Discovering Latent Variables for the Tasks With Confounders in Multi-Agent Reinforcement Learning

Kun Jiang, Wenzhang Liu, Yuanda Wang, Lu Dong, *Member, IEEE*, and Changyin Sun, *Senior Member, IEEE*

***Abstract*—Efficient exploration in complex coordination tasks has been considered a challenging problem in multi-agent reinforcement learning (MARL). It is significantly more difficult for those tasks with latent variables that agents cannot directly observe. However, most of the existing latent variable discovery methods lack a clear representation of latent variables and an effective evaluation of the influence of latent variables on the agent. In this paper, we propose a new MARL algorithm based on the soft actor-critic method for complex continuous control tasks with confounders. It is called the multi-agent soft actor-critic with latent variable (MASAC-LV) algorithm, which uses variational inference theory to infer the compact latent variables representation space from a large amount of offline experience. Besides, we derive the counterfactual policy whose input has no latent variables and quantify the difference between the actual policy and the counterfactual policy via a distance function. This quantified difference is considered an intrinsic motivation that gives additional rewards based on how much the latent variable affects each agent. The proposed algorithm is evaluated on two collaboration tasks with confounders, and the experimental results demonstrate the effectiveness of MASAC-LV compared to other baseline algorithms.**

***Index Terms*—Latent variable model, maximum entropy, multi-agent reinforcement learning (MARL), multi-agent system.**

I. Introduction

REINFORCEMENT learning (RL) has been recognized as an effective method in solving control and navigation problems [1]–[6]. Therefore, RL algorithms are extended from single-agent systems to multi-agent systems to solve some cooperative and competitive tasks [7]–[10]. In a multi-agent system, the non-stationarity [11], [12] during the learning process and the challenge in effectively exploring the environment [13]–[15] are considered two great challenges. The method based on the independent Q-function [16] enables each agent to solve the optimal policy under local observation independently, regardless of the policies and observations of other agents. This makes it impossible to share information between agents, and it is difficult to ensure optimal overall performance. Hence, the paradigm of centralized training with decentralized execution (CTDE) is proposed as a general approach to learning optimal joint policies and stabilizing the training [17]. Besides, some methods based on value decomposition [18]–[20] and maximum entropy [21], [22] are believed to be helpful for the agent to explore the complex environment. The methods mentioned above are all under the observable system, and the generation mechanism of the reward function is deterministic. Even if the system's state is locally observable, the global state can be extracted from the historical sequence using recurrent neural network technology. However, for some scenarios with latent variables that cannot be directly observed by the agent, e.g., the tasks with latent probability distributions and a non-deterministic reward function generation mechanism, it is more difficult for the agent to explore the joint optimal action space. For example, the predator-prey task with confounding factor is shown in Fig. 1. In a traditional predator-prey task, the predators are rewarded if any of them collide with the prey. However, this is unrealistic because the behavior of only one predator colliding with prey cannot be called capture behavior. On the other hand, this behavior will alert the prey and affect the behavior of multiple predators to capture the prey. In this paper, the predators collaboratively obtain a positive reward k when they collide with one prey at the same time. If only one or two predators collide with prey at the same time, the predator gets a different negative reward h and l, respectively. This non-monotonic reward makes it more difficult to explore the environment and complete the task.

The traditional RL algorithms based on the Markov decision process (MDP) have difficulty solving such tasks because they assume that the generation mechanism of the reward

Manuscript received April 11, 2023; revised June 23, 2023, December 23, 2023, January 23, 2024; accepted January 28, 2024. This work was supported in part by the National Natural Science Foundation of China (62136008, 62236002, 61921004, 62173251, 62103104), the "Zhishan" Scholars Programs of Southeast University, and the Fundamental Research Funds for the Central Universities (2242023K30034). Recommended by Associate Editor Nian Zhang. *(Corresponding author: Changyin Sun.)*

Citation: K. Jiang, W. Liu, Y. Wang, L. Dong, and C. Sun, "Discovering latent variables for the tasks with confounders in multi-agent reinforcement learning," *IEEE/CAA J. Autom. Sinica*, vol. 11, no. 7, pp. 1591–1604, Jul. 2024.

K. Jiang and Y. Wang are with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: kjiang@seu.edu.cn; wangyd@seu.edu.cn).

W. Liu is with the School of Artificial Intelligence, Anhui University, Hefei 230601, China (e-mail: wzliu@ahu.edu.cn).

L. Dong is with the School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: ldong90@seu.edu.cn).

C. Sun is with the School of Automation, Southeast University, Nanjing 210096, and also with the Engineering Research Center of Autonomous Unmanned System Technology, Ministry of Education, Hefei 230601, China (e-mail: cysun@seu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2024.124281

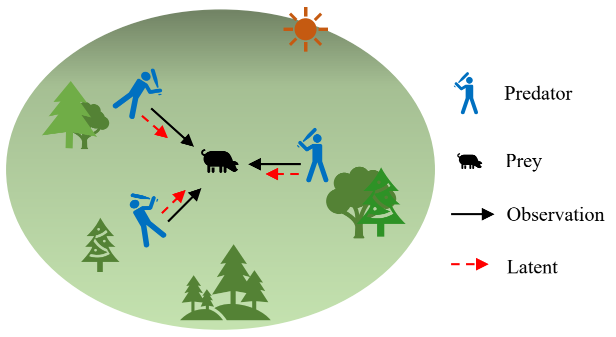

Fig. 1. Schematic diagram of the predator-prey environment, where these predators get a positive reward when they collide with the prey at the same time. The solid black arrows represent the influence of the received observations on the policy, and the dashed arrows represent the influence of invisible latent variables on the policy.

function is deterministic, so it is necessary to improve the MDP-based algorithm for the tasks with latent variables.

A direct improvement is one where the agent utilizes a latent variables discovery model to represent the space of the latent variables in the task. The predicted latent variables are used to expand each agent's observation range and generate a policy that can solve complex tasks with confounding factors. Therefore, the existing latent variable models based on variational inference [23] and causal inference [24] are used to learn statistical latent variable representations. Those methods try to learn the probability distribution of latent variables in a large number of offline experiences, and then infer the latent cause in a separate state, to improve the efficiency of the agent's exploration of the environment. However, most of the above methods focus on the single-agent domain, and they do not evaluate the specific impact of latent variables on the agent's policy. In addition, some methods are based on counterfactual inference to obtain internal rewards, which can improve the efficiency of agents in exploring complex environments, but it is also difficult to solve complex tasks with confounding factors [25], [26].

In a multi-agent system, the interaction between the agents and the environment is more complex, and the latent variables in the environment have different effects on each agent. Therefore, it is more difficult to infer the latent variable space of the task and measure the impact of latent variables on each agent in a multi-agent system. In this paper, the aim is to improve the efficiency of agents exploring complex tasks by discovering the latent variable space of the task and quantifying the impact of latent variables on each agent's policy. Specifically, we propose a new multi-agent reinforcement learning (MARL) algorithm for those tasks with continuous action and state spaces based on the soft actor-critic framework. Further, the amortized variational inference method is adopted to learn the latent variables discovery model for each agent, which is trained by maximizing the evidence lower bound (ELBO) with a large number of offline experiences [27]. The latent variable model assigns an inference network for each agent to produce a posterior distribution, which combines historical information and the current state to predict the latent variables at each time step. Then, we augment the input of the agent's policy with the predicted latent variables to obtain an actual adjusted policy that can represent any conditional distribution over actions. The evaluation of the value function is based not only on the observations and actions of all agents but also on the sampled latent variables.

To measure the influence of the predicted latent variables on each agent's policy, the Monte Carlo method is applied to derive the counterfactual policy, which is a probability distribution with no latent variables as input. Then, a distance function is used to measure the difference between the actual adjusted policy and the counterfactual policy. We make this difference as an intrinsic reward to motivate the agent to explore the environment. The intrinsic rewards provide a more coherent exploration of each agent's policy and measure latent variables' differential impact on each agent. The main contributions of this paper are summarized as follows.

1) To discover latent variables in complex tasks with confounders, a new algorithm called the multi-agent soft actor-critic with latent variables (MASAC-LV) using the CTDE framework is proposed. The inference network in the latent variable model fits a posterior distribution to construct the latent variable space for the task at hand. The predicted latent variables broaden the observation space of the agent's policy and improve the agent's exploration efficiency for complex tasks.

2) Considering that the interaction between agents and the environment is more complex in multi-agent systems, the Monte Carlo statistical method is applied to derive the counterfactual policies without latent variables as input. Then, the difference between the actual policy and the counterfactual policy is measured to represent the different effects of the latent variables on each agent.

The following contents are organized as follows. Section II mainly includes the related works on MARL and latent variables model. The research background and problem description are introduced in Section III. In Section IV, the details about the proposed algorithm are provided. The simulation results of the proposed algorithm and the baselines are compared in Section V. Finally, we summarize this paper in Section VI.

II. Related Work

Maximum entropy reinforcement learning has shown a strong exploratory ability in dealing with single-agent tasks, as it optimizes the policy to maximize the expected reward and the entropy of policy [28]. Haarnoja *et al.* [29] combines the maximum entropy with the actor-critic framework and off-policy method to propose the soft actor-critic (SAC) algorithm. They found that the SAC is a more stable and scalable algorithm, and it exceeded the final performance of the classic actor-critic algorithm and deep deterministic policy gradient (DDPG) in experiments [30]. The success of these algorithms has also sparked more interest in MARL. For example, independent Q-learning (IQL) directly extends deep Q-learning to multi-agent systems, where each agent learns an independent Q-function according to the information it receives [31]. Considering that the IQL may lead to non-stability and local optimal solutions, the CTDE framework is proposed to stabilize the training of each agent [32]. Multi-agent deep deterministic policy gradient (MADDPG) learns a centralized critic for each agent whose input includes the observations and actions of all agents, while the input of each agent's policy network is only its observation [33]. For instance, Hua *et al.* [34] originated MADDPG by introducing a hand-shaking strategy to ascertain different learning agents achieving energy management collaboratively. In addition, QMIX [35], COMA [36] and AWRMIX [37] algorithms are based on the CTDE framework to solve the credit assignment. The maximum entropy method still plays an important role in MARL, Ryu *et al.* [38] combine the multi-agent soft actor-critic scheme with a hierarchical graph attention network to facilitate the transfer of learned policies to new tasks.

