recurrent unit (GRU) (Cho et al., 2014) to record long-term temporal information. In conclusion, HGRN is a spatial-temporal aware network that can make decisions based on information aggregated from the agent's heterogeneous neighbors and long-term memory.

Apart from the network structure, the exploration strategy is also critical in POMDP. Instead of learning deterministic policies with heuristic exploration methods such as ϵ-greedy, as many previous works have done (Jiang et al., 2020; Ryu et al., 2020), we propose a maximum-entropy method that can learn stochastic policies of a configurable target action entropy. As the exploration level varies in different scenarios, the optimal target action entropy will change accordingly. Therefore, our strategy is more interpretable and convenient to find the optimal setting compared with the previous maximum-entropy MARL method (Iqbal and Sha, 2019) that learns stochastic policies with a fixed temperature parameter. We name the value-based policies trained in this way as Soft-HGRN, and its actor-critic variant as SAC-HGRN.

The main contributions of this study are summarized as follows:

1. We introduce a novel network structure named HGRN that first combines the advantages of a graph convolutional network and a recurrent unit in MADRL. It uses spatio-temporal information to handle heterogeneous partially observable environments.

2. We propose two maximum-entropy MADRL algorithms (Soft-HGRN and SAC-HGRN) that introduce a learnable temperature parameter to learn our HGRN-structured policy with a configurable target action entropy.

3. Experiments show that our approach outperforms four state-of-the-art MADRL baselines in several homogeneous and heterogeneous environments. Case studies and ablation studies are implemented to validate the effectiveness of each component in our methods.

2 Related works

To solve the problem of multi-agent cooperation, the simplest and most straightforward method is to use a single-agent deep reinforcement

learning (SADRL) algorithm to train each agent independently. This method belongs to the decentralized training and decentralized execution (DTDE) paradigm and is known as independent learning. However, training multiple policies simultaneously may make the environment too unstable to converge. To solve the environmental instability, MADDPG (Lowe et al., 2017) extends DDPG (Lillicrap et al., 2015) by learning a centralized critic network with full observation to update the decentralized actor network with partial observability. This paradigm is known as CTDE and leads to many variants such as multi-actor-attention-critic (MAAC) (Iqbal and Sha, 2019) and PEDMA (Ye ZH et al., 2022a). However, because the input space of the centralized critic expands exponentially to the scale of the multi-agent system, it cannot converge easily in large-scale multi-agent tasks. As a consequence, many large-scale multi-agent cooperative tasks (Rui, 2010; Chu et al., 2020) are still handled by independent learning methods such as deep Q-network (DQN) (Mnih et al., 2015) and A3C (Mnih et al., 2016).

Communication learning aims to learn a communication protocol among agents to enhance cooperation. DIAL (Foerster et al., 2016) is the first work that proposes a learnable communication protocol in the multi-agent partially observable environment. CommNet (Sukhbaatar et al., 2016) uses the average embedding of all agents as the global communication value. Both DIAL and CommNet assume that communication is available among all agents, which is essentially impractical. The development of the graph neural network (GNN) in recent years offers a scalable and flexible communication structure for MADRL. Networked agents (Zhang KQ et al., 2018) construct the multi-agent system in a graph and transfer the network parameters along the edges. DGN (Jiang et al., 2020) stacks two graph attention network (GAT) layers to achieve inter-agent communication in a two-hop perception region. HAMA (Ryu et al., 2020) proposes a novel GNN structure that achieves communication among heterogeneous agents with an agent-level and a group-level self-attention structure, sequentially.

Similar to DGN and HAMA, the proposed Soft-HGRN and SAC-HGRN use the graph attention mechanism for inter-agent communication. However, DGN considers only communication among homogeneous agents, and HAMA has not devoted much attention to designing the overall network structure. By contrast, Soft-HGRN improves the graph convolution structure of HAMA to better communicate with heterogeneous agents and to properly design the overall network structure. As for memory modules, DRQN (Hausknecht and Stone, 2015), QMIX (Rashid et al., 2018), and R-MADDPG (Wang et al., 2020) also use recurrent units to store historical information to address POMDP problems. The novelty of our approach is that the stored historical information is the aggregated information obtained by graph convolution. In MAAC, Iqbal and Sha (2019) used maximum-entropy learning (Haarnoja et al., 2017, 2018) to train stochastic policies and set a fixed temperature parameter to control exploration; in our approach, the temperature parameter is learned according to the configurable target action entropy, which shows better interpretability and controllability.

3 Preliminaries and notations

Partially observable Markov game (POMG) is a multi-agent extension of the Markov decision process. At each timeslot of a POMG environment with N agents, each agent i obtains a local observation $\boldsymbol{o}^i$ and executes an action a^i, and then receives a scalar reward r^i from the environment. The objective of reinforcement learning (RL) is to learn a policy $\pi_i(a^i|\boldsymbol{o}^i)$ for agent i that maximizes its discounted reward $\mathbb{E}[R_t] = \mathbb{E}[\sum_{t=0}^{T}\gamma^t r_t^i]$, where $\gamma \in [0,1]$ is a discounting factor and T is the total interaction steps of the Markov decision process. Our work is based on the POMG framework with extra neighboring communication.

Q-learning (Watkins and Dayan, 1992) is a popular method in RL and has been widely used in multi-agent domains (Claus and Boutilier, 1998). It learns a value function $Q(\boldsymbol{o}, a)$ that estimates the expected return $\mathbb{E}[\sum_{t=\tau}^{T}\gamma^t r_t^i]$ after taking action a under observation $\boldsymbol{o}$, which could be recursively defined as $Q(\boldsymbol{o}_t, a_t) = \mathbb{E}_{a_{t+1}}[r + Q(\boldsymbol{o}_{t+1}, a_{t+1})]$. DQN (Mnih et al., 2015) is the first work that learns a Q-function using a deep neural network as its approximator, which introduces experience replay (Lin, 1992) and a target network to stabilize the training. At each environmental timeslot t, it stores the transition tuple (namely, the experience) $(\boldsymbol{o}_t, a_t, r_t, \boldsymbol{o}_{t+1})$ in a large

sliding window container (namely, the replay buffer), and resamples a mini-batch of experience from the replay buffer every τ steps. Then it updates the Q-function by minimizing the loss function as follows:

$$Q_{\text{loss}} = \left(r_t + \max_{a_{t+1}} Q'(\boldsymbol{o}_{t+1}, a_{t+1}) - Q(\boldsymbol{o}_t, a_t) \right)^2, \tag{1}$$

where Q' is the target network whose parameters are periodically updated by copying the parameters of the learned network Q. Now that an action-value function Q is trained, an optimal policy can be obtained by selecting the action with the largest Q-value: $\pi^*(a|\boldsymbol{o}) = \arg\max_a Q(\boldsymbol{o}, a)$. As the greedy policy can easily converge to sub-optimum, the DQN is trained and generally executed with heuristic exploration strategies, such as ϵ-greedy.

