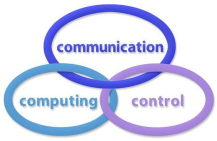
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 18, Issue: 1, Month: February, Year: 2023
Article Number: 5062, https://doi.org/10.15837/ijccc.2023.1.5062



CCC Publications

A Graph-Based Soft Actor Critic Approach in Multi-Agent Reinforcement Learning

Wei Pan, Cheng Liu

Wei Pan
School of Computer Science
Northwestern Polytechnical University, China
Xi'an 710072, China
panwei@nwpu.edu.cn

Cheng Liu
School of Computer Science
Northwestern Polytechnical University, China
Xi'an 710072, China
lion_liucheng@163.com

Abstract

Multi-Agent Reinforcement Learning (MARL) is widely used to solve various real-world problems. In MARL, the environment contains multiple agents. A good grasp of the environment can guide agents to learn cooperative strategies. In Centralized Training Decentralized Execution (CTDE), a centralized critic is used to guide cooperative strategies learning. However, having multiple agents in the environment leads to the curse of dimensionality and influence of other agents' strategies, resulting in difficulties for centralized critics to learn good cooperative strategies. We propose a graph-based approach to overcome the above problems. It uses a graph neural network, which uses partial observations of agents as input, and information between agents is aggregated by graph methods to extract information about the whole environment. In this way, agents can improve their understanding of the overall state of the environment and other agents in the environment while avoiding dimensional explosion. Then we combine a dual critic dynamic decomposition method with soft actor-critic to train policy. The former uses individual and global rewards for learning, avoiding the influence of other agents' strategies, and the latter help to learn an optional policy better. We call this approach Multi-Agent Graph-based soft Actor-Critic (MAGAC). We compare our proposed method with several classical MARL algorithms under the Multi-agent Particle Environment (MPE). The experimental results show that our method can achieve a faster learning speed while learning better policy.

Keywords: Multi-Agent System, Deep Reinforcement Learning, Graph Neural Network, Soft Actor-Critic.

1 Introduction

Reinforcement Learning (RL) learns strategies by interacting with the environment[1, 2]. In recent years, deep learning has been widely used in various fields, such as garbage classification[3], fault detection[4], stock prediction[5], image processing[6, 7], etc. Combining deep learning and reinforcement learning effectively solves the curse of dimensionality in reinforcement learning. It enables reinforcement learning to be widely used in various complex tasks. Deep Reinforcement Learning (DRL) has achieved excellence in areas such as Go[8] and robot control[9] especially in video games[10], even beyond human performance. DRL is currently used for many more complex real-world tasks, such as control design for unmanned aerial vehicle[11], resilient road network recovery[12] and navigation control[13]. Multi-agent reinforcement learning is a branch of RL that aims to jointly control multiple agents in an environment so that agents learn to cooperate with other agents to accomplish tasks. Many algorithms in MARL are influenced by high-dimensional state and action spaces because it is difficult for agents to learn good cooperation policies in high-dimensional state spaces. The combination of MARL and deep learning can overcome the problem of high-dimensional state spaces to some extent due to the powerful representational capabilities of deep learning. Nevertheless, it is still an open question of how to process a large amount of information in an environment with many agents so that agent learns good cooperation strategies.

To solve the problems mentioned above, many MARL methods have been proposed. Using a single agent-based RL algorithm[14] to train each agent independently is a natural idea. However, in this approach, agents treat other agents as part of the environment. In such an unstable environment, it is difficult for independent single-agent RL algorithms to learn good policies, let alone how to collaborate with other agents. Multi Agent Deep Deterministic Policy Gradient (MADDPG)[15] proposes an actor-critic architecture-based MARL algorithm based on Deep Deterministic Policy Gradient (DDPG), which learns the cooperation of multiple agents through a centralized training, decentralized execution paradigm. MADDPG constructed a centralized critic, which uses the observations and actions of all agents as input to assess the value. Actors in decentralized implementation use their observations as input to learning collaborative strategies under the guidance of centralized critics. This paradigm is known as centralized training and decentralized execution. In one case, the good action of the current agent may not achieve good returns because of the action of other agents. COMA[16] further proposes a CTDE-based counterfactual baseline approach to predict a virtual return as a baseline. With a counterfactual baseline, COMA achieves a credit assignment under multi agents, avoiding strategic interference from other agents and helping agents learn cooperative strategies while minimizing the influence of other agents as much as feasible.

MADDPG and COMA need to use all agents' state and observation data when constructing their centralized critic networks. As the number of agents increases and a large amount of information is used to evaluate the network, it becomes difficult for the critic network to focus on the most important parts among a large amount of redundant information. The network becomes increasingly difficult to be trained. When the number of agents increases, in order to filter out the most important information for the current agent from a large amount of information, MAAC[17] introduces an attention mechanism to the centralized critic network to solve this problem. MAAC takes the state of the current agent as a query and the state of other agents as the key. The states and actions of other agents is the value. Attention weights are calculated by comparing the states between different agents. The attention weights are used to weigh the states and actions of all other agents to obtain highly encoded global information, which is used for value assessment by critics after concatenating the observations of the current agent. MFMARL[18] proposes the mean-field approach, which solves the problem of strategic interference of large-scale agents from another perspective. MFMARL proposes the hypothesis of a multi-intelligent system: For a particular agent, the effect of all other agents on it can be replaced by an average effect. The interaction between an agent and its neighboring agents is considered an interaction between two agents. ATOC[19] proposes a soft attention-based communication mechanism to decide when an agent should communicate with other agents.

Graph neural network (GNN)[20] is a special kind of neural network structure designed for tasks with graphical relations. GNN was originally used for social networking, point clouds, and other issues. In recent years, some advanced work has introduced GNN into drug development and computer vision

and achieved remarkable results. Many real-world problems, such as social networks and point clouds, can be constructed as graphs. A multi-agent system can use a graph to depict the relationships between the agents. Due to this unique ability of GNN to be compatible with multi-agent systems, the powerful information construction capability of the graph has attracted the attention of many MARL researchers. Sequence invariance means that changing the agent order will give the same output value in a centralized critic network. However, the critic network of multilayer perceptron (MLP) used in MADDPG uses a concatenation of all agents' observations and actions, so it does not have sequence invariance. PIC[21] experimentally demonstrated that changing the order of different agents in a sample can improve sample efficiency. The method constructs a centralized critic using GNN, which achieves close to MLP and performs better in scenarios with a larger number of agents. GCS[22] uses GNN to generate a cooperation graph that identifies the neighboring agents that each agent needs to interact with and then forms a graph-based cooperation policy based on this.