For tasks with latent information unobservable by the agent, some existing model-free works in the single-agent field learn the latent variables discovery model to solve the latent-space partially observable Markov decision process (POMDP). Lee *et al.* [23] proposed a stochastic latent actor-critic algorithm to learn latent representation from lots of high-dimensional images, which is proven to be sampling efficient and accelerates the learning process. Haarnoja *et al.* [39] learned a hierarchical reinforcement learning framework for solving complex sparse-reward tasks, where each layer is augmented with latent random variables sampled from the trained prior distribution. There are also some model-based reinforcement learning methods to expand the observation space of the agent by learning the latent-space dynamics system models [40]–[43]. Besides, causal inference is introduced as a very useful method to solve the MDP with confounded observational data, where its main concept is to infer the representation of latent variables by building a causal structural model [44]–[47]. For example, Madumal *et al.* [48] learned the causal models to derive causal explanations of the behavior of model-free RL agents and encoded causal relationships between variables of interest. Sontakke *et al.* [24] proposed a hierarchical manner to infer the causal factor in the dynamic of the environment and introduce the causal curiosity as a novel intrinsic reward to motivate the exploration of the agent.

However, in a multi-agent system, the interaction between agents is more complex, and the latent variables in the task have different effects on each agent [49]–[51]. Therefore, it is more difficult to discover the latent variable space in MARL. For challenging social dilemma environments, Jaques *et al.* [25] reward those agents whose actions have causal effects on other agents to enhance coordination between all agents, where the causal influence is assessed using counterfactual reasoning. Zheng *et al.* [26] introduced a novel Episodic MARL algorithm with curiosity-driven exploration, where they make prediction errors of individual Q-values as intrinsic rewards for coordinated exploration. Those methods are reward-shaping methods by rewarding agents to motivate all agents to explore the environment and cooperate.

III. MARL and Statistical Variational Inference

A. Decentralized Partially Observable Markov Decision Process

A fully cooperative multi-agent task can be described as a decentralized partially observable Markov decision process (Dec-POMDP). Correspondingly, the Dec-POMDP can be formulated as a tuple $\langle \mathcal{I}, \mathcal{S}, \mathcal{O}, \mathcal{A}, R, P, \gamma \rangle$. $\mathcal{I} = \{1, \ldots, N\}$ is the set of agents. $\mathcal{S}$ is the true state space of the environment. $\mathcal{O}^i \in \mathcal{O} = \{\mathcal{O}^1, \ldots, \mathcal{O}^N\}$ is the observation space of agent i, and $\mathcal{A}^i \in \mathcal{A} = \{\mathcal{A}^1, \ldots, \mathcal{A}^N\}$ is the action space of agent i. The multi-agent team shares the same reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. $\gamma \in [0,1)$ is the discount factor. At each certain step $\boldsymbol{s}_t \in \mathcal{S}$, the agent i chooses the action a_t^i based on its local observation o_t^i. Then, the joint action $\boldsymbol{a}_t$ at time t is applied to the environment, which leads to a state transition and an immediate reward r_{t+1} according to the transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$. Solving the Dec-POMDP problem can be understood as the optimization of maximizing the discounted reward $\sum_t \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho}[\gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)]$, where $\boldsymbol{a}_t$ is the joint action of all agents and ρ denotes the state marginals $\rho(\boldsymbol{s}_t)$ of the trajectory distribution induced by the joint policy $\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t)$. We consider the framework of CTDE, therefore, the stochastic policy $\pi(a_t^i|o_t^i)$ of agent i can only condition on its independent history observation and action information. All agents learning the centralized action-value function $Q_{\boldsymbol{\pi}}(\boldsymbol{s}_t, a_t^1, \ldots, a_t^N)$ to evaluate the discounted reward $\mathbb{E}[G]$.

B. Maximum Entropy Multi-Agent Reinforcement Learning

Standard reinforcement learning aims to learn the policy $\pi(a_t^i|o_t^i)$ to maximize the expected cumulative reward. On this basis, maximum entropy reinforcement learning considers an additional maximum entropy objective, so that each agent's stochastic policy tends to maximize the expected entropy $\mathcal{H}(\pi(a_t^i|o_t^i)$. Thus, the multi-agent team attempts to find an optimal policy to maximize the following objective:

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \sum_{t=0}^{T} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho}[\gamma^t (r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha \mathcal{H}(\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t))] \quad (1)$$

where $\boldsymbol{\pi}^*$ is the optimal joint policy, T is the number of timesteps, and α is the temperature parameter representing the relative importance of the entropy term to the reward. $\mathcal{H}(\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t))$ is the entropy of the joint policy and can be expressed as $\mathcal{H}(\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t)) = -\log \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t)$. To obtain the optimal joint policy in (1), the soft actor-critic algorithm derives soft policy iteration to alternate between policy evaluation and policy improvement under the maximum entropy framework. Correspondingly, the Q-function is called the soft Q-function and is optimized to minimize the following soft Bellman residual:

$$\begin{aligned} J_Q(\theta^i) = \frac{1}{2}(Q_{\boldsymbol{\pi}}^i(\boldsymbol{s}_t, a_t^1, \ldots, a_t^N) - (r_t \\ + \gamma \mathop{\mathbb{E}}_{\boldsymbol{a}_{t+1} \sim \boldsymbol{\pi}}[\bar{Q}_{\boldsymbol{\pi}}^i(\boldsymbol{s}_{t+1}, a_{t+1}^1, \ldots, a_{t+1}^N) \\ - \alpha \log \pi(a_{t+1}^i|o_{t+1}^i)]))^2 \end{aligned} \quad (2)$$

where θ^i are the parameters of the soft Q-function of agent i, and $\bar{Q}_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}, a_{t+1}^1, \ldots, a_{t+1}^N)$ is the target soft Q-function of agent i. The policy parameter ϕ^i of agent i is updated according to the gradient of the soft Q-function, and the corresponding policy loss of agent i is expressed as

$$J_\pi(\phi^i) = \mathop{\mathbb{E}}_{\boldsymbol{a}_t\sim\boldsymbol{\pi}}[\alpha\log\pi(a_t^i|o_t^i) - Q_{\boldsymbol{\pi}}^i(\boldsymbol{s}_t, a_t^1, \ldots, a_t^N)]. \tag{3}$$

C. Statistical Variational Inference in POMDP

To learn augmented representation for the tasks with latent variables, we consider training a latent variable model to infer the latent variables in the task. Specifically, the statistical amortized variational inference method is employed to learn the probability distribution of latent variables. The trained latent variables model can decouple entangled state information $\boldsymbol{s}$ into task-relevant representations z and is optimized by maximizing the probability of observed state $\boldsymbol{s}$ under the equation $p(\boldsymbol{s}) = \int p(\boldsymbol{s}|z)p(z)dz$. In statistical variational inference, this objective is often transformed into the ELBO for the log-likelihood [27]

$$\begin{aligned}\log p(\boldsymbol{s}) &\geq E_{z\sim q}[\log p(\boldsymbol{s}|z) - D_{KL}(q(z|\boldsymbol{s})||p(z)]\\ &= ELBO(q)\end{aligned} \tag{4}$$

where $q(z|\boldsymbol{s})$ is the recognition model and also is called a probabilistic encoder, $p(\boldsymbol{s}|z)$ is the probabilistic decoder, therefore, $D_{KL}(q(z|\boldsymbol{s})||p(z)$ denotes the Kullback-Leibler (KL) divergence between the two distributions and $p(z)$ is the prior distribution of latent variables. Correspondingly, the loss function of the entire stochastic latent variable model can be expressed as

$$\mathcal{L}_{\text{latent}}(\psi) = E_{z\sim q}[\log p_\psi(\boldsymbol{s}|z) - D_{KL}(q_\psi(z|\boldsymbol{s})||p_\psi(z)] \tag{5}$$

where ψ are the parameters of all distributions. The encoder $q(z|\boldsymbol{s})$ and the decoder $p(\boldsymbol{s}|z)$ are directly optimized according to the stochastic gradient form loss function.

IV. Modeling and Learning Probabilistic Latent Variable Space

In a multi-agent system, the connections between agents and the interaction between agents and the environment are more complex, so it is more difficult to discover latent variables in a multi-agent system with confounding variables. To learn the knowledge that is beneficial to completing the current task under the statistical latent variable z, based on which we adjust the policy of each agent to $\pi(a_t^i|o_t^i, z_t^i)$ so that the actions of each agent adapt to the complex tasks. We learn the latent variable discovery model from a large amount of offline experience and then infer the latent variable value at the current time-step through the posterior distribution $q(z|\boldsymbol{s})$. In the following section, the specific structure of the latent variable discovery model will be described, and the corresponding MASAC-LV algorithm will be given.

A. Latent Variable Inference Model

To effectively learn the improved stochastic policy $\pi(a_t^i|o_t^i, z_t^i)$ in a complex MARL task with confounders, the learned latent variable model must tease out the salient information about the task into a disentangled latent representation. Under the influence of latent variables, the Markov decision chain of each agent can be represented by Fig. 2. It can be known that the latent variable z^i of each agent is contained in the global state s of the environment, and the undiscovered latent variable z^i will affect the policy of each agent and ultimately affect the reward function. Therefore, our purpose is to discover latent variable space in multi-agent tasks with confounders and adjust the stochastic policy of each agent.

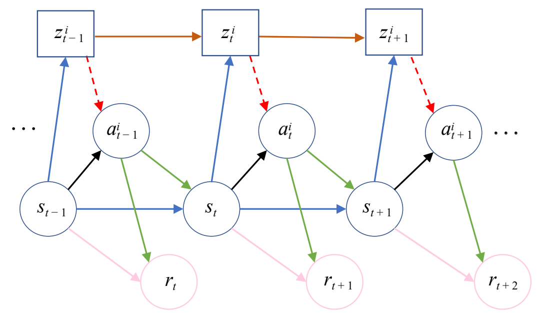

Fig. 2. The Markov chain of each agent, including the global state s of the environment, the action a^i of each agent, latent variable z^i and reward r in the environment, where the red dashed arrows represent the influence of the latent variable z^i on the policy of each agent.

