Modeling Decision Process in Multi-Agent Systems: A Graphical Markov Game based Approach

Hao Li*, Yuejiang Li* and H.Vicky Zhao *
* Department of Automation, Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing, P. R. China
E-mail: lihao2020@ia.ac.cn, lyj18@mails.tsinghua.edu.cn, vzhao@tsinghua.edu.cn

***Abstract*—Multi-agent decision processes, where multiple agents interact with each other and make decisions independently, can be seen everywhere in life. Many of the multi-agent systems in reality have underlying topological structures, which constraint the interactions and decision makings of agents such as social networks, computer networks, and cognitive radio networks. Some works considered the topological structure between agents, while the decision process of each agent is modeled as a simple imitation of neighbors. Thus, it is of critical importance to study and model how agents interact with each other with consideration about long-term rewards and how the system evolves when considering the topological structure between agents. In this paper, we consider the topological structure between agents and formulate the graphical Markov game. In graphical Markov game, each agent can only observe the actions of neighbors and make decisions based on the interactions with them. The goal of each agent is to maximize its long-term cumulative reward. To find the optimal policy of each agent, we implement a policy gradient based algorithm. We compare our framework with graphical evolutionary game theory where agents only consider the current rewards through experiments of different game settings.**

Keywords: multi-agent system, topological structure, multi-agent reinforcement learning, Markov game

I. INTRODUCTION

People make decisions all the time, such as whether to forward a message in a social network, how to invest money, and these decisions may have long-term effects. Modeling decision processes is important to study human behavior and offers important guidelines to the design of more efficient and personalized services.

To model sequential decision processes of a single agent, a series of work [1–5] formulated Markov decision processes (MDPs). In MDPs, the agents sequentially make decisions based on the current environment state. Every time the agent makes a decision, it obtains a corresponding reward. The goal is to maximize the expected long-term cumulative reward.

However, in some scenarios, there are more than one agents who interact with each other frequently and influence each other's decisions. Such systems are called multi-agent systems. Game theory is used to model the decision processes in multi-agent systems where two or more agents make decisions, and their decisions together determine the reward of each agent. In game theory, the agents are considered to be rational: they know the structure of the game and aim to maximize their own utilities. The solution to such games is the Nash equilibrium, where no agent could gain more rewards by changing only its own decision [6]. However, the assumed rationality of agents in conventional game theory is not often satisfied. To model the decision processes of not fully rational agents, the authors in [7] proposed evolutionary game theory (EGT). In EGT, a game is played over and over again by biologically or socially conditioned agents who are randomly drawn from large populations [8]. The actions of the agents who have higher rewards would be more likely to be adopted and replicated in the populations. EGT studies how the proportion of actions converge to a stable equilibrium, which would be restored even if a small proportion of agents randomly deviate. Such a stable equilibrium is called an evolutionary stable state (ESS) [9].

To consider the impact of long-term rewards, the work in [10] proposed Markov games as a combination of game theory and MDPs. In Markov games, each agent is assumed to choose actions independently based on the observation of the environment. According to whether the observation contains complete information about the environment, Markov games are divided into fully observable and partially observable Markov games. The actions of all agents together decide the transition of the environment. Same as in MDPs, each agent in Markov games obtains a reward every time it makes decisions and aims to maximize its expected long-term cumulative reward.

However, the above works did not consider the topological constraint among agents. That is, each agent could only interact with its neighbors. Such topology among agents is common in our real life. For example, in social networks, users are often greatly influenced by friends, while the influence of unfamiliar ones is negligible. In a computer network, each machine can only directly communicate with its neighbors. The work in [11] modeled the topology among agents as an undirected graph and proposed graphical EGT. In the graph, vertices represents agents, and edges represents interactions among agents. Each agent in graphical EGT copies actions from its neighbors according to different strategy update rules. These rules have a common feature: the actions of agents who receive larger rewards are more likely to be adopted and imitated by others. Graphical EGT focuses on the evolutionary dynamic and ESS in EGT. It is shown in [12] that the cooperation action is more popular in the Prisoner's Dilemma and the Snow Drift

This work is supported by the National Key Research and Development Program of China (2017YFB1400100).

game when there are topological constraints among agents. Similar conclusions are also drawn in [13–15], which indicates that topological structure among agents significantly influences the evolutionary stable states of the agents' interactions.

Most existing works consider the long-term reward of agents and the topological structures among agents in multi-agent systems separately. However, these two ingredients are often mixed together in common decision scenarios. In this paper, we formulate graphical Markov games. In graphical Markov games, the topological constraint among agents is also modeled using an undirected graph structure. Each agent chooses an action simultaneously according to the observation of its neighbors and then obtains a reward based on the action and the state. The goal of each agent is to maximize its expected long-term cumulative reward. We use the policy gradient algorithm in multi-agent reinforcement learning (MARL) to find the optimal policy for agents and analyze the stable state.