GAT (Veličković et al., 2018) is a remarkable GNN and serves as a powerful network structure for calculating the relationships between agents in MADRL. Generally, agent i in the environment can be represented as a node with its local information $\boldsymbol{e}^i$ as the node embedding. The connection between nodes can be determined by distance, network connectivity, or other metrics. For convenience, we use G_i to represent the set of agent i and its neighbors. To aggregate valuable information from its neighbors, agent i would calculate its relationship to each neighbor j ($j \in G_i$) with a bilinear transform (Vaswani et al., 2017).

Important notations used in this study are presented in Table 1.

Table 1 List of important notations in the study

Notation	Explanation
N, K	Number of agents, number of agent groups
$\boldsymbol{o}_t^i, a_t^i, r_t^i$	Observation, action, and reward of agent i at time t
G_t^i	Set of agent i and its neighbors at time t
C^k	Set of agents in group k
$\mathcal{O}_t^i$	Set of observation of agent i and its neighbors at time t
Q, V	Q-function, value function
π	Policy
$\boldsymbol{e}_t^i$	Encoded embedding of the local observation $\boldsymbol{o}_t^i$
Q', π'	Target Q network, target actor network
$\boldsymbol{g}_t^i$	Output of HGAT of agent i at time t
$\boldsymbol{h}_t^i$	GRU's hidden state of agent i at time t
S	Batch size
$\mathcal{H}$, $\mathcal{H}_{\text{target}}$	Action entropy, target action entropy of the policy
p_α	Percentage of $\mathcal{H}_{\text{target}}$ to $\max \mathcal{H}$
α	Temperature parameter in soft Q-learning
γ	Discounting factor

HGAT: hierarchical graph attention network; GRU: gated recurrent unit

4 Soft hierarchical graph recurrent networks

In this section, we introduce our MADRL approach for large-scale partially observable problems, including a novel network structure named HGRN and two maximum-entropy MADRL methods named Soft-HGRN and SAC-HGRN.

4.1 HGRN: aggregating information from neighbors and histories

The design of the HGRN is inspired by human society, in which each individual can communicate with his/her (logical or physical) neighbors and recall valuable information from his/her memory. We construct the multi-agent system in a graph, where each agent in the environment is represented as a node and its local information is the node embedding $\boldsymbol{e}^i$. The node embedding could pass through the edge during forward propagation. For each agent i, we define G_i as the set of agent i and its neighboring agents that interconnect to agent i. As for heterogeneous environments, there could be K different groups of agents, and we represent the set of agents in group k as C^k.

There are mainly two challenges in designing a network structure for communication among heterogeneous agents. First, the graph structure will dynamically change over time, which requires each node to have good scalability and robustness to process the neighboring nodes' information. Second, the feature representation between different agent groups may vary greatly, which makes it challenging to use the features from heterogeneous agents.

To handle the first challenge, we adopt the idea of HGAT in HAMA (Ryu et al., 2020) to achieve communication among heterogeneous agents. The key idea of HGAT is to communicate by group. Specifically for agent i, HGAT extracts information from agent group k and computes a separate node embedding vector $\boldsymbol{g}_i^k$ ($k = 1, 2, \ldots, K$). We first compute the individual relationship between agent i and each of its neighbors j in group k, represented

as

$$\alpha_{ij}^{k}=\frac{\exp\left((\boldsymbol{W}_{\mathrm{K}}^{k}\boldsymbol{e}_{j})^{\mathrm{T}}\boldsymbol{W}_{\mathrm{Q}}^{1}\boldsymbol{e}_{i}\right)}{\sum_{j\in G_{i}\cap C^{k}}\exp((\boldsymbol{W}_{\mathrm{K}}^{k}\boldsymbol{e}_{j})^{\mathrm{T}}\boldsymbol{W}_{\mathrm{Q}}^{1}\boldsymbol{e}_{i})}. \tag{2}$$

Then we aggregate the information of each neighbor j by

$$\boldsymbol{g}_{i}^{k}=\sum_{j\in G_{i}\cap C^{k}}\alpha_{ij}^{k}\boldsymbol{W}_{\mathrm{V}}^{k}\boldsymbol{e}_{j}, \tag{3}$$

where $\boldsymbol{W}_{\mathrm{Q}}^{k}$, $\boldsymbol{W}_{\mathrm{K}}^{k}$, and $\boldsymbol{W}_{\mathrm{V}}^{k}$ are learnable matrices of group k that transform the node embedding into the query, key, and value vector, respectively.

To address the second challenge mentioned above, different from the ordinary HGAT method that uses a shared self-attention unit to integrate the group-level embeddings, we concatenate the embeddings of each group and process them with a linear transform:

$$\boldsymbol{g}_{i}=W\left(\boldsymbol{g}_{i}^{1}|\boldsymbol{g}_{i}^{2}|\cdots|\boldsymbol{g}_{i}^{K}\right), \tag{4}$$

where $(\cdot|\cdot)$ denotes the concatenation operation. This is because the characteristics of different groups of agents can be quite different, and using shared parameters to process their embedding directly may lead to training instability. Note that our version of HGAT is equivalent to GAT in homogeneous environments. The detailed structure of our modified HGAT is presented in Fig. 1.

After using HGAT to realize the spatial information aggregation, we hope that the agent can retain temporal information in memory, which requires the agent to judge and select critical information in its history. To this end, we apply the GRU to process the node embedding $\boldsymbol{g}_{t}^{i}$ at time t:

$$\boldsymbol{h}_{t}^{i}=\mathrm{GRU}(\boldsymbol{g}_{t}^{i}|\boldsymbol{h}_{t-1}^{i}), \tag{5}$$

where $\boldsymbol{h}_{t-1}^{i}$ is the hidden state stored in the GRU at time $t-1$, which records valuable information in historical node embeddings.