In this paper, the amortized variational inference approach in Section IV is adopted to infer the latent variable z_t^i of each agent. To maximize ELBO in (4), it needs to build function approximators to evaluate the prior distribution $p(\boldsymbol{z})$, posterior distribution $q(\boldsymbol{z}|\boldsymbol{s})$, and the data generator $p(\boldsymbol{s}|\boldsymbol{z})$. This posterior distribution is considered an encoder, while the data generator is considered a decoder. We consider a task as a multi-agent MDP, which consists of a series of actions, states, a bounded reward function, a transition function, etc. For a model-free reinforcement learning problem, the transition and reward functions can be reconstructed from a sequence of unordered transitions. It follows that a collection of those transitions is sufficient to infer the space of the latent variables for the tasks with confounders. The latent variables contained in the state information of the environment are different for each agent, so it needs to learn a different latent variable model for each agent. At the same time, we augment the input of the prior and posterior distribution of each agent as $p(z_t^i|z_{t-1}^i, a_{t-1}^i)$ and $q(z_t^i|z_{t-1}^i, a_{t-1}^i, s_t)$ based on the time sequence information in a fully observed MDP. In a multi-agent system, the global state of the environment is difficult to obtain, but the observations of all agents almost contain the overall state information in the environment. Therefore, the global state s in the above probability distribution is replaced with the observations $\boldsymbol{o}_t$ of all agents, and the corresponding posterior distributions becomes $q(z_t^i|z_{t-1}^i, a_{t-1}^i, \boldsymbol{o}_t)$.

Those distributions are considered as the multivariate diagonal Gaussian, and we train the inference network $q_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i, \boldsymbol{o}_t)$ and the generative network $p_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i)$ parameterized by ψ to estimate the posterior and prior distributions, respectively. When designing the architecture of the inference network, we would like it to be expressive enough to capture the task-relevant information, without modeling irrelevant dependencies. The architecture of the designed inference network is shown in Fig. 3, and a parameter-sharing trick is used, where all agents share an inference network. The inference network $q_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i, \boldsymbol{o}_t)$ takes as input the action a_{t-1}^i and latent variable z_{t-1}^i of each agent, as well as the observations $\boldsymbol{o}_t$ of all agents. It outputs the multidimensional mean μ_t^i and variance σ_t^i, forming a multivariate diagonal Gaussian as in (6) to predict the distribution of the latent variable.

$$q_\psi^i(z_t^i|z_{t-1}^i, a_{t-1}^i, \boldsymbol{o}_t) = \mathcal{N}(\mu_t^i, \sigma_t^i). \quad (6)$$

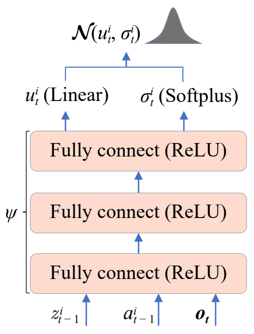

Fig. 3. The architecture of the inference network for each agent, which consists of three layers of fully connected neural networks. $\mathcal{N}(\mu_t^i, \sigma_t^i)$ is a multivariate diagonal Gaussian.

If more historical sequence information is considered, the fully connected layer in the inference network can also be replaced by a long short-term memory (LSTM) layer. As for the architecture of the generative model $p_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i)$, the difference between it and the inference model is that its input has no observations $\boldsymbol{o}_t$ of all agent. The KL divergence term in ELBO can also be understood as an information bottleneck constraining the relationship between the latent variable and other information. The first term of ELBO is considered as $p_\psi(\boldsymbol{o}_t|\boldsymbol{z}_t)$, where $\boldsymbol{z}_t$ is the latent variables for all agents. $p_\psi(\boldsymbol{o}_t|\boldsymbol{z}_t)$ is estimated by a multi-layer perception (MLP), in which the hidden layer uses the ReLU activation function, and the last layer uses the linear and Softplus activation functions to output the mean and variance respectively. According to the ELBO in (4), the parameters ψ of the latent variable model are optimized with the following objective:

$$\mathcal{L}_{\text{latent}}(\psi) = E_{z\sim q_\psi}[\log p_\psi(\boldsymbol{o}_t|\boldsymbol{z}_t) - D_{KL}(q_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i, \boldsymbol{o}_t))||p_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i)]. \quad (7)$$

The difference between the above equation and (4) is that when calculating the KL divergence term, the latent variable contained in the two distributions is the latent variable of each agent, not the latent variable of all agents. Because the decoder must decode the latent variables of all agents to output the distribution about $\boldsymbol{o}_t$, the latent variables of each agent can be inferred from the observations $\boldsymbol{o}_t$ of all agents.

Next, the ELBO will be derived in multi-agent reinforcement learning. To make the following expression more concise, we define $q_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i, \boldsymbol{o}_t) = q_\psi(z_t^i|c_t^i, \boldsymbol{o}_t)$ and $p_\psi(z_t^i|z_{t-1}^i, a_{t-1}^i) = p_\psi(z_t^i|c_t^i)$. For the convenience of subsequent derivation, we temporarily use $p(z_t^i|c_t^i, \boldsymbol{o}_t)$ to replace $q_\psi(z_t^i|c_t^i, \boldsymbol{o}_t)$. Firstly, for a general problem, $\boldsymbol{o}_t$ is the observed variable from the environment, c_t^i is other information of agent i, z_t^i is the latent variable encoded from $\boldsymbol{o}_t$ and c_t^i, the posterior distribution is computed by the following Bayesian model:

$$p(z_t^i|c_t^i, \boldsymbol{o}_t) = \frac{p(z_t^i)p(c_t^i, \boldsymbol{o}_t|z_t^i)}{p(c_t^i, \boldsymbol{o}_t)} \quad (8)$$

where $p(z_t^i)$ is the prior distribution of latent variable, $p(c_t^i, \boldsymbol{o}_t|z_t^i)$ is the likelihood function, and $p(c_t^i, \boldsymbol{o}_t) = \int p(z_t^i)\times p(c_t^i, \boldsymbol{o}_t|z_t^i)$, called evidence. Since $p(c_t^i, \boldsymbol{o}_t)$ includes non-integrable multiple integrals, this results in no analytical solution to the posterior distribution $p(z_t^i|c_t^i, \boldsymbol{o}_t)$. The variational inference method assumes that this posterior distribution $p(z_t^i|c_t^i, \boldsymbol{o}_t)$ is approximated by a variational distribution $q(z_t^i)$, thus constructing the following optimization problem:

$$q^*(z_t^i) = \arg\min L(q(z_t^i), p(z_t^i|c_t^i, \boldsymbol{o}_t)) \quad (9)$$

where L represents the distance function, which is usually considered as the KL divergence; this distribution $q^*(z_t^i)$ is easier to solve than the posterior distribution $p(z_t^i|c_t^i, \boldsymbol{o}_t)$.

Theorem 1: For the optimization problem in (9), the posterior distribution $q(z_t^i) \in \mathcal{Q}$, can directly optimize the ELBO of $q(z_t^i)$ to solve the optimal $q^*(z_t^i)$, which means

$$\begin{aligned} q^*(z_t^i) &= \arg\min_{q(z_t^i)\in\mathcal{Q}} D_{KL}(q(z_t^i)||p(z_t^i|c_t^i, \boldsymbol{o}_t)) \\ &= \arg\max_{q(z_t^i)\in\mathcal{Q}} ELBO(q) \\ &= \arg\max_{q(z_t^i)\in\mathcal{Q}} E_q[\log p(c_t^i, \boldsymbol{o}_t|z_t^i)] \\ &\quad - D_{KL}(q(z_t^i)||p(z_t^i|c_t^i, \boldsymbol{o}_t)). \end{aligned} \quad (10)$$

Proof: For (9), we expand the D_{KL} term as

$$\begin{aligned} D_{KL}(q(z_t^i)||p(z_t^i|c_t^i, \boldsymbol{o}_t)) &= -\int_{z_t^i} q(z_t^i)\log[\frac{p(z_t^i|c_t^i, \boldsymbol{o}_t)}{q(z_t^i)}]dz_t^i \\ &= \int_{z_t^i} q(z_t^i)\log q(z_t^i)dz_t^i - \int_{z_t^i} q(z_t^i)\log p(z_t^i|c_t^i, \boldsymbol{o}_t)dz_t^i. \end{aligned} \quad (11)$$

The integral of the above formula about $q(z_t^i)$ to z_t^i is the expectation about $q(z_t^i)$. We express the above formula with the following expectation form:

$$\begin{aligned} D_{KL}(q(z_t^i)||p(z_t^i|c_t^i, \boldsymbol{o}_t)) &= \mathbb{E}_q[\log q(z_t^i)] - \mathbb{E}_q[\log p(z_t^i|c_t^i, \boldsymbol{o}_t)] \\ &= \mathbb{E}_q[\log q(z_t^i)] - \mathbb{E}_q[\log[\frac{p(c_t^i, \boldsymbol{o}_t, z_t^i)}{p(c_t^i, \boldsymbol{o}_t)}]] \\ &= \mathbb{E}_q[\log q(z_t^i)] - \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t, z_t^i)] \\ &\quad + \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t)]. \end{aligned} \quad (12)$$

Since $\log p(c_t^i, \boldsymbol{o}_t)$ is independent of the expected object $q(z_t^i)$, the expected symbol can be removed directly, so we get

$$\begin{aligned} D_{KL}(q(z_t^i)||p(z_t^i|c_t^i, \boldsymbol{o}_t)) &= \mathbb{E}_q[\log q(z_t^i)] \\ &\quad - \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t, z_t^i)] + \log p(c_t^i, \boldsymbol{o}_t). \end{aligned} \quad (13)$$

The first two terms of the above formula are called ELBO. In the actual calculation, ELBO can be expressed in the following form for calculation:

$$
\begin{aligned}
ELBO(q) &= \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t, z_t^i)] - \mathbb{E}_q[\log q(z_t^i)] \\
&= \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t | z_t^i) p(z_t^i)] - \mathbb{E}_q[\log q(z_t^i)] \\
&= \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t | z_t^i)] + \mathbb{E}_q[\log p(z_t^i)] \\
&\quad - \mathbb{E}_q[\log q(z_t^i)] \\
&= \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t | z_t^i)] + \mathbb{E}_q\Big[\frac{\log p(z_t^i)}{\log q(z_t^i)}\Big] \\
&= \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t | z_t^i)] + \int_{z_t^i} q(z_t^i) \frac{\log p(z_t^i)}{\log q(z_t^i)} dz_t^i \\
&= \mathbb{E}_q[\log p(c_t^i, \boldsymbol{o}_t | z_t^i)] - D_{KL}(q(z_t^i) \| p(z_t^i)). \qquad (14)
\end{aligned}
$$

Through the derivation of the above formula, ELBO is consistent with the description in (4). Moreover, $\log p(c_t^i, \boldsymbol{o}_t)$ in (13) includes the statistical information about the observed variables, so it is a constant. Our initial goal was to minimize $D_{KL}(q(z_t^i) \| p(z_t^i | c_t^i, \boldsymbol{o}_t))$, which can now be transformed to maximize the ELBO, which results in

$$
\begin{aligned}
q^*(z_t^i) &= \arg \min_{q(z_t^i) \in Q} D_{KL}(q(z_t^i) \| p(z_t^i | c_t^i, \boldsymbol{o}_t)) \\
&= \arg \max_{q(z_t^i) \in Q} ELBO(q). \qquad (15)
\end{aligned}
$$