The remaining parts of the paper are organized as follows. In Section 2, we introduce related works, including graphical EGT, partially observable Markov games, and policy gradient algorithm. In Section 3, we introduce the formulation of graphical Markov games. We introduce a MARL method for graphical Markov games in Section 4. In Section 5, we show the simulation results of graphical Markov game and graphical EGT with different game settings and different number of agent types. Conclusions are drawn in Section 6.

II. Related Works

A. Graphical Evolutionary Game Theory

Graphical EGT is used to model decision scenarios where there are not fully rational agents with the topological structure. In graphical EGT, agents are located on the vertices of a graph, and the edges determine the interactions among agents [16]. Agents in graphical EGT imitate actions from neighbors according to different strategy update rules, and actions of the agents who have larger fitness are more likely to be imitated by others. An agent's fitness f is defined as a linear combination of the baseline fitness B and the reward R received from interactions with its neighbors:

$$f = (1-\alpha)\cdot B + \alpha \cdot R. \tag{1}$$

B represents the agent's inherent property, e.g., its social status, and in this work, we assume that B is the same for all agents. In the graphical evolutionary game with w possible actions, the payoff matrix $\mathbf{U}$ is defined as:

$$\mathbf{U} = \begin{bmatrix} u_{11} & \cdots & u_{1w} \\ \vdots & \ddots & \vdots \\ u_{w1} & \cdots & u_{ww} \end{bmatrix},$$

whose entry u_{ij} is the reward an agent receives when it chooses action i and interacts with a neighbor with action j. R is the sum of rewards that an agent obtains from interactions with all its neighbors:

$$R = \sum_{j=1}^{w} |n_j| \cdot u_{ij}, \tag{2}$$

where i represents the action of the agent and $|n_j|$ represents the number of neighbors with action j. In (1), $0 \leq \alpha \leq 1$ is defined as the selection strength, which controls how much the rewards that an agent obtains from interactions with neighbors contribute to its fitness.

Death-Birth (DB) update rule is widely used to character how each agent is influenced by neighbors and updates its action: one agent is chosen randomly as the focal agent and it imitates one of its neighbors' actions with probability proportional to their fitness. There are also the Birth-Death update rule and the Imitation update rule, and the analysis is similar and omitted. The whole population evolves under specific update rules. Graphical EGT focuses on such evolution dynamic and ESS, which would be restored even if a small proportion of agents randomly deviate their actions.

The original graphical EGT assumes that all agents use the same payoff matrix. However, agents are heterogeneous in many scenarios. For example, the fans of a singer are more likely to obtain high reward from forwarding the messages of him/her, while others cannot obtain such a high reward even if they choose the same action. To model such heterogeneous agents, the work in [17] divided agents into different types, and different types of agents had different payoff matrices. It is assumed in [17] that the type of agents was unknown to others and modified the DB update rule for heterogeneous agents. As the focal agent has no information about the type of neighbors, the focal agent in the DB update rule regards the type of all neighbors the same as itself, estimates the fitness of neighbors, and then copies one of neighbors' actions with the probability proportional to the estimated fitness.

B. Partially Observable Markov Games

MDP is a discrete-time model where an agent sequentially chooses actions at each time in an environment. At each time, the agent fully observes the state of the environment s, chooses an action a based on the state, and gains a corresponding reward that is defined by a reward function $R(s,a)$. After the agent chooses its action, the state of the environment s changes to s' with probability defined by the transition probability function $\mathcal{T}(s',s,a)$. The policy is a rule guiding the agent to choose the action at different states. The goal of the agent is to find the optimal policy π^* that maximizes the expected long-term cumulative reward:

$$\pi^* = \arg\max_{\pi} E(\sum_{t=0}^{\infty} \gamma^t R^{(t)}), \tag{3}$$

where $R^{(t)}$ is the reward at time t. $\gamma \in (0,1)$ is a discounting factor, indicating the agent cares more about the current reward than future rewards. For more details of MDPs, we refer readers to [3].

Partially observable Markov game is an extension of MDPs in multi-agent systems, where multiple agents make decisions simultaneously and each agent can only learn partial information about the environment from its observation. Let $\mathcal{S}$ represent the set of states of the environment. At state s,

each agent i receives a private observation $o_i(s) \in \mathcal{O}_i$ and chooses an action $a_i \in \mathcal{A}_i$ based on its policy π_i. π_i can be a deterministic policy $\pi_i : \mathcal{O}_i \mapsto \mathcal{A}_i$; it can also be a stochastic policy $\pi_i : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$, where $\pi_i(o_i, a_i)$ represents the probability to choose action a_i with observation o_i. After all N agents simultaneously choose their actions, each agent gains a reward based on the reward function $R_i : \mathcal{S} \times \mathcal{A}_i \mapsto \mathbb{R}$, and the state s changes to s' with probability $\mathcal{T}(s', s, a_1, a_2, ..., a_N)$. Same as in MDPs, each agent i in Markov games aims to find the optimal policy that maximizes its expected long-term cumulative reward $E(\sum_{t=0}^{\infty} \gamma^t R_i^{(t)})$, where γ is the discounting factor, and $R_i^{(t)}$ is the reward that agent i gets at time t.