The overall network structure of our proposed HGRN is shown in Fig. 2. For scalability and sample efficiency, agents in the same group share the same parameters. To reduce the computational

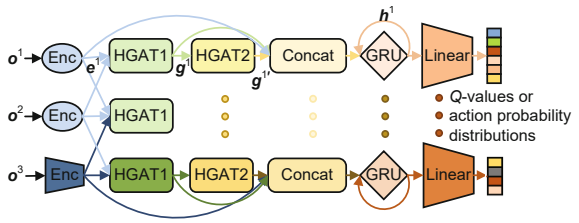

Fig. 2 Overall structure of the hierarchical graph recurrent network (HGRN) for heterogeneous environments. For simplicity, we consider three agents from two groups: agent 1 and agent 2 from the same group, and agent 3 from the other group. Agent 2 is interconnected with agent 1 and agent 3. Enc: encoding; HGAT: hierarchical graph attention; Concat: concatenation; GRU: gated recurrent unit

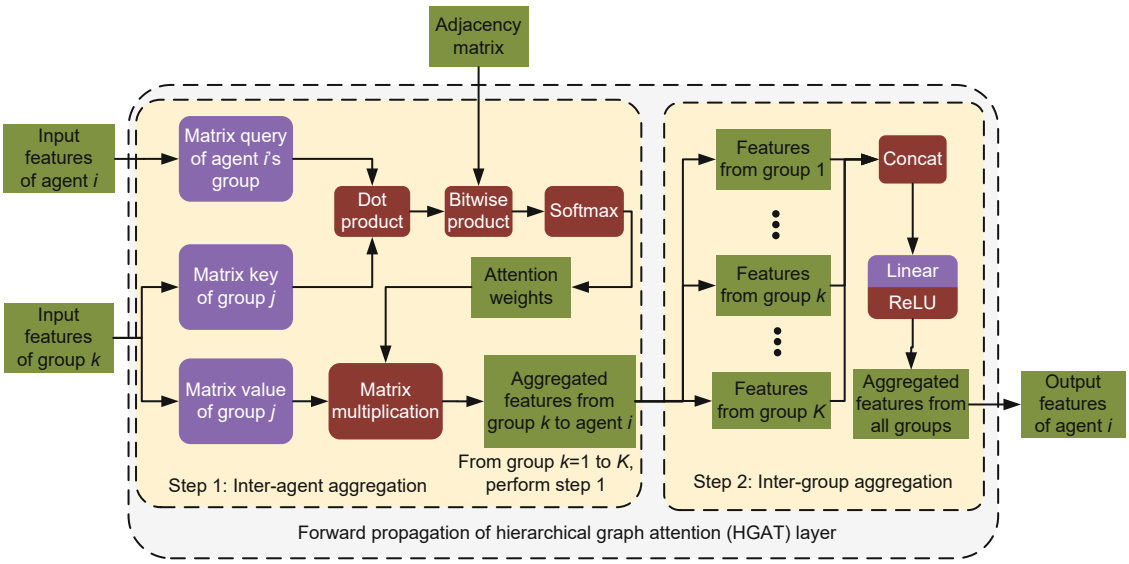

Fig. 1 Forward propagation of the proposed hierarchical graph attention (HGAT) layer to calculate agent i's node embedding. The process is composed of two steps: the inter-agent and inter-group aggregation. Parameters in the purple rectangles are trainable. References to color refer to the online version of this figure

complexity, HGRN takes only observations as the input and outputs the Q-value for each possible action. We use linear encoders to map the raw observation of heterogeneous agents to the same dimension, and then use encoding as the node embedding in HGAT. We perform network architecture search to design the HGAT-based communication structure, and details can be found in Section 5.4. Compared with HAMA (Ryu et al., 2020), which uses only one HGAT layer, our proposed HGRN stacks two HGAT layers and therefore has a two-hop perceptive field. Skip connections (He et al., 2016) are also created over the two HGAT layers to achieve better performance and faster convergence. After the GRU, a linear transform is applied to infer the Q-values for all actions.

For clarity, we use $\mathcal{O}_t^i$ to represent the set of observations of agent i and its neighbors at time t, i.e., $\mathcal{O}_t^i = \{\boldsymbol{o}_t^j | \forall j \in G_t^i\}$. Hence, the HGRN-structured Q-function could be represented as $Q^i : \mathcal{O}_t^i \to \mathbb{R}^{\mathcal{A}}$, where $\mathcal{A}$ is the dimension of the action space.

4.2 Soft-HGRN: learning energy-based policies with configurable action entropy

Deterministic policies, such as DQN and DGN, can easily fall into a local optimum, so ϵ-greedy is often used in the training phase and execution phase to encourage exploration. However, the exploration induced by ϵ-greedy is completely random and may incur performance degradation during execution. Our intuition is that stochastic policy can be used to replace ϵ-greedy to help exploration, which allows the model to learn when and how to explore. To this end, we adopt maximum-entropy learning in the multi-agent setting to train an energy-based stochastic model. Specifically, we first obtain the probability of the action $\pi(a_t|\mathcal{O}_t)$ by using softmax to process the Q-value produced by HGRN. Then we redefine the value function $V(\mathcal{O}_t)$ to learn the energy-based policy. Detailed derivations can be found in supplementary materials.

Then the Q-function is updated by minimizing the mean squared temporal-difference error (TD-error):

$$Q_{\text{loss}} = \frac{1}{S} \sum \Big(r_t + V(\mathcal{O}_{t+1}) - Q(\mathcal{O}_t, a_t) \Big)^2, \quad (6)$$

where S is the size of the mini-batch and $y_t = r_t + V(\mathcal{O}_{t+1})$ is the learning target of $Q(\mathcal{O}_t, a_t)$.

In general, action entropy is widely used to measure the degree of exploration of a policy, which is defined as the information entropy of the policy's action probability:

$$\mathbb{E}[\mathcal{H}_\pi(\mathcal{O})] = \mathbb{E}\left[-\sum_a \pi(a|\mathcal{O}) \ln \pi(a|\mathcal{O}) \right]. \quad (7)$$

As is mentioned above, the action entropy is controlled by the temperature hyper-parameter α. However, the effect of α on the action entropy varies with different network structures and environments. By contrast, setting the action entropy as the goal of the learned model is more intuitive and interpretable. Therefore, we set a target action entropy $\mathcal{H}_{\text{target}}$, and then adaptively adjust the temperature parameter α to approximate the goal of $\mathbb{E}[\mathcal{H}_\pi(\mathcal{O})] \to \mathcal{H}_{\text{target}}$. Specifically, we learn α with gradient descent by

$$\nabla \alpha = f\big(\mathcal{H}_{\text{target}} - \mathbb{E}[\mathcal{H}_\pi(\mathcal{O})]\big), \quad (8)$$

where f is a predefined function that meets the condition of $f(x) \cdot x \le 0$. Also note that $\mathcal{H}_{\text{target}}$ is the target entropy value for the policy to approximate and we generally represent it as $\mathcal{H}_{\text{target}} = p_\alpha \cdot \max \mathcal{H}_\pi$, where $\max \mathcal{H}_\pi$ is determined by the action space of policy π, and $p_\alpha \in [0, 1]$ is a new hyper-parameter to be tuned. We call the approach that learns the HGRN-based stochastic model with configurable action entropy as Soft-HGRN. An intuitive flowchart of the Soft-HGRN learning process is shown in Fig. 3.

