Neurocomputing 459 (2021) 383–394

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Structural relational inference actor-critic for multi-agent reinforcement learning

Xianjie Zhang [a], Yu Liu [a,*], Xiujuan Xu [a], Qiong Huang [b], Hangyu Mao [c], Anil Carie [d]

[a] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116620, China
[b] Okinawa Institute of Science and Technology Graduate University, 1919-1 Tancha, Onna-son, Kunigami-gun, Okinawa 904-0495, Japan
[c] Huawei Noah's Ark Lab, No. 3 Xinxi Road, Haidian District, Beijing 100085, China
[d] School of Computer Science, VIT-AP, Amaravathi 522237, India

ARTICLE INFO

Article history:
Received 8 September 2020
Revised 20 April 2021
Accepted 4 July 2021
Available online 7 July 2021
Communicated by Zidong Wang

Keywords:
Multi-agent systems
Deep reinforcement learning
Variational autoencoder
Actor-critic
Graph neural network

ABSTRACT

Multi-agent reinforcement learning (MARL) is essential for a wide range of high-dimensional scenarios and complicated tasks with multiple agents. Many attempts have been made for agents with prior domain knowledge and predefined structure. However, the interaction relationship between agents in a multi-agent system (MAS) in general is usually unknown, and previous methods could not tackle dynamical activities in an ever-changing environment. Here we propose a multi-agent Actor-Critic algorithm called Structural Relational Inference Actor-Critic (SRI-AC), which is based on the framework of centralized training and decentralized execution. SRI-AC utilizes the latent codes in variational autoencoder (VAE) to represent interactions between paired agents, and the reconstruction error is based on Graph Neural Network (GNN). With this framework, we test whether the reinforcement learning learners could form an interpretable structure while achieving better performance in both cooperative and competitive scenarios. The results indicate that SRI-AC could be applied to complex dynamic environments to find an interpretable structure while obtaining better performance compared to baseline algorithms.

1. Introduction

Inference of relations, as one of the most ordinary things to human beings, is a core competence of human cognition [1]. This cognitive ability enables human beings to establish a partnership with others or realize the potential relationship between objects according to certain laws [2]. It can provide a strong inductive bias to describe the world [3] in a structured way. In multi-agent settings, agents can also take advantage of potential relations to form a cooperative structure and achieve better performance.

Recently, reinforcement learning (RL) has shown exciting success in solving cooperative multi-agent problems. The development of deep learning has promoted the application of RL in many fields, such as games [4,5] and robotics [6]. Applying deep RL technologies into the multi-agent application has become a popular trend. Multi-agent reinforcement learning (MARL) has recently drawn attention to various complex tasks, including traffic light control [7], autonomous driving [8], and network packet delivery [9–11].

The multi-agent systems can be similar to our human activities. When facing a task, human beings first establish a cognitive model of the task, then, determine which partners are needed to interact with the current situation. This kind of collaborative relationship usually changes with time and task status. Earlier work mainly establishes fixed coordination graphs (CGs) and value decomposition approach to increase the payoff of the overall system [12,13]. However, CG-based approaches can only be applied to tasks for which there is an existing cooperative structure between agents. Recent work uses manually defined topologies to establish relationships between multiple agents [14–16]. In particular, NCC-MARL [17], which is also based on the existing relationships between agents, uses graph convolutional networks (GCN) and VAE models to make the agent and neighbors have consistent cognition. These pre-defined topologies cannot be applied to special scenarios that need agent interaction. Therefore, the multi-agent system needs a dynamic reasoning mechanism to infer the relationship between agents in an automatic way.

There have also been some attention-based methods to infer the relationship between agents. The first one is MAAC [18], which uses a multi-headed attention to learn a centralized critic. AHAC [19] improves the MAAC, and allows the agent to have different attention weights for teammates and enemies through hierarchical

* Corresponding author.
E-mail address: yuliu@dlut.edu.cn (Y. Liu).

https://doi.org/10.1016/j.neucom.2021.07.014
0925-2312/

attention. However, the invisible observations of enemies are fed into critic network in AHAC, where it is impossible to see the enemy's observations in the real competitive environments. DAACMP [20] is using the attention to explicitly model the dynamic joint policy of teammates in an adaptive manner, but agents designed in the algorithm do not execute in a decentralized way. These attention-based methods learn the importance distribution of other agents for each agent. However, these methods cannot learn the real relationship between agents, and cannot ignore irrelevant agents to simplify policy learning.

In this paper, we propose a MARL framework Structural Relational Inference Actor-Critic (SRI-AC) framework, which can automatically infer the pairwise interaction between agents and learn a state representation. We compare the SRI-AC algorithm with the existing multi-agent algorithms based on global information. These algoritms include centralized training and decentralized execution [21,22], and obtain other agent information through communication [23–27]. Our model can identify the agents that need to interact in advance, and then feed the most relevant agent observation information to the critic network. There are three crucial components in our model.

(1) Centralized critic and decentralized actor framework [21]. In SRI-AC, each agent has a critic which uses information that conditioned on the joint action and relevant observational information during training. To avoid the agent laziness problem, we share the parameters among all critics.
(2) Variational autoencoder (VAE) model. To process relational learning, we use a variational autoencoder (VAE) model to infer the pairwise interaction, as well as to learn a state representation from observed data. The latent code represents the pairwise interaction, while the reconstruction is based on graph neural networks (GNN).
(3) Graph attention network (GAT) model. More importantly, we utilize the learned relationship to form the adjacency matrix between our agents, and our critic network uses a graph attention network (GAT) [28] to integrate the information from neighbor agents.

We evaluate our methods by four challenging tasks, three of which are based on the multi-agent particle environment (MPE) [29], and the other is a fully cooperative football game [30]. Experiments show that our algorithm can form an effective interactive network, leading to a higher reward compared to the baseline algorithms.

The remainder of this paper is organized as follows. Section 2 introduces the background about RL and structured models. In Section 3, we list related work including policy-based and value-based MARL algorithms. Section 4 presents the proposed method, where we outlined our approach in Section 4.1. The details of the VAE model and the GAT model are presented in Section 4.2 and Section 4.3, respectively. The examples and results are presented in Section 5. The conclusion and future work are given in Section 6.

2. Background

We consider a cooperative multi-agent setting that can be formulated as the Decentralized Partially Observable Markov Decision Process (DEC-POMDP) [31]. It is defined by a tuple $\langle S, O, U, R, P, \gamma, N\rangle$. We assume each agent holds a partially observable state, $o_i \in O$, which contains partial information from the global state, $s \in S$. Action set for N agents is $U = A_1 \times \cdots \times A_N$, at which every time step agent i chooses an action, $a_i \in A_i$. State transition function from state s to s' is followed by: $P(s'|s, a_1, \ldots, a_N) : S\times \quad U \times S \to [0, 1]. \quad R = \langle r_1, \ldots, r_N\rangle : S \times A_1 \times \ldots \times A_N \to \mathbb{R}^N$ is the joint reward function. After taking joint action $a = \langle a_i, \ldots, a_N\rangle$, the agents receive a joint reward R, and transform the state s to s'. The agents aim to learn a policy $\pi_i(a_i|o_i)$ that maximizes their expected discounted return $\mathbb{E}_{a_1\sim\pi_1,\ldots,a_N\sim\pi_N,s\sim P}\left[\sum_{t=0}^{\infty}\gamma^t r_{i,t}(s_t, a_{1,t}, \ldots, a_{N,t})\right]$, where agent i can only observe its state o_i, and $\gamma \to [0, 1]$ is the discount factor.

2.1. Reinforcement Learning (RL)

Deep Q-network (DQN). A well-understood value-based RL algorithm for single agent learning is Q-learning [32,33]. In practice, the Q-value (or, action-value) function is defined as $Q(s, a) = \mathbb{E}[G|S = s, A = a]$, which can be recursively rewritten as $Q(s, a) = \mathbb{E}_{s'}[r(s, a) + \gamma\mathbb{E}_{a'}[Q(s', a')]]$. The Q-learning updates the state-action value using value iteration. To apply it to high-dimensional state, Deep Q-Network (DQN) [5] combines reinforcement learning and deep neural networks. The DQN can be written as $Q(s, a; w)$, where w is network parameters. The parameters w are updated by minimizing the loss: $L(w) = \mathbb{E}_{s,a,r,s'}[y' - Q(s, a; w)]$, where y' is computed by a target network $y' = r+\gamma\max_{a'} Q(s', a'; \bar{w})$. $\bar{w}$ is a target network parameter updated by copying w.