C. Policy Gradient Algorithm

Reinforcement learning (RL) studies how to learn by trial and error and is usually used to solve MDPs. Policy gradient algorithm is a classic RL algorithm without any prior knowledge about the environment, and performs well to approximate the optimal stochastic policy in MDPs with a continuous set of actions. Policy gradient algorithm represents the stochastic policy by a parameter vector θ and adjusts θ in the gradient direction of expected long-term cumulative reward [18]. As the state transition is unknown or cannot be expressed mathematically, the gradient cannot be obtained directly. Policy gradient algorithm uses the action-value function $Q(s, a)$ and the discounted weighting of states $\rho(s)$ to estimate the gradient. $Q(s, a)$ is the expected long-term cumulative reward with initial state s and action a:

$$Q(s, a) = E(\sum_{t=0}^{\infty} \gamma^t R^{(t)} | s^{(0)} = s, a^{(0)} = a), \tag{4}$$

where $s^{(t)}$ and $a^{(t)}$ are the state and action at time t, respectively. $\rho(s)$ is defined as:

$$\rho(s) = E(\sum_{t=0}^{\infty} \gamma^t p(s^{(t)} = s)). \tag{5}$$

$\rho(s)$ could been seen as the weights of the reward at state s while considering the expected long-term cumulative reward. The work in [18] proved that for the stochastic policy, the gradient is equal to:

$$\begin{aligned} \nabla_\theta E(\sum_{t=0}^{\infty} \gamma^t R^{(t)}) &= \int_{\mathcal{S}} \rho(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(s, a) \cdot Q(s, a) \mathrm{d}a \mathrm{d}s \\ &= E_{s \sim \rho, a \sim \pi_\theta}(\nabla_\theta \ln \pi_\theta(s, a) \cdot Q(s, a)), \end{aligned} \tag{6}$$

where $\mathcal{S}$ is the set of states and $\mathcal{A}$ is the set of actions. Using (6), the gradient can be estimated with samples of $\theta(s, a) \cdot Q(s, a)$. However, $Q(s, a)$ is not explicitly known due to ignorance of the state transition. An alternative approach to estimate $Q(s, a)$ is Monte-Carlo method. It repeatedly generates a T-long trajectory $\left\{(s^{(t)}, a^{(t)}, R^{(t)})|_{t=0}^{T-1}\right\}$ with the initial state $s^{(0)} = s$ and action $a^{(0)} = a$, and estimates $Q(s, a)$ as the average of the long-term cumulative reward $\sum_{t=0}^{T-1} \gamma^t R^t$.

III. Graphical Markov Game Formulation

In this work, we extend partially observable Markov game by considering the topological structure among agents, and propose graphical Markov game model. In an N-agent graphical Markov game, the topological structure among agents is modeled as an undirected graph G with N vertices. The vertices represent agents, and the edges represent interactions among agents. The whole decision process is split into time steps. In each time step, each agent i chooses an action from its action set $\mathcal{A}_i$ based on the observations of its neighbors. In this paper, we consider a simple scenario where all agents have the same binary action set: $\mathcal{A}^* = \mathcal{A}_1 = ... = \mathcal{A}_N = \{c, d\}$. The system state can be described by all agents' actions at the last time step: $s^{(t)} = \left[a_1^{(t-1)}, a_2^{(t-1)}, ..., a_N^{(t-1)}\right] \in \{c, d\}^N$, where $s^{(t)}$ is the state at time step t and $a_i^{(t-1)}$ is the action of agent i at time step $t-1$. Due to the topological structure, the observation that agent i receives at state s, $o_i(s)$, is defined as the proportion of neighbors who take action c in the last time step.

In each time step, agent i interacts with each of its neighbors, and obtains rewards from these interactions. The payoff matrix of agent i, U_i, is defined as follows:

$$U_i = \begin{bmatrix} u_{cc}^i & u_{cd}^i \\ u_{dc}^i & u_{dd}^i \end{bmatrix},$$

where u_{cd}^i is the payoff that agent i gets when it chooses action c and interacts a neighbor with action d. u_{cc}^i, u_{dc}^i and u_{dd}^i are similarly defined. Same as in [17], we also assume that the payoff matrices show the intrinsic property of agents, and different types of agents have different payoff matrices. The reward that agent i gets with action a_i at state s, $R_i(s, a_i)$, is defined as the sum of the payoffs that agent i gets from all interactions with its neighbors:

$$R_i(a, s) = \begin{cases} |n_i| \cdot \left[u_{cc}^i \cdot o_i(s) + u_{cd}^i \cdot (1 - o_i(s))\right], a = c, \\ |n_i| \cdot \left[u_{dc}^i \cdot o_i(s) + u_{dd}^i \cdot (1 - o_i(s))\right], a = d, \end{cases} \tag{7}$$

where $|n_i|$ is the total number of neighbors of agent i.