■

B. Intrinsic Reward Motivate Agent Exploration

Considering the fact that latent variables have different effects on each agent's policy, we derive the following intrinsic reward r_{int}^i to motivate the agents strongly influenced by latent variables to explore. Firstly, our actual policy is $\pi(a_t^i | o_t^i, z_t^i)$, where z_t^i is sampled from the posterior distribution $q_\psi(z_t^i | c^i, \boldsymbol{o}_t)$. We want to determine the counterfactual policy $\pi(a_t^i | o_t^i)$ whose input has no latent variables z_t^i, and then compare the gap between the actual policy $\pi(a_t^i | o_t^i, z_t^i)$ and the counterfactual $\pi(a_t^i | o_t^i)$, to determine how much the latent variable affects each agent. Therefore, it needs to infer from the statistical actual policy $\pi(a_t^i | o_t^i, z_t^i)$ to $\pi(a_t^i | o_t^i)$. The latent variable z_t^i can be understood as a cause of $\pi(a_t^i | o_t^i)$, and deduce the following total probability formula about the latent variable z_t^i:

$$
\pi(a_t^i | o_t^i) = \int_z \pi(a_t^i | o_t^i, z_t^i) q_\psi(z_t^i | c^i, \boldsymbol{o}_t) dz_t^i. \qquad (16)
$$

This generating process is intractable to compute due to the marginalization of the latent variables z_t^i. It instead uses the Monte Carlo method to estimate the above integral. Firstly, we sample n independent and identically distributed random latent values z_{sam}^j from the distribution $q_\psi(z_t^i | c^i, \boldsymbol{o}_t)$, where $j = 1, \dots, n$. Further, these random latent variables z_{sam}^j are used to calculate the n independent action values $\pi(a_t^i | o_t^i, z_{\text{sam}}^j)$.

Lemma 1: Given the posterior distribution $q_\psi(z_t^i | c^i, \boldsymbol{o}_t)$ and the actual policy $\pi(a_t^i | o_t^i, z_t^i)$, sample n independent and identically distributed random latent value z_{sam}^j from $q_\psi(z_t^i | c^i, \boldsymbol{o}_t)$, and the expectation $\mathbb{E}[\pi(a_t^i | o_t^i, z_t^i)]$ of these random action values exist, then the following equation holds [52]:

$$
\lim_{n \to \infty} \frac{1}{n} \sum_{j=1}^{n} \pi(a_t^i | o_t^i, z_{\text{sam}}^j) = \mathbb{E}[\pi(a_t^i | o_t^i, z_t^i)]. \qquad (17)
$$

By apply the Lemma 1, we can approximate (16) by the n calculated action values $\pi(a_t^i | o_t^i, z_t^j)$

$$
\frac{1}{n} \sum_{j=1}^{n} \pi(a_t^i | o_t^i, z_{\text{sam}}^j) \approx \int_z \pi(a_t^i | o_t^i, z_t^i) q_\psi(z_t^i | c^i, \boldsymbol{o}_t) dz_t^i. \qquad (18)
$$

Therefore, the action value output by the counterfactual policy $\pi(a_t^i | o_t^i)$ can also be approximated by the following process:

$$
\begin{aligned}
\pi(a_t^i | o_t^i) &= \int_z \pi(a_t^i | o_t^i, z_t^i) q_\psi(z_t^i | c^i, \boldsymbol{o}_t) dz_t^i \\
&\approx \frac{1}{n} \sum_{j=1}^{n} \pi(a_t^i | o_t^i, z_{\text{sam}}^j). \qquad (19)
\end{aligned}
$$

Next, the gap between the actual action value and the counterfactual action value needs to be measured, and it can be defined as $L(\pi(a_t^i | o_t^i, z_t^i), \pi(a_t^i | o_t^i))$. Considering our statistical policy, the KL divergence is used to compute $L(\pi(a_t^i | o_t^i, z_t^i), \pi(a_t^i | o_t^i))$

$$
L(\pi(a_t^i | o_t^i, z_t^i), \pi(a_t^i | o_t^i)) = D_{KL}(\pi(a_t^i | o_t^i, z_t^i) \| \pi(a_t^i | o_t^i)). \qquad (20)
$$

Further, this KL divergence is taken as an intrinsic reward for each agent

$$
\begin{aligned}
r_{\text{int}}^i &= L(\pi(a_t^i | o_t^i, z_t^i), \pi(a_t^i | o_t^i)) \\
&= D_{KL}(\pi(a_t^i | o_t^i, z_t^i) \| \pi(a_t^i | o_t^i)). \qquad (21)
\end{aligned}
$$

In this paper, it just needs to take n to be a large integer, so that the estimation of the intrinsic reward is more accurate by sampling a large number of latent variable samples. After some tests, we take the sampling size n as 200. However, what needs to be emphasized is that the latent variable sampling size n will only affect the trade-off between the amount of computation and accuracy during the training process, and it will not affect the final convergence of our latent variable discovery algorithm. Besides, to avoid deriving an excessively large intrinsic reward that would harm the learning process of each agent, we truncate it after deriving the intrinsic reward. That is, if the derived r_{int}^i value is greater than $r_{\max}$, then r_{int}^i takes $r_{\max}$. This $r_{\max}$ has different values according to the environmental rewards in different tasks. There is no need to train a neural network to fit the counterfactual policy, thus reducing the amount of computation.

C. Statistical Latent Actor and Critic

By modeling the latent variable model above, it can take full advantage of posterior sampling to effectively explore the tasks with confounders. Our MASAC-LV method directly infers the posterior distribution over the latent variable z_t^i via the ELBO, which reconstructs the MDP and optimizes the stochastic policy and value functions of each agent. The posterior distribution provides an extended exploration by the adjusted policy $\pi(a_t^i | o_t^i, z_t^i)$ conditioned on historical experience. It means that each agent can act to test hypotheses based on the sampled latent variable z_t^i, even if the action is not immediately informative of the task. Based on the multi-agent

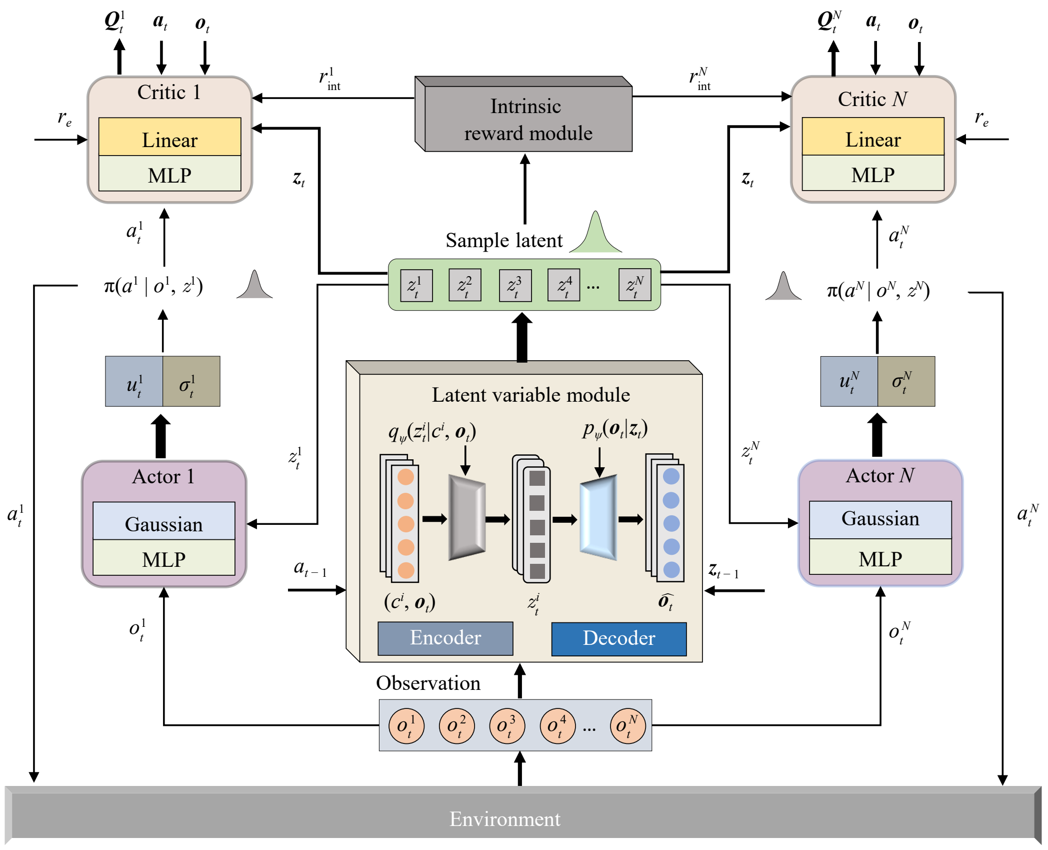

Fig. 4. The framework of our MASAC-LV algorithm, where r^i_{int} is the intrinsic reward in Section IV-B, and the latent variable model is shown in Section IV-A. The inference network outputs the posterior distribution $q_\psi(z^i_t|c^i_t, \boldsymbol{o}_t)$ from which the latent variables z^i_t of each agent are sampled. The intrinsic reward module derives the intrinsic rewards for each agent.

actor-critic framework, the whole proposed architecture is shown in Fig. 4.