Policy Gradient (PG). Different from value-based algorithms, policy gradient methods' primary purpose is to directly adjust the policy network θ to maximize the agent's expected return $J(\pi_\theta)$. In the classical algorithm REINFORCE [34], the gradient can be written as: $\nabla_\theta J(\pi_\theta) = \sum_{t'}\gamma^t r_{t'}(s_{t'}, a_{t'})\nabla_\theta \log(\pi_\theta(a_t|s_t)$. This algorithm has a problem of high variance, which is caused by the cumulative reward term $\sum_{t'}\gamma^t r_{t'}(s_{t'}, a_{t'})$. Actor-Critc [35] replaces the return term with an approximate $Q_w(s_t, a_t) = \mathbb{E}\left[\sum_{t'}\gamma^t r_{t'}(s_{t'}, a_{t'})\right]$,

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \log(\pi_\theta(a_t|s_t))Q_w(s_t, a_t) \tag{1}$$

This formula is updated in the same way as Q-learning, i.e., by minimizing the temporal-difference loss: $L(w) = \mathbb{E}_{s,a,r,s'}[y' - Q_w(s, a; w)]$, where $y' = r + \gamma\mathbb{E}_{a'\sim\pi(s')}[Q(s', a'; \bar{w})]$. Deterministic Policy Gradient (DPG) [36] extends Actor-Critic to deterministic policies $a = \mu_\theta(s)$. Here we optimize the actor network in the direction of the gradient of Q to maximize the action-value function. For state s, the parameters θ are updated along the gradient direction $\nabla_\theta Q^\mu(s, \mu_\theta(s))$. The objective takes the average over the state distribution $\nabla_\theta J(\theta) = \mathbb{E}_{s,a,r,s'}\left[\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)}\right]$. To ensure the existence of gradient of the action-value with respect to actions, the action needs to be continuous. Based on DPG, Deep Deterministic Policy Gradient (DDPG) is proposed by Lillicrap et al. [37], using deep neural networks to approximate the actor $\mu_\theta(s)$ and the critic $Q(s, a; w)$. It updates the actor and the critic network by:

$$\begin{aligned}&\text{Critic}:\\ &L(w) = \mathbb{E}_{s,a,r,s'}[y' - Q(s, a; w)]\\ &\quad y' = r + \gamma\mathbb{E}_{a'\sim\mu(s')}\left[Q(s', a'; \bar{w})|_{a'=\mu_{\bar{\theta}}(s')}\right]\\ &\text{Actor}:\\ &\nabla_\theta J(\theta) = \mathbb{E}_{s,a,r,s'}\left[\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s, a; w)|_{a=\mu_\theta(s)}\right]\end{aligned} \tag{2}$$

2.2. Structured models

The modeling structure of interacting multi-object or multi-agent systems can achieve higher predictive accuracy by considering the structured nature of such systems [38,2,39]. Jaques et al. [40] proposed a model based on the encoder-decoder architecture. Based on this architecture, this paper extracts the position and speeds of each object from the image, using a differentiable physics engine to calculate the position change of the object. This architec-

ture also can be applied to the three-body problem and the spring problem. C-SWMs [39] uses a graph neural network to find pairwise interaction relations between object states, and discover objects from raw pixels. There are also many previous works about state representation learning through Variational Autoencoder (VAE). For example, World Model [3] uses a standard VAE to calculate the latent space of the image and utilize this latent space to abstract our compressed state.

3. Related work

MARL has been extensively studied in a limited state and action space [41,42]. Independent Q-learning [43] uses an independent controller for each agent, which ignores the non-stationarity caused by the actions of other agents. Based on coordinate graphs, Guestrin et al. [12] proposed a variable elimination (VE) algorithm utilizing conditional independence properties between agents, in order to achieve maximization joint payoffs. Max-plus [13] also uses coordination graphs to find an approximately maximizing joint action by payoff propagation. However, these methods must be based on the prior knowledge of known graph structure. Compared with these traditional methods, SRI-AC can learn a collaborative graph in a dynamic environment.

Recent work on learning skills in high-dimensional complex environments in the Actor-Critic framework is more relevant. Lowe et al. [21] and Foerster *et al.* [22] proposed methods using centralized critic and decentralized actors for continuous actions and discrete actions, respectively. The core idea of MADDPG is to learn a centralized critic with global information and take the policy with only local observation. Foerster *et al.* [22] proposed COMA for solving the credit assignment problem by letting each agent know its relative contribution to the team with counterfactual reward. MAAC [18] also applies centralized critics with attention mechanism to handle a non-stationary environment. Another way to decentralized agents is to use factored value function. Examples are VDN [44], QMIX [45], and QTRAN [46]. VDN uses a simple mix Q-network which is the sum of independent utility functions. QMIX ensures that each independent Q increases monotonically for the overall effect by feeding global state in mix-network. However, these methods do not consider the structural information between agents. DCG [16] combines a static coordination graph and deep learning, factoring the joint value function into payoffs between pairs of agents. In DCG, the static coordination graph is manually defined. None of these methods explicitly learn the cooperative relationship between agents, so the coordination they can achieve is limited.

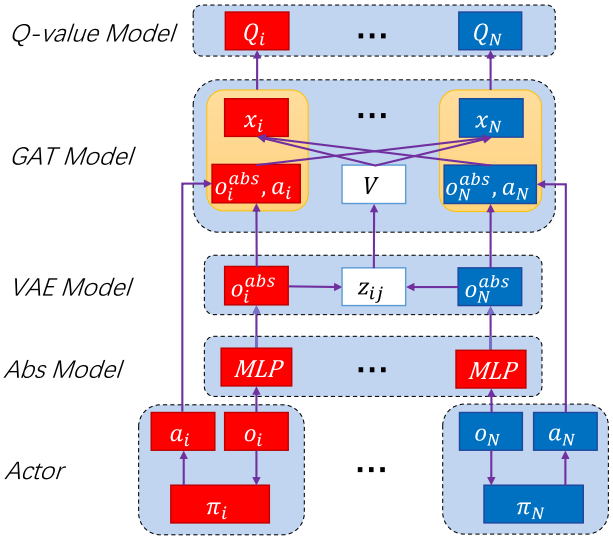


Fig. 1. The network structure of SRI-AC. Note that V is the adjacency matrix composed of pairs of relations z_{ij}.

4. Methods

In this section, we implement Actor-Critic for multi-agent games with centralized critic, which can infer the interaction relationship between agents. To learn the relationship, we model all agents into the graph structure $G = (\mathscr{V}, \mathscr{E})$, where the node $i \in \mathscr{V}$ represents one agent in G, and the edge $z_{ij} \in z$ indicates the relationship between agent i and j. To begin with, we assume that the agents can get the cognition of the world model through local observation. We build a variational autoencoder (VAE) [47,48] model to predict the interactions z with agents' partial observation information. In the end, our centralized critic takes advantage of Graph Attention Network (GAT) [28], getting extra information based on a dynamic graph, composed of interactions z.

4.1. Overall design of SRI-AC

Our SRI-AC network structure consists of the following parts, as shown in Fig. 1.

(1) State Abstraction (Abs Model)

With the game running, we can get each agent's replay buffer which includes states o_i, actions a_i, reward r_i, and follow-up states o_i' that constitute tuples $\mathbb{D} = \{o_i, a_i, r_i, o_i'\}_{t=0}^{T}$. Based on the fully connected state abstract model, the raw high-dimensional observation state o_i is compressed into low-dimensional abstract information $o_{i,abs}$. We use a multilayer perceptron (MLP) network to eliminate redundant information and obtain embedding abstraction $o_{i,abs}$.