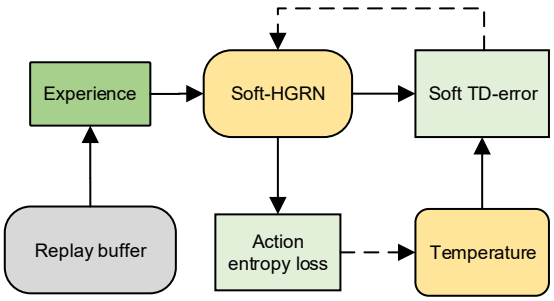

Fig. 3 Flowchart of the Soft-HGRN learning process. Solid lines denote the forward propagation to calculate the loss, and the dashed lines denote backward propagation (HGRN: hierarchical graph recurrent network; TD: temporal-difference)

4.3 SAC-HGRN: actor-critic styled stochastic policy

In some complicated scenarios, decoupling the task of value estimation and action selection can

improve the performance of the model. Adapted from soft actor-critic (SAC) (Haarnoja et al., 2018), we design an actor-critic styled variant named SAC-HGRN. The detailed derivations and a pseudocode for training SAC-HGRN are available in supplementary materials.

5 Experiments

We applied our methods in four multi-agent partially observable environments with the main goal of answering the following questions:

Q1: How do Soft-HGRN and SAC-HGRN perform in homogeneous and heterogeneous tasks compared with other state-of-the-art baseline algorithms?

Q2: How does the HGRN structure extract the necessary features for effective cooperation?

Q3: How does the learnable temperature of soft Q-learning work to control the action entropy and improve the performance of the policy?

We also perform additional experiments including network architecture search, parameter analysis, interpretability study, and ablation study.

5.1 Environments

In this subsection, we illustrate each environment in detail. The four simulation environments are all partially observable, including three homogeneous scenarios (unmanned aerial vehicle - mobile base station (UAV-MBS), Surviving, and Pursuit) and one heterogeneous scenario (cooperative treasure collection, CTC). Fig. 4 shows screenshots of the four tested environments.

1. UAV-MBS (Ye ZH et al., 2022b): a group of UAVs (solid blue circles) collaboratively serve as mobile base stations to fly around a target region to cover and provide communication services to the randomly distributed ground users (green squares). The dashed green circles, yellow circles, and blue circles represent the coverage range, observation range, and communication range of each UAV, respectively. The dashed blue arrow between UAV 0 and UAV 1 represents GAT connectivity.

2. Surviving (Jiang et al., 2020): a group of agents (green circles) cooperate to explore a big map and collect the randomly distributed food to prevent starvation. The dashed green square is the agent's observation area. Green squares with different shades denote the different numbers of food

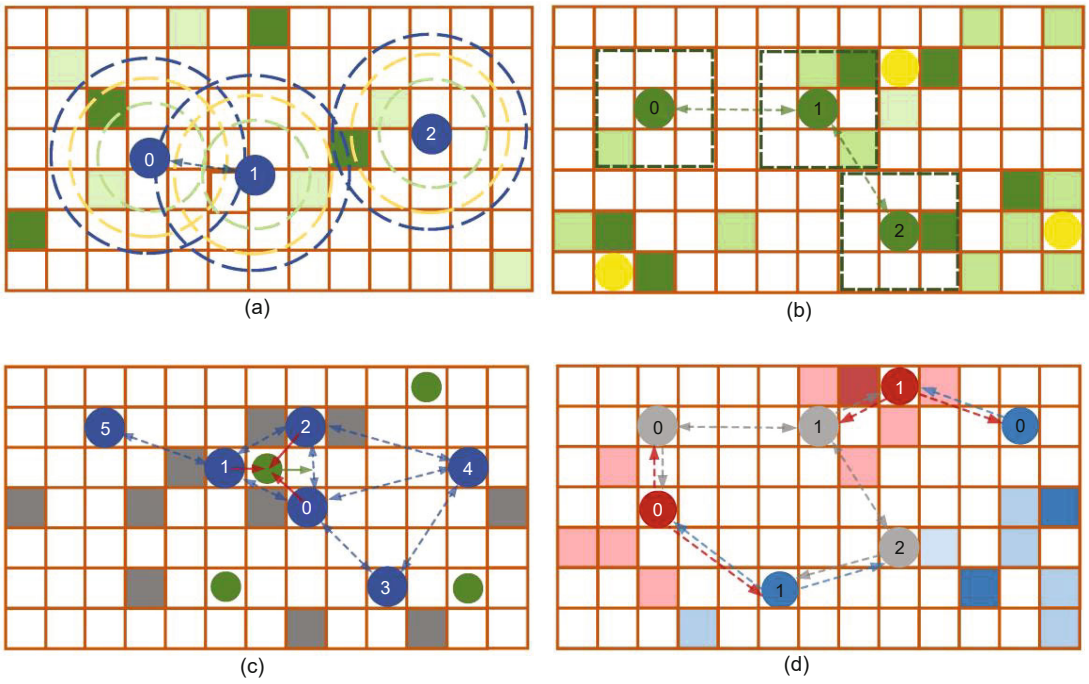

Fig. 4 Screenshots of the tested simulation environments: (a) UAV-MBS; (b) Surviving; (c) Pursuit; (d) cooperative treasure collection (CTC). References to color refer to the online version of this figure

items that are scattered out from the food resource (yellow squares).

3. Pursuit (Zheng et al., 2018): an adversarial environment where a group of predators (blue circles) controlled by the model is rewarded by attacking the prey (green circle). The prey is trained to escape from the predator. The gray square represents the obstacle, which can be used by the predator to lock the prey.

