2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636636

Multi-agent Collaborative Learning with Relational Graph Reasoning in Adversarial Environments*

Shiguang Wu[1,2], Tenghai Qiu[1], Zhiqiang Pu[1,2], and Jianqiang Yi[1,2], *Senior Member, IEEE*

***Abstract*— This paper proposes a collaborative policy framework via relational graph reasoning for multi-agent systems to accomplish adversarial tasks. A relational graph reasoning module consisting of an agent graph reasoning module and an opponent graph module, is designed to enable each agent to learn mixture state representation to enhance the effectiveness of the policy. In particular, for each agent, the agent graph reasoning module is designed to infer different underlying influences from different opponents and generate agent-level state representation. The opponent graph reasoning module is creatively designed for the opponents to reason relations from their surrounding objects including the agents and the opponents based on their latent features and then predict the future state of the opponents. It forms an opponent-level state representation. Besides, in order to effectively predict the state of the opponents, an intrinsic reward based on prediction error is designed to motivate the policy learning. Furthermore, interactions among agents are utilized to transmit messages and fuse information to promote the cooperative behaviors among the agents. Finally, various representative simulations on two multi-agent adversarial tasks are conducted to demonstrate the superiority and effectiveness of the proposed framework by comparison with existing methods.**

I. Introduction

Multi-agent systems have received increasing attention from researchers due to their broad applications in several domains. Their applications can be found in warehousing logistics [1], disaster relief [2], formation control [3], satellite cluster [4], and so on. In a multi-agent system, agents need to cooperatively accomplish a complex task that a single agent can not complete. The collaboration among the agents becomes more significant especially in adversarial environments, where an agent needs to cooperate with other agents of the same team to fight against the opponent team. Meanwhile, in some special cases, the number of opponents is uncertain and changing since the opponents may be killed in competing. This requires that the policy of the agents has better adaptability. Therefore, for each agent, deriving a distributed, efficient, and collaborative policy in multi-agent adversarial environments remains challenging.

*This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102404, the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA27000000, the National Natural Science Foundation of China under Grant 62073323, and the Innovation Academy for Light-duty Gas Turbine, Chinese Academy of Sciences, No.CXYJJ19-ZD-02 and No.CXYJJ20-QN-05.

[1]Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, e-mail: shiguang.wu@outlook.com, tenghai.qiu@ia.ac.cn, zhiqiang.pu@ia.ac.cn, jianqiang.yi@ia.ac.cn

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China.

Recently, deep reinforcement learning (DRL) becomes a research focus and shows the potentialities and learning ability [5]–[7]. Owing to the huge potential, a natural idea is to extend reinforcement learning to multi-agent settings. Some multi-agent reinforcement learning (MARL) methods [8], [9] are used to enable agents to collaborate and achieve common goals. However, most of them treat opponents as a part of environments by directly using or stacking the states of the opponents, which have the difficulty of transferability and scalability. Besides, they ignore different influences from different opponents, which has an impact on decision-making agents. A hierarchical graph neural network is utilized to effectively model inter-agent and inter-group relationships in [10]. However, it only considers different influences from different opponents to agents without considering different influences from the agents to the opponents in the opponent perspective.

The opponent modeling in the opponent perspective can make multi-agent systems effectively accomplish an adversarial task. Some works focus on the opponent modeling using deep learning architectures [11]–[14]. In [13], agent modeling is used to learn an opponent policy to improve the expected rewards. It takes the prediction of the opponent policy as an auxiliary task to train the policy of the agent simultaneously. However, it predicts the opponent policy based on the representation of the agent without designing corresponding representation in the opponent perspective. In [14], modeling opponent policy using variational autoencoders is utilized to learn the opponent representation based on the trajectories of the opponent or the agent. These methods consider opponent modeling with fixed number opponents, which can not adapt to changing number opponents. This also makes these methods suffer from the difficulty of transferability and adaptability.

Inferring underlying relations among agents and opponents can provide valuable information for the decision-making of the agents [15]. Relations and objects can usually be represented as a connected graph. Hence, graph-based learning methods are developed to model objects and their relations, such as graph neural networks (GNNs) [16]. GNN is an effective method to extract features by taking objects as nodes and relations as edges. In [12], an agent-interaction graph is designed to describe the relations among objects and then generate new features. This motivates us to reason about the relations among agents and opponents then generate high-dimensional features based on the relational reasoning and GNN models.