(2) Variational Autoencoder (VAE) Model

In this model, we implement our variational autoencoder with an encoder-decoder framework which includes an abstract state reconstruction error and a KL-divergence. The encoder learns the pairwise interaction posterior $q_\phi(z|o)$ with all agents' partial observation information. Under the condition of action a, the decoder can predict the next moment abstract state $o'_{i,abs}$ which can be written as $o_{i,abs} \times a_i \rightarrow o'_{i,abs}$. We formulate the decoder as $p(o'_{abs}|(o_{abs}, a), z)$. For all of them, we use the Graph Neural Network (GNN) message passing operation to capture all agents' structural information.

(3) GAT Model

All the agents can be considered as nodes of graph G, based on agent's interactions z_{ij}. We combine all pairwise relations z_{ij} to form adjacency matrix V. More importantly, we aggregate information from other agents with graph attention under the adjacency matrix V. In the process of updating, agent i can see the observations of neighbor agents and obtain the attention coefficient α_{ij}, which is the importance of the neighbor agents $j \in N_i$.

(4) Q-value Model

We calculate the action-value $Q_i(o, a)$ for every agent, while updating together to minimize a joint loss function of a mix-Q. The Q-value model captures aggregate information from all agents' action $a = (a_1, \cdots, a_N)$ and observation $o = (o_1, \cdots, o_N)$. It can be written as: $Q_i(o, a) = Q_i((o_i, a_i), x_i)$, where x_i is the result of agent i's neighbor information obtained by GAT model. The Q-value model adopts parameter sharing to address the "lazy agent" problem and this design also reduces the parameter size.

4.2. VAE model

We model the structural inference as a VAE as shown in Fig. 2. For dynamic multi-agent systems, there are latent interaction z

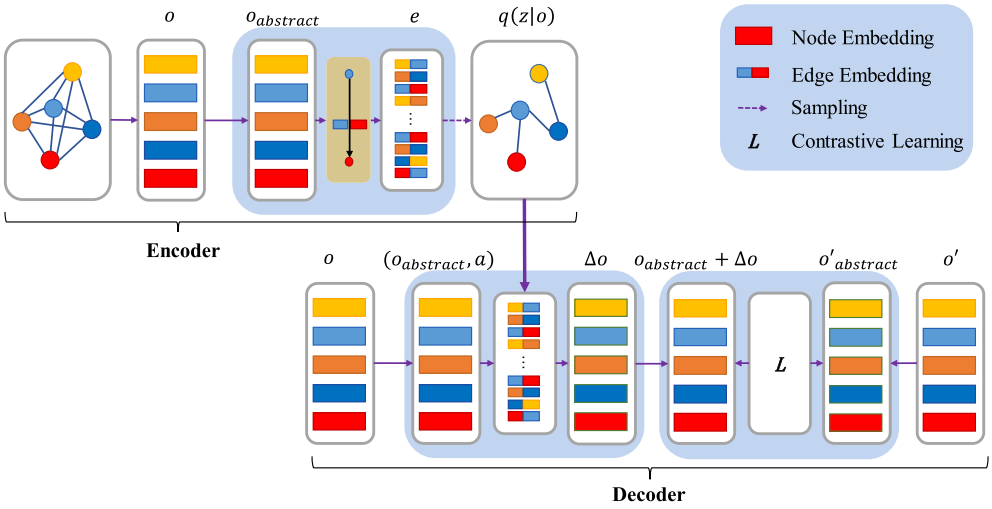


Fig. 2. There are two parts in VAE model: An encoder that predicts the interactions from agents' observations; and a decoder that contrasts between contrasts between the reconstructed and the real observations given the agents' interactions.

between agents, changing over time, given from every agent's local observation o_i. We construct our model which includes approximate posterior distribution $q_\phi(z|o)$ as inference neural networks with trainable parameters ϕ. Based on this potential interaction, the local relationship z_{ij} can realize the collaboration of the entire multi-agent system. We consider that each agent has a partially observable observation o_i, and the number of agents is N. At time t, we use a graph neural networks (GNN) network to capture our structure with the set of observations of all N agents: $o = \{o_1, \ldots, o_N\}$. Finally, through the GNN network, we generate the representation e_{ij} of our edges. We can obtain $N \times (N-1)/2$ edges, in which each edge z_{ij} connects agents v_i and v_j. An edge is a binary variable sampled as $z_{ij} \in \{0, 1\}$. There is one path from v_i to v_j if and only if $z_{ij} = 1$. All edges can form a real graph structure G.

We need to calculate the agent interaction structure under the condition of local observation o, which can be written as $p(z|o)$:

$$p_\theta(z|o) = \frac{p_\theta(o|z)p_\theta(z)}{p_\theta(o)} = \frac{p_\theta(o|z)p_\theta(z)}{\int_z p_\theta(o|z)p_\theta(z)\,dz} \tag{3}$$

For computing the denominator, we need to integrate all the terms with respect to z. The above calculation is unrealistic, hence we need to find approximate distribution $q(z|o)$. The approximate probability must be as close to the real posterior distribution $p(z|o)$ as possible.

$$\begin{aligned}\log p_\theta(o) &= \int q_\phi(z|o)\log p_\theta(o)dz \\ &= D_{kl}\left(q_\phi(z|o)\|p_\theta(z|o)\right) + \int q_\phi(z|o)\log\frac{p_\theta(z,o)}{q_\phi(z|o)}dz\end{aligned} \tag{4}$$

Since the probability $p_\theta(o)$ is fixed, we can maximize the evidence lower bound (ELBO) [47,48] to make the probability $q_\phi(z|o)$ and $p_\theta(z|o)$ as close as possible.

$$\max_{\theta,\phi}\int q_\phi(z|o)\log\frac{p_\theta(z,o)}{q_\phi(z|o)}dz = \max_{\theta,\phi}\left\{-D_{kl}\left(q_\phi(z|o)\|p_\theta(z)\right) + E_{q_\phi(z|o)}[\log p_\theta(o|z)]\right\} \tag{5}$$

We formalize our VAE loss function as L_{vae}:

$$L_{vae} = D_{kl}\left(q_\phi(z|o)\|p_\theta(z)\right) - E_{q_\phi(z|o)}[\log p_\theta(o|z)] \tag{6}$$

There are two parts in our VAE loss function: the first part is an encoder that predicts the interactions from agents' observations, and the other part is reconstruction error that contrasts between reconstruction and real observation error given the agents' interaction. The model is shown in Fig. 2.

4.2.1. Encoder - edges inference

The first item is KL divergence in the loss function L_{vae}:

$$\begin{aligned}D_{kl}\left(q_\phi(z|o)\|p_\theta(z)\right) &= -H\left(q_\phi(z|o)\right) - \int_z q_\phi(z|o)\log p_\theta(z) \\ &= -H\left(q_\phi(z|o)\right) - const\end{aligned} \tag{7}$$

We have a constant term "$-const$", which is generated by the probability of the uniform prior $p_\theta(z)$. In order to infer the pairwise interaction relations z_{ij} between agents from observation set $o = \{o_1, \ldots, o_N\}$, where we draw inspiration from NRI [2]. This model takes fully-connected graph as input of GNN to predict the latent graph structure $q_\phi(z|o)$, where ϕ are the parameters of the fully-connected graph neural networks. Firstly, we concatenate the features of the pairwise agents as their edges' representation, $e_{ij} = f_e([o_i, o_j])$, where f_e is a MLP network, and e_{ij} is the representation of the edge connecting agent i and j. Then we can use softmax function to calculate the probability of each edge. We formalize the posterior probability of the edges as $q_\phi(z_{ij}|o) = \mathrm{SoftMax}(e_{ij})$.