In some environments where individual rewards are available, direct single-agent reinforcement learning for training treats all other agents as part of the environment. Although strategic interference is avoided in CTDE, in this case, while simple tasks can be learned, good cooperative strategies cannot be learned. DE-MADDPG[23] uses two critics to focus on individual and collective rewards, combined with the MADDPPG method to avoid strategy interference effectively. DD-MADDPG[24] proposes a dynamic decomposition method based on DE-MADDPG, which gives different weighted attention to the two assessment networks at different stages.

The performance of methods such as MADDPG can break down due to strategic interference, while GNN-based methods can overcome this problem very well. We propose a GNN-based approach to overcome inter-agent strategic interference. The method consists of three components based on a stochastic policy, which includes partial observation encoder for the homogeneous agent, graph-based state encoder, maximum entropy reinforcement learning combined with the dynamic decomposition of the dual critic. A partial observation encoder encodes partial observations of a homogeneous agent using an encoder with the same parameters, mapping them to a uniform dimension for easy processing by subsequent modules. In this module, information from other agents is aggregated to the node after propagation of the K-layer graph. The GNN in our state encoder takes its inspiration from GraphSage[25]. At each layer, the summary information of the upper layer features of this node is concatenated with the upper layer features of other nodes. Through this GNN state encoding module, the state information of other agents can be aggregated. When facing many agents, the proposed method can avoid curse of dimensionality, maintain good feature extraction ability, and avoid strategic interference from other agents. Soft Actor-Critic (SAC)[26] is a stochastic policy algorithm that introduces maximum entropy reinforcement learning. The deterministic policy algorithm learns a policy such that the cumulative reward expectation is maximized. As for the SAC, in addition to the above basic objectives, it requires that the action entropy of each output of the policy is maximized. That is, the probability of each output action is spread out as much as possible rather than being concentrated on one action.

Our contributions are as follows:

(1) A graph-based state encoder that extracts agent states in a multi-agent environment while avoiding the curse of dimensionality. The features extracted by this module that contain a good grasp of the environment lead to good cooperation strategies.

(2) Based on (1), a combination of soft actor criticism and dynamic decomposition dual critic methods is proposed. The method trains the optimal cooperative strategy while avoiding the strategic influence of other agents.

We tested our proposed MAGAC with two classical algorithms of MARL in an MPE environment. We examined the algorithm's robustness in relation to the number of agents. Experiments show that the results of our proposed MAGAC method outperform other algorithms.

The rest of the article will be organized as follows. The second section will present all the prerequisite knowledge used. The third section will present our approach. The fourth section will compare our approach with other MARL algorithms in a simulation environment. Finally, the fifth section will give the conclusion.

2 Background

2.1 Markov games and multi-agent reinforcment learning

The Markov decision process can be extended to partially observable Markov decisions when the number of agents is more than a single. In a Markov game with N agents at time t, There are defined with state s^t, which describes the status of the entire environment at time t. The MA system gets the joint observations of time t: $\mathbf{o}^t = (o_1^t, \cdots, o_i^t, \cdots, o_N^t)$form state s, which is partial information of state s^t. Agents in the environment take joint action: $\mathbf{a}^t = (a_1^t, \cdots, a_i^t, \cdots, a_N^t)$. After taking action, the environment can be moved to a new state according to a state transfer function: $T = P\left(s^{t+1} | s_t \times a_1^t \times \ldots \times a_N^t\right)$. in the next time. As states are changing, rewards are given in the environment in relation to the preceding and following states and the action being taken $r^t : s^t \times a_1^t \times \ldots \times a_N^t \to R$, which joint reward:$\mathbf{r}^t = (r_1^t, \cdots, r_i^t, \cdots, r_N^t)$. In an experimental scenario with fully shared rewards, $r_1^t = \cdots = r_i^t = \cdots = r_N^t$. Each agent learns a policy $\pi_i : o_i \to P\left(a_i\right)$ that maximizes the cumulative expected reward based on the observed selection action. where the cumulative expected reward is as:

$$J_i\left(\pi_i\right) = E_{a_1 \sim \pi_1, \ldots, a_N \sim \pi_N, s \sim T}\left[\sum_{t=0}^{\infty} \gamma^t r_i^t\left(s_t, a_1^t, \ldots, a_N^t\right)\right] \tag{1}$$

Where $\gamma \in [0, 1]$ determines the strategy's balance of forwarding and near-term rewards. The state transfer function $T = P\left(s^{t+1} | s_t \times a_1^t \times \ldots \times a_N^t\right)$ determines the reward. From the equation, we can see that the policies of other agents influence agent i in the learning process because the joint action of all agents determines the state transfer from moment t to t+1, which is one of the core problems in MARL. Off-policy reinforcement learning uses a replay buffer to improve sample utilization. The system interacts to generate transactions and samples the transactions in the experience pool to update the parameters of the policy.

2.2 Maximum entropy reinforcement learning and soft actor-critic

We discuss maximum entropy reinforcement learning and soft actor-critic in the context of single agent reinforcement learning.

2.2.1 maximum entropy reinforcement learning

Unlike standard reinforcement learning that maximizes the expected return $\sum_t E_{(s_t, a_t) \sim \rho_\pi}\left[r\left(s_t, a_t\right)\right]$, maximum entropy reinforcement learning adds a political entropy term to the expected return which mean that:

$$J\left(\pi\right) = \sum_{t=0}^{T} E_{(s_t, a_t) \sim \rho_\pi}\left[r\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)\right] \tag{2}$$

the term $\mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)$ implies a more uniform action of policy output in the face of the same state. It encourages the exploration of policy rather than resting on local optimal policy, which means that the strategy can learn more near-optimal actions. So there may be more than one action that is optimal in some states. The more uniform the probability distribution of the optimal action, the global optimal policy can be learned. The parameter α controls the importance of the relative reward to the stochastic policy. The larger the parameter α, the more the policy converges to the optimal policy and the objective changes to classical reinforcement learning when $\alpha \to 0$.