4. Cooperative treasure collection (CTC) (Iqbal and Sha, 2019): a heterogeneous environment contains three types of controllable agents, the treasure hunter (gray circles), red bank (red circles), and blue bank (blue circles). The goal of the agents is to collect the colored treasure (represented as the square) and reposit it in the bank with the correct color. The dashed colored arrow denotes HGAT connectivity.

Detailed descriptions and settings for these environments are available in supplementary materials.

5.2 Baseline methods

We compared our approach with four MADRL baselines, including DQN, CommNet, MAAC, and DGN.

DQN is a simple but strong decentralized approach for large-scale multi-agent tasks. CommNet is a centralized approach that performs communication among all agents. Thus, we used it for comparison to show the superiority of our HGAT-based communication structure. MAAC is a popular CTDE method that learns the centralized critic with full observability to update the decentralized stochastic policy. Therefore, we used MAAC for comparison to show the need to perform communication during execution, and the fact that there is no need to provide global information during training. DGN is a state-of-the-art MADRL algorithm that also uses GAT to communicate. We compared DGN with HGN (DGN with HGAT) to show the benefit of enabling communication with heterogeneous agents, and compared HGN with Soft-HGRN to examine the necessity of GRU and the maximum-entropy-based stochastic policy. A summary of the comparison of these algorithms is shown in Table 2.

5.3 Implementation details

We conducted experiments on the four environments with our methods and four baselines. Each model was updated until convergence. We used Adam (Kingma and Ba, 2015) as the network optimizer, and used stochastic gradient descent (SGD) with the same learning rate to update the learnable temperature parameter α. To make a fair comparison, the communication structure of DGN and MAAC was implemented as two stacked GAT layers with skip-connection, which is similar to HGRN as shown in Fig. 2. The hyper-parameter settings of all tested models in all environments are available in supplementary materials.

Table 2 Comparison of the characteristics of each algorithm

Algorithm	DT	DE	Comm	Hetero
DQN	√	√	×	×
CommNet	×	×	√	×
MAAC	×	√	×	×
DGN	√	√	√	×
Soft-HGRN (ours)	√	√	√	√
SAC-HGRN (ours)	√	√	√	√

DT means decentralized training; DE means decentralized execution; Comm denotes whether communication occurs during execution; Hetero denotes the ability to communicate with heterogeneous agents

5.4 Network architecture search of communication structure

In the HGRN network, we processed each node's local observation with an encoder, and then aggregated information from its (homogeneous/heterogeneous) neighbors through our HGAT-based communication structure. During the design process of the communication structure in our network, we tested three candidates: (1) one HGAT layer; (2) two stacked HGAT layers; (3) two stacked HGAT layers with skip-connection.

We tested each version of the communication structure with Soft-HGRN in UAV-MBS. Two stacked HGAT layers with skip-connection was proved to converge faster and perform better. Therefore, we applied the third communication structure in DGN, MAAC, Soft-HGRN, and SAC-HGRN. The evaluation results are available in supplementary materials.

5.5 Performance evaluation and ablation studies

After designing the overall network structure, we first compared the performance of Soft-HGRN

and SAC-HGRN with other baselines to answer Q1.

For homogeneous environments, we used the mean episodic reward over 2000 testing episodes as our evaluation metric. We performed ablation studies by removing each component (HGAT, GRU, and stochastic policy) from our approach and validating the necessity of each component. For each case, we conducted three repeated runs and calculated the average results along with their standard deviations (Table 3). Note that HGAT is equivalent to GAT in homogeneous tasks, and hence DGN is equivalent to Soft-HGRN-G and Soft-HGRN-S in the table. It can be seen that our approach outperforms other baselines by a significant margin, and removing any component would cause performance degradation. All learning curves are available in supplementary materials.

We also noticed that the necessity of each component varies in different environments, possibly due to the different nature of each task. First, removing GRU in UAV-MBS and Pursuit significantly degraded performance, whereas it was not that obvious in Surviving. This finding shows that the temporal memory is needed in UAV-MBS and Pursuit, possibly because the position of ground users in UAV-MBS is fixed during an episode, and predators in Pursuit should construct consistent closure to lock the prey, whereas in Surviving, the food will be quickly consumed and regenerated in another position. Second, HGAT-based communication is extremely important in Surviving, possibly because the dynamic food distribution requires cooperative exploration among the agents. A notable finding is that disabling communication leads to the best performance in Pursuit. Our insight is that the key factor in this task is to form stable closure with neighboring predators, which requires only information of the very nearest neighbors in the local observation range, while information from far away may distract the agent. Third, the performance of the stochastic policies trained with maximum-entropy learning outperforms the corresponding deterministic policies, proving its capability to substitute ϵ-greedy to provide more intelligent exploration. Finally, we discussed the difference between our proposed Soft-HGRN and SAC-HGRN. It can be seen in Table 3 that SAC-HGRN outperforms Soft-HGRN in every homogeneous and heterogeneous scenario. This is possibly because Soft-HGRN uses the Q-value with softmax to output the probability of actions, which leads to a small temperature value (typically around 10^{-3}) in environments where the action-value is very close; hence, the performance may be sensitive and fragile to the update of α.

In the heterogeneous environment CTC, we focused mainly on the impact of HGAT's communication between different types of agents. To this end, we also tested the DGN with HGAT (namely HGN), and Soft-HGRN/SAC-HGRN without heterogeneous communication (namely Soft-DGRN and SAC-DGRN). The learning curves of all tested models are shown in Fig. 5. It can be seen that every

Table 3 Evaluated episodic reward of different methods in three homogeneous tasks

Algorithm	Reward		
	UAV-MBS	Surviving	Pursuit
DQN	2314 ± 177	-7453 ± 74	5007 ± 150
CommNet	2377 ± 116	-7511 ± 82	5222 ± 60
MAAC	3435 ± 88	-10 ± 46	5927 ± 91
DGN	2808 ± 109	-286 ± 67	4552 ± 50
Soft-HGRN	$\underline{4051 \pm 75}$	$\underline{194 \pm 41}$	6840 ± 20
Soft-HGRN-G	3751 ± 59	-118 ± 90	$\underline{7138 \pm 96}$
Soft-HGRN-R	2935 ± 134	-26 ± 61	5649 ± 166
Soft-HGRN-S	3925 ± 46	-89 ± 35	5570 ± 30
SAC-HGRN	$\mathbf{4072 \pm 77}$	$\mathbf{325 \pm 48}$	7033 ± 55
SAC-HGRN-G	3580 ± 75	-36 ± 63	$\mathbf{7183 \pm 77}$
SAC-HGRN-R	3052 ± 22	141 ± 13	5797 ± 36
SAC-HGRN-S	3723 ± 82	-376 ± 27	-76 ± 1

