A Variational Approach to Mutual Information-Based Coordination for Multi-Agent Reinforcement Learning

Anonymous Authors[1]

Abstract

In this paper, we propose a new mutual information (MMI) framework for multi-agent reinforcement learning (MARL) to enable multiple agents to learn coordinated behaviors by regularizing the accumulated return with the mutual information between multi-agent actions. By introducing a latent variable to induce nonzero mutual information between multi-agent actions and applying a variational bound, we derive a tractable lower bound on the considered MMI-regularized objective function. Applying policy iteration to maximize the derived lower bound, we propose a practical algorithm named variational maximum mutual information multi-agent actor-critic (VM3-AC). We evaluated VM3-AC for several games requiring coordination, and numerical results show that VM3-AC outperforms other MARL algorithms in multi-agent tasks requiring coordination.

1. Introduction

With the success of RL in the single-agent domain (Mnih et al., 2015; Lillicrap et al., 2015), MARL is being actively studied and applied to real-world problems such as traffic control systems and connected self-driving cars, which can be modeled as multi-agent systems requiring coordinated control (Li et al., 2019; Andriotis & Papakonstantinou, 2019). The simplest approach to MARL is independent learning, which trains each agent independently while treating other agents as a part of the environment, but this approach suffers from the problem of non-stationarity of the environment. A common solution to this problem is to use fully-centralized critic in the framework of centralized training with decentralized execution (CTDE) (OroojlooyJadid & Hajinezhad, 2019; Rashid et al., 2018; Lowe et al., 2017; Iqbal & Sha, 2018; Foerster et al., 2018). For example, MADDPG (Lowe et al., 2017) uses a centralized critic to train a decentralized policy for each agent, and COMA (Foerster et al., 2018) uses a common centralized critic to train all decentralized policies. However, these approaches assume that decentralized policies are independent and hence the joint policy is the product of each agent's policy. Such non-correlated factorization of the joint policy limits the agents to learn coordinated behavior due to negligence of the influence of other agents (Wen et al., 2019; de Witt et al., 2019). Recently, *mutual information* (MI) between multiple agents' actions has been considered as an effective intrinsic reward to promote coordination in MARL (Jaques et al., 2018). In (Jaques et al., 2018), MI between agents' actions is captured as social influence and the goal is to maximize the sum of accumulated return and social influence between agents' actions. It is shown that the social influence approach is effective for sequential social dilemma games. In this framework, however, causality between actions under coordination is required, and it is not straightforward to coordinate multi-agents' simultaneous actions. In certain multi-agent games, coordination of simultaneous actions of multiple agents is required to achieve cooperation for a common goal. For example, suppose that a pack of wolves try to catch a prey. To catch the prey, coordinating simultaneous actions among the wolves is more effective than coordinating one wolf's action and other wolves' actions at the next time because the latter case causes delay in coordination. In this paper, we propose a new approach to the MI-based coordination for MARL to coordinate simultaneous actions among multiple agents under the assumption of the knowledge of timing information among agents. Our approach is based on introducing a common latent variable to induce MI among simultaneous actions of multiple agents and on a variational lower bound on MI that enables tractable optimization. Under the proposed formulation, applying policy iteration by redefining value functions, we propose the VM3-AC algorithm for MARL to learn coordination of simultaneous actions among multiple agents. Numerical results show its superior performance on cooperative multi-agent tasks requiring coordination.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Related Work

MI is a measure of dependence between two variables (Cover & Thomas, 2006) and has been considered as an effective intrinsic reward for MARL (Wang et al., 2019; Jaques et al., 2018). (Mohamed & Rezende, 2015) proposed an intrinsic reward for empowerment by maximizing MI between agent's action and its future state. (Wang et al., 2019) proposed two intrinsic rewards capturing the influence based on a decision-theoretic measure and MI between an agent's current actions/states and other agents' next states. In particular, (Jaques et al., 2018) proposed a social influence intrinsic reward, which basically captures the mutual information between multiple agents' actions to achieve coordination, and showed that the social influence formulation yields good performance in sequential social dilemma environments. The difference of our approach from the social influence framework to MI-based coordination will be explained in Section 6.

Some previous works approached correlated policies from different perspectives. (Liu et al., 2020) proposed explicit modeling of correlated policies for multi-agent imitation learning, and (Wen et al., 2019) proposed a recursive reasoning framework for MARL to maximize the expected return by decomposing the joint policy into own policy and opponents' policies. Going beyond adopting correlated policies, our approach maximizes the MI between multiple agents' actions which is a measure of correlation.