Motivated by the above discussion, a collaborative policy

framework via relational graph reasoning is proposed to enable agents to efficiently accomplish adversarial tasks in this paper. The main innovation of this framework is to design a relational graph reasoning module including an agent graph reasoning module and an opponent graph reasoning module, which enables the agents to learn mixture state representation to enhance the effectiveness of the policy. Specifically, the agent graph reasoning module with agent relation attention, the opponent graph reasoning module with opponent relation modeling, and state prediction are designed for each agent to represent the state representation about its surroundings in the agent-level and opponent-level, respectively. Moreover, to promote the cooperative behaviors among the agents, interactions among the agents are adopted to transmit messages and fuse information. The main contributions in this paper are listed as follows:

1) A novel collaborative policy framework via relational graph reasoning is proposed for multi-agent systems to efficiently accomplish adversarial tasks.
2) Different from some existing methods, the proposed opponent graph reasoning module describes the different influences from the agents to the opponents in the opponent perspective and then predicts the future state of the opponents. This strongly promotes the effectiveness of the policy.
3) To effectively predict the state of the opponents, an intrinsic reward based on the prediction error is designed to motivate the agents to predict the state of the opponents and promote the policy learning.

II. Background

A. Markov Games

In this paper, we consider the framework of Markov Games [17], which is an multi-agent extension of Markov Decision Process. A Markov Game for N agents is defined by a global state S, a set of observations $O_1, \cdots, O_N$, and a set of actions $A_1, \cdots, A_N$. At each time-step, each agent chooses an action using a learned policy π_i, which determines the next state according to state transition model $T : S \times A_1 \times \cdots \times A_N \rightarrow S'$. Each agent i receives an reward denoted as r_i after all agents execute the actions. The agent i aims to maximize its own expected discounted return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ with the learned policy, where $\gamma \in [0, 1]$ is a discount factor and T is the time horizon.

B. Proximal Policy Optimization

The proximal policy optimization (PPO) [18] is adopted as a training algorithm in this paper. It addresses the problem that the policy updating is unstable in the gradient descent method and the problem of low sampling efficiency through clip operation and importance sample. Let

$$l_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{1}$$

be the likelihood ratio. π_θ is the policy with parameters θ. Then, the optimization objective of PPO is as follow:

$$\begin{aligned} L(\theta) = E[\min(&l_t(\theta)\hat{A}_t^{\theta_{old}}(s_t, a_t), \\ &clip(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\theta_{old}}(s_t, a_t))] \end{aligned} \tag{2}$$

where $\hat{A}_t^{\theta_{old}}(s_t, a_t)$ is an estimator of the generalized advantage. To ensure the rationality of importance sampling, $l_t(\theta)$ should be limited in the range $[1-\epsilon, 1+\epsilon]$ with parameters ϵ through the clip operation. It ensures the difference between π_θ and $\pi_{\theta_{old}}$ not too big.

III. Method

For an adversarial scenario, there are at least two or more participating teams. In this paper, we consider that one agent team including N^a agents competes with opponent team including N^o opponents in a two-dimensional continuous space. Then, we introduce our approach from the perspective of the agent team. Let s_i denotes the state of the i^{th} agent in the agent team, which includes the position and velocity of the agent. Similarly, let s_{oppj} denotes the state of the j^{th} opponent in the opponent team. In addition, the dynamic model of each agent is modeled as a second-order integrator. The purpose of this paper is to design a policy for agent i, $\pi_i(t) : O_i(t) \rightarrow a_i(t)$ to select an appropriate action to efficiently accomplish multi-agent adversarial tasks, where $O_i(t) = [S(t), S^a_{opp}(t)]$, $S(t)$, $S^a_{opp}(t)$ denote the set of agents' state and the set of alive opponents' state, respectively.

A. Overall Structure

In this section, a collaborative policy framework consisting of three components, i,e., relational graph reasoning, interaction, and actor-critic, is given in Fig. 1. First, a relational graph reasoning module is designed for each agent to learn mixture state representation to enhance the effectiveness of policy, which consists of two components, i.e., agent graph reasoning module and opponent graph reasoning module. In particular, the agent graph reasoning aims to infer different underlying influences from different opponents to the agent. It represents the state representation in the perspective of the agent. Then, the opponent graph reasoning module is designed for the opponents to model relations from their surrounding objects including the agents and the opponents and then predict the future state of the opponents in the opponent's perspective. It generates the state representation in the opponent-level. Besides, in order to effectively predict the opponents' state, an intrinsic reward based on prediction error is designed to motivate the agents to predict the opponents' state and promote the policy learning simultaneously. Moreover, by fusing two components, the agents can have a mixture state representation for surrounding environment. Furthermore, interactions among the agents are used to transmit messages and fuse different representations.