"-G" means disabling HGAT-based communication among agents; "-R" means removing the GRU-based memory unit from the policy model; "-S" denotes training a deterministic policy instead of a stochastic policy. The best and second best models are indicated by bold and single underline, respectively

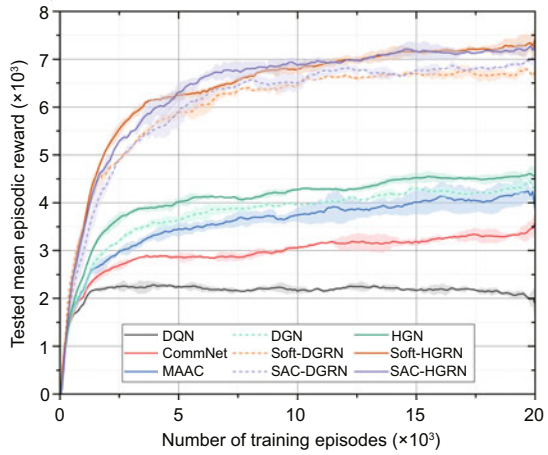

Fig. 5 Learning curves of various algorithms in cooperative treasure collection (CTC). References to color refer to the online version of this figure

HGAT-based model (the solid line) performs better than its GAT-based variant (the dashed line), which demonstrates the effectiveness of our designed hierarchical communication structure.

5.6 Case studies about interpretability

In this subsection, we perform a series of case studies to investigate the principle behind the performance of our approach.

5.6.1 How does HGRN extract valuable spatio-temporal features?

To answer Q2, we studied the effectiveness of the proposed HGRN structure, which includes two key components, HGAT and GRU. Ideally, each HGRN agent can extract valuable embedding from neighbors and also recall necessary information from its memory, to construct features for decision-making. To provide an intuitive example and get some meaningful insights, we considered a special situation in Surviving. A screenshot of the considered case is shown in Fig. 6a, in which we controlled agent 0 to pass through agent 1 and agent 2. Then we inspected the graph attention weights (which describe the predicted importance of each agent's information to agent 0) and the mean value of GRU's reset gate (which controls the weight of history features to form the output embedding). The results are shown in Figs. 6b and 6c. At time 2, we observed that HGRN managed to pay the most attention to agent 1, who observed the most food. Between time 2 and time 4, in which the HGAT connection was established, the first observation is that the agent pays more attention to itself, possibly because HGRN has well stored the information from its neighbors; meanwhile, the reset gate value of GRU grows rapidly, which denotes that the model relies more on its memory to make the decision.

5.6.2 How does the self-adaptive temperature work?

To answer Q3, we first illustrated the training loop of the learnable temperature parameter α. As can be seen in Fig. 7, during the Soft-HGRN training process, when the action entropy is smaller than the target entropy, the model tends to increase α to encourage exploration and vice versa. To prevent instability, we clipped the gradient of α when it exceeded the value of 0.01α, which contributes to stable convergence of the action entropy. As for the principle behind the better performance of the stochastic policy, our insight is that it enables the agent to learn when and how to explore and exploit.

To support our idea, we considered the same Surviving situation as shown in Fig. 6a to demonstrate that our approach enables the agent to learn when and how to explore and exploit. We tested the Soft-HGRN model in this scenario and inspected its output action probability distribution and its corresponding action entropy, as shown in Figs. 6d and 6e. Note that we trained Soft-HGRN with $p_\alpha = 0.9$ and the dimension of the action space is 5; thus, the target entropy of the policy is $\mathcal{H}_{\text{target}} = 0.9 \max \mathcal{H} = 1.448\,494$. At time 0 and time 1, because agent 0 cannot observe any food or neighbor, its action probability approximated a uniform distribution that had a large action entropy. At time 2, agent 0 established GAT connection with agent 1 and agent 2, and because agent 1 observed more food, agent 0 tended to go to the right or upward to approach agent 1. At time 3, agent 1 was on the upward side; hence, "upwards" was the action with the largest probability, and the output action probability distribution had smaller action entropy. These observations demonstrate that Soft-HGRN has learned when to explore (by controlling the action entropy of the output action distribution based on the observation), and how to explore (by giving the supposed optimal action with a relatively large probability).

In addition, we noticed that the optimal action entropy factor p_α is dependent on the exploration demand of the environment. For instance, intuitively