In our approach, the MI between agents' action distributions is decomposed as the sum of each agent's action entropy and a variational term related to prediction of other agents' actions. Hence, our framework can be interpreted as enhancing correlated exploration by increasing the entropy of own policy (Haarnoja et al., 2018) while decreasing the uncertainty about other agents' actions. Some previous works proposed other techniques to enhance correlated exploration (Zheng & Yue, 2018; Mahajan et al., 2019). MAVEN addressed the poor exploration problem of QMIX by maximizing the mutual information between the latent variable and the observed trajectories (Mahajan et al., 2019). However, MAVEN does not consider the correlation among policies.

3. Background

Setup We consider a Markov Game (Littman, 1994), which is an extention of Markov Decision Process (MDP) to multi-agent setting. An N-agent Markov game is defined by an environment state space $\mathcal{S}$, action spaces for N agents $\mathcal{A}_1, \cdots, \mathcal{A}_N$, a state transition probability $p_{\mathcal{T}} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to [0, 1]$, where $\boldsymbol{\mathcal{A}} = \prod_{i=1}^{N} \mathcal{A}_i$ is the joint action space, and a reward function $\mathcal{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$. At each time step t, Agent i with policy π^i executes action $a_t^i \in \mathcal{A}_i$ based on state $s_t \in \mathcal{S}$. The actions of all agents $\boldsymbol{a}_t = (a_t^1, \cdots, a_t^N)$ yield the next state s_{t+1} according to $p_{\mathcal{T}}$ and shared common reward r_t according to $\mathcal{R}$ under the assumption of fully-cooperative MARL. The discounted return is defined as $R_t = \sum_{n=t}^{\infty} \gamma^n r_n$, where $\gamma \in [0, 1)$ is the discounting factor.

We assume centralized training and decentralized execution with timing information (CTDE/TI), which does not require communication among agents but requires synchronized timing information during the execution phase. Under CTDE/TI each agent can access all information including the environment state, observations and actions of other agents in the training phase, whereas the policy of each agent can be conditioned only on its own observation o_t^i and timing information in the execution phase. The goal of fully cooperative MARL is to find the optimal joint policy $\boldsymbol{\pi}^*$ that maximizes the objective $J(\boldsymbol{\pi}) = E_{\tau_0 \sim \boldsymbol{\pi}}[R_0]$, where $\tau_t = (s_t, \boldsymbol{a}_t, s_{t+1}, \boldsymbol{a}_{t+1}, \cdots)$ and $\boldsymbol{\pi} = (\pi^1, \cdots, \pi^N)$ denotes the joint policy of all agents.

Mutual Information-Based Coordination for MARL MI between agents' actions has been considered as an intrinsic reward to promote coordination in MARL (Jaques et al., 2018). Under this framework, one basically aims to find the policy that maximizes the weighted sum of the cumulative return and the MI between multi-agent actions. Thus, the MI-regularized objective function for joint policy $\boldsymbol{\pi}$ is given by

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\tau_0 \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_t + \alpha \sum_{(i,j)|i \neq j} I(a_t^i; a_t^j | s_t) \Big) \right], \tag{1}$$

where $I(a_t^i; a_t^j | s_t)$ is the MI between $a_t^i \sim \pi^i(\cdot|s_t)$ and $a_t^j \sim \pi^j(\cdot|s_t)$, and α is the temperature parameter that controls the relative importance of the MI against the reward. It is known that by regularization with MI in the objective function (1), the policy of each agent is encouraged to coordinate with other agents' policies. There are several approaches to implement (1). Under the social influence framework in (Jaques et al., 2018), the MI is decomposed as

$$I(a_t^i; a_t^j | s_t) = \int_{a_t^i, a_t^j} p(a_t^i, a_t^j | s_t) \log \frac{p(a_t^i, a_t^j | s_t)}{p(a_t^i | s_t) p(a_t^j | s_t)} \tag{2}$$

$$= \int_{a_t^i} p(a_t^i | s_t) \int_{a_t^j} p(a_t^j | a_t^i, s_t) \log \frac{p(a_t^j | a_t^i, s_t)}{p(a_t^j | s_t)} \tag{3}$$

$$= \int_{a_t^i} p(a_t^i | s_t) \underbrace{D_{KL}(p(a_t^j | a_t^i, s_t) || p(a_t^j | s_t))}_{\text{social influence of } i \text{ on } j}, \tag{4}$$

where $D_{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence.Thus, in this decomposition, influencing Agent i's policy is given by $\pi^i = p(a_t^i | s_t)$ and influenced Agent j's policy is given by $\pi^j = p(a_t^j | a_t^i, s_t)$. Hence, at time step t, influencing