4.2.2. Decoder - contrastive learning

In the second part, we use the contrastive learning loss as the reconstruction error in our VAE model. By sampling from continuous relaxation, we can get the dynamic interaction graph structure $z_{ij} \in \{0, 1\}$ between agents through the encoder. Based on the interaction relationship z_{ij}, we construct the graph structure G. The reconstruction error is to predict the dynamics of observations of the agents at the next moment $o'_{abs} = f_\theta((o_{abs}, a), z)$. In our setting, the last item of VAE loss function, $E_{q_\phi(z|o)}[\log p_\theta(o|z)]$, can be written as follow:

$$E_{q_\phi(z|o)}\log p_\theta(o|z) \doteq -\|f_\theta((o_{abs}, a), z) - o'_{abs}\| \tag{8}$$

Similar to NRI, we model the decoder with interaction neural networks f_θ, where the θ denotes the parameters of our networks. Firstly, we obtain the concatenation of abstracted state o_{abs} and

action a as $\widehat{o} = (o_{abs}, a)$ as the initial information of each node in the graph G. For each type of edge, we specify a MLP network. The same types share the same parameters. We can obtain the representation of edges $\widehat{e}_{(i,j)} = \sum_k z_{ij,k} \widehat{f}_e^k([\widehat{o}_i, \widehat{o}_j])$, where k is the k-th type of edges. In our setting, we use categorical type edges, so this formula can select useful edges. In the end, all information is aggregated to vertices of the edges $\Delta o_i = \widehat{f}_v\left(\sum_{j \neq i} \widehat{e}_{(i,j)}\right)$, where Δo_i is update. For agent i, we can predict the next step state abstraction $f_\theta((o_{abs}, a), z) = o_{i,abs} + \Delta o_i$. The contrastive loss can be obtained as follow: $\|f_\theta((o_{abs}, a), z) - o'\| = \sum_i \|o_{i,abs} + \Delta o_i - o'_{i,abs}\|$, so we can rewrite the reconstruction error as:

$$E_{q_\phi(z|o)} \log p_\theta(o|z) \doteq \sum_i \|o_{i,abs} + \Delta o_i - o'_{i,abs}\| \tag{9}$$

4.2.3. Differentiable sampling - gumbel softmax

In our model, the interaction between agents is categorical type. However, after sampling directly from probability, the model will lose the differentiability. To obtain this categorical interaction, we utilize gumbel-softmax to sample the edges z of our interaction structure. The method can obtain a continuous relaxation result, recently proposed by [49,50] , which can be effectively applied to our setting.

$$\mathbb{G}(\log p)_k = \frac{\exp((\log p_k + \xi)/\tau)}{\sum_{j=0}^{K} \exp((\log p_j + \xi)/\tau)} \tag{10}$$

where ξ can be sampled from distribution $Gumble(0,1), \xi = -\log(-\log(u)), u \sim U[0,1]$. τ is a temperature parameter that controls softmax approaching argmax. When the parameter ξ approaches 0, our distribution will turn to one-hot coding. After obtaining our discrete distribution, the whole model is able to use the backpropagation algorithm.

4.3. GAT model

We obtain interaction z_{ij} with the VAE model, which can integrate as the adjacency matrix V of the interaction graph structure. Based on the generated graph structure, we can use a GAT model to pay attention to neighborhoods' information of an agent, which will specify different weights to different agents in one agent's neighborhood. In traditional GNN, node embedding update in a graph is as follow:

$$H^{l+1} = \sigma\left(VH^l W^l\right) \tag{11}$$

where hidden layer embedding of each node H^l performs a linear transformation with a weight matrix W^l. Multiplication with the adjacency matrix V means that this formula combines all the neighbor node information. To that end, using an activation function, we will get the next step hidden embedding H^{l+1}.

In our model, we consider that agents have different attention to nearby agents, therefore we use graph attention network to capture observations of nearby agents, which is written as

$$H_i^{(l+1)} = \|_{k=1}^{K} \sigma\left(\sum_{j \in N(i)} \alpha_{ij,k}^{(l)} H_j^{(l)} W_k^{(l)}\right) \tag{12}$$

where $\alpha_{ij} = \sigma((H_i W) C (H_j W))$ is an attention coefficient which indicates the importance between i-th and j-th node, and C is a coupling matrix corresponding to the dictionaries of pairwise interaction relations [51]. We use a multi-head attention, therefore $\|$ is used to connect the features from different channels.

We can take advantage of GAT model to calculate the contribution of other agents' states and actions feed into critic. Each agent has a Q-value function $Q_i(o, a)$, which receives the observations $o = (o_1, \ldots, o_N)$, and actions, $a = (a_1, \ldots, a_N)$. In our setting, we can use the following formula to calculate the contribution of all agents feed into critic $Q_i(o, a)$:

$$Q_i(o, a) = Q_i((o_i, a_i), x_i) \tag{13}$$

where (o_i, a_i) is concatenation of agent i' state and action, and x_i indicates the information of other agents calculated from GAT model. For calculating x_i, we can rewrite Eq. (12) as:

$$x_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in N(i)} \alpha_{ij,k}^{(l)} m_j W_k\right) \tag{14}$$

where agent i's neighbor node information, $m_j, j \in N(i)$, can be written as $m_j = (o_j, a_j)$. The attention coefficient becomes as:

$$\alpha_{ij} = \sigma\left((m_i W) C (m_j W)^T\right) \tag{15}$$

The activation functions $\sigma(\sim)$ are tanh for Eq. (14) and Leaky ReLU for Eq. (15), respectively. In the end, we get the concatenation vector of the contributions from all heads.

4.4. Training SRI actor-critic

Our algorithm has a centralized critic like the MADDPG [21] and MAAC [18], and the actors are independent with each other. Fig. 1 shows the critic $Q_i, (i \in N)$, where we need two loss functions for calculating our ultimate Q-value. The first loss is a VAE loss:

$$\begin{aligned} L_{vae} &= \mathbb{E}_{(o,o') \sim D} D_{kl}(q_\phi(z|o) \| p_\theta(z)) - E_{q_\phi(z|o)}[\log p_\theta(o|z)] \\ &= \mathbb{E}_{(o,o') \sim D}\left[-H\left(q_\phi(z|o) + \sum_i \|o_{i,abs} + \Delta o_i - o'_{i,abs}\| - const\right)\right] \end{aligned} \tag{16}$$

The second loss is a temporal-difference loss, the parameters of which is shared in the agent's critic networks [44]. This design can avoid lazy agents in a cooperative setting. We update all critic networks with a mix-Q model, which is the accumulation of total agents' Q-value. Mix-Q of SRI-AC is trained by:

$$\begin{aligned} L_Q(w) &= \sum_{i=1}^{N} \mathbb{E}_{(o,a,r,o') \sim D}\left(Q_i^w(o, a) - y_i\right)^2 \text{where}: \quad y_i \\ &= r_i + \gamma \mathbb{E}_{a' \sim \pi_{\bar{\theta}}(o')}\left[Q_i^{\bar{w}}(o', a')\right] \end{aligned} \tag{17}$$

We then combine two Eqs. (16) and (17) as our total loss:

$$L_{total} = \alpha L_{vae} + L_Q(w) \tag{18}$$

Our algorithm uses an Actor-Critic framework, hence the training process needs to take Q-value and its observation. We extend Eq. (1) into the multi-agent setting as follows:

$$\nabla_{\theta_i} J(\pi_\theta) = \mathbb{E}_{o \sim D, a \sim \pi}\left[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i)) Q_i^w(o, a)\right] \tag{19}$$

The pseudo-code for algorithm is presented in Algorithm 1 and Algorithm 2.

Algorithm 1: Structural Relational Inference Actor-Critic (SRI-AC)

1: Initialize parallel environments E for all agents
2: Initialize replay buffer, D
3: $T_{update} \leftarrow 0$
4: **for** each training episode *epi* **do**
5: Reset environment, and get intial obs_i^e for each agent i
6: **for** $t = 1 \ldots$steps per *epi* **do**

(*continued on next page*)

a (*continued*)

Algorithm 1: Structural Relational Inference Actor-Critic (SRI-AC)

```
    choose actions a_i^e ~ π(·|obs_i^e) for each agent
    Get reward r_i and next obs'_i^e for all agents
    Add episode to buffer, D
    T_update = T_update + E
    if T_update ⩾ BatchSize then
      for j = 1 ... num updates do
        Sample minibatch, B
        Get latent interactions, z_ij ~ q(·|B)
        Combine all z_ij as adjacency matrix, V
        UPDATECRITICAndSRI(B,V)
        UPDATEPOLICIES(B, V)
      end for
      Update target parameters
      T_update ← 0
    end if
  end for
end for
```