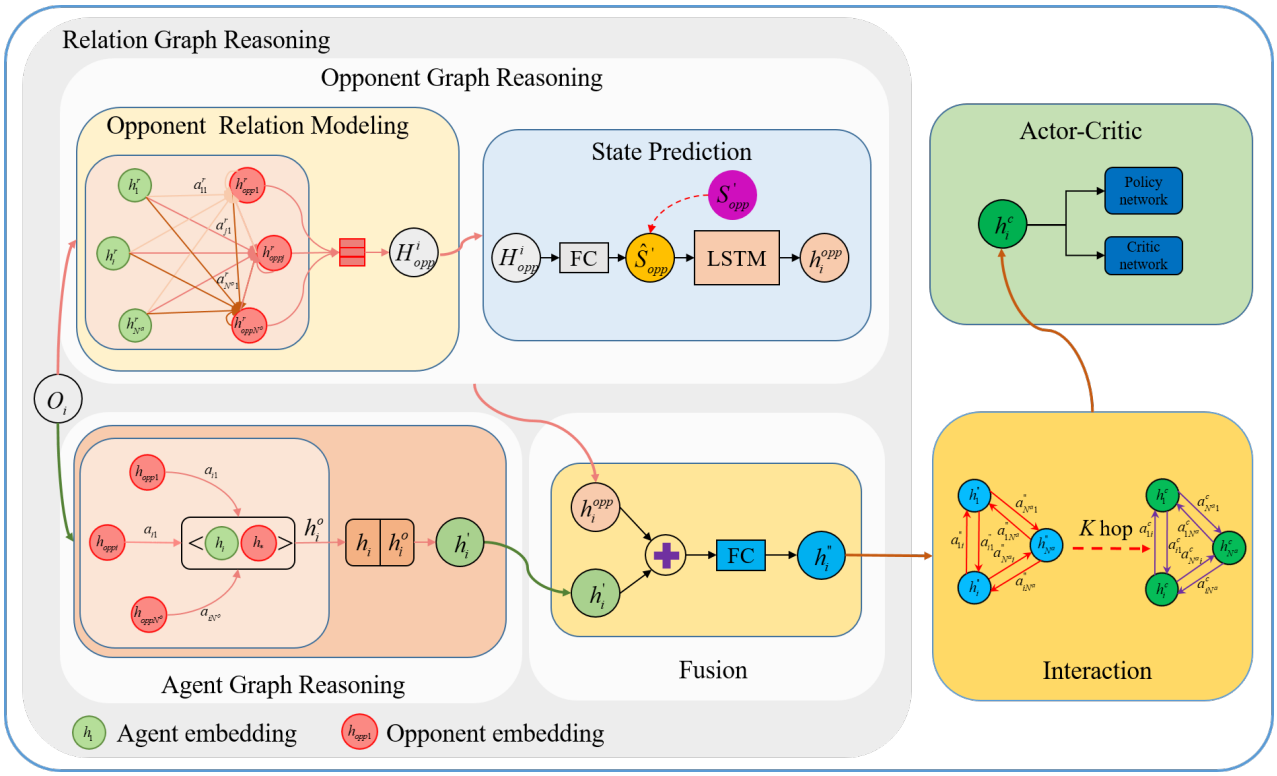


Fig. 1. The collaborative policy framework with relational graph reasoning.

B. Agent Graph Reasoning

In an adversarial scenario, agents should make decisions based on the state of opponents in real-time. Intuitively, different opponents have different influences on the behavior of the agents. Besides, the number of the opponents is changing because the opponents may be killed in competing. Therefore, to address these issues, an agent graph reasoning module is proposed for each agent to reason the different influences from the opponents in its own perspective.