In this work, we consider the scenario where all agents adopt stochastic policy, that is, agent i chooses action $a_i \in \{c, d\}$ at state s with probability $\pi_i(o_i(s), a_i)$. The goal of each agent i is to find the optimal policy π_i^* that maximizes its expected long-term cumulative reward $L_i = E(\sum_{t=0}^{\infty} \gamma^t R_i^{(t)})$:

$$\pi_i^* = \arg\max_{\pi_i} L_i, \tag{8}$$

where $0 < \gamma < 1$ is the discounting factor, and $R_i^{(t)}$ is the reward for agent i at time step t.

IV. Policy Gradient for Graphical Markov Game

In this work, to find the optimal policy in the graphical Markov game, we adopt the policy gradient algorithm and the Monte-Carlo method. We use a 2-layer fully connected neural network, which is called the policy network, to represent the stochastic policy for agents. In the policy network, the

activation function of the hidden layer is ReLu, while the activation function of the output layer is softmax. We let π_θ denote the policy network with parameters θ.

According to the formulation of graphical Markov game in Section III, the input of the policy network for an agent is its observation, that is, the proportion of neighbors who take action c, while the output is the probabilities to choose c and d, respectively. We update the policy parameters of each agent i, θ_i, using the gradient ascent method:

$$\theta_i = \theta_i + \lambda \cdot \nabla_{\theta_i} L_i, \tag{9}$$

where λ is the learning rate and controls the update rate of π_{θ_i}. L_i is the expected long-term cumulative reward of agent i. $\nabla_{\theta_i} L_i$ is estimated using (6), and the Monte-Carlo method is used to estimate the action-value function $Q(o, a)$ with a T-long trajectory of agent i, $\left\{(o_i^{(t)}, a_i^{(t)}, R_i^{(t)})|_{t=1}^{T}\right\}$:

$$\begin{aligned}\nabla_{\theta_i} L_i &= E_{o\sim\rho^{\pi_{\theta_i}}, a\sim\pi_{\theta_i}}(\nabla_{\theta_i} \ln \pi_{\theta_i}(o,a) \cdot Q(o,a)) \\ &\approx \sum_{t=1}^{T} \nabla_{\theta_i} \ln \pi_{\theta_i}(o_i^{(t)}, a_i^{(t)}) \cdot \sum_{j=t}^{T} \gamma^{j-t} R_i^{(j)}.\end{aligned} \tag{10}$$

However, there are two main problems: instability and expensive computational cost. Policy gradient algorithm applies to MDPs where the environment is stable. However, in graphical Markov games, all agents simultaneously update their policies, and the environment for each agent includes all other agents and is unstable. On the other hand, the policy gradient algorithm requires a large number of trajectories to converge, which causes expensive computational cost if we find the optimal policy for each agent respectively.

To address thees two issues, we use parameter sharing to reduce the computational cost, and use two policy networks for each type of agents to improve stability. Fig. 1 illustrates the whole training process, and fig. 2 shows details of the policy gradient step.

A. Parameter Sharing

Parameter sharing means that the same type of agents share the same policy network, that is, they use the same policy. This strategy is widely used to reduce the computational cost in MARL, when there are considerable amount of agents in each type [19–21]. We let $m(i)$ denote the type of agent i, and $\pi_{\theta_m^*}$ denotes the policy network for type-m agents, whose parameters are θ_m^*. With parameter sharing, each trajectory of type-m agents may be used to update θ_m^*. In non-uniform degree networks, the same type of agents may have different number of neighbors, and the agents who have more neighbors are more likely to obtain higher rewards. It leads to large variance of rewards and may result in oscillation in training. To eliminate the impact of the neighbor's number, we normalize R_i^t with $|n_i|$. $\theta_{m(i)}^*$ is updated with a T-long trajectory

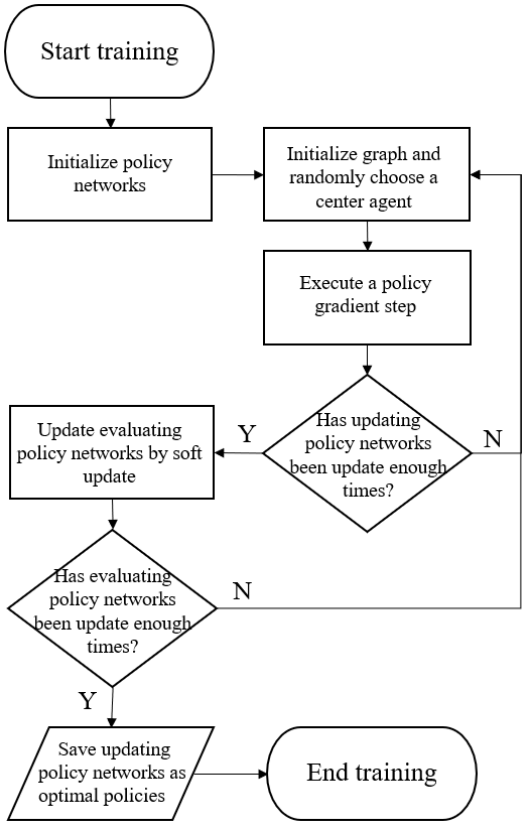


Fig. 1. Training process.