The MASAC-LV algorithm consists of the following three components: 1) Critic networks; 2) Actor networks; and 3) Latent variable model. The critic network is used to evaluate the centralized Q-value function, and its inputs include observations $\boldsymbol{o}$, actions $\boldsymbol{a}$, and latent $\boldsymbol{z}$ of all agents. The critic network is also approximated by an MLP with parameters θ^i_c, the critic network's hidden layer employs the rectified linear unit (ReLU) activation function, while the final layer utilizes the linear activation function to output the Q-value. When updating the parameters of critic networks, the environment reward r_e and intrinsic reward r^i_{int} are used to compute the target centralized Q-value function. Therefore, the loss function of the critic network is changed to the following equation:

$$J_Q(\theta^i_c) = \frac{1}{2}\Big(Q^i_{\boldsymbol{\pi}}(\boldsymbol{o}_t, z_t, a^1_t, \ldots, a^N_t) - (\lambda_1 r_e + \lambda_2 r^i_{\text{int}} + \gamma \mathop{\mathbb{E}}_{\boldsymbol{a}_{t+1}\sim\boldsymbol{\pi}}[\bar{Q}^i_{\boldsymbol{\pi}}(\boldsymbol{o}_{t+1}, \boldsymbol{z}_{t+1}, a^1_{t+1}, \ldots, a^N_{t+1}) - \alpha \log \pi(a^i_{t+1}|o^i_{t+1}, z^i_{t+1})])\Big)^2 \quad (22)$$

where $\lambda_1, \lambda_2 \in (0, 1)$ are used to control the proportion of environment reward r_e and intrinsic reward r^i_{int}, respectively. In the following experimental results part, we set λ_1 to 1 and then selected different λ_2 values in different environments through advanced testing. In addition, we also perform ablation experiments on the λ_2 value to show the impact of different λ_2 values on the performance of the algorithm. The actor network of agent i outputs the action based on the received observations o^i_t and the inferred latent variable z^i_t. To meet the requirements of z^i_t from random sampling, the reparameterization trick is applied to sample z^i_t from the posterior distribution $q_\psi(z^i_t|c^i, \boldsymbol{o}_t)$; these networks can still be trained through backpropagation. The actor network is defined as $a_t = \pi_{\phi^i_a}(o^i_t, z^i_t)$, where $\pi_{\phi^i_a}$ is approximated by a MLP with parameters ϕ^i_a. The hidden layer within the actor network employs the ReLU activation function, while the output layer utilizes both the linear activation function and the Softplus activation function to respectively generate the policy's mean and variance. The adjusted policy parameters are still optimized according to the gradient from the soft Q-function, and the loss function of the actor network is expressed as

$$J_\pi(\phi^i_a) = \mathop{\mathbb{E}}_{a_t\sim\boldsymbol{\pi}}[\alpha \log \pi(a^i_t|o^i_t, z^i_t) - Q^i_{\boldsymbol{\pi}}(\boldsymbol{o}_t, z_t, a^1_t, \ldots, a^N_t)]. \quad (23)$$

The purpose of the latent variable model is to obtain the inference network through training and learning, and then output the posterior distribution $q_\psi(z^i_t|c^i, \boldsymbol{o}_t)$. The latent variable z^i_t sampled in the posterior distribution adds more exploration to the policy network. We use the collected experience to update the posterior distribution and continue to make inferences about the latent variables z^i_t of the current state based on the historical sequence information of the episodes. We update the actor and critic networks with off-policy data N^{ac}_{batch} sampled uniformly from the entire replay buffer $\mathcal{D}$. It should be noted that the off-policy data used to train the latent variable model is different from the off-policy data used to update the actors and critics. In addition, unlike the global state s in the Markov chain in Fig. 2, we use the joint observation information $\boldsymbol{o}$ of all agents in the algorithm framework of Fig. 4. This is because it is difficult for the multi-agent system to obtain the global state information in the environment in a real task. The multi-agent system can only form a joint observation space based on the information observed by each agent for the inference of the latent variable space. Specifically, to improve the accuracy and timeliness of the inferred latent variables z^i_t in the current state, we uniformly sample mini-batch N^l_{batch} from the most recent data collected in the last 1000 steps. The pseudocode of our MASAC-LV algorithm is shown in Algorithm 1.

Algorithm 1 MASAC-LV

```
1: **Input:** Agent number $N_a$, max trajectory length $T$, episode number $M$, mini-batch size $N^{ac}_{\text{batch}}$ of actor and critic, mini-batch size $N^l_{\text{batch}}$ of latent variable model.
2: **Initialize:** Replay buffer $\mathcal{D}$, the parameters of latent variable model's networks, actor networks, critic networks, and target networks.
3: for $episode = 1$ to $M$ do
4:     Initialize a random process $\mathcal{N}$ for action exploration.
5:     Receive initial observation $\boldsymbol{o}$ for all agents.
6:     for $t = 1$ to $T$ do
7:         For each agent, sample latent $z^i_t \sim q_\psi(z^i_t|c^i, \boldsymbol{o}_t)$
8:         Select action $a^i_t = \boldsymbol{\pi}_{\phi^i_a}(o^i_t, z^i_t)$.
9:         Execute actions $\boldsymbol{a_t} = (a^i_t, \dots, a^N_t)$, then get the reward $r_e$ in environment, and observations $\boldsymbol{o}_{t+1}$ of all agents.
10:        Compute intrinsic reward $\boldsymbol{r}_{\text{int}} = (r^1_{\text{int}}, \dots, r^N_{\text{int}})$.
11:        Store $\boldsymbol{o}_t, \boldsymbol{z}_t, \boldsymbol{a}_t, r_e, \boldsymbol{r}_{\text{int}}, \boldsymbol{o}_{t+1}\}$ into $\mathcal{D}$.
12:        Sample a random mini-batch of $N^{ac}_{\text{batch}}$ and $N^l_{\text{batch}}$ samples.
13:        for agent $i = 1$ to $N_a$ do
14:            Update critic by minimizing (22) with the sampled data $N^{ac}_{\text{batch}}$.
15:            Update actor according to the gradient (23) with the sampled data $N^{ac}_{\text{batch}}$.
16:        end for
17:        Update the networks of the latent variable module by minimizing (7) with the sampled data $N^l_{\text{batch}}$.
18:        Update target networks.
19:    end for
20: end for
```

V. Experiment Results

In this section, the MASAC-LV algorithm is evaluated on two partially observable tasks with confounders, which include simplified particle predator-prey and box-pushing environments. The specific details of the simulation environment will be covered in subsequent subsections. We extend the SAC algorithm to the multi-agent field and make it a baseline, called the multi-agent SAC (MASAC) algorithm. Besides, MADDPG, multi-agent proximal policy optimization (MAPPO), and independent SAC (ISAC) algorithms are also used as baselines. We also use the social causal reward method in [46] as a baseline, named the basic social influence (BSI) algorithm. We run 8 trials for each algorithm, where the hyperparameters and basic settings of all algorithms remain the same. Our simulations run on the Ubuntu 20.04 operating system with an Intel Core i7-7700k CPU@4.20 GHz.

A. Predator-Prey With Confounders

Firstly, the predator-prey task in Fig. 1 is transformed into a structured vector space, and the reward function of the predator-prey task is designed as

$$r_{\text{base}} = \begin{cases} h, & n_{\text{count}} = 1 \\ l, & n_{\text{count}} = 2 \\ k, & n_{\text{count}} = 3 \end{cases} \tag{24}$$

where n_{count} is the number of predators colliding with one prey at the same time. The reward function is expressed as a vector $\boldsymbol{w} = [h, l, k]^T$. Further, we design the basis vectors according to the reward function as the following formula:

$$\boldsymbol{r}_p = \begin{cases} [1,0,0]^T, & vn_{\text{count}} = 1 \\ [0,1,0]^T, & n_{\text{count}} = 2 \\ [0,0,1]^T, & n_{\text{count}} = 3. \end{cases} \tag{25}$$

Therefore, the structured space of the reward function for the predator-prey task is $r = \boldsymbol{w}^T \boldsymbol{r}_p$, and w controls the specific form of the reward function. Considering the above predator-prey task with the structured reward function, only when three predators collide with one prey at the same time, can the predators receive a positive reward, otherwise, the reward is negative. This positive and negative reward tends to confuse the predators, who do not know what is the correct behavior to engage in capturing prey. This phenomenon can be understood as the predators being affected by confounding factors in the environment. Besides that, considering the uncertainty in the real environment, for more difficult predator-prey tasks with confounders, h, l, and k are not even constant values but obey some probability distribution $p(h)$, $p(l)$, and $p(k)$. This means that the generation mechanism of the reward function is uncertain, and these uncertain probability distributions further increase the confounding factors in the task.

We redesigned two simplified particle predator-prey tasks with different difficulties. As shown in Fig. 5(a), both predator and prey can observe the position and velocity information of all predator and prey, and the prey moves faster than the predator. The reward function considers the number of predators colliding with prey at the same time in (24), where all predators receive an additional reward $-d$ based on the sum of the distance between each prey to its nearest predator. The prey is considered to be captured only when three predators collide with prey at the same time. All the prey are trained by the ISAC policy to escape the predators, and all the predators are trained by the MASAC-LV policy and other baselines. The relevant training hyperparameters are shown in Table I. For each algorithm, we run 8 experiments with the same hyperparameters but different initialized network parameters. After training our MASAC-LV algorithm and other baseline

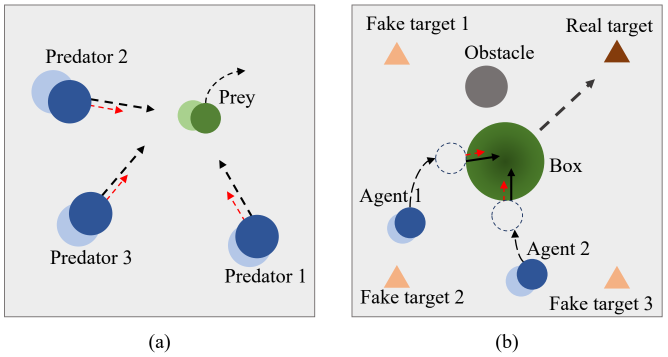

Fig. 5. Two tasks with confounders: (a) Predator-prey; (b) Box-pushing. As shown in Fig. 1, the black solid arrows and the red dashed arrows represent the influence of the observed variables and the latent variables of the task on the policy of each agent, respectively.

TABLE I
PARAMETER SETTINGS FOR PREDATOR-PREY ENVIRONMENT

Parameter	Value
Episode number (M)	15 000
Max trajectory length	120
Discount factor (γ)	0.95
Learning rate (actor)	0.001
Learning rate (critic)	0.001
Learning rate (latent)	0.005
Soft update factor (τ)	0.001
Batch size (actor, critic, N^{ac}_{batch})	1 024
Batch size (latent, N^{l}_{batch})	512
Replay buffer size	300 000
Hidden layer units (actor)	(64, 64)
Hidden layer units (critic)	(180, 180)
Hidden layer units (the posterior distribution, q_ψ)	(150, 150)

algorithms, we choose the model of each algorithm that performs best to validate the performance of our proposed algorithm. Specifically, we test the average number of times N_{capt} that the prey is captured and the average number of times N_{coll} that only one predator collides with the prey over 1000 episodes. Prey is captured only when three predators collide with the prey at the same time. For the derived intrinsic reward r^i_{int} in the method part, $r_{\max}$ is set to 1 to limit its maximum value.