In particular, each agent i receives an observation O_i. Then, the agent and the opponents can be represented as a graph $G^a = (V^a, E^a)$ called an agent relation graph, where $|V^a| = N_o + 1$. The s_i, s_{oppj} are encoded into embedding h_i, h_{oppj}, respectively through different fully connected networks. Then, the normalized attention weight can be computed using a dot product attention, i,e., a_{ij}. It illustrates agent i how much attention it pays to opponent j.

$$e_{ij} = \frac{1}{d_1} < h_i, h_{oppj} >, \tag{3}$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{j \in N_a^o} \exp(e_{ij})}, \tag{4}$$

$$h_i^o = \sum_{j \in N_a^o} a_{ij} h_{oppj}, \tag{5}$$

where $<>$ denotes vector dot product. N_a^o is the number of the alive opponents. d_1 is the dimension of h_i. h_i^o represents the embedding from the opponents in the perspective of agent i. Finally, h_i^o is concatenated with h_i to generate an agent-level state representation h_i'.

C. Opponent Graph Reasoning

Inferring opponent behaviors is an important ability for agents by constructing models that make predictions about the opponents in adversarial tasks. [13] uses an auxiliary task to learn a response and an opponent model in the agent's perspective without considering the influences from surrounding objects to the opponents. In fact, the influences from the agents to the opponents are important for the agents for predicting the opponents' future state. Therefore, an opponent graph reasoning module is designed for the opponents to reason the relations among the agents and the opponents and then predict the future state of the opponents. Previous work does not model the influences for the opponents to predict the opponents' state. Here, we model the agents and opponents as a graph, reason about the relations among the agents and the opponents, and use graph attention network (GAT) [19] to compute the influences, which is further to predict the future state of the opponents.

Opponent relational modeling. The key challenge in opponent reasoning is to learn the influences for the opponents from their surrounding objects. Graph neural network is used to learn the relations between the opponents and the objects and calculate new state representation with learned attention in [12]. This provides a new sight to model the relations between the opponents and the objects. Therefore, the agents and the opponents can be modeled as a directed

graph $G^r = (V^r, E^r)$ called an opponent relational graph, where $|V^r| = N_a + N_o$. The relational graph is unidirectional, where the edges point to the opponents. The edge e_{ji} indicates how much influence agent i to opponent j or the importance of agent i to opponent j. The relation is not known as a prior, so it is inferred through learning. After inferring the relations among the agents and the opponents, the information is propagated from one node to another by a graph neural network. Then, the state representation for the opponents can be computed. In particular, since the agents and the opponents contain different level state information, we use two fully connected networks f_a and f_o to map the state of agent k and opponent j into a latent space denoted h^r_k, h^r_{oppj}. Given the latent features, a relation weight is computed using a linear learnable weight matrix W_r, i.e., $e^r_{jk} = f^r_a(h^r_{oppj}, h^r_k, W_r)$ describing influence from the surrounding agents k to the opponent j. The normalized relation weight is computed by the softmax function

$$a^r_{jk} = \frac{\exp(\sigma_0(e^r_{jk}))}{\sum_{k \in N^o_a + N^a} \exp(\sigma_0(e^r_{jk}))}, \tag{6}$$

where σ_0 is $LeakyReLU$ activation function. N^o_a is the number of the alive opponents. With the latent features h^r_k and normalized relation weight a^r_{jk}, the state representation of the opponent j in the perspective of agent i is computed by $h^i_j = \sigma(\sum_{k \in N^o_a + N^a} a^r_{jk} W^T_r h^r_k)$, where σ is the $RELU$ activation function. Finally, by stacking the state representation of the opponents, the state representation matrix of the opponents in the perspective of agent i denoted as H^i_{opp} can be obtained.

State prediction. After obtaining the state representation of the opponents by the opponent relation modeling, a state prediction module is designed to predict the future state of the opponents. It is different from other works where action prediction is considered. In fact, the action space of the opponents can not be known. Therefore, the state prediction is more realistic and reasonable. Specifically, a state prediction model uses the state representation of the opponents to predict their next state $\hat{S}^{i'}_{opp} = f_p(H^i_{opp})$, where $\hat{S}^{i'}_{opp}$ is the predicted state for the opponents in the perspective of agent i. The state prediction network f_p is modeled with a fully connected network. In order to make the state prediction valid, the prediction error is used as an intrinsic reward [20] to guide agent i to reason the future state of the opponents.