$\left\{(o_i^{(t)}, a_i^{(t)}, R_i^{(t)})|_{t=1}^{T}\right\}$ of agent i using:

$$\theta_{m(i)}^* = \theta_{m(i)}^* + \lambda \cdot \sum_{t=1}^{T} \nabla_{\theta_{m(i)}^*} \ln \pi_{\theta_{m(i)}^*}(o_i^{(t)}, a_i^{(t)}) \cdot \sum_{j=t}^{T} \gamma^{j-t} \frac{R_i^{(j)}}{|n_i|}. \tag{11}$$

As the topological structure G is static and $|n_i|$ is a constant, dividing R_i^t by $|n_i|$ would not change the optimal policy of agent i.

B. Two Policy Networks For Each Type Of Agents

For one agent, changes in other agents' policy cause the environment instability in training. To overcome it, only one agent, which is called the center agent, updates its policy network in a policy gradient step. Thus each type of agents have two policy networks, called the updating policy network and the evaluating policy network, respectively. In policy gradient steps, updating policy networks are updated while evaluating policy networks keep constant. As shown in fig. 2, only the center agent uses the updating policy network, while others use the evaluating policy network. Thus the environment is stable for the center agent. Similar strategy is used in [22].

Updating policy networks and evaluating policy networks are updated in different ways. Updating policy networks are updated by (11) in policy gradient steps. After the policy

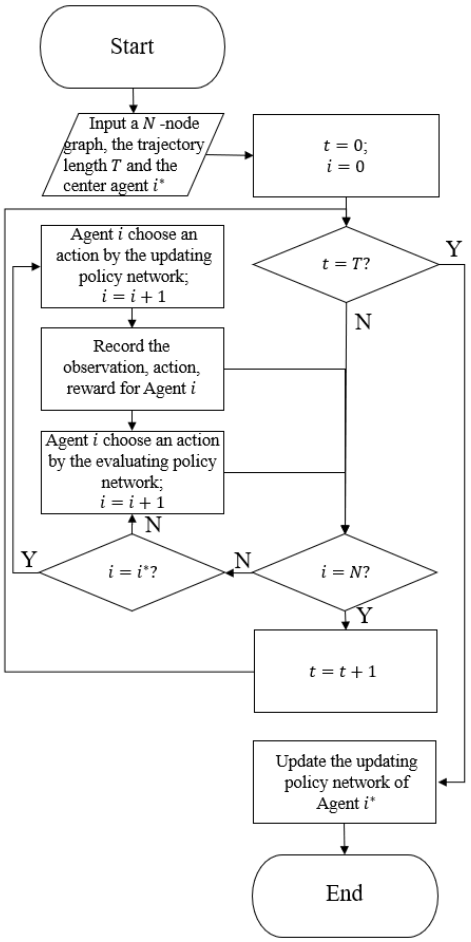


Fig. 2. The policy gradient step, where updating policy networks are updated and evaluating policy networks keep constant.

gradient step is executed several times, the evaluating policy network is updated by soft update as in [22]:

$$\theta_m^e = (1-\tau)\theta_m^e + \tau\theta_m^u \tag{12}$$

where $\tau \ll 1$, and θ_m^u and θ_m^e represent the parameters of the updating policy network and the evaluating policy network for type-m agents, respectively. By using soft update rather than direct copy, the evaluating policy network tracks the updating policy network slowly, which improves the stability of training.

V. Experiments

In this paper, we design experiments with different games and different number of agent types, and compare the results of graphical Markov game with graphical EGT. The policy gradient algorithm is used to find the optimal policy in graphical Markov game, while the DB update rule is used in graphical EGT.

Table I shows the hyper-parameters that are shared in all experiments. In policy gradient algorithm, the hidden layer of policy networks has five neurons, and there are 1000 iterations in training and each iteration has 10 policy gradient steps.