2.2.2 Soft Actor-Critic

The Actor-critic method[27] uses neural networks Q to approximate the expected returns, effectively overcoming the large variance problem of the expected returns. The neural network Q can also be called ad critic. The learning of critic network can be minimized by minimizing the following loss through the Td-error method:

$$\mathcal{L}_Q\left(\theta^Q\right) = E_{(s,a,r,s')\sim B}\left[\left(Q_{\theta^Q}(s,a) - y\right)^2\right] \tag{3}$$

$$y = r(s,a) + \gamma E_{a'\sim\pi(s')}\left[Q_{\theta^{\bar{Q}}}\left(s', a'\right)\right] \tag{4}$$

where B denotes the mini-batch of experiences sampled from the replay buffer in the off-policy RL. $Q_{\theta^{\bar{Q}}}$ denotes the target Q-network[28], which is used to maintain algorithm stability. The same approach is not used in the policy network. With the help of a critic network, policy learning with Actor-critic can be expressed as:

$$\nabla_{\theta^\pi} J\left(\pi_{\theta^\pi}\right) = E_{s\sim D, a\sim\pi}\left[\nabla_{\theta^\pi} log\left(\pi_{\theta^\pi}(a|s)\right) Q_{\theta^Q}(s,a)\right] \tag{5}$$

Actor-critic can be improved to Soft Actor-Critic by adding the maximum entropy term to the learning objective of the policy network.

$$\nabla_{\theta^\pi} \mathrm{J}\left(\pi_{\theta^\pi}\right) = E_{s\sim D, a\sim\pi}\left[\nabla_\theta \log\left(\pi_{\theta^\pi}(a \mid s)\right)\left(-\alpha \log\left(\pi_{\theta^\pi}(a \mid s)\right) + Q_\theta(s,a)\right)\right] \tag{6}$$

2.3 Graph neural networks

Unlike MLPs, graph neural networks are deep networks that operate on graph structures. The inputs to the network are nodes with a set of edges. In MADDPG's Critic, swapping observation of two other agents changes the network's input, while GNN has Sequence Invariance, which improves sample efficiency. There are currently many variants of the GNN approach. Graph Convolutional Nets[29], GraphSAGE, which uses a limited number of interacting nodes, allowing training to extract some subgraphs from the entire set for mini-batch, extending the application of the graph set approach to large-scale graph data. GAT improves the graph propagation mechanism by using the attention mechanism, which gives different attention to different graph nodes. The input graph of GNN can be expressed as $G = (V, E)$. That is, we define a Graph G as a set of nodes V, with a set of edges E connecting them. E represents the connection relation on the graph. Each node can be represented as a vector h. GNN iterates over the node vector and aggregates node information to update node h in each iteration.

$$h_i^{(k)} = \sum_{j\in N(i)} FC^{(k)}\left(h_i^{(k-1)}, h_j^{(k-1)}\right) \tag{7}$$

3 Algorithmic contribution

MAGAC is based on a maximum entropy Actor-critic method. It consists of three parts: a partial observation encoder, a GNN-based state encoder, and an Actor-critic based on maximum entropy and dynamic decomposition methods. We will introduce each of these three parts and the whole model training method in this section.

3.1 Problem formulation and partial observation encoder

In the MPE environment, an agent's observation consists of several components. As shown in Figure 1, the blue circles indicate the agents, and the black circles indicate landmarks in the environment. All agents and landmarks are contained within a two-dimensional graph.

All agents and landmarks are randomly initialized in each episode. Each agent can obtain its absolute position relative to origin and velocity. The absolute position of agent i is $p_i = (p_i^x, p_i^y)$, which contains two directions of position. Similarly, the velocity of A is expressed as $v_i = (v_i^x, v_i^y)$.The relative positions of landmarks O_i and A are denoted as $p_{oj}^i = \left(p_{oj}^{ix}, p_{oj}^{iy}\right)$.The positions of all landmarks in the environment relative to A are expressed as $p_o^i = (p_{o1}^i, \ldots p_{oi}^i \ldots p_{oM}^i)$. M denotes the number of landmarks observed in the environment by agent i. Similarly, the position of other agents relative to A can be expressed as $p_a^i = \left(p_{a1}^i, \ldots p_{aj}^i \ldots p_{aN}^i\right)$. N denotes the number of agents that can be observed

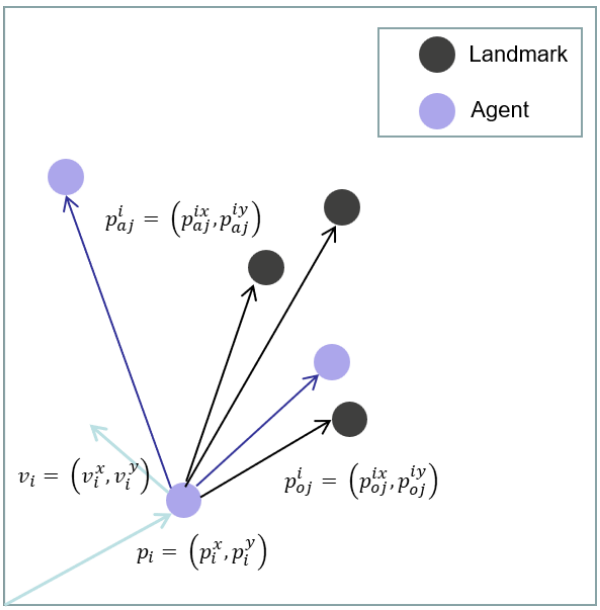


Figure 1: Description of the environment.

in the environment by agent i. In the fully observed setting, it is also possible to obtain the relative positions of other agents in the environment by agent i. The physics engine model of the environment is a second-order integral model, the environment has inertia, and agent collision with landmarks will generate mutual forces. In a scenario with N agents and M landmarks, the observation of agent i can be expressed as o_i. $o_i = [p_i, v_i, p_o^i, p_a^i]$ in methods such as MADDPG and MAAC that use full observation. We use partial of the observations $o_i = [p_i, v_i, p_o^i, p_a^i]$, as initial graph node information, and the information between agents is summarized by the graph. The graph uses graph feature nodes of uniform dimension. In contrast, the observation dimension is different for different agents, so the observations of agents need to be preprocessed by an partial observation encoder and compressed to a fixed size and length. The partial observation encoder uses the same parameter for the homogeneous agent. The encoder can be expressed as follow:

$$h_i = e\left(o_i | w_c\right) \tag{8}$$

Where c indicates the category to which the agents belong. The parametric scale of the observation encoder is only related to the number of classes of agents, so it does not increase as the number of agents increases.

3.2 Graph based state encoder

In RL systems, the agent interacts with the environment to maximize the expected reward continuously. Because there are other agents in the MARL environment; the environment becomes more unstable, which makes learning cooperative strategy more difficult. During the graph propagation accompanied by message passing between nodes, GNN aggregates information from other agents in the environment and iterates into a new feature h. Unlike the classical GNN, our state encoder is inspired by GraphSage and uses concatenation in each layer of iteration. The state encoder takes the output of the partial state encoder h_i as the input node of the graph h_i^0. State encoder updates graph nodes using K aggregation layers: $\left\{f^{(1)}, \ldots, f^{(K)}\right\}$. Correspondingly there are K aggregation functions, denoted by AGGREGATE $_k, \forall k \in \{1, \ldots, K\}$ which aggregates information from other agents. These aggregate functions will aggregate information from other agents.