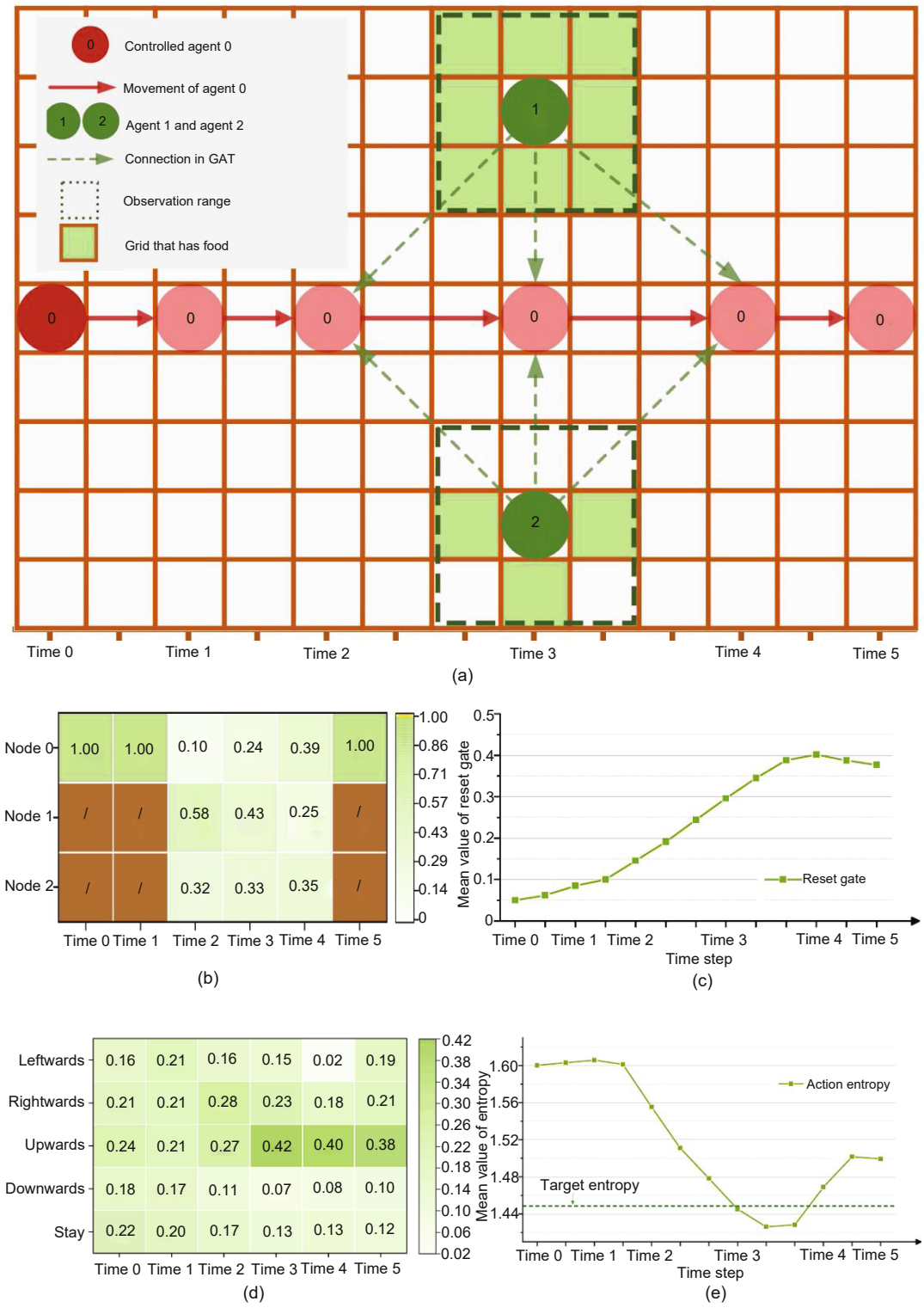

Fig. 6 An intuitive example to show how the hierarchical graph recurrent network (HGRN) extracts valuable features from neighbors and histories: (a) a screenshot of the intuitive example; (b) graph attention weights of agent 0; (c) mean value of GRU's reset gate; (d) action probability distribution of agent 0; (e) action entropy of agent 0. GAT: graph attention network; GRU: gated recurrent unit

in Surviving, we needed to consistently explore the food so the action entropy should be large; in Pursuit, a stable closure created by multiple agents is required to lock the prey, and thus the action entropy should be small. To demonstrate this intuition, and to choose the optimal entropy target p_α for our proposed stochastic policies, we performed parameter analysis to figure out how the target entropy factor p_α affects the performance in each environment, as shown in Fig. 8.

5.7 Transferability

Weak generalization ability is an urgent problem to be solved by RL algorithms. To test the generalization ability of our model, we trained all models in the Surviving environment with 100 agents, and then deployed them to environments with different agent scales for testing, as shown in Fig. 9. It can be seen that our approach performs better than the baselines under all tested agent scales. An interesting finding is that Soft-HGRN performs better when the agent scale is <100, and SAC-HGRN shows the best performance when the scale is >100. Our insight is that it is easier for Soft-HGRN to estimate the return value of a situation with a smaller agent scale, while the discriminative policy of SAC-HGRN is better when the environment becomes too unstable to estimate the return due to the ultra-large agent scale.

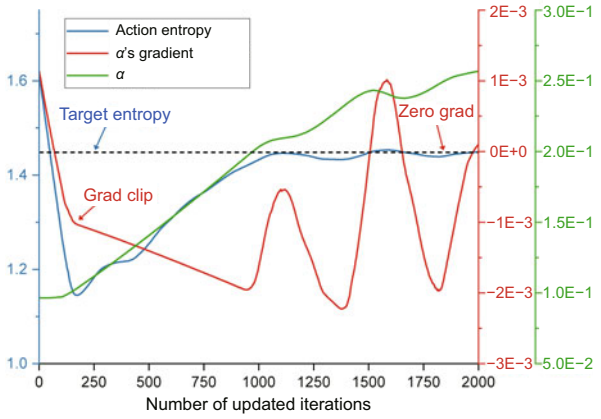

Fig. 7 Learning curves of action entropy, temperature parameter α, and its gradient $\nabla\alpha$ during the training process of Soft-HGRN in Surviving (References to color refer to the online version of this figure)

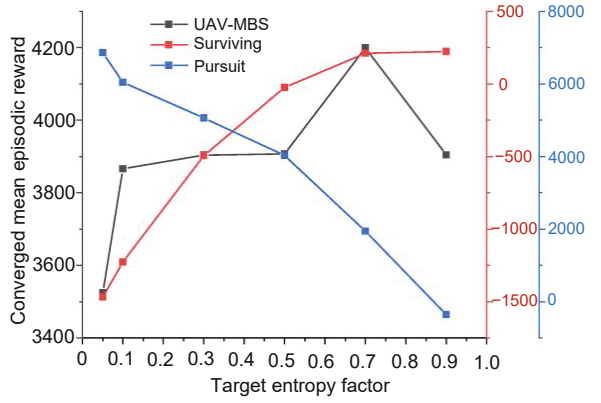

Fig. 8 Performance of Soft-HGRN with different target entropy factors in UAV-MBS, Surviving, and Pursuit (References to color refer to the online version of this figure)

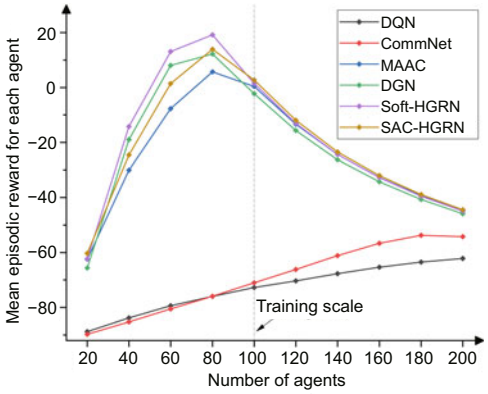

Fig. 9 Comparison of different algorithms in Surviving with various agent scales (References to color refer to the online version of this figure)

6 Conclusions

In this paper, we have proposed a value-based MADRL model (namely, Soft-HGRN) and its actor-critic variant SAC-HGRN, to address the large-scale multi-agent partially observable problem. The proposed approach consists of two key components, a novel network structure named HGRN that could aggregate information from neighbors and history, as well as a maximum-entropy learning technique that could self-adapt the temperature parameter to learn a stochastic policy with configurable action entropy. The experiments on four multi-agent environments demonstrated that the learned model outperforms other MADRL baselines, and ablation studies showed the necessity of each component in our approach. We have also analyzed the interpretability, scalability, and transferability of the learned model. As for future work, we will investigate to transfer

our methods into continuous space and implement more challenging adversarial environments, such as StarCraft II.