Algorithm 2: Update Procedure for Critic, VAE and Polices

```
function UPDATECRITICandSRI(B, V)
  Unpack minibatch
  (o^B_1...N, a^B_1...N, r^B_1...N, o'^B_1...N) ← B
  Calculate Q_i^w(o^B_1...N, a^B_1...N, V) for every agent on the
  condition of adjacency matrix V
  Calculate a'^B_i ~ π_i^θ(o'^B_i) using target policies
  Calculate obs', a' using target polices
  Calculate Q_i^w̄(o'^B_1...N, a'^B_1...N, V) for all agents using target
  critic
  Calculate ELBO L_vae
  Calculate TD-error L_Q
  Update critic using ∇(L_Q + L_vae)
end function

function UPDATEPOLICIES(B, V)
  Calculate a^B_1...N ~ π_i^θ̄(o'^B_1...N), i ∈ 1...N
  Calculate Q_i^w(o^B_1...N, a^B_1...N, V) for all agent
  Update policy using ∇J(π_θ)
end function
```

5. Experiments

In order to validate the performance of our algorithm, we test SRI-AC in four experimental environments. The first three scenarios are multi-agent particle environment proposed in [21,29,52]. In our setting, we change the first three into a two-dimensional world with discrete-time, continuous space and discrete action spaces. Agents can execute five actions, including up, down, left, right, or stay. In the first three scenarios, we choose DDPG [37] with discrete actions to train the opponent's policy. For comparing the performance between SRI-AC and baseline algorithms, we fix the parameters of DDPG's module when opponents learn how to escape or attack. The last environment is google research football environment [30], which is an advanced, physics-based 3D simulator with an open-source license. The football scenario supports multi-agent settings, which is challenging for the study of reinforcement learning. The details of the experimental environments are as follows.

5.1. The testing environments

5.1.1. Predator-prey

We choose the predator-prey scenario which is one of the Multi-Agent Particle Environments (MPE) [21]. Fig. 3(a) illustrates one of our game fields, in which we extended it as cooperative-competitive environments. This game contains two types of agents: predator and prey. To increase the difficulty of the game, we place random landmarks on the game field, and the landmarks don't take actions. As shown in Fig. 3(a), grey circles represent landmarks, green circles are the good agents (prey) and red circles are the adversary agents (predator). Our algorithm needs to control adversary agents to capture the good agents collaboratively, while the good agents learn a pattern to escape from adversary agents. To verify the collaborative interaction structure, the game needs more than one good agent. Taking the scenario of Fig. 3(a) as an example, we have five homogeneous adversary agents and two good agents. We use SRI-AC to learn adversary agents' strategies. Agents have more than one target, which promotes interaction between agents. The adversary agents' acceleration and speed are all slower than the good agents, hence they need to learn a mechanism of coordination to capture the faster good agents. Each time one adversary agent catches the good agents, this adversary agent gets a positive reward, +10, while the good agent obtains a negative reward, −10.

5.1.2. Grassland

This game consists of two kinds of animals, sheep and wolves, where sheep move faster than wolves [52]. We also set a fixed number of green landmarks as food for sheep, which is shown in Fig. 3(b). When a wolf collides (eats) with a sheep, it will be rewarded. The (eaten) sheep will be rewarded negatively and become inactive (dead). A grass pellet will be collected and regenerated in another random position when a sheep encounters it, and the sheep will get a positive reward. In order to incentivize the sheep work as a team, we set a shared reward value: when a sheep eats grass, others can get part of the reward; when one sheep dies, the whole sheep herd will be punished.

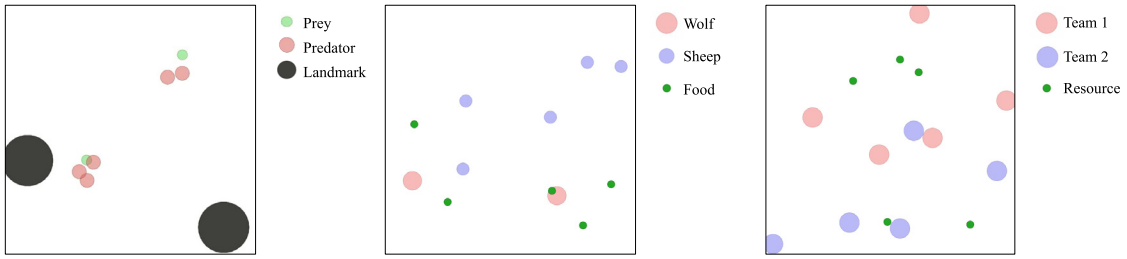
5.1.3. Adversarial battle

As shown in Fig. 3(c), we have two opposite teams in this scenario. The two teams need to attack each other and compete for resources to improve their teams' reward score. As in the grassland scenario, resources will appear in random positions after being collected, and the total number of resources will not change. The agent who collects resources will get a reward value, and his team will also receive a shared reward value. We also design another mechanism for two teams of agents to attack each other. When more than two agents from team one collide with one agent in the other team at the same time, the alone agent will die and his whole team will get negative rewards, while the attacking agent and their whole team will get positive rewards [52].

5.1.4. Google football

This environment is an open-source virtual football game for reinforcement learning research, which enables us to quickly verify and test the designed algorithm in sports games. The physical 3D simulation model of the game scenario is very close to the real football game [30].

In the simulator, we can set two football teams, from which we can choose one to select players that we can control. This soccer environment supports three state representations: pixels, super-mini map, and floats. We choose the floats as features, which can



(a) Predator-Prey 5v2. (b) Grassland 5v2. (c) Adversarial 5v5.

Fig. 3. All scenarios are based on the multi-agent particle environment (MPE). (a) is predator-prey scenario, where we can see 5 adversary agents chasing 2 good agents and 2 large dark circles indicate landmarks impeding the way. (b) is the grassland scenario. This game has 5 sheep, 2 wolves and fixed amount of grass pellets as green landmarks. (c) is adversarial scenario. In this game, we have two teams of agents and fixed number of resources as green landmarks.

represent the current critical state of all games with 115-dimensional floating-point vector. It contains position, speed, and direction of all athletes, as well as the game's information of one-hot vector. In order to make the observation of each agent different, we add the coordinates of each agent relative to other entities to the original 115-dimensional vector. Finally, we form 161-dimensional vector. These floating-points numbers are critical features for algorithm training. This representation greatly reduces the difficulty of feature extraction. We can focus on the design of the reinforcement learning algorithm. In the original game, there are 21 basic actions. In order to improve the convergence speed of the algorithm, we simplify the actions into several basic actions, including idle, top right, right, bottom right, high pass, and shot. We choose scoring and checkpoints as agents' reward. When a controlled team scores a goal, the reward is +1 for all members of the team. If an agent brings the ball close to the goal, he can also get checkpoint reward.

5.2. Comparison with the baselines

5.2.1. Baselines

We compare with 6 algorithms in our examples, three of which are recently proposed algorithms based on the framework of centralized training and decentralized execution: MADDPG [21], ATT-MADDPG [11], NCC-MARL [17], AHAC [19], MAAC [18], and COMA [22]. Apart from these three algorithms, we also designed several structurally simplified algorithms: the Centralized-AC that feeds the concatenated observation of all agents to the centralized critics, two single-agent RL methods DDPG and IAC that make each agent learn independently with its own actor and critic. All of our environments use discrete action spaces, while the original DDPG, MADDPG and ATT-MADDPG use continuous actions. In the process of training, if we sample discrete actions directly, the model will lose the differentiability. In order to transfer the gradient to the actor network, we use the Gumbel-Softmax [50] to sample the actions, and transform the original algorithm into DDPG (Discrete), MADDPG (Discrete) and ATT-MADDPG(Discrete). All of our baseline algorithms have the same network parameters in the actor network. All kinds of critic networks have the same hyper-parameters. SRI-AC is robust enough, so the hyper-parameters are consistent in all environments. A list of the final hyper-parameters for our tests is shown in Table 1.