$$r^i_{in} = E_{\hat{s}^{i'}_{opp} \sim S^a_{opp}}[(\hat{s}^{i'}_{opp} - s^{'}_{opp})^2], \tag{7}$$

where $s^{'}_{opp}$ is the actual state of the opponent, which represents the position of the opponent. It can obtained by some sensors. S^a_{opp} is the set of the alive opponents. Then the total reward for agent i is computed as:

$$r^i = r^i_{ex} - \alpha r^i_{in}, \tag{8}$$

where r^i_{ex} is an extrinsic or environmental reward, and α is a tunable parameter that balances the extrinsic team reward and the intrinsic reward. In the process of competing, the number of opponents is changing. Therefore, for agent i, long short term memory (LSTM) [21] is adopted to handle the changing opponents and encode the opponent team, which forms an opponent-level state representation denoted as h^{opp}_i. Finally, fusing the agent-level state representation and the opponent-level state representation, the agents can have a mixture state representation denoted as $h^{''}_i$ about surrounding environments, which promotes the policy learning.

D. Interaction

After obtaining the mixture state representation h'', an interaction module is adopted to selectively aggregate the incoming messages, instead of stacking all interaction messages. For the agents in the team, a communication graph $G^c = (V, E)$ is defined. Each node denotes the agents, and each edge denotes the interaction between different agents. Next, the mixture state representation is regarded as messages to be transmitted in this graph. Transformer attention [22] is implemented to extract the interaction vector h^c_i, from the graph by:

$$h^c_i = \sigma\left(\sum_{k \in \mathcal{N}^c_i} \frac{\exp((W^Q h^{''}_k)(W^K h^{''}_i)/d_K)}{\sum_{k \in \mathcal{N}^c_i} \exp((W^Q h^{''}_k)(W^K h^{''}_i)/d_K)} W^V h^{''}_i\right) \tag{9}$$

where $\mathcal{N}^c_i$ is the neighborhood of agent i, W^Q, W^K, W^V is the linear weight matrix. d_K is the dimensionality of keys. σ is a nonlinear action function. Besides, the K-hop [23] communications is used to enlarge the receptive field of agent i. With the interaction, the opponent information can be fused with other agents, which promotes the agents cooperation.

E. Training Method

The final embedding h^c_i of agent i is passed to a policy network and a value network. The policy network outputs a categorical distribution in discrete action space. The value network outputs a scalar value to evaluate the behavior of the policy network. The PPO algorithm in the Actor-critic framework is adopted to train the proposed framework denoted as CPF-RGR. The parameter-sharing method is applied to train all the agents in a decentralized framework. Besides, multiple parallel environments are conducted to speed up the training. Moreover, the policy is trained by minimizing the total loss L_{total} consisting of weighted value loss L_V, policy loss L_π and entropy H. The training algorithm is shown in Algorithm 1.

IV. Simulation

In this section, simulations including predator-prey task and defender-attacker task as shown in Fig. 2 are conducted to show the effectiveness of the proposed framework. Besides, to validate the superiority of the proposed framework, the baseline algorithm TRANSFER [23] is taken for comparisons, which do not adopt the opponent graph reasoning.

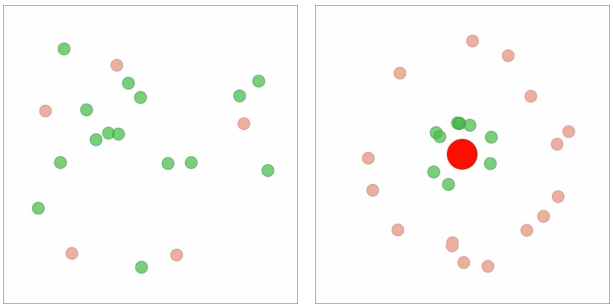
Algorithm 1 Training Algorithm for **CPF-RGR**