Two more sets of weights are needed in each layer iteration:W_v^k, $\forall k \in \{1, \ldots, K\}$, and W_f^k, $\forall k \in \{1, \ldots, K\}$. The former is used to process the aggregated information, while the latter is used to process the concatenated information.

Algorithm1 describes the processing of the graph state encoding module: in each iteration, nodes

Algorithm 1 Graph-based state encoder.

Define:
input features : $h_i, \forall i \in \mathcal{N}$
K := times of GNN iteration
$W_v^k, \forall k \in \{1, \ldots, K\}$
$W_f^k, \forall k \in \{1, \ldots, K\}$
$\sigma := non - linearity$
$AGGREGATE_k, \forall k \in \{1, \ldots, K\}$
$h_i^0 \leftarrow h_i \ \forall i \in \mathcal{N}$
for k = 1...K do:
for i in $\mathcal{N}$:
$h^k \leftarrow AGGREGATE_k \left(W_V^k h_j^{k-1}, \forall j \in \mathcal{N}\right)$
$h_i^k \leftarrow \sigma\left(W^k \cdot CONCAT\left(h_i^{k-1}, h_N^k\right)\right)$
end for
end for
Output := $s_i \leftarrow h_i^K, \forall i \in \mathcal{N}$

aggregate information from their other nodes, and as this process iterates, the nodes gain more information from other agents in the environment. k denotes the number of graph iterations, and we choose K = 3. To compute the aggregation we use average function, which brings together the vectors of all other nodes on average. This aggregation is the same as the convolutional propagation of GCN. After aggregation, h^{k-1} is concatenated with h_i^{k-1} after a full concatenation, and the concatenated features are subjected to a full concatenation and activation function. The obtained output is used as the new node feature h_i^k. The node feature h_i^K after K iterations is used as the output s_i of the whole module. The only parameters that need to be trained in this module are $W_v^k, \forall k \in \{1, \ldots, K\}$ and $W_f^k, \forall k \in \{1, \ldots, K\}$ for each layer, which has the advantage that the number of parameters does not increase rapidly when the number of agents increases.

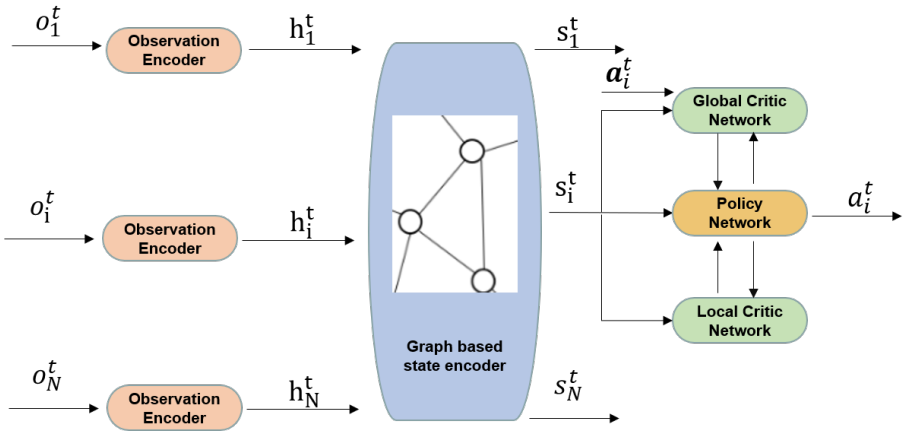


Figure 2: Overall architecture of the algorithm.

3.3 SAC with dynamic decomposition of dual-critic

In many scenarios, the agent's individual rewards are available. To reduce the impact of other agents' policies on the current agent training, based on GNN's state encoder module, we will introduce the dynamic decomposition of the dual-critic method. The algorithm uses individual reward critics and collective reward critics to evaluate policy. In the early stages of training, more attention is given to individual rewards, and agents are instructed to learn simple tasks first. During this process, the weight of collective rewards increases, and agents are instructed to learn teamwork. The importance of the dual-critic changes dynamically in the training process. It enables agents to transition from individual

to collective learning adaptively. We set up two critics. One uses a single agent's observations to evaluate actions, called the local critic. The local critic uses s_i to evaluate the individual rewards for an action. Another critic, called the global critic, evaluates the collective reward of an action using s_i and the actions of all other agents as input. Dynamic adjustment of the importance of two critics to guide policy learning. The gradient of local critics is

$$\nabla^l_{\theta^\pi_i} J\left(\pi_{\theta^\pi_i}\right) = E_{a\sim\pi_{\theta^\pi_i}}\left[\nabla_{\theta^\pi_i} log\left(\pi_{\theta^\pi_i}\left(a_i|s_i\right)\right)\left(\alpha\log\left(\pi_{\theta^\pi_i}\left(a_i \mid s_i\right)\right) - Q_i\left(s_i, a_i\right)\right)\right] \tag{9}$$

The gradient of global critics is

$$\nabla^g_{\theta^\pi_i} J\left(\pi_{\theta^\pi_i}\right) = E_{a\sim\pi_{\theta^\pi_i}}\left[\nabla_{\theta^\pi_i} log\left(\pi_{\theta^\pi_i}\left(a_i|s_i\right)\right)\left(\alpha\log\left(\pi_{\theta^\pi_i}\left(a_i \mid s_i\right)\right) - Q^g_{\theta^Q}\left(s_i, a\right)\right)\right] \tag{10}$$

Total gradient:

$$\nabla_{\theta^\pi_i} J\left(\pi_{\theta^\pi_i}\right) = w_1 * \nabla^g_{\theta^\pi_i} J\left(\pi_{\theta^\pi_i}\right) + w_2 * \nabla^l_{\theta^\pi_i} J\left(\pi_{\theta^\pi_i}\right) \tag{11}$$

w_1 and w_2 denote attention to different critics, respectively. w_2 is initialized to 1. and decays with the training process.

$$w_2 = w_2 * \text{ decay} \quad and \quad w_1 = 1 - w_2 \tag{12}$$

The updates to the local critics are as follows

$$\mathcal{L}_{Q_i}\left(\theta^Q_i\right) = E_{s,a,r,s'}\left[\left(Q_{\theta^Q_i}\left(s_i\right) - y_i\right)^2\right] \tag{13}$$

Where

$$y_i = r_i + \gamma\left[Q_{\theta^Q_i}\left(s'_i, a'_i\right) - \alpha\log\left(\pi_{\theta^\pi_i}\left(a'_i \mid s'_i\right)\right)\right] \tag{14}$$

The updates to the global critic are as follows

$$\mathcal{L}_{Q^g}\left(\theta^Q\right) = E_{s,a,r,s'}\left[\left(Q^g_{\theta^Q}\left(s_i, a_i\right) - y_g\right)^2\right] \tag{15}$$

Where

$$y_g = r_g + \gamma\left[Q^g_{\theta^Q}\left(s'_i, a'_i\right) - \alpha\log\left(\pi_{\theta^\pi_i}\left(a'_i- \mid s'_i\right)\right)\right] \tag{16}$$

Algorithm 2 Graph-Based Soft Actor Critic for MARL.