1) Scenario 1

In this scenario, we set a fixed non-monotonic reward function, which means that the h, l, and k in (24) remains constant, and the designed reward function is as follows:

$$r_{\text{base}} = \begin{cases} -1.0, & n_{\text{count}} = 1 \\ -0.2, & n_{\text{count}} = 2 \\ +7.0, & n_{\text{count}} = 3. \end{cases} \tag{26}$$

The corresponding experimental results of all algorithms are shown in Fig. 6. From Fig. 6(a), it can be noticed that the N_{capt} value of our proposed MASAC-LV algorithm is the largest, indicating that predators trained by our MASAC-LV algorithm are the best at capturing prey. In addition, the statistical policies MASAC-LV, MASAC, and BSI are better than the deterministic policy MADDPG, which means the statistical policies have advantages when dealing with tasks with confounders. The predators trained with the ISAC algorithm have difficulty cooperating to capture prey. Fig. 7(b) shows that our MASAC-LV algorithm has the largest average N_{coll} value within one episode. This is because the MASAC-LV algorithm results in each predator trying to capture the prey to obtain the maximum reward; Thus, the predator will collide with the prey more frequently. Furthermore, the number of times, N_{capt}, that the predators trained by the ISAC algorithm capture the prey is close to 0, but the number of times, N_{coll}, where only one predator collides with the prey is close to 10. This can be explained by the fact that each predator still occasionally collides with the prey due to the existence of the based reward $-d$.

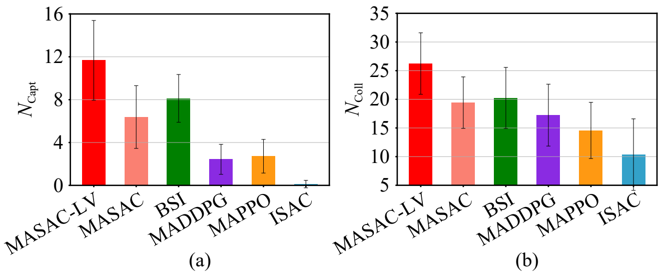

Fig. 6. The experimental results of Scenario 1 in the predator-prey environment: (a) The average number of times N_{capt} that prey is captured per episode; and (b) The average number of times N_{coll} that only one predator collides with the prey.

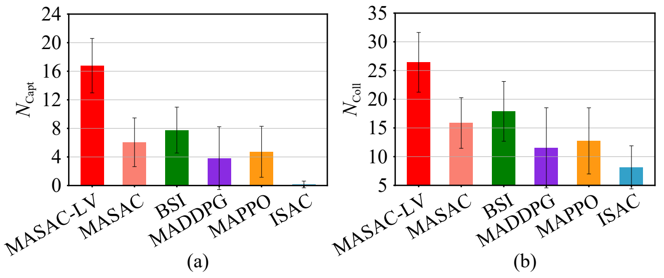

Fig. 7. The experimental results of Scenario 2 in the predator-prey environment: (a) The average number of times N_{capt} that prey is captured per episode; and (b) The average number of times N_{coll} that only one predator collides with the prey.

2) Scenario 2

Compared with the fixed non-monotonic reward value in Scenario 1, we further adjusted the difficulty of the task by taking into account the uncertainties in reality and then changed h, l, and k in (24) to randomly sampled values. Specifically, we set h, l, and k to randomly sample from vectors $[h_1,h_2,h_3]$, $[l_1,l_2,l_3]$ and $[k_1,k_2,k_3]$, respectively, and the corresponding reward function is designed as

$$r_{\text{base}} = \begin{cases} \text{Sample from } [-0.9,-1.1,-1.3], & n_{\text{count}} = 1 \\ \text{Sample from } [-0.2,-0.3,-0.4], & n_{\text{count}} = 2 \\ \text{Sample from } [+6.0,+7.0,+8.0], & n_{\text{count}} = 3. \end{cases} \tag{27}$$

TABLE II
THE RELATIVE PERCENTAGES P_r OF EACH BASELINE ALGORITHM WITH MASAC-LV ALGORITHM IN THE PREDATOR-PREY ENVIRONMENT, AMONG THEM, ITEM MASAC-LV INDICATES THE PERCENTAGE $P_r = N_{\text{capt}}^{\text{LV}}/N_{\text{capt}}^{\text{LV}}$ VALUE OF THE MASAC ALGORITHM COMPARED WITH ITSELF

Scenario	MASAC-LV	MASAC	BSI	MADDPG	MAPPO	ISAC
Scenario 1	100.00%	54.67%	67.58%	20.82%	23.31%	1.03%
Scenario 2	100.00%	36.05%	41.26%	22.82%	26.13%	0.89%

In this scenario with more confounders, the performance of our MASAC-LV algorithm and all baselines are shown in Fig. 7.

Besides, to more intuitively reflect the performance gap between our algorithm and the baseline algorithms in Scenarios 1 and 2, we also compare the relative percentages P_r of the baseline algorithms with our MASAC-LV algorithm on the N_{capt} value in the two scenarios. P_r can be calculated by $P_r = N_{\text{capt}}^{\text{base}}/N_{\text{capt}}^{\text{LV}}$, where $N_{\text{capt}}^{\text{LV}}$ and $N_{\text{capt}}^{\text{base}}$ are the average numbers of times that the predators trained by our algorithm and the predators trained by the baseline algorithms capture the prey in each episode, respectively. The relative percentages P_r of each baseline algorithm with the MASAC-LV algorithm are shown in Table II. Obviously, the MASAC-LV algorithm has a $P_r = N_{\text{capt}}^{\text{LV}}/N_{\text{capt}}^{\text{LV}}$ value of 100.00% compared to itself.

Comparing the experimental data in Figs. 6 and 7, we find that the N_{capt} and N_{coll} value of MASAC-LV algorithm is the largest. From Table II, it can be seen that the performance of the MADDPG and MAPPO algorithms does not change much in the two scenarios, and they are already quite different from our algorithm in performance. ISAC algorithm is almost completely useless in this predator-prey environment with confounders. The relative percentages P_r value of the MASAC algorithm drops from 54.67% to 36.05% in Table II, and the value of the BSI Algorithm drops from 67.67% to 41.26%. This worsening performance means that the MASAC and BSI algorithms have difficulty coping with the tasks with more confounders due to random sampling of non-monotonic reward values, and the social causal reward in the BSI algorithm cannot solve complex problems with latent variable distribution. Our MASAC-LV algorithm still outperforms in these two scenarios with different difficulty confounders.

From the data in Table II, it can be seen that the performance gap between our MASAC-LV algorithm and other algorithms in Scenario 2 is notably greater than that in Scenario 1. That's because the random non-monotonic rewards in Scenario 2 greatly increase the confounding factors in the predator-prey task, making the gap between the performance of other baseline algorithms and our algorithm larger.

Therefore, it can be concluded that the proposed MASAC-LV algorithm outperforms the MASAC, BSI, MAPPO, MADDPG, and ISAC algorithms under the same task setting and algorithm parameter setting. Our algorithm has a notable performance improvement based on the MASAC algorithm for those tasks with latent variables.

B. Cooperative Box-Pushing

In a cooperative box-pushing environment, two agents cooperate to push the box to the target position. As shown in Fig. 5(b), the box can only move if two agents exert force on it at the same time, we also design different target positions, including one real target position and three fake target positions. The two agents can only get the maximum reward by pushing the box to the real target point and get different penalties for pushing the box to the fake target point. This is equivalent to increasing the confounding variables and also makes the task more challenging. In addition, the agent will also be penalized by -1 when it collides with an obstacle. The continuous observation space of the agent includes information about the positions of all entities and the velocity of the agent and the box. To guide the agent to move towards the box, the agents can get a +1 reward by touching the box. The real target point is extracted from the original four target points according to a certain probability distribution $\boldsymbol{p}_{\text{tar}} = [p_1, p_2, p_3, p_4]$, where the sum of p_1, p_2, p_3, and p_4 is 1, corresponding to the four original target points respectively. This method of setting the target points according to a certain probability distribution $\boldsymbol{p}_{\text{tar}}$ increases the non-stationarity of the environment. Only by discovering the latent variables behind the unstable task can the agent improve the probability of pushing the box to the target point. Like the vectorized rewards in Section III, we also design the base reward for pushing the box to the target point as follows:

$$r_{\text{base}} = \begin{cases} h, & \text{to fake target 1} \\ l, & \text{to fake target 2} \\ k, & \text{to fake target 3} \\ g, & \text{to real target} \end{cases} \tag{28}$$

where g is the maximum reward +20, and h, l and k are all randomly sampled from the vector $[-1.5, -2, -2.5]$. The base reward is expressed as a vector as $\boldsymbol{w} = [h, l, k, g]^T$, and the corresponding basis vectors can be written as

$$\boldsymbol{r}_p = \begin{cases} [1,0,0,0]^T, & \text{to fake target 1} \\ [0,1,0,0]^T, & \text{to fake target 2} \\ [0,0,1,0]^T, & \text{to fake target 3} \\ [0,0,0,1]^T, & \text{to real target.} \end{cases} \tag{29}$$

Then, the structured reward space of the box-pushing task is $r = \boldsymbol{w}^T \boldsymbol{r}_p$. The above reasonable processing greatly increases the confounding factor in the box-pushing task. The agent can only push the box to the real target point to get the maximum reward by continuously exploring and discovering the latent variables of the task. Since the maximum reward value that can be obtained in the box-pushing environment is relatively large, for the derived intrinsic reward r_{int}^i, we set $r_{\max}$ to be 2 to limit its maximum value. We train the MASAC-LV and other baseline algorithms in this task, and the relevant hyper-parameters are shown in Table III.

TABLE III
PARAMETER SETTINGS FOR BOX-PUSHING ENVIRONMENT

Parameter	Value
Episode number (M)	10 000
Max trajectory length	100
Discount factor (γ)	0.95
Learning rate (actor)	0.001
Learning rate (critic)	0.001
Learning rate (latent)	0.005
Soft update factor (τ)	0.001
Batch size (actor, critic, $N_{\text{batch}}^{\text{ac}}$)	1024
Batch size (latent, N_{batch}^{l})	512
Replay buffer size	200 000
Hidden layer units (actor)	(90, 90)
Hidden layer units (critic)	(180, 180)
Hidden layer units (the posterior distribution, q_ψ)	(160, 160)

Four different $\boldsymbol{p}_t$ are designed to verify the performance of our MASAC-LV algorithm, including $\boldsymbol{p}_{\text{tar}}^1 = [1.0, 0, 0, 0]$, $\boldsymbol{p}_{\text{tar}}^2 = [0.1, 0.7, 0.1, 0.1]$, $\boldsymbol{p}_{\text{tar}}^3 = [0.05, 0.3, 0.6, 0.05]$ and $\boldsymbol{p}_{\text{tar}}^4 = [0.2, 0.2, 0.2, 0.4]$. The maximum probability values of the four groups of $\boldsymbol{p}_t$ are from large to small and correspond to different original target points respectively. For example, when $\boldsymbol{p}_{\text{tar}}^1 = [1.0, 0, 0, 0]$, this indicates that the real target point must be at the original target point 1. When $\boldsymbol{p}_{\text{tar}}^2 = [0.1, 0.7, 0.1, 0.1]$, this indicates that the real target point has a 70% probability of being at the original target point 2. After training all algorithms, it shows the reward curves of all algorithms in Fig. 8.