Initialize the parameters of actor π, critic V, memory $\mathcal{D}$, initial observation $\boldsymbol{o} = (o_1, o_2, \cdots, o_N)$
Build W parallel environments
for episode =1 **to** num-episodes **do**
  for step =1 **to** num-steps **do**
    Select a action by running actor π
    Model the opponent relation and Predict next state of the alive opponent $\hat{s}'$
    Execute the action $\boldsymbol{a} = (a_1, a_2, \cdots, a_N)$
    Receive the external reward r_{ex} and new observations $\boldsymbol{o'} = (o'_1, o'_2, \cdots, o'_N)$
    Compute the intrinsic reward r_{in} with mean squared losses: $r_{in} = E_{\hat{s}'_{opp} \sim S^a_{opp}}[(\hat{s}'_{opp} - s'_{opp})^2]$, where S_a is is the set of alive opponent.
    Compute the total reward $r = r_{ex} - \alpha r_{in}$
    $\boldsymbol{o} \leftarrow \boldsymbol{o'}$
  end for
  Compute advantages estimates $\hat{A}_t$
  Compute action-values function Q_t
  Store $(\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{o'}, \boldsymbol{Q}, \hat{\boldsymbol{A}})$ in a memory $\mathcal{D}$
  $\pi_{old} \leftarrow \pi$
  for k = 1 **to** ppo-epoch **do**
    Sample a random minibatch from memory $\mathcal{D}$
    Calculate total loss: $L_{total} = \beta_1 L_V + \beta_2 L_\pi - \beta_3 H$
    Update actor and critic network by minimizing L_{total} with gradient descent algorithm
  end for
  Clear memory $\mathcal{D}$
end for

TABLE I

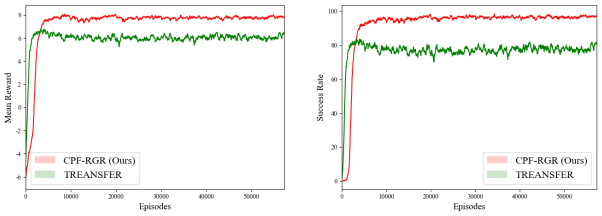
SIMULATION TEST PERFORMANCE OF OUR METHOD AND BASELINE METHOD IN PREDATOR-PREY TASK. *Mean Reward (MR): the mean of rewards for each episode of the agents. Success Rate (S%): the percentage of episodes the agents completed the task. Episode Length (EL): the mean of episode length.*

	Method	Mean Reward	Success Rate	Episode Length
N^a=5, N^o=2	CPF-RGR (Ours)	**7.68**	**96.2**	**30.5**
	TRANSFER	5.88	76.2	39.80
N^a=7, N^o=3	CPF-RGR (Ours)	**7.44**	**96.5**	**31.47**
	TRANSFER	2.72	51.2	44.96
N^a=10, N^o=4	CPF-RGR (Ours)	**7.33**	**98.9**	**27.73**
	TRANSFER	0.6	40.3	46.82
N^a=15, N^o=5	CPF-RGR (Ours)	**6.67**	**99.6**	**25.07**
	TRANSFER	-0.09	49.5	45.19
N^a=20, N^o=6	CPF-RGR (Ours)	**6.35**	**99.9**	**22.42**
	TRANSFER	-0.34	60.3	43.91



(a) Predator-prey (b) Defender-attacker

Fig. 2. The simulation tasks.



(a) Mean Reward (b) Success Rate

Fig. 3. The training curves in 5 vs. 2 predator-prey.

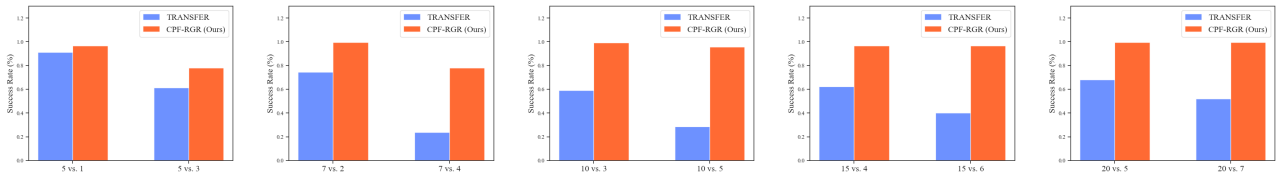
A. Simulation Setting

In these simulations, all tasks are implemented based on Multi-Agent Particle Environment (MAPE) developed by Open AI. The environment size with 4×4 box is considered. The policy is updated by collecting a total of 16384 time-step experience (128 time-steps on 128 parallel environments). The tunable parameter α is chosen as 0.01 for the two tasks. K = 3 is set as K-hop communications between the agents. The parameters of the PPO algorithm β_1, β_2, β_3 are chosen as 0.5, 1, 0.01, respectively. In addition, to speed up the training, curriculum learning [24] based on model load is adopted to train the policy from a small number of agents to a large number of agents.

TABLE II

SIMULATION TEST PERFORMANCE OF OUR METHOD AND BASELINE METHOD IN DEFENDER-ATTACKER TASK.