TABLE I
HYPER-PARAMETERS SHARED IN EXPERIMENTS

parameter	physical meaning	value
T	the length of trajectories	50
λ	learning rate in the policy algorithm	0.001
τ	soft update coefficient	0.1
γ	the discounting factor in graphical Markov game	0.9
α	the selection strength in graphical EGT	0.15

A. Stag Hunt Game

In this experiment, all agents are of the same type and use the same payoff matrix U that satisfies: $u_{cc} > u_{dc} \geq u_{dd} > u_{cd}$. The game between two agents is called the stag hunt game and action c can be seen as the cooperative action. We normalize $u_{cc} = 1$ and $u_{cd} = 0$, and assume $u_{dc} = u_{dd}$. In the game, the reward for agent i, R_i, is:

$$\frac{R_i(a,s)}{|n_i|} = \begin{cases} o_i(s), & a = c, \\ u_{dd}, & a = d, \end{cases} \tag{13}$$

where s is the state of the environment, and a is the action that agent i chooses. With $o_i(s)$ increases, indicating that there are more cooperative neighbors, agent i would obtain a larger reward with the cooperative action but a constant reward with the uncooperative action. This leads to a positive feedback: if the cooperative agents obtain more rewards or long-term cumulative rewards than uncooperative agents, agents would be more likely to choose the cooperative action and the advantage of the cooperative agents would increase at the next time step. Thus, whether the cooperative action has an advantage or not at time step 0 influences the proportion of cooperative agents at the following time steps.

To reduce the random error caused by policy initialization in our method, the policy network is initialized using randomly chosen actions, that is, agents would choose action c or d with probability 0.5 respectively. We obtain the initial policy network by supervised learning: we first randomly initialize a policy network and define the loss function as the square of the difference between the probability to choose action c and 0.5. Then we repeatedly generate a random input between 0 and 1, calculate the loss function, use the back-propagation algorithm to find the gradient of the loss function with respect to the parameters of the policy network, and update the parameters in the opposite direction to the gradient. The above steps are repeated 10,000 times.

Let $p_c^{(0)}$ denote the proportion of agents who begin with action c, and let p_c^* denote the proportion of agents that choose action c in the final stable state. $p_c^{(0)}$ determines the probability distribution of the state and the observations at time step 0. The experimental results show that with the payoff matrix of a stag hunt game, the graphical Markov game and the graphical EGT both have two kinds of stable equilibrium: $p_c^* = 0$ and $p_c^* = 1$. Table II shows the number of times of reaching the full cooperation stable state with $p_c^* = 1$ of two models in a uniform degree graph with different u_{dd} and

$p_c^{(0)}$. Here, the uniform degree graph has 500 nodes, and each node has 20 neighbors. From Table II, both models would be more likely to achieve the cooperative stable equilibrium $p_c^* = 1$ when u_{dd} are smaller and when $p_c^{(0)}$ are larger. At time step 0, with higher $p_c^{(0)}$, cooperative agents are more likely to interact with cooperative neighbors and obtain higher rewards. A smaller u_{dd} means that the reward that agents obtain with uncooperative actions is smaller. Those intuitively lead to the advantage of the cooperative action at time step 0 and the trend of cooperation.

TABLE II
THE NUMBER OF TIMES OF REACHING THE $p_c^* = 1$ STABLE STATE IN THE STAG HUNT GAME

u_{dd}	$p_c^{(0)}$	graphical Markov game	graphical EGT
0.4	0.1	10/10	0/10
0.4	0.5	10/10	10/10
0.5	0.1	7/10	0/10
0.5	0.5	10/10	9/10
0.55	0.1	1/10	0/10
0.55	0.5	1/10	6/10

When $p_c^{(0)}$ decreases from 0.5 to 0.1 with $u_{dd} = 0.5$, the number of times of $p_c^* = 1$ in graphical EGT decreases much faster than graphical Markov game. But when u_{dd} increase from 0.5 to 0.55 with $p_c^{(0)} = 0.5$, the number of times of $p_c^* = 1$ in the graphical EGT decreases less than that in the graphical Markov game. This difference is due to the influence of u_{dd} and $p_c^{(0)}$ on the current and future rewards: $p_c^{(0)}$ directly affects the reward at time step 0 and indirectly affects the following rewards by affecting the proportion of agents who choose action c in the following time steps. But u_{dd} directly affects the rewards at all time steps. As the agents in graphical Markov games consider both the current reward and the future rewards, while agents in graphical EGT only consider the current reward, the advantage of the cooperative action in graphical Markov game at time step 0 is influenced more by u_{dd} and less by $p_c^{(0)}$ than in graphical EGT.

B. Prisoner's Dilemma

In this experiment, all agents are of the same type and the game played between two neighboring agents is defined as a prisoner's dilemma with payoff matrix U:

$$U = \begin{bmatrix} 1 + u_{cd} & u_{cd} \\ 1 & 0 \end{bmatrix},$$

where $0 > u_{cd} > -1$. The action c is defined as cooperation in the game, while the action d is defined as defection. u_{cd} can been seen as the current cost of cooperation. In the game, the reward for agent i, R_i, is:

$$\frac{R_i(a,s)}{|n_i|} = \begin{cases} o_i(s) + u_{cd}, & a = c, \\ o_i(s), & a = d. \end{cases} \tag{14}$$

As $u_{cd} < 0$, action c is dominated by action d, that is, action d brings larger current reward than action c in any scenarios. And agent i would obtain higher rewards with larger $o_i(s)$, which means more cooperative neighbors. In this experiment, the graph has 100 nodes and each node has 5 neighbors. Each agent would choose c as the initial action with probability 0.1.