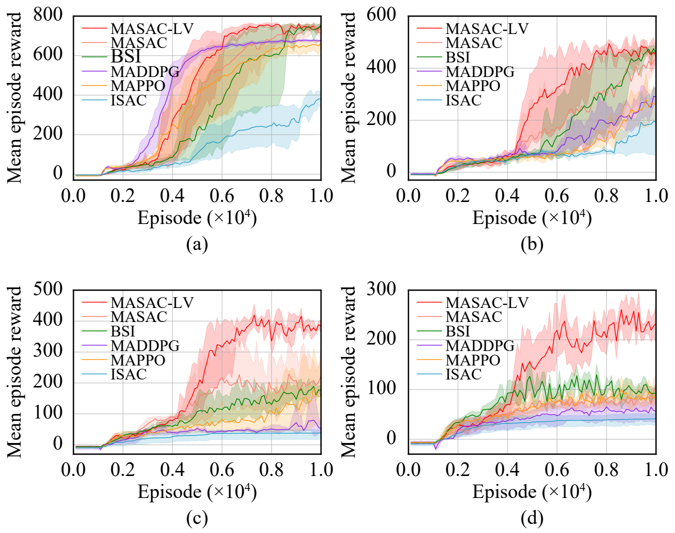

Fig. 8. Learning curves of MASAC-LV and other baselines for the box-pushing scenario: (a) $\boldsymbol{p}_{\text{tar}}^1 = [1.0, 0, 0, 0]$; (b) $\boldsymbol{p}_{\text{tar}}^2 = [0.1, 0.7, 0.1, 0.1]$; (c) $\boldsymbol{p}_{\text{tar}}^3 = [0.05, 0.3, 0.6, 0.05]$; and (d) $\boldsymbol{p}_{\text{tar}}^4 = [0.2, 0.2, 0.2, 0.4]$.

When $\boldsymbol{p}_{\text{tar}}^1 = [1.0, 0, 0, 0]$, we compare the MASAC-LV algorithm with other baselines in Fig. 8(a). It is found that the performance of our algorithm is only slightly better than the MASAC, BSI, MADDPG, and MAPPO algorithms. This is because the tasks in this set of experiments are deterministic and there are no confounders, resulting in little difference between the four traditional MDP-based algorithms. Moreover, the agent trained by the ISAC algorithm can still get certain rewards. When $\boldsymbol{p}_{\text{tar}}^2$ is set to $[0.1, 0.7, 0.1, 0.1]$, it can be found that the advantages of our algorithm begin to manifest in Fig. 8(b), and the performance of other baselines is notably lower than that of our algorithm. It can also be seen that our algorithm learns faster and can converge to the maximum value at about the eighth thousandth episode. Further, we continue to increase the uncertainty of the latent variables of the boxing-pushing task by setting the $\boldsymbol{p}_{\text{tar}}^3$ and $\boldsymbol{p}_{\text{tar}}^4$. Therefore, this task is full of confounding factors, which will lead to confusion in the process of the agent pushing the box to the real target point. As shown in Figs. 8(c) and 8(d), it can be seen that the performance of other baselines is greatly degraded, but our algorithm is greatly better than other traditional reinforcement learning algorithms in both learning speed and final performance, especially when $\boldsymbol{p}_{\text{tar}}^4 = [0.2, 0.2, 0.2, 0.4]$.

It has been verified that our algorithm outperforms other baselines in overall performance by analyzing the learning curves during the training of all algorithms. Next, we further analyze the success rate T_{suc} of each algorithm in pushing the box to the real target point and the average number of steps T_{step} required to push the box to the real target point. Specifically, we select the model with the best performance among the models trained by each algorithm, and then test it for 1000 episodes to evaluate the T_{suc} and T_{step}. The statistical data is shown in Tables IV and V. It can be seen that our algorithm has the highest success rate in all four scenarios, especially when $\boldsymbol{p}_{\text{tar}}^3 = [0.05, 0.3, 0.6, 0.05]$, and $\boldsymbol{p}_{\text{tar}}^4 = [0.2, 0.2, 0.2, 0.4]$, the success rate of our algorithm is much higher than other baselines (as shown in Table IV). In addition, the success rate of our algorithm in the four scenarios is close to the maximum probability value in the probability distribution $\boldsymbol{p}_{\text{tar}}$, indicating that our algorithm has learned the distribution of the latent variables behind the task with confounders. Finally, as shown in Table V, it can be seen that the average number of steps T_{step} of our MASAC-LV algorithm is the smallest, which means that the agents trained by the MASAC-LV algorithm can push the box to the real target point more quickly.

In conclusion, it has been verified the superiority of our algorithm in solving the box-pushing task full of confounders through the above four sets of experiments. The MASAC-LV algorithm plays an important role in discovering the space of the latent variables behind the task.

C. Ablation

In the loss function of the critic network in (22), we set the hyperparameters λ_1 and λ_2 to weight the environmental reward and intrinsic reward respectively. In this part, we set the parameter λ_1 to 1 and explore the influence of the size of the intrinsic reward on the performance of our algorithm in different scenarios by setting different values of λ_2. Specifi-

TABLE IV
THE SUCCESS RATE T_{suc} OF AGENTS COOPERATING TO PUSH THE BOX TO THE REAL TARGET POINT

p_t	MASAC-LV	MASAC	BSI	MADDPG	MAPPO	ISAC
p_t^1	**100.00%**	99.80%	99.90%	99.60%	99.40%	74.80%
p_t^2	**71.20%**	64.30%	65.40%	60.30%	61.50%	0.74%
p_t^3	**60.20%**	43.20%	45.80%	0.17%	37.50%	0.00%
p_t^4	**38.90%**	9.20%	12.30%	0.02%	0.01%	0.00%

TABLE V
AVERAGE NUMBER OF STEPS T_{step} REQUIRED FOR THE AGENTS TO COOPERATE TO PUSH THE BOX TO THE REAL TARGET POINT, WHERE — MEANS THAT THE BOX IS NOT PUSHED TO THE REAL TARGET POINT ONCE

p_t	MASAC-LV	MASAC	BSI	MADDPG	MAPPO	ISAC
p_t^1	**41.60**	47.88	47.15	53.30	49.77	65.59
p_t^2	**49.64**	58.08	56.45	66.90	62.91	69.46
p_t^3	**51.01**	59.26	62.23	88.06	78.75	—
p_t^4	**67.05**	75.63	75.34	93.50	95.00	—

cally, we set λ_2 to be 0.0, 0.1, 0.4, 0.7, and 1.0 respectively, where $\lambda_2 = 0$ means no intrinsic reward is added in the training process. We choose the random non-monotonic reward setting of Scenario 2 in the predator-prey environment and the parameter setting of $\boldsymbol{p}_{\text{tar}}^3 = [0.05, 0.3, 0.6, 0.05]$ with more confounder in the box-pushing environment. The ablation results in the two environments are shown in Fig. 9.

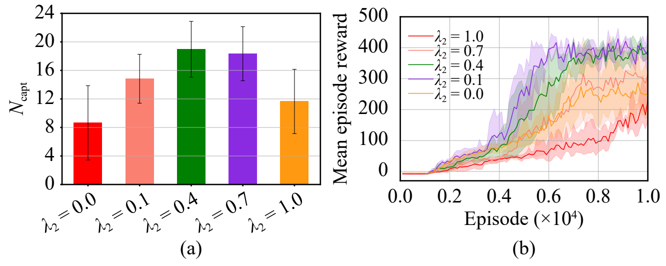

Fig. 9. Ablation data on hyperparameter λ_2: (a) The average number of times N_{capt} that prey is captured per episode in Scenario 2 of the predator-prey environment; and (b) Training curves at $\boldsymbol{p}_{\text{tar}}^3 = [0.05, 0.3, 0.6, 0.05]$ setting in the box-pushing environment.

From Fig. 9(a), it can be seen that when $\lambda_2 = 0.4$, the value of N_{capt} is the largest, but when $\lambda_2 = 0.7$, the value of N_{capt} is also very large, indicating that the best value of λ_2 may be between 0.4 and 0.7. From the results of $\lambda_2 = 0.1$ and $\lambda_2 = 1.0$, it can be known that the results obtained by setting the value of λ_2 too large or too small are not ideal. From the performance curves of different λ_2 value settings in Fig. 9(b), it can be seen that the performance of the algorithm is the best when $\lambda_2 = 0.1$ and $\lambda_2 = 0.4$. When λ_2 takes a larger value of 0.7 and 1.0, the performance of the algorithm is worse. The above analysis shows that when $\boldsymbol{p}_{\text{tar}}^3 = [0.05, 0.3, 0.6, 0.05]$ in the box-pushing environment, a smaller value of λ_2 can result in good performance.

In addition, when $\lambda_2 = 0$, our proposed algorithm does not perform very well in the above two environments, indicating that the performance of the algorithm will be greatly reduced if the intrinsic reward is removed. When $\lambda_2 = 0$, we further compared the data graphs in Fig. 9 under different scenarios with the algorithm performance of MASAC in Figs. 7(a) and 8(c) and found that the performance of our algorithm is still better than MASAC. This shows that even without the derived intrinsic reward, our inferred latent variable space can still improve the performance of the algorithm in those tasks with confounders. Comprehensively analyzing the above results, it can be concluded that the inference of latent variable space and the derivation of intrinsic rewards are indispensable, and they both play a positive role in the performance of the entire MASAC-LV algorithm framework.

VI. CONCLUSION

In this paper, the multi-agent soft actor-critic with the latent variable (MASAC-LV) algorithm is proposed under the centralized training with decentralized execution framework to discover the latent variable space in MARL tasks with confounders. The posterior distribution approximated by the inference network can effectively predict the latent variables in the environment at the current time step. The predicted latent variables are augmented into the observation space of the statistical policy and used directly for the evaluation of the value function. Further, to measure the different effects of latent variables on each agent in the complex MARL task with confounders, we derive the difference between the actual policy and the counterfactual policy and make it an intrinsic reward. The intrinsic reward can provide targeted exploration according to the different states of each agent in the environment. The MASAC-LV algorithm is evaluated in complex predator-prey and box-pushing tasks. We increase the difficulty of these two tasks by adding probabilistic confounders, where the experimental results have shown that our proposed algorithm is superior to other baseline algorithms in terms of convergence and final performance. Our ablation experiments demonstrate that both our inferred latent variable space and derived intrinsic rewards can play a positive role in the overall algorithmic framework.