Initialize all network and replay buffer
for episode e = 1 to T do:
  for time steps t = 1 to episode length do:
    Get union observation $\boldsymbol{o}^t$ from environment
    Get $\boldsymbol{h}^t$ by Equation 8
    Get $\boldsymbol{s}^t$ by Algorithm 1
    For each agent i, select action $a^t_i = \pi_{\theta^\pi_i}\left(s^t_i\right)$
    Execute joint action $\boldsymbol{a}^t$ and get union observation $\boldsymbol{o}^{t+1}$ and $\boldsymbol{r}^t$
    Store transaction to replay buffer
  if e % d == 0 then:
    For each agent i update local critic by minimizing Equation 13
    Update global critic by minimizing Equation 13
    For each agent i update policy according Equation 11
    Update w_1 and w_2 according Equation 12
    Update all target network parameters

The overall architecture of the algorithm is shown in Figure 2. $\boldsymbol{a}^t_i$ donates the joint action at time t except for the action of agent i. The partial observation gets preprocessed to h^t_i, which has a fixed

dimension. Then graph-based state encoder uses $\boldsymbol{h}^t$ extract $\boldsymbol{s}^t$. Policy network gets action by s_i^t and is trained with the dynamic decomposition of duel critic. The detailed training process is shown in Algorithm 2.

4 Experiments

This section describes the setup of the experiment, the baseline used for comparison, and the hyperparameters used in the experiment. We tested and evaluated our algorithm in multi-agent particle environment. A diagram of the application of MAGAC in the environment is shown in figure3.

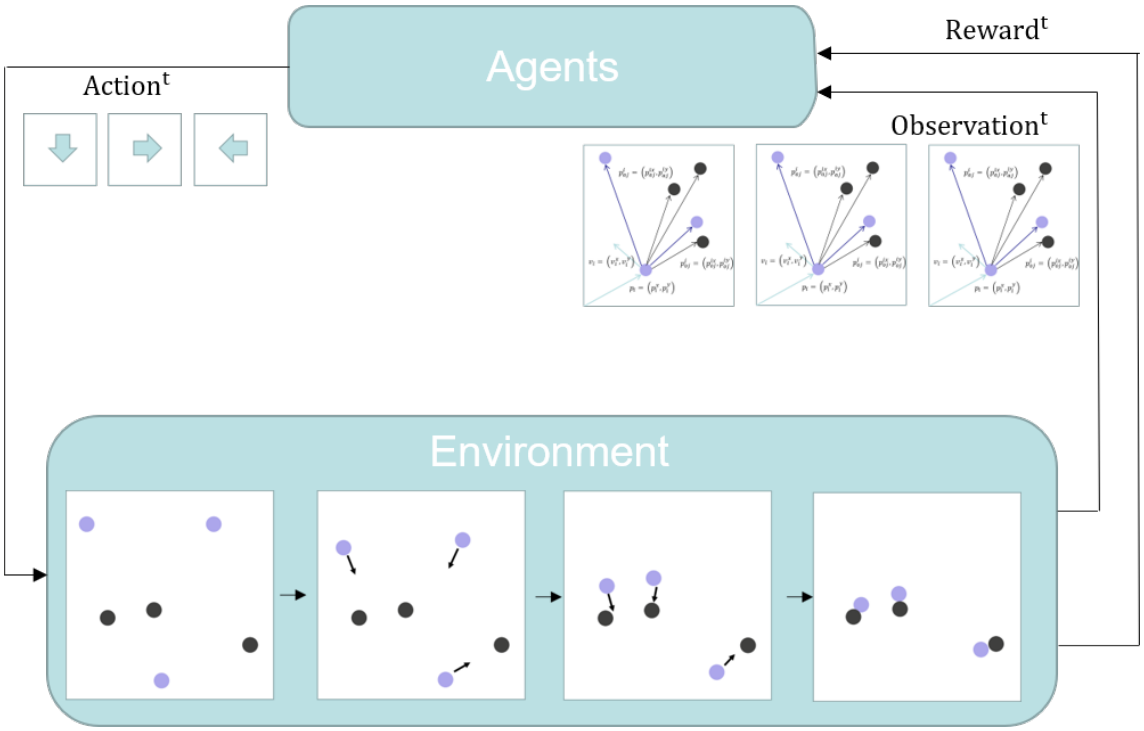
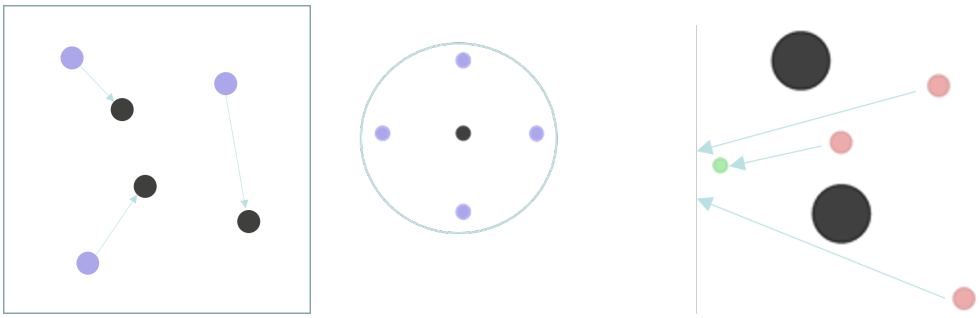


Figure 3: Overall architecture of the algorithm.

4.1 Experimental environment

We tested three scenarios under MPE, with the following details for each scenario, as shown in Figure4. The scenarios focus on target tracking and formation and can be used as a reference for practical applications.



(a) Cover control (b) Formation Control (c) Predator-prey

Figure 4: An overview of the MPE environments used in the experiments.

4.1.1 Cover control

Here there are N agents and N landmarks. Agents are rewarded if they approach the corresponding landmarks and punished if they collide with other agents. Agents need to learn to cover all the landmarks while avoiding collisions with other agents.

4.1.2 Formation Control

Here there are N agents and one landmark, and the agents are distributed as evenly as possible around the landmark to form an equilateral polygon. The reward increases according to the similarity of the agents to the equilateral polygon.