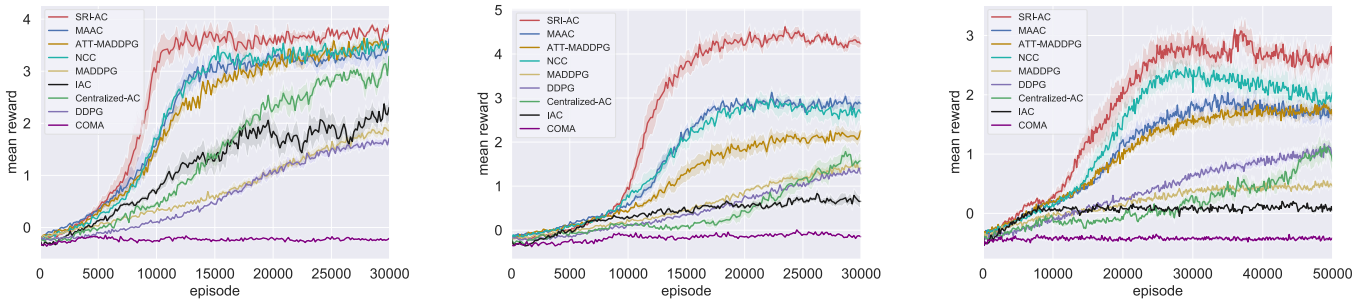
5.2.2. Comparison

Predator-Prey. We compare our method with a series of baselines in predator-prey scenario. In this experiment, we first train 10,000 steps with DDPG to help the good agents learn how to escape. In order to control variables in all algorithms, we fix the policy of good agents, and then train SRI-AC and baseline algorithms for adversary agents. Fig. 4(a) and Fig. 4(b) show the learning curves of 30,000 episodes in 4v2 and 5v2 scenarios, respectively. Because more agents in 6v2 scenario lead to difficult training, we set 50,000 episodes in this scenario. The curve is the mean reward over a sliding window of 500 steps. Shaded regions are one standard deviation over 5 runs. The predator-prey scenario require agents form connect into pair-wise groups to capture multiple targets. As can be seen from Fig. 4, benefited from the relational inference scheme, our algorithm always outperforms its baselines in predator-prey scenarios, when it confronts with DDPG. We also compare with the state-of-the-art algorithm AHAC. In AHAC, the observations of the enemy are fed into critic network, however, those observations are often invisible in the real scene. Therefore, we compared the performance of this case separately in the most difficult scenario (6v2). To evaluate the performance of SRI-AC and AHAC, we set the algorithm SRI-AC (with adv-obs). In SRI-AC (with adv-obs), not only can the structure between agents be established, but also the enemy's observations are also fed into the critic network with the attention mechanism. As shown in Fig. 5, SRI-AC (with adv-obs) still gets more rewards than AHAC. In Section 5.3, we analyze the effectiveness of our algorithm as the number of agents changes in predator-prey scenarios.

Table 1
The hyperparameters used in games.

Hyperparameters	Description	Value
α	Learning rate for the L_{vae}	0.01
γ	Discount factor	0.95
lr_actor	Learning rate for the actor	0.01
lr_critic	Learning rate for the critic	0.01
optimizer	Optimizers for all the networks	Adam [53]
batch_size	How many tuples to sample for each update	1024

Grassland. In this game, we also use DDPG algorithm to train the policies of the wolves. We calculate the average reward of sheep in one episode of the game. Our environment is competitive and part of the reward value is negative. In order to compare the performance of the algorithms, we add 20 to all the original reward. We set up two scenarios, 4v2 and 5v2, in this environment with different numbers of sheep. As shown in Fig. 6, we can see that our algorithm is better than other baseline algorithms. The interesting phenomenon is that the IAC algorithm is better than other baseline algorithms because a single agent only needs to escape the wolf attack and eat the nearest grass, which will lead to a higher reward value. The network does not need information from special agents, and local observations provide enough information for agents. SRI-AC can also automatically infer agents that need to cooperate, so it can effectively filter some unnecessary information. Fig. 4 shows that SRI-AC performs substantially better than all alternative approaches in the 5v2 and 6v2 scenarios, and performance similar to MAAC on 4v2 scenarios. When the number of



(a) Predator-Prey 4v2. (b) Predator-Prey 5v2. (c) Predator-Prey 6v2.

Fig. 4. Learning curves of our method and baselines on predator-prey scenarios. (a) shows the result of 4 adversary agents with 2 good agents training on 30000 episodes. (b) shows the result of 5 adversary agents with 2 good agents training on 30000 episodes. (c) shows the result of 6 adversary agents with 2 good agents training on 50000 episodes. Error bars are one standard deviation over 5 runs.

Fig. 5. Learning curves of SRI-AC (with adv-obs) and AHAC [19]. In SRI-AC (with adv-obs), the observations of the enemy are fed into critic network, which is the same as AHAC.

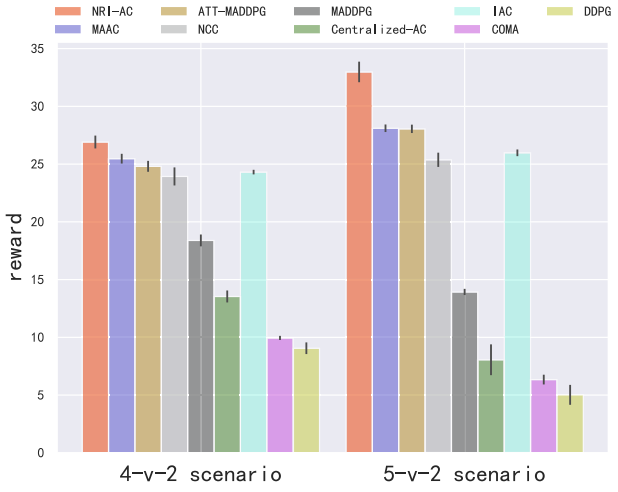


Fig. 6. Grassland scenarios: the average reward of one episode.

agents increases, SRI-AC can avoid collision and performs better. Note that COMA fails to achieve good results in our experiments, because COMA uses a single critic network, which would achieve good results for agents with a global reward value and a single goal. However, in our experiment, there are multiple sheep to chase, so it does not perform very well.

Adversarial Battle. In this game, we set up scenarios 5v2 and 5v5 with 2 opponents and 5 opponents, respectively. The opponents' policy also trained 10000 steps with DDPG algorithm. It can be seen from Fig. 7 that the reward of SRI-AC is higher than other baselines. Whereas, the MAAC algorithm does not produce good results compared to other baseline algorithms. In this adversarial game, It requires two agents to destroy an opponent at the same time to destroy it. Nevertheless, MAAC merges information from all agents, which cannot make good use of local information.

Google Football. Fig. 8 is the scenario of the google football. In this scenario, we limit the maximum number of steps per episode to 150. We calculate the average reward value return at each step in the entire episode as shown in Fig. 9. As the curves show, we can see that our algorithm can also achieve good results in complex environment compared with the baseline algorithm MADDPG.

5.3. Increasing the number of agents

In predator-prey scenarios, we test agents' performance when the number of adversary agents changes from 4 to 6. In Fig. 4(a), for the small number of agents, our algorithm does not show obvious advantages. Because this topology is rather simple, and all methods can find a not-so-bad control policy, regardless of whether the methods adopt advanced relational mechanisms. In Fig. 4(b-c), with the increase of agents, MAAC has a good reward value because of the self-attention mechanism. However, SRI-AC shows a better performance than all baseline algorithms. Com-

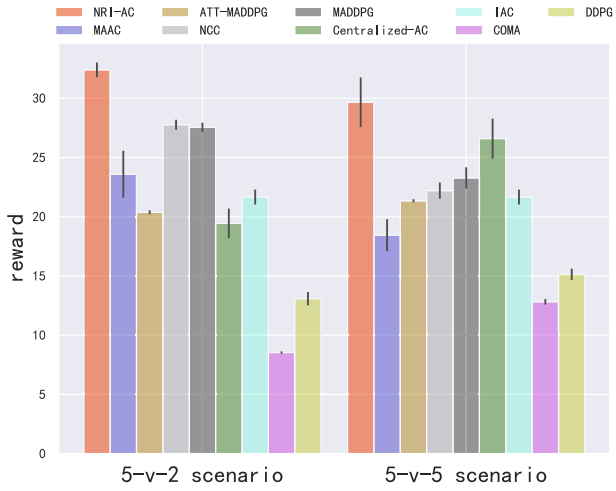


Fig. 7. Adversarial scenarios: the average reward of one episode.

Fig. 8. Football 4v3.

Fig. 9. The average result of football 4v3 scenario.

pared with other algorithms, we find that the performances of Centralized-AC and MADDPG do not perform well as the number of agents increases. The reason is that the critic concatenates feature of all agents, and the dimension of it becomes relatively large.