TABLE III
THE NUMBER OF TIMES OF REACHING THE STABLE STATE $p_c^* = 1$ IN THE PRISONER'S DILEMMA GAME

u_{cd}	graphical Markov game	graphical EGT
-0.1	10/10	5/10
-0.13	6/10	4/10
-0.15	3/10	2/10
-0.2	0/10	1/10

Our experiments show that with a randomly initialized policy network, p_c^* would always be 0. A randomly initialized policy network can hardly satisfy that the probability to choose action c increases fast enough with higher proportion of cooperative agents in neighbors. This makes cooperative agents can hardly have enough more cooperative neighbors in the future and obtain enough more future rewards to make up for the disadvantage on current rewards. Therefore, agents would be more inclined to defect and aggravates the disadvantage, which finally leads p_c^* to 0.

Essentially, each agent is in a repeated prisoner's dilemma game with each neighbor. In [23], it is shown that in the 2-agent repeated game of prisoner's dilemma, "tit for tat", which means first cooperating and then subsequently replicating the other agent's previous action, can successfully stimulate cooperation and punish non-cooperative actions. Inspired by this, we use an initial policy that each agent chooses the cooperative action with probability that equals to the proportion of cooperative neighbors. Such an initial policy is obtained by supervised learning: we first randomly initialize a policy network and define the loss function as the square of the difference between the probability to choose action c, and the input. Then we repeatedly generate a random input between 0 and 1, calculate the loss function, and use the back-propagation algorithm to calculate the gradient of the loss function with respect to the parameters of the policy network and update the parameters in the opposite direction to the gradient. The above steps are repeated 10,000 times.

The p_c^* of graphical Markov game and graphical EGT have the same two possible value, 0 and 1. Table III shows the number of times of reaching the fully cooperative stable state $p_c^* = 1$ in the two models. In both models, p_c^* is more likely to be 1 when u_{cd} is larger. The agents are more likely to choose cooperative actions when the additional future rewards that cooperative actions bring can make up for the current lose u_{cd}. Larger u_{cd} means less current cost of cooperation, which intuitively leads to cooperation in the graphical Markov game. For analysis on prisoner's dilemma in the graphical EGT, we refer readers to [12]. The results also show that p_c^* in graphical Markov game is influenced more by u_{cd} than in graphical EGT: when u_{cd} decreases from -0.1 to -0.2, the number of

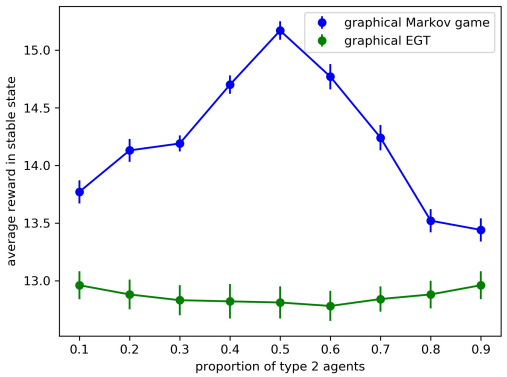


Fig. 3. average rewards in the stable state with different proportions of type-2 agents.

times of reaching $p_c^* = 1$ in graphical Markov game decreases faster than in graphical EGT.

C. Two Types Of Nodes

In this experiment, there are two types of agents. Type 1 agents and type 2 agents have the two different payoff matrices respectively as:

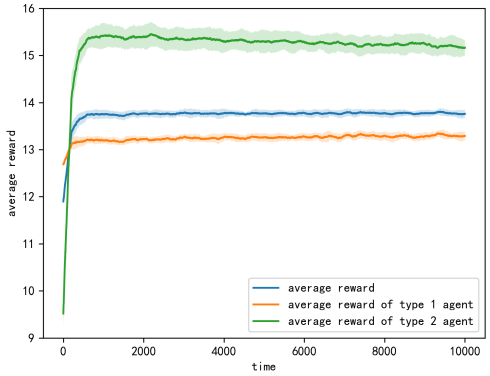
$$U_1 = \begin{bmatrix} 0.4 & 0.8 \\ 0.8 & 0.6 \end{bmatrix}, \quad \text{and} \quad U_2 = \begin{bmatrix} 0.6 & 0.8 \\ 0.8 & 0.4 \end{bmatrix}.$$

Here, the graph has 200 nodes and each node has 20 neighbors.