4.1.3 Predator-prey

Predator-prey scenario is means chase, with three red hunters, several black roadblocks, and red prey. The prey takes inspired action, and the hunters need to learn to cooperate to complete the chase of the prey. If the hunters collide with each other, they are punished. Each agent can see the relative positions of the other agents. The prey is faster than the hunters, so the hunters need to learn to cooperate.

4.1.4 Experimental Details setting

Table 1: The parameters used for all algorithms in the experiments.

	MASAC	MAAC	MADDPG
buffer size	1e6	1e6	1e6
mini batch size	128	128	128
lr-actor	0.001	0.001	0.001
lr-critic	0.001	0.001	0.001
decay	0.001	0.001	0.001
gamma	0.99	0.99	0.99
Temperature parameters α	0.02	0.02	0

MADDPG and MAAC are both classical MARL algorithm. All three experimental algorithms are based on the AC structure. MAAC and MAGAC introduce a maximum entropy reinforcement learning method based on the AC structure. All three algorithms use the same training method and hyperparameter settings, and all employ techniques such as target networks to verify the effectiveness of our methods. The parameters are set as shown in the table 1. Each hidden layer consists of 128 cells. We use the adam optimizer to train all neural networks, which automatically adjusts the learning rate during training to improve the stability of the training. For all scenarios, we used a length such that each scene had 25 steps and trained 5000 episodes for each scenarios. For each scenario, five random seeds were run.

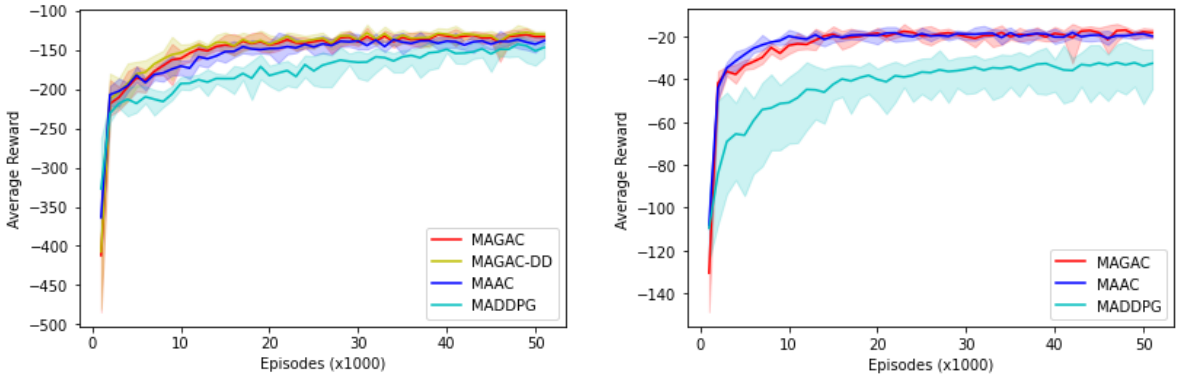
4.2 Result and analysis

We evaluate the algorithm's performance mainly regarding learning stability and global reward. The global reward plot can directly demonstrate how effectively the agent learns the policy during the different training phases. In addition, the average reward and confidence interval of the last episode can be digitized as a better indicator of the final strategy learned by the algorithm. To verify the robustness of the proposed method when the number of agents increases, we compare the algorithm's performance by increasing the number of agents in the same experimental scenario.

Figures 5, 6 and 7 show how the global reward of the agents varies with the training process. The lighter area is the error band under the five random seeds, and the darker area is the average of the five experiments. Table 2 shows the last episode's average reward and confidence interval for the three

Table 2: The mean global reward in the last episodes on 1000 times. In scenarios A and B, where individual rewards are available, we tested MAGAC in combination with the Dual critic Dynamic decomposition method, named MAGAC-DD. * means increasing the number of agents in the same scenario. Cover* and Formation* contain 10 agents, Predator-prey* contains 8 agents

	MASAC	MASAC-DD	MAAC	MADDPG
Cover	-133.22	-131.02	-138.4	-147.09
Formation	-18.18		-19.84	-31.03
Predator-prey	128.12	141.30	67.32	82.24
Cover*	-1052.87	-1011.04	-1507.21	-1159.07
Formation*	-47.23		-52.75	-88.31
Predator-prey*	604.43	759.09	376.53	236.49

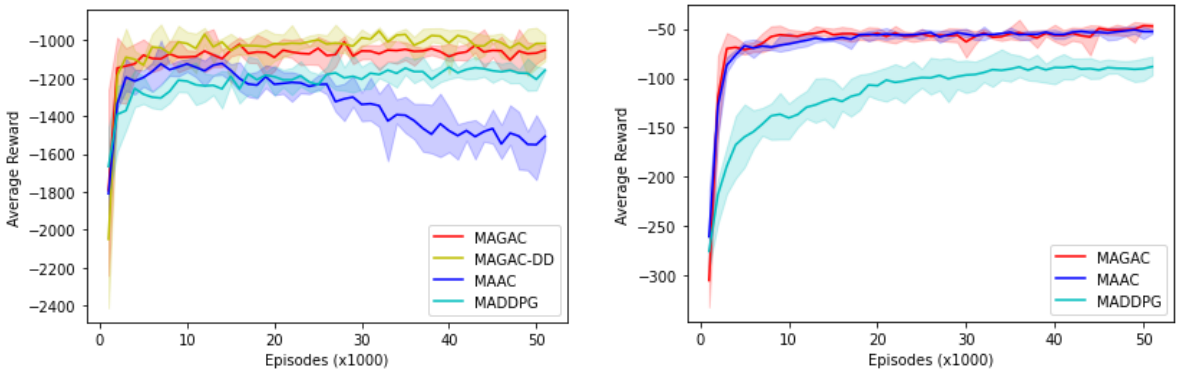


(a) Cover control (b) Formation Control

Figure 5: Graph of average global rewards with training episodes.

algorithms under different experimental scenarios. Overall, our MAGAS algorithm's learning stability and global reward generally outperformed the baseline algorithm to a greater extent.

For the Cover task, as shown in Figure 5a, MAGAC slightly outperforms the MADDPG method in terms of global reward and has some advantages compared to MAAC. In the global average reward metric of the final episode, MAGAC achieves -133.22, MAAC achieves -138.40, and MADDPG achieves 147.90. MADDPG is significantly slower than the other two in convergence speed, and MAAC is closer to our MASAC. It is because MAAC and MAGAC use the maximum entropy reinforcement learning method. Meanwhile, in the Formation environment, the performance of the three algorithms is similar to that in the cover scenario, with MAAC and MAGAC performing similarly and MADDPG performing somewhat worse.