5.4. Ablation models

In order to verify that our model can generate effective connections, we design fully connected and linearly connected structures. The adjacency matrix of the fully connected graph is a matrix with 0 in every diagonal entry, and 1 in every off-diagonal entry. The linearly connected graph is the sequential connection from the first node to the last node. They are all undirected graphs, so we set them as undirected and symmetric. The adjacency matrices (5 agents) are as follows:

$$M_f = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{20}$$

$$M_l = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{21}$$

where $M_f$and M_l represent fully connected and linearly connected adjacency matrices, respectively. Different from the fixed topology connection, SRI-AC can automatically learn a dynamic relationship. The dynamic relationship forms the adjacency matrix between our agents, which can affect the GAT model by filtering out useless information. The average rewards across 5 runs for different topologies are shown in Fig. 10. We can see that our algorithm has obvious advantages over manually defined structure. This result shows that SRI-AC can learn dynamically changing connections compared to manually defined structure-base algorithms, helping critics get information from neighbor agents, thereby increasing rewards.

5.5. Visualization

We visualized the adversarial scenario. As shown in Fig. 11, our algorithm learned a clever strategy. When the scenario is initialized, our agents (brown) are distributed in the field. Then, these agents quickly concentrate in the middle of the field. They can unite well to avoid being eaten by another team, while a collaborative relationship can be formed in pairs to attack the opponents.

It can be seen from Fig. 12 that, in the predator-prey scenarios, adversary agents can coordinate to prey good agents. At first, there are four agents in order to hunt one good agent. When adversary agent senses that, good agent is only hunted by one hunter at that time, A1 and A2 adjust the pursuit target (good agent) to form a hunting relationship of 2 agents and 3 agents, which can reduce the collision between hunters and increase the reward value of the team.

6. Conclusion and future work

In this paper, we propose the Structural Relational Inference Actor-Critic (SRI-AC), a novel multi-agent deep reinforcement algorithm for collaborative tasks. SRI-AC uses a variational inference model that allows agents to find the necessary cooperative partners automatically in training phase. The algorithm has the characteristics of centralized learning and decentralized execution, allowing agents to take actions in a completely decentralized way in execution phase. SRI-AC is evaluated by four tasks. Experimental results demonstrate that this method can achieve better results than a range of baseline MARL algorithms. The ablation study shows that our model can learn effective interactive topology and promote training.

The advantage of using our algorithm is that when our critic uses the inference relationship, the critic can collect the most relevant neighbor agents' information to estimate a value function, and then optimize the policy network. However, we found that

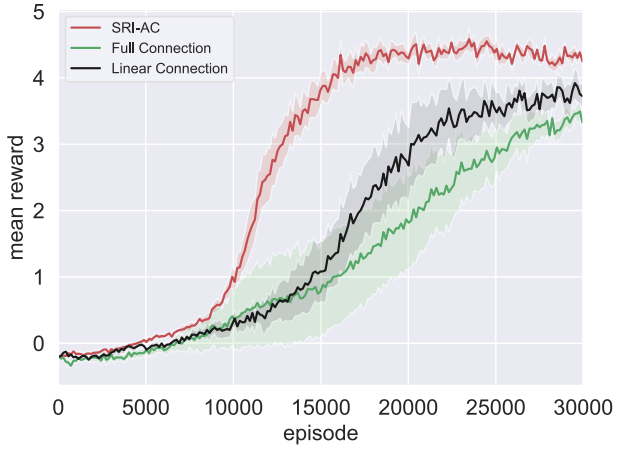


Fig. 10. Comparison between topologies: SRI-AC, linear connection and full connection.

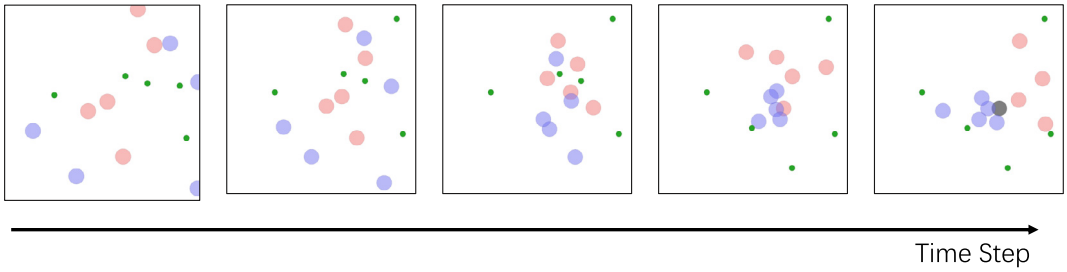


Fig. 11. A convergent joint policy learned by SRI-AC under an instance of the adversarial scenario. Note that dark particles are dead agents.

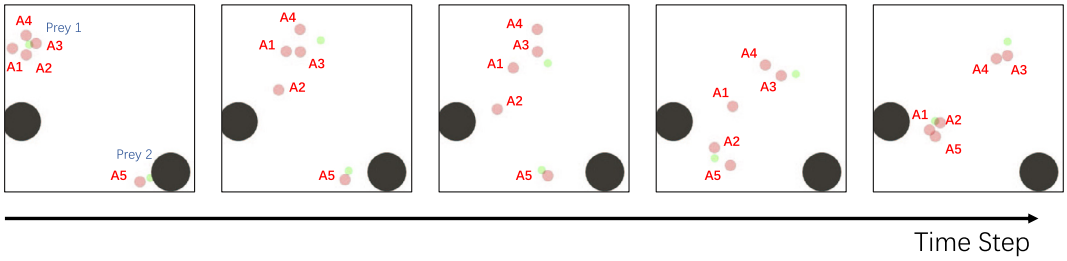


Fig. 12. A convergent joint policy learned by SRI-AC under an instance of the predator-prey scenarios.

when the number of agents increases, the impact on VAE model limits the speed of convergence. Therefore, future research can improve the exploration efficiency of generating interactive relationships. In addition, Communication between agents can also be considered, which allows agents to share high-dimensional information when executing actions. In the end, it can also be integrated with the recursive neural networks (RNN) to improve the utilization of local observations.

CRediT authorship contribution statement

Xianjie Zhang: Conceptualization, Methodology, Software, Validation, Writing - original draft. **Yu Liu:** Supervision, Resources, Funding acquisition, Writing - review & editing. **Xiujuan Xu:** Supervision, Writing - review & editing. **Qiong Huang:** Methodology, Writing - review & editing. **Hangyu Mao:** Methodology. **Anil Carie:** Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant: 61672128) and the Fundamental Research Fund for Central University (Grant: DUT20TD107). The contact author is Yu Liu.