REFERENCES

[1] K. Wang and C. Mu, "Learning-based control with decentralized dynamic event-triggering for vehicle systems," *IEEE Trans. Industrial Informatics*, vol. 19, no. 3, pp. 2629–2639, 2023.

[2] C. Mu, K. Wang, and T. Qiu, "Dynamic event-triggering neural learning control for partially unknown nonlinear systems," *IEEE Trans. Cyber.*, vol. 52, no. 4, pp. 2200–2213, 2022.

[3] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Robotics Research*, vol. 40, no. 4–5, pp. 698–721, 2021.

[4] P. Liu, D. Tateo, H. B. Ammar, and J. Peters, "Robot reinforcement learning on the constraint manifold," in *Proc. 5th Conf. Robot Learning*, 2022, vol. 164, pp. 1357–1366.

[5] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2017, pp. 31–36.

[6] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[7] O. Vinyals, I. Babuschkin, W. M. Czarnecki, *et al.*, "Grandmaster level in starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[8] C. Sun, W. Liu, and L. Dong, "Reinforcement learning with task decomposition for cooperative multiagent systems," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 5, pp. 2054–2065, 2021.

[9] H. J. Bae and P. Koumoutsakos, "Scientific multi-agent reinforcement learning for wall-models of turbulent flows," *Nature Communications*, vol. 13, no. 1, pp. 1–9, 2022.

[10] W. Liu, W. Cai, K. Jiang, G. Cheng, Y. Wang, J. Wang, J. Cao, L. Xu, C. Mu, and C. Sun, "XUANCE: A comprehensive and unified deep reinforcement learning library," arXiv preprint arXiv: 2312.16248, 2023.

[11] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," [Online], Available: https://arxiv.org/abs/1906.04737, 2019.

[12] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, p. 11, 2021.

[13] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11853–11864, 2020.

[14] W. Liu, L. Dong, J. Liu, and C. Sun, "Knowledge transfer in multi-agent reinforcement learning with incremental number of agents," *J. Systems Engineering and Electronics*, vol. 33, no. 2, pp. 447–460, 2022.

[15] W. Liu, L. Dong, D. Niu, and C. Sun, "Efficient exploration for multiagent reinforcement learning via transferable successor features," *IEEE CAA J. Autom. Sinica*, vol. 9, no. 9, pp. 1673–1686, 2022.

[16] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. Int. Conf. Machine Learning*, 2017, pp. 2681–2690.

[17] C. S. de Witt, B. Peng, P. Kamienny, P. H. S. Torr, W. Böhmer, and S. Whiteson, "Deep multi-agent reinforcement learning for decentralized continuous cooperative control," [Online], Available: https://arxiv.org/abs/2003.06709, 2020.

[18] J. Su, S. C. Adams, and P. A. Beling, "Value-decomposition multi-agent actor-critics," in *Proc. 35th AAAI Conf. Artificial Intelligence*, 2021, pp. 11352–11360.

[19] J. Li, K. Kuang, B. Wang, F. Liu, L. Chen, C. Fan, F. Wu, and J. Xiao, "Deconfounded value decomposition for multi-agent reinforcement learning," in *Proc. Int. Conf. Machine Learning*, 2022, vol. 162, pp. 12843–12856.

[20] R. Zohar, S. Mannor, and G. Tennenholtz, "Locality matters: A scalable value decomposition approach for cooperative multi-agent reinforcement learning," in *Proc. AAAI Conf. Artificial Intelligence*, 2022, vol. 36, no. 8, pp. 9278–9285.

[21] T. Zhang, Y. Li, C. Wang, G. Xie, and Z. Lu, "FOP: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning," in *Proc. 38th Int. Conf. Machine Learning*, 2021, vol. 139, pp. 12491–12500.

[22] Y. Chen, K. Wang, G. Song, and X. Jiang, "Entropy enhanced multiagent coordination based on hierarchical graph learning for continuous action space," [Online], Available: https://arxiv.org/abs/2208.10676, 2022.

[23] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," *Advances in Neural Information Processing Systems*, vol. 33, pp. 741–752, 2020.

[24] S. A. Sontakke, A. Mehrjou, L. Itti, and B. Schölkopf, "Causal curiosity: RL agents discovering self-supervised experiments for causal representation learning," in *Proc. 38th Int. Conf. Machine Learning*, 2021, vol. 139, pp. 9848–9858.

[25] N. Jaques, A. Lazaridou, E. Hughes, Ç. Gülçehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *Proc. 36th Int. Conf. Machine Learning*, 2019, vol. 97, pp. 3040–3049.

[26] L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiositydriven exploration," in *Proc. Advances in Neural Information Processing Systems*, 2021, pp. 3757–3769.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv preprint arXiv: 1312.6114, 2014.

[28] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. 34th Int. Conf. Machine Learning*, 2017, vol. 70, pp. 1352–1361.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Offpolicy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Machine Learning*, 2018, vol. 80, pp. 1856–1865.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv: 1509.02971, 2016.

[31] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Machine Learning*, 1993, pp. 330–337.

[32] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *J. Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

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

[34] M. Hua, C. Zhang, F. Zhang, Z. Li, X. Yu, H. Xu, and Q. Zhou, "Energy management of multi-mode plug-in hybrid electric vehicle using multiagent deep reinforcement learning," *Applied Energy*, vol. 348, p. 121526, 2023.

[35] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Machine Learning*, Jul. 2018, vol. 80, pp. 4295–4304.

[36] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32th AAAI Conf. Artificial Intelligence*, 2018, pp. 2974–2982.

[37] K. Jiang, W. Liu, Y. Wang, L. Dong, and C. Sun, "Credit assignment in heterogeneous multi-agent reinforcement learning for fully cooperative tasks," *Applied Intelligence*, vol. 53, no. 23, pp. 29205–29222, 2023.

[38] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network," in *Proc. 34th AAAI Conf. Artificial Intelligence*, 2020, pp. 7236–7243.

[39] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, "Latent space policies for hierarchical reinforcement learning," in *Proc. 35th Int. Conf. Machine Learning*, 2018, vol. 80, pp. 1846–1855.

[40] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Advance In Neural Information Processing Systems*, 2015, pp. 2746–2754.

[41] O. Rybkin, C. Zhu, A. Nagabandi, K. Daniilidis, I. Mordatch, and S. Levine, "Model-based reinforcement learning via latent-space collocation," in *Proc. 38th Int. Conf. Machine Learning*, 2021, vol. 139, pp. 9190–9201.

[42] D. Hafner, T. P. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Proc. 36th Int. Conf. Machine Learning*, 2019, vol. 97, pp. 2555–2565.

[43] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *Proc. Int. Conf. Machine Learning*, 2019, pp. 7444–7453.

[44] M. Gasse, D. Grasset, G. Gaudron, and P.-Y. Oudeyer, "Causal reinforcement learning using observational and interventional data," arXiv preprint arXiv:2106.14421, 2021.

[45] S. Lee and E. Bareinboim, "Structural causal bandits: Where to intervene?" in *Proc. Advance in Neural Information Processing Systems*, 2018, vol. 31, pp. 2573–2583.

[46] L. Wang, Z. Yang, and Z. Wang, "Provably efficient causal reinforcement learning with confounded observational data," in *Proc. Advances in Neural Information Processing Systems*, 2021, pp. 21164–21175.

[47] M. Seitzer, B. Schölkopf, and G. Martius, "Causal influence detection for improving efficiency in reinforcement learning," in *Proc. Advances in Neural Information Processing Systems*, 2021, pp. 22905–22918.

[48] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," in *Proc. 34th AAAI Conf. Artificial Intelligence*, 2020, pp. 2493–2500.

[49] N. Jaques, A. Lazaridou, E. Hughes, Ç. Gülçehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas, "Intrinsic social motivation via causal influence in multi-agent RL," [Online], Available: https://arxiv.org/abs/1810.08647, 2018.

[50] S. J. Grimbly, J. P. Shock, and A. Pretorius, "Causal multi-agent reinforcement learning: Review and open problems," [Online], Available: https://corr.org/abs/2111.06721, 2021.

[51] N. Gruver, J. Song, M. J. Kochenderfer, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning with latent variables," in *Proc. 19th Int. Conf. Autonomous Agents and Multiagent Systems*, 2020, pp. 1855–1857.

[52] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*. Boston, USA: Cengage Learning, 2015.

Kun Jiang received the B.S. degree in energy and power engineering and M.S. degree in transport engineering from the School of Energy and Power Engineering, Wuhan University of Technology in 2017 and 2020, respectively. He is currently a Ph.D. candidate in control science and engineering with the School of Automation, Southeast University. His current research interests include machine learning, deep reinforcement learning, and multi-agent cooperative control.

Wenzhang Liu received the B.S. degree in engineering from the College of Communication Engineering, Jilin University in 2016, and the Ph.D. degree in control science and engineering from the School of Automation, Southeast University in 2021. He is currently working as a Post-Doctoral Researcher with the School of Artificial Intelligence, Anhui University. His current research interests include deep reinforcement learning, multi-agent reinforcement learning, transfer learning, and robotics.

Yuanda Wang received the B.S. degree in automation from Nanjing University of Information Science and Technology in 2014, and the Ph.D. degree in control science and engineering from Southeast University in 2020. He is currently working as a Post-Doctoral Researcher with the School of Automation, Southeast University. He has been a visiting Ph.D. student with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, USA, from 2016 to 2018. His current research interests include deep reinforcement learning, robotics, and multiagent systems.

Lu Dong (Member, IEEE) received the B.S. degree in physics and the Ph.D. degree in electrical engineering from Southeast University in 2012 and 2017, respectively. She is currently an Associate Professor with the School of Cyber Science and Engineering, Southeast University. Her current research interests include adaptive dynamic programming, event triggered control, and MARL.

Changyin Sun (Senior Member, IEEE) received the B.S. degree in applied mathematics from the College of Mathematics, Sichuan University in 1996, and the M.S. and the Ph.D. degrees in electrical engineering from Southeast University in 2001 and 2004, respectively. He is currently a Professor with the School of Automation, Southeast University. From 2022, he has served as the Vice President of Anhui University, Dean of the Future College, and Dean of the School of Artificial Intelligence. His current research interests include neural networks, intelligent control, optimal theory.