	Method	Mean Reward	Success Rate
N^a=2, N^o=3	CPF-RGR (Ours)	6.22	97.0
	TRANSFER	6.10	96.2
N^a=3, N^o=5	CPF-RGR (Ours)	6.42	96.5
	TRANSFER	6.23	94.1
N^a=5, N^o=8	CPF-RGR (Ours)	6.80	98.6
	TRANSFER	6.60	96.8
N^a=7, N^o=10	CPF-RGR (Ours)	6.79	99.2
	TRANSFER	6.57	98.1
N^a=10, N^o=15	CPF-RGR (Ours)	6.87	98.6
	TRANSFER	6.72	97.7



(a) $N^a = 5, N^o = 2$ (b) $N^a = 7, N^o = 3$ (c) $N^a = 10, N^o = 4$ (d) $N^a = 15, N^o = 5$ (e) $N^a = 20, N^o = 6$

Fig. 4. Generalization performance of CPF-RGR and baseline method in predator-prey task. The policy trained in the scenario with N^a predators and N^o prey is directly evaluated for other scenarios that fix the number of the predators and change the number of the preys.

TABLE III

GENERALIZATION PERFORMANCE OF OUR METHOD AND BASELINE METHOD IN DEFENDER-ATTACKER TASK. *The policy trained in the scenario with N^a defenders and N^o attackers is directly evaluated for other scenarios that fix the number of the defenders and change the number of the attackers.*

Method	$N^a = 2, N^o = 3$ MR/S(%)		$N^a = 3, N^o = 5$ MR/S(%)		$N^a = 5, N^o = 8$ MR/S(%)		$N^a = 7, N^o = 10$ MR/S(%)		$N^a = 10, N^o = 15$ MR/S(%)	
	2 vs. 2	2 vs. 4	3 vs. 4	3 vs. 6	5 vs. 7	5 vs. 9	7 vs. 9	7 vs. 11	10 vs. 14	10 vs. 16
CPF-RGR (Ours)	99.6/6.17	83.2/5.10	99.2/6.47	92.5/6.20	99.4/6.66	95.8/6.64	99.1/6.65	96.8/6.72	98.9/6.82	96.5/6.84
TRANSFER	99.5/6.11	82.6/5.04	98.7/6.40	91.9/6.16	98.8/6.65	93.3/6.40	98.1/6.57	95.6/6.56	97.7/6.72	96.3/6.83

B. Predator-prey task

As shown in Fig. 2(a), green predators team hopes to capture pink preys team. The preys move faster than the predators. In this section, the predators with learning policy are taken into consideration. Then, we consider a rule-based opponents that the preys move with a scripted policy based on the repulsion forces from the predators and the walls with a repulsive range. The predators get a positive reward when all the preys are eaten. Besides, to guide the predators to learn an effective policy, the predators also are rewarded to get close to the preys team. Therefore, to maximize the accumulated rewards, the predator team should coordinate in order to capture the preys. When the preys are captured or the time-step reaches the maximum steps, the task will be terminated.

The training curves in 5 vs. 2 predator-prey is shown in Fig. 3. Although CPF-RGR learns slower than TRANSFER in the early stage, it can achieve higher performance in the stable stage. This is because the proposed framework needs to spend more time to learn the policy. The final results illustrate the effectiveness and superiority of the proposed framework. This is due to that the opponent graph reasoning module promotes the policy learning. Furthermore, to fully demonstrate the superiority and transferability of the proposed framework, five scenarios, where, N^a predators hunt N^o preys, where, $N^a = 5, 7, 10, 15, 20$, $N^o = 2, 3, 4, 5, 6$, respectively, are conducted by the curriculum learning. Specifically, the policy with fewer predators and preys trained by 5000 updates is transferred to a new scenario with more predators and preys. The comparison results obtained by testing 1000 episodes with the trained policy are shown in Table I. As expected, it is indicated that in any scenario, CPF-RGR has higher performance than TRANSFER in the evaluation criteria of success rate, mean reward, and episode length. The success rate obtained by CPF-RGR are almost twice as much as TRANSFER in the scenario $N^a = 7, 10, 15, 20$, which means that CPF-RGR actually can help agents learn a stable policy in complex scenarios. It is noted that the performance of TRANSFER is greatly degraded in complicated scenarios. However, the proposed framework also has satisfying performance in complex scenarios, especially in large-scale agent scenarios, verifying the transferability of the proposed framework. Therefore, the above results fully reflect the effectiveness and superiority of CPF-RGR.