References

[1] E.S. Spelke, K.D. Kinzler, Core knowledge, Developmental science 10 (1) (2007) 89–96.
[2] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, R. Zemel, Neural relational inference for interacting systems, in: International Conference on Machine Learning (ICML), PMLR, 2018, pp. 2688–2697.
[3] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, in: Advances in Neural Information Processing Systems (NIPS), 2018, pp. 2450–2462..
[4] 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 529 (7587) (2016) 484–489.
[5] 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 518 (7540) (2015) 529–533.
[6] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (1) (2016) 1334–1373.
[7] J. Jin, X. Ma, Hierarchical multi-agent control of traffic lights based on collective learning, Eng. Appl. Artif. Intell. 68 (2018) 236–248.
[8] Y. Cao, W. Yu, W. Ren, G. Chen, An overview of recent progress in the study of distributed multi-agent coordination, IEEE Trans. Ind. Inf. 9 (1) (2012) 427–438.
[9] D. Ye, M. Zhang, Y. Yang, A multi-agent framework for packet routing in wireless sensor networks, Sensors 15 (5) (2015) 10026–10047.
[10] H. Mao, Z. Gong, Z. Zhang, Z. Xiao, Y. Ni, Learning multi-agent communication under limited-bandwidth restriction for internet packet routing, arXiv preprint arXiv:1903.05561 (2019)..
[11] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, Modelling the dynamic joint policy of teammates with attention multi-agent ddpg, in: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2019, pp. 1108–1116.
[12] C. Guestrin, M. Lagoudakis, R. Parr, Coordinated reinforcement learning, in: International Conference on Machine Learning (ICML), Vol. 2, Citeseer, 2002, pp. 227–234..
[13] J.R. Kok, N. Vlassis, Sparse cooperative q-learning, in: Proceedings of the Twenty-First International Conference on Machine Learning, Association for Computing Machinery, New York, NY, USA, 2004, p. 61..
[14] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: International Conference on Machine Learning (ICML), 2018, pp. 5571–5580.
[15] J. Jiang, Z. Lu, Learning attentional communication for multi-agent cooperation, in: Advances in Neural Information Processing Systems (NIPS), Curran Associates Inc, 2018, pp. 7254–7264..
[16] W. Boehmer, V. Kurin, S. Whiteson, Deep coordination graphs, in: Proceedings of the 37th International Conference on Machine Learning (ICML), PMLR, 2020, pp. 980–991.
[17] H. Mao, W. Liu, J. Hao, J. Luo, D. Li, Z. Zhang, J. Wang, Z. Xiao, Neighborhood cognition consistent multi-agent reinforcement learning, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020, pp. 7219–7226..
[18] S. Iqbal, F. Sha, Actor-attention-critic for multi-agent reinforcement learning, in: International Conference on Machine Learning (ICML), Vol. 97, PMLR, 2019, pp. 2961–2970..
[19] Y. Wang, D. Shi, C. Xue, H. Jiang, G. Wang, P. Gong, AHAC: actor hierarchical attention critic for multi-agent reinforcement learning, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2020, pp. 3013–3020..
[20] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, Y. Ni, Learning multi-agent communication with double attentional deep reinforcement learning, Auton. Agent. Multi-Agent Syst. 34 (1) (2020) 32.

[21] R. Lowe, Y. Wu, A. Tamar, J. Harb, O.P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems (NIPS), 2017, pp. 6379–6390.
[22] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI Press, 2018, pp. 2974–2982..
[23] J.N. Foerster, Y.M. Assael, N. de Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems (NIPS), 2016, pp. 2137–2145..
[24] A. Singh, T. Jain, S. Sukhbaatar, Learning when to communicate at scale in multiagent cooperative and competitive tasks, in: International Conference on Learning Representations (ICLR), 2019.
[25] S. Sukhbaatar, A. Szlam, R. Fergus, Learning multiagent communication with backpropagation, in: Advances in Neural Information Processing Systems (NIPS), 2016, pp. 2244–2252..
[26] Q. Huang, E. Uchibe, K. Doya, Emergence of communication among reinforcement learning agents under coordination environment, in: 2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), IEEE, 2016, pp. 57–58.
[27] Q. Huang, D. Kenji, An experimental study of emergence of communication of reinforcement learning agents, in: International Conference on Artificial General Intelligence, Springer, 2019, pp. 91–100.
[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations (ICLR), 2018.
[29] I. Mordatch, P. Abbeel, Emergence of grounded compositional language in multi-agent populations (2018) 1495–1502..
[30] K. Kurach, A. Raichuk, P. Stanczyk, M. Zajac, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, S. Gelly, Google research football: A novel reinforcement learning environment (2020) 4501–4510..
[31] D.S. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of markov decision processes, Math. Oper. Res. 27 (4) (2002) 819–840.
[32] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[33] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[34] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256.
[35] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, in: Advances in neural information processing systems (NIPS), 2000, pp. 1008–1014..
[36] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M.A. Riedmiller, Deterministic policy gradient algorithms, in: Proceedings of the 31th International Conference on Machine Learning (ICML), Vol. 32, JMLR.org, 2014, pp. 387–395..
[37] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2016.
[38] S. Sukhbaatar, R. Fergus, et al., Learning multiagent communication with backpropagation, in: Advances in neural information processing systems (NIPS), 2016, pp. 2244–2252..
[39] T.N. Kipf, E. van der Pol, M. Welling, Contrastive learning of structured world models, in: 8th International Conference on Learning Representations (ICLR), 2020.
[40] M. Jaques, M. Burke, T.M. Hospedales, Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video, in: 8th International Conference on Learning Representations (ICLR), 2020.
[41] L. Busoniu, R. Babuska, B. De Schutter, A comprehensive survey of multiagent reinforcement learning, IEEE Trans. Syst. Man Cybern. C 38 (2) (2008) 156–172.
[42] E. Yang, D. Gu, Multiagent reinforcement learning for multi-robot systems: A survey, Tech. rep., tech. rep (2004). .
[43] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents, in: Proceedings of the tenth international conference on machine learning, 1993, pp. 330–337.
[44] P. Sunehag, G. Lever, A. Gruslys, W.M. Czarnecki, V.F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J.Z. Leibo, K. Tuyls, T. Graepel, Value-decomposition networks for cooperative multi-agent learning based on team reward, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2018, pp. 2085–2087.
[45] T. Rashid, M. Samvelyan, C.S. De Witt, G. Farquhar, J. Foerster, S. Whiteson, Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, in: Proceedings of the 35th International Conference on Machine Learning (ICML), Vol. 80, 2018, pp. 4292–4301..
[46] K. Son, D. Kim, W.J. Kang, D.E. Hostallero, Y. Yi, Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning, in: Proceedings of the 36th International Conference on Machine Learning (ICML), Vol. 97, 2019, pp. 5887–5896..
[47] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: International Conference on Learning Representations (ICLR), 2014.
[48] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International Conference on Machine Learning (ICML), Vol. 32, 2014, pp. 1278–1286..
[49] C.J. Maddison, A. Mnih, Y.W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, in: International Conference on Learning Representations (ICLR), 2017..
[50] E. Jang, S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, in: 5th International Conference on Learning Representations (ICLR), 2017.
[51] S. Ryu, J. Lim, S.H. Hong, W.Y. Kim, Deeply learning molecular structure-property relationships using attention-and gate-augmented graph convolutional network, arXiv preprint arXiv:1805.10988 (2018)..
[52] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, X. Wang, Evolutionary population curriculum for scaling multi-agent reinforcement learning, in: International Conference on Learning Representations (ICLR), 2020..
[53] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations (ICLR), 2015.

Xianjie Zhang is currently pursuing the Ph.D. degree with the School of Software Technology, Dalian University of Technology, China. His research interest includes reinforcement learning, multi-agent systems, and generative models.

Yu Liu is currently a full Professor in Software School and Head of Institute of Machine intelligence and Informetrics, Dalian University of Technology, China. He received the Ph.D. degree from Xi'an Jiaotong University, China in 2006 and worked as a postdoctoral research fellow at Department of Computer Science and Technology, Tsinghua Unversity from 2006 to 2008. Professor Liu also worked at Harvard University as a visiting research scientist in 2012. His research interests include swarm intelligence, evolutionary computation, computational intelligence, and Web data mining.

Xiujuan Xu received the Ph.D. degree in College of Computer Science and Technology from Jilin University, Changchun, China, in 2008. She is currently an Associate Professor in School of Software, Dalian University of Technology. Dr Xu worked at School of Computer Science, the University of Adelaide as a visiting scholar in 2016. Her research interests include data mining, intelligent transportation systems, recommender systems, and social network analysis. Dr Xu is the author of more than 50 publications.

Qiong Huang received the B.S. and M.S. degrees in electrical engineering from University of Electronic Science and Technology of China, and is currently a Ph. D. candidate in Okinawa Institute of Science and Technology Graduate University, Japan. Her research interests include reinforcement learning, multi-agent system, emergence of communication and applications of deep reinforcement learning with energy storage systems.

Hangyu Mao received the PhD degree from Peking University, in July 2020. He is currently a researcher with the Huawei Noah's Ark Lab. His research interests include Multi-agent System, Reinforcement Learning, and various intelligent decision-making and reasoning issues. He is a member of the AAAI.

Anil Carie received Ph.D. degree in Software Engineering from Dalian University of Technology, Dalian, China. Currently is Assistant Professor in School of Computer Science and Engineering, VIT-AP, Amaravati, India and Serving as Post-Doctoral in Nanjing Agriculture University, Nanjing, China. His research interests include common control channel design for MAC and routing protocols in Cognitive radio ad-hoc networks, Game Theory, Vehicular Ad hoc Networks, Internet of Things.