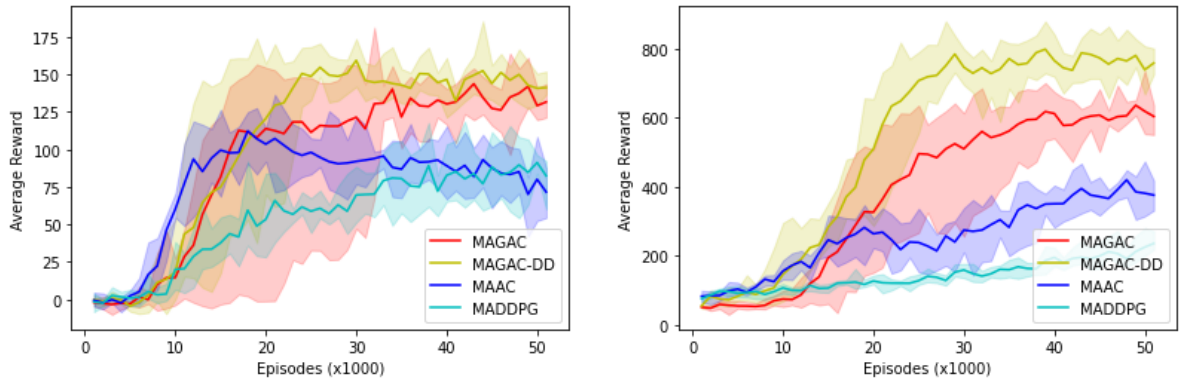


(a) Cover control* (b) Formation Control*

Figure 6: Graph of average global rewards with training episodes when number of agent increase.

In the predator-prey scenario, the reward plot is not as smooth as in the previous environment for several reasons.1: in the predator-prey scenario, the reward is sparse. In the case of sparse rewards, the

predator can only get rewards after cooperating to complete the capture of the prey.2: The scenario is competitive. The same predator's policy will act differently when dealing with different prey, so the reward of the scenario is more random, which leads to different performances of the same policy in a different episode. In contrast, in the cover scenario, the reward gradually increases with the distance to the target, and the reward changes continuously.



(a) predator-preyl (b) predator-prey*

Figure 7: Graph of average global rewards with training episodes in predator-prey of different number of agent.

One possible reason for the poor performance of MAAC and MADDPG in the Predator-prey scenario is that it is difficult to differentiate information about other agents in the evaluation network of MAAC and MADDPG. The actions taken by prey agents are difficult to be learned by their centralized review network, which leads to difficulty in converging the critic network.

In the same environment, we expand the number of agents to verify the robustness of our approach when the number of agents increases. Under the cover task, as shown in the figure5, the advantage of MAGAC over MAAC and MADDPG becomes more obvious. The learning of MADDPG is relatively stable, but the effect of MAAC starts to drop sharply after a training period, and the effect eventually collapses. The reason may be that the attention weights in the attention mechanism in MAAC are difficult to train when there are too many agents.

We also conducted ablation experiments on the dual critic dynamic decomposition method in cover control scenario and predator-prey scenario. The experimental results show that the DD method can show a slight advantage in the original scene, which is more obvious when the number of agents increase.

5 Conclusion

We propose a graph-based soft actor-critic approach for the curse of dimensionality and influence of other agents' strategies in multi-agent reinforcement learning. In MARL, a good grasp of the environment and the state information of all agents can guide agents to learn cooperative strategy better. This method uses the agent's partial observation as the input feature node of the graph and aggregates other agent information through the graph neural network. After several graph iterations, the feature encoding of the agent state is obtained, which contains an understanding of the whole environment from the current agent perspective. This module better extracts the state of an agent in a multi-agent environment while avoiding the curse of dimensionality when there are too many agents. Then a combination of soft actor-critic and dynamic decomposition dual evaluation network methods is used to train the policy. This method trains optimal cooperative strategies while avoiding the influence of the strategy of other agents.

We have conducted comparison experiments with classical baseline algorithms under MPE, which show that our algorithm can obtain higher rewards and has a faster learning speed. The advantage of our approach is more pronounced when the number of agents increases means that we effectively overcome the curse of dimensionality and the influence of other agents' strategies associated with the increasing number of agents.

Funding

This work was supported in part by the National Defense Science and Technology Key Laboratory Fund under Grant 61421030302012109 in part by National Natural Science Foundation of China under Grant 61976178 and 62076202.

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no conflict of interest.

References

[1] Mnih, Volodymyr.; Kavukcuoglu, Koray.; Silver, David.; Rusu, Andrei A.; Veness, Joel.; Bellemare, Marc G.; Graves, Alex.; Riedmiller, Martin.; Fidjeland, Andreas K.; Ostrovski, Georg. (2015). Human-level control through deep reinforcement learning, *nature*, 518(7540), 529–533, 2015.

[2] Duan, Yan.; Chen, Xi.; Houthooft, Rein.; Schulman, John.; Abbeel, Pieter. (2016). Benchmarking deep reinforcement learning for continuous control, *International conference on machine learning*, 1329–1338, 2016.

[3] Ma, X.; Li, Z.; Zhang, L. (2022). An Improved ResNet-50 for Garbage Image Classification, *Tehnički vjesnik*, 29(5), 1552-1559, 2022.

[4] Baressi Šegota*, S.; Anđelić, N.; Car, Z.; Šercer, M. (2021). Neural Network-Based Model for Classification of Faults During Operation of a Robotic Manipulator, *Tehnički vjesnik*, 28(4), 1380-1387, 2021.

[5] Ozgur, Cemile.; Sarikovanlik, Vedat. (2022). Forecasting BIST100 and NASDAQ Indices with Single and Hybrid Machine Learning Algorithms, *Economic Computation And Economic Cybernetics Studies And Research*, 56(3), 235–250, 2022.

[6] Lee, Y.S. (2022). A study on abnormal behavior detection in CCTV images through the supervised learning model of deep learning, *Journal of Logistics, Informatics and Service Science*, 9(2), 196–209, 2022.

[7] Afify, Heba M.; Mohammed, Kamel K.; Hassanien, Aboul Ella. (2020). Multi-images recognition of breast cancer histopathological via probabilistic neural network approach, *Journal of System and Management Sciences*, 10(2), 53–68, 2020.

[8] Silver, David.; Huang, Aja.; Maddison, Chris J.; Guez, Arthur.; Sifre, Laurent.; Van Den Driessche, George.; Schrittwieser, Julian.; Antonoglou, Ioannis.; Panneershelvam, Veda.; Lanctot, Marc. (2016). Mastering the game of Go with deep neural networks and tree search, *nature*, 529(7587), 484–489, 2016.