In order to verify the generalization of the proposed framework, the learned policy trained from old scenarios is directly used to accomplish new scenarios without any fine-tuning. The generalization performance of our method and baseline method is shown in Fig. 4. The results show that the proposed framework has better adaptability than the baseline method, especially in the large-scale agent scenarios.

C. Defender-attacker

As shown in Fig. 2(b), there are two groups; one is the green defender, the other is the pink attacker. The task of the defenders is to eliminate the attackers and prevent them from attacking the red base. The task of the attackers is to avoid eaten by the defenders and attack the base. We consider the defenders with learning policy, while the attackers adopt the Velocity Obstacle (VO) algorithm [25], which is proposed in the navigation problem. The termination is triggered when the base is attacked, or the attackers are eliminated. When all the attackers are eliminated, the defenders get a positive reward. Besides, we consider the case that the number of defenders is less than the attackers. Therefore, the defenders are set to move faster than the attackers. The initial positions of the attackers are set to satisfy the condition that the distance between the attackers and the base should be in the range $[1.5, 2]$. Similarly, the distance between the defenders

and the base should be in the range $[0.4, 0.5]$. The radius of the base is set as 0.25. Therefore, the defenders need to reason about the future state of the attackers and then timely coordinate to accomplish the task.

In this part, five scenarios, where N^a defenders block N^o attackers, where $N^a = 2, 3, 5, 7, 10, N^o = 3, 5, 8, 10, 15$, are conducted to verify the proposed framework through curriculum learning. The simulation results as shown in Table II. It is indicated that CPF-RGR has higher performance than TRANSFER in the mean reward and success rate evaluation criteria. This further shows that the proposed framework can enable the agents to learn an effective policy, owing to reasoning about the opponent state. Furthermore, the generalization tests are also conducted to verify the adaptability of the proposed framework. The comparative results are shown in Table III. When the scenario is complicated, the improvement is significant compared with the baseline method. Therefore, the effectiveness and superiority of CPF-RGR are further validated.

V. Conclusion

In this paper, a collaborative policy framework via relational graph reasoning is proposed for multi-agent systems to accomplish adversarial tasks. A relational graph reasoning module is creatively designed to enable the agents to learn mixture state representation to enhance the effectiveness of the policy. Specifically, an agent graph reasoning with agent relation attention, an opponent graph reasoning with opponent relation modeling, and state prediction are designed to represent the state features of surroundings. Furthermore, to promote the cooperative behaviors among the agents, interactions among agents are used to transmit messages and fuse information. Simulation results verify the effectiveness and superiority of the proposed framework compared with existing methods in several challenging tasks.

References

[1] J. Fischer, C. Lieberoth-Leden, J. Fottner, and B. Vogel-Heuser, "Design, application, and evaluation of a multiagent system in the logistics domain," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1283–1296, 2020.

[2] T. Yang, Z. Jiang, R. Sun, N. Cheng, and H. Feng, "Maritime search and rescue based on group mobile computing for unmanned aerial vehicles and unmanned surface vehicles," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7700–7708, 2020.

[3] Z. Sui, Z. Pu, J. Yi, and S. Wu, "Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[4] H. Zhang and P. Gurfil, "Distributed control for satellite cluster flight under different communication topologies," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 617–627, 2016.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] Y. Zhou, F. Lu, G. Pu, X. Ma, R. Sun, H.-Y. Chen, and X. Li, "Adaptive leader-follower formation control and obstacle avoidance via deep reinforcement learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4273–4280.

[8] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[9] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[10] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7236–7243.

[11] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4257–4266.

[12] A. Grover, M. Al-Shedivat, J. K. Gupta, Y. Burda, and H. Edwards, "Learning policy representations in multiagent systems," in *ICML*, 2018.

[13] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Agent modeling as auxiliary task for deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 15, no. 1, 2019, pp. 31–37.

[14] G. Papoudakis and S. V. Albrecht, "Variational autoencoders for opponent modeling in multi-agent systems," *arXiv preprint arXiv:2001.10829*, 2020.

[15] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 007–10 013.

[16] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[17] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[20] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3040–3049.

[21] A. Graves, "Long short-term memory," in *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 37–45.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[23] A. Agarwal, S. Kumar, K. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent teams," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1741–1743.

[24] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 41–48.

[25] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.