Contributors

Zhenhui YE, Yixiang REN, and Guanghua SONG designed the research. Zhenhui YE and Yixiang REN processed the data. Zhenhui YE drafted the paper. Yixiang REN helped organize and revise the paper. Yining CHEN, Xiaohong JIANG, and Guanghua SONG revised and finalized the paper.

Compliance with ethics guidelines

Yixiang REN, Zhenhui YE, Yining CHEN, Xiaohong JIANG, and Guanghua SONG declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Adler JL, Satapathy G, Manikonda V, et al., 2005. A multi-agent approach to cooperative traffic management and route guidance. *Trans Res Part B*, 39(4):297-318. https://doi.org/10.1016/j.trb.2004.03.005

Cho K, van Merriënboer B, Gulcehre C, et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proc Conf on Empirical Methods in Natural Language Processing, p.1724-1734. https://doi.org/10.3115/v1/D14-1179

Chu TS, Wang J, Codecà L, et al., 2020. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Trans Intell Transp Syst*, 21(3):1086-1095. https://doi.org/10.1109/TITS.2019.2901791

Claus C, Boutilier C, 1998. The dynamics of reinforcement learning in cooperative multiagent systems. Proc 15th National Conf on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conf, p.746-752.

Foerster JN, Assael YM, de Freitas N, et al., 2016. Learning to communicate with deep multi-agent reinforcement learning. Proc 30th Int Conf on Neural Information Processing Systems, p.2145-2153.

Haarnoja T, Tang HR, Abbeel P, et al., 2017. Reinforcement learning with deep energy-based policies. Proc 34th Int Conf on Machine Learning, p.1352-1361.

Haarnoja T, Zhou A, Abbeel P, et al., 2018. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. Proc 35th Int Conf on Machine Learning, p.1861-1870.

Hausknecht M, Stone P, 2015. Deep recurrent Q-learning for partially observable MDPs. Proc AAAI Fall Symp Series, p.29-37.

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

Iqbal S, Sha F, 2019. Actor-attention-critic for multi-agent reinforcement learning. Proc 36th Int Conf on Machine Learning, p.2961-2970.

Jiang JC, Dun C, Huang TJ, et al., 2020. Graph convolutional reinforcement learning. Proc 8th Int Conf on Learning Representations.

Kingma DP, Ba J, 2015. Adam: a method for stochastic optimization. Proc 3rd Int Conf on Learning Representations.

Lillicrap TP, Hunt JJ, Pritzel A, et al., 2015. Continuous control with deep reinforcement learning. Proc 4th Int Conf on Learning Representations.

Lin LJ, 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach Learn*, 8(3-4):293-321. https://doi.org/10.1007/BF00992699

Lowe R, Wu Y, Tamar A, et al., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Proc 31st Int Conf on Neural Information Processing Systems, p.6382-6393.

Mnih V, Kavukcuoglu K, Silver D, et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533. https://doi.org/10.1038/nature14236

Mnih V, Badia AP, Mirza M, et al., 2016. Asynchronous methods for deep reinforcement learning. Proc 33rd Int Conf on Machine Learning, p.1928-1937.

Rashid T, Samvelyan M, Schroeder C, et al., 2018. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. Proc 35th Int Conf on Machine Learning, p.4295-4304.

Rui P, 2010. Multi-UAV formation maneuvering control based on Q-learning fuzzy controller. Proc 2nd Int Conf on Advanced Computer Control, p.252-257. https://doi.org/10.1109/ICACC.2010.5486961

Ryu H, Shin H, Park J, 2020. Multi-agent actor-critic with hierarchical graph attention network. Proc AAAI Conf Artif Intell, p.7236-7243. https://doi.org/10.1609/aaai.v34i05.6214

Sukhbaatar S, Szlam A, Fergus R, 2016. Learning multi-agent communication with backpropagation. Proc 30th Int Conf on Neural Information Processing Systems, p.2252-2260.

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. Proc 31st Int Conf on Neural Information Processing Systems, p.6000-6010. https://doi.org/10.5555/3295222.3295349

Veličković P, Cucurull G, Casanova A, et al., 2018. Graph attention networks. Proc 6th Int Conf on Learning Representations.

Wang RE, Everett M, How JP, 2020. R-MADDPG for partially observable environments and limited communication. https://doi.org/10.48550/arXiv.2002.06684

Watkins CJCH, Dayan P, 1992. *Q*-learning. *Mach Learn*, 8(3-4):279-292. https://doi.org/10.1007/BF00992698

Ye DY, Zhang MJ, Yang Y, 2015. A multi-agent framework for packet routing in wireless sensor networks. *Sensors*, 15(5):10026-10047. https://doi.org/10.3390/s150510026

Ye ZH, Chen YN, Jiang XH, et al., 2022a. Improving sample efficiency in multi-agent actor-critic methods. *Appl Intell*, 52(4):3691-3704. https://doi.org/10.1007/s10489-021-02554-5

Ye ZH, Wang K, Chen YN, et al., 2022b. Multi-UAV navigation for partially observable communication coverage by graph reinforcement learning. *IEEE Trans Mobile Comput*, early access. https://doi.org/10.1109/TMC.2022.3146881

Zhang KQ, Yang ZR, Liu H, et al., 2018. Fully decentralized multi-agent reinforcement learning with networked agents. Proc 35th Int Conf on Machine Learning, p.5872-5881.

Zhang Y, Mou ZY, Gao FF, et al., 2020. UAV-enabled secure communications by multi-agent deep reinforcement learning. *IEEE Trans Veh Technol*, 69(10):11599-11611. https://doi.org/10.1109/TVT.2020.3014788

Zheng LM, Yang JC, Cai H, et al., 2018. MAgent: a many-agent reinforcement learning platform for artificial collective intelligence. Proc 32nd AAAI Conf on Artificial Intelligence, p.8222-8223.

List of supplementary materials