[9] Shi, Haobin.; Lin, Zhiqiang.; Zhang, Shuge.; Li, Xuesi.; Hwang, Kao-Shing. (2018). An adaptive decision-making method with fuzzy bayesian reinforcement learning for robot soccer, *Information Sciences*, 436, 268–281, 2018.

[10] Justesen, Niels.; Bontrager, Philip.; Togelius, Julian.; Risi, Sebastian. (2019). Deep learning for video game playing, *IEEE Transactions on Games*, 12(1), 1–20, 2019.

[11] Din, Adnan Fayyaz Ud.; Mir, Imran.; Gul, Faiza.; Nasar, Al.; Rustom, Mohammad.; Abualigah, Laith. (2022). Reinforced Learning-Based Robust Control Design for Unmanned Aerial Vehicle: Accurate forecasts of vehicle motion, *Communications of the ACM*, DOI: 10.1007/s13369-022-06746-0, 1–16, 2022.

[12] Fan, Xudong.; Zhang, Xijin.; Wang, Xiaowei.; Yu, Xiong. (2022). A Novel Deep Reinforcement Learning Model for Resilient Road Network Recovery from Multiple Hazards, *Journal of Infrastructure Preservation and Resilience*, DOI: doi: 10.21203/rs.3.rs-2052084/v1, 2022.

[13] Shi, Haobin.; Shi, Lin.; Xu, Meng.; Hwang, Kao-Shing. (2019). End-to-end navigation strategy with deep reinforcement learning for mobile robots, *IEEE Transactions on Industrial Informatics*, 16(4), 2393–2402, 2019.

[14] Lillicrap, Timothy P.; Hunt, Jonathan J.; Pritzel, Alexander.; Heess, Nicolas.; Erez, Tom.; Tassa, Yuval.; Silver, David.; Wierstra, Daan. (2015). Continuous control with deep reinforcement learning, *arXiv preprint arXiv:1509.02971*, 2015.

[15] Lowe, Ryan.; Wu, Yi I.; Tamar, Aviv.; Harb, Jean.; Pieter Abbeel, OpenAI.; Mordatch, Igor. (2017). Multiagent actor-critic for mixed cooperative-competitive environments, *Advances in neural information processing systems*, 30, 6379–6390, 2017.

[16] Foerster, Jakob.; Farquhar, Gregory.; Afouras, Triantafyllos.; Nardelli, Nantas.; Whiteson, Shimon. (2018). Counterfactual multi-agent policy gradients, *Proceedings of the AAAI conference on artificial intelligence*, 32(1), 2974-2982, 2018.

[17] Iqbal, Shariq.; Sha, Fei. (2019). Actor-attention-critic for multi-agent reinforcement learning, *International conference on machine learning*, 2961–2970, 2019.

[18] Yang, Yaodong.; Luo, Rui.; Li, Minne.; Zhou, Ming.; Zhang, Weinan.; Wang, Jun. (2018). Mean field multi-agent reinforcement learning, *International conference on machine learning*, 5571–5580, 2018.

[19] Jiang, Jiechuan and Lu, Zongqing. (2018). Learning attentional communication for multi-agent cooperation, *Advances in neural information processing systems*, 31(6), 7265-7275, 2018.

[20] Scarselli, Franco.; Gori, Marco.; Tsoi, Ah Chung.; Hagenbuchner, Markus.; Monfardini, Gabriele. (2008). The graph neural network model, *IEEE transactions on neural networks*, 20(1), 61–80, 2008.

[21] Liu, Iou-Jen.; Yeh, Raymond A.; Schwing, Alexander G. (2020). PIC: permutation invariant critic for multi-agent deep reinforcement learning, *Conference on Robot Learning*, 590–602, 2020.

[22] Ruan, Jingqing.; Du, Yali.; Xiong, Xuantang.; Xing, Dengpeng.; Li, Xiyun.; Meng, Linghui.; Zhang, Haifeng.; Wang, Jun.; Xu, Bo. (2022). GCS: Graph-Based Coordination Strategy for Multi-Agent Reinforcement Learning, *International Conference on Autonomous Agents and Multiagent Systems*, 1128–1136, 2022.

[23] Sheikh, Hassam Ullah.; Bölöni, Ladislau. (2020). Multi-agent reinforcement learning for problems with combined individual and team reward, *International Joint Conference on Neural Networks* , 1–8, 2020.

[24] Pan, Wei.; Wang, Nanding.; Xu, Chenxi.; Hwang, Kao-Shing. (2021). A dynamically adaptive approach to reducing strategic interference for multi-agent systems, *IEEE Transactions on Cognitive and Developmental Systems*, 14(4), 1486–1495, 2022.

[25] Hamilton, Will.; Ying, Zhitao.; Leskovec, Jure. (2017). Inductive representation learning on large graphs, *Advances in neural information processing systems*, 30, 2017.

[26] Haarnoja, Tuomas.; Zhou, Aurick.; Abbeel, Pieter.; Levine, Sergey. (2020). Off-policy maximum entropy deep reinforcement learning with a stochastic actor, *Proceedings of Machine Learning Research*, 80, 1861–1870, 2018.

[27] Konda, Vijay and Tsitsiklis, John. (1999). Actor-Critic Algorithms, *Advances in neural information processing systems*, 12, 1008–1014, 1999.

[28] Van Hasselt, Hado.; Guez, Arthur.; Silver, David. (2016). Deep reinforcement learning with double q-learning, *Proceedings of the AAAI conference on artificial intelligence*, 30(1), 2159–5399, 2016.

[29] Kipf, Thomas N.; Welling, Max. (2016). Semi-supervised classification with graph convolutional networks Learning, *arXiv preprint arXiv:1609.02907*, 2016.

[30] Shi, Haobin.; Li, Jingchen.; Mao, Jiahui.; Hwang, Kao-Shing. (2021). Lateral transfer learning for multiagent reinforcement learning, *IEEE Transactions on Cybernetics*, DOI: 10.1109/TCYB.2021.3108237, 2021.

[31] Li, Jingchen and Shi, Haobin and Hwang, Kao-Shing. (2021). An explainable ensemble feedforward method with Gaussian convolutional filter, *Knowledge-Based Systems*, 225, 107103.1-107103.11, 2021.

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:

Pan, W.; Cheng, L. (2023). A Graph-Based Soft Actor Critic Approach in Multi-Agent Reinforcement Learning, *International Journal of Computers Communications & Control*, 18(1), 5062, 2023.

https://doi.org/10.15837/ijccc.2023.1.5062