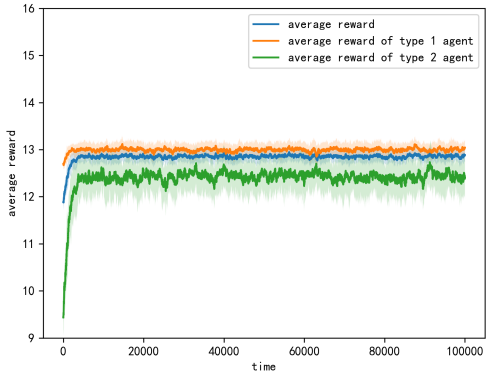
We let p_2 denote the proportion of type-2 agents. Fig. 3 shows the average rewards of the two models in the stable state with different p_2. Agents in graphical Markov game have higher average rewards than in graphical EGT, especially when p_2 is close to 0.5. Fig. 4 shows the dynamic of average rewards in the two models during testing when p_2 is 0.25. The results shows that both type-1 and type-2 agents in graphical Markov game always gain more rewards than in graphical EGT. Those differences are caused by the difference in the strategy update rules/polices. In graphical Markov game, agents choose actions based on the optimal policy that maximizes long-term cumulative rewards; while in graphical EGT, agents simply imitate neighbors' actions with specific update rules, which only consider current rewards.

VI. Conclusion

In this paper, we formulate graphical Markov game, which considers both the topological structure among agents and long-term rewards. In graphical Markov game, each agent can only observe the actions of neighbors and choose actions simultaneously based on the observations. The goal of each agent is to maximize its expected long-term cumulative reward. To find the optimal policy of each agent, we implement a policy-gradient-based algorithm with parameter sharing and two policy networks. We design experiments of different game settings and compare our framework with graphical EGT.



(a) graphical Markov game



(b) graphical EGT

Fig. 4. average reward of two models with $p_2 = 0.25$.

When interactions between agents is defined by a stag hunt game or prisoner's dilemma, due to the influence of future rewards on the agent's actions, cooperation is more likely to occur in our framework with large encough reward of cooperation. We also find that when there are two types of agents, the agents in our framework obtain higher reward than in graphical EGT.

References

[1] R. A. Howard, *Dynamic programming and Markov processes*. MIT Press, 1960.

[2] A. G. Barto, R. S. Sutton, and C. Watkins, *Learning and sequential decision making*. University of Massachusetts Amherst, 1989.

[3] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[4] D. P. Bertsekas, *Dynamic programming and optimal control*, 2nd ed. Athena Scientific, 2000.

[5] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: theory and practice*. Elsevier, 2004.

[6] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT press, 1994.
[7] J. M. Smith and G. R. Price, "The logic of animal conflict," *Nature*, vol. 246, no. 5427, pp. 15–18, 1973.
[8] R. Cressman, *Evolutionary dynamics and extensive form games*. MIT Press, 2003.
[9] J. M. Smith and J. M. M. Smith, *Evolution and the theory of games*. Cambridge university press, 1982.
[10] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of The Eleventh International Conference on International Conference on Machine Learning*, 1994, pp. 157–163.
[11] M. A. Nowak and R. M. May, "Evolutionary games and spatial chaos," *Nature*, vol. 359, no. 6398, pp. 826–829, 1992.
[12] H. Ohtsuki and M. A. Nowak, "The replicator equation on graphs." *Journal of Theoretical Biology*, vol. 243, no. 1, pp. 86–97, 2006.
[13] F. C. Santos and J. M. Pacheco, "Scale-free networks provide a unifying framework for the emergence of cooperation." *Physical Review Letters*, vol. 95, no. 9, p. 98104, 2005.
[14] F. C. Santos, J. M. J. Pacheco, and T. Lenaerts, "Evolutionary dynamics of social dilemmas in structured heterogeneous populations," *the National Academy of Sciences of the United States of America*, vol. 103, no. 9, pp. 3490–3494, 2006.
[15] G. Szabó and G. Fáth, "Evolutionary games on graphs," *Physics Reports*, vol. 446, no. 4, pp. 97–216, 2007.
[16] F. Fu, L. Wang, M. A. Nowak, and C. Hauert, "Evolutionary dynamics on graphs: efficient method for weak selection," *Physical Review E*, vol. 79, no. 4, pp. 46 707–46 707, 2009.
[17] X. Cao, Y. Chen, C. Jiang, and K. J. R. Liu, "Evolutionary information diffusion over heterogeneous social networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 595–610, 2016.
[18] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of Advances in Neural Information Processing Systems 12*, 1999, pp. 1057–1063.
[19] J. K. Gupta, M. Egorov, and M. J. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, 2017, pp. 66–83.
[20] Y. Yang, R. Luo, M. Li, J. Wang, and W. Zhang, "Mean field multi-agent reinforcement learning," in *Proceedings of The Thirty-fifth International Conference on Machine Learning*, 2018, pp. 5567–5576.
[21] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *Proceedings of The Eighth International Conference on Learning Representations*, 2020.
[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[23] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *Science*, vol. 211, no. 4489, pp. 1390–1396, 01 1981.