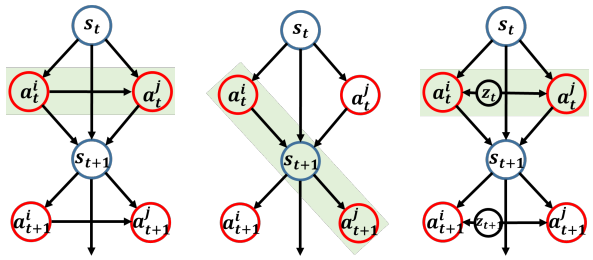


Figure 1. Causal diagram: (a) basic social influence, (b) social influence of modeling other agents, and (c) the proposed approach

Agent i acts first and then influenced Agent j acts based on a_t^i after Agent i acts, as shown in Fig. 1(a). Agents i and j cannot perform actions simultaneously. One way to remove this action ordering is to model other agents (Jaques et al., 2018). In this case, the causal influence of action a_t^i of Agent i at time t on action a_{t+1}^j of Agent j at time $t+1$ is considered, as shown in Fig. 1(b), i.e., the social influence $D_{KL}(p(a_{t+1}^j|a_t^i, s_t)||p(a_{t+1}^j|s_t))$ instead of the influence term in (4) is considered based on modeling $p(a_{t+1}^j|a_t^i, s_t)$ so that actions a_t^i and a_t^j can be performed simultaneously without ordering. In this case, however, the actually considered MI is $I(a_t^i; a_{t+1}^j|s_t)$ and is not the MI between a_t^i and a_t^j occurring at the same time. In this paper, we propose a different approach to MI regularization which enables simultaneous coordination between actions a_t^i and a_t^j both at time t without action ordering.

4. The Proposed Approach

We assume that the environment is fully observable, i.e., each agent can observe the environment state s_t for theoretical development in this section, and will consider partially observable environment for practical algorithm construction under CTDE/TI in the next section.

4.1. Formulation

Our approach to induce MI between concurrent two actions a_t^i and a_j^t of Agents i and j at time t is to introduce a common latent variable Z_t, as shown in Fig. 1(c). We assume that the latent variable Z_t has a prior distribution $p_Z(z_t)$ and that actions a_t^i and a_t^j are generated from the state variable s_t and the latent random variable Z_t. Thus, Agent i's action a_t^i at time t is drawn from the policy distribution of Agent i as

$$a_t^i \sim \pi^i(\cdot|S_t = s_t, Z_t), \quad i = 1, 2, \cdots, N, \tag{5}$$

where we use the upper case for random variables and the lower case as realization for the notation in the conditioning input terms for clarification. In case of stochastic policy, there is randomness in a_t^i even for given $S_t = s_t$, and furthermore we have additional randomness in a_t^i due to the random input Z_t since a function of random variable is a random variable. One can view the randomness due to Z_t as a perturbation to nominal a_t^i for given $S_t = s_t$. With the common perturbation-inducing variable Z_t, two random variables a_i^t and a_j^t conditioned on $S_t = s_t$ is correlated due to common Z_t, and nonzero MI $I(a_t^i; a_t^j|s_t)$ between concurrent a_t^i and a_t^j is induced. We aim to exploit this correlation for action coordination. (See Appendix A for a simple example and explanation of our basic idea with the simple example.)

With nontrivial MI $I(a_t^i; a_t^j|s_t)$, we now express this MI. First, note in (4) that we need $p(a_t^j|a_t^i, s_t)$ to compute the MI but we do not want to use $p(a_t^j|a_t^i, s_t)$ directly. For this, we adopt a variational distribution $q(a_t^j|a_t^i, s_t)$ to approximate $p(a_t^j|a_t^i, s_t)$ and derive a lower bound on the MI $I(a_t^i; a_t^j|s_t)$ as follows: $I(a_t^i; a_t^j|s_t) =$

$$\begin{aligned}
&\int_{a_t^i, a_t^j} p(a_t^i, a_t^j|s_t) \log \frac{p(a_t^i, a_t^j|s_t)}{p(a_t^i|s_t)p(a_t^j|s_t)} \\
&= \int_{a_t^i, a_t^j} p(a_t^i, a_t^j|s_t) \log \frac{p(a_t^i|s_t)p(a_t^j|a_t^i, s_t)q(a_t^j|a_t^i, s_t)}{p(a_t^i|s_t)p(a_t^j|s_t)q(a_t^j|a_t^i, s_t)} \\
&= \mathbb{E}_{p(a_t^i, a_t^j|s_t)} \left[\log \frac{q(a_t^j|a_t^i, s_t)}{p(a_t^j|s_t)}\right] \\
&\qquad \times \mathbb{E}_{p(a_t^i|s_t)} \left[D_{KL}(p(a_t^j|a_t^i, s_t)\|q(a_t^j|a^i, s_t)\right] \\
&\geq H(a_t^j|s_t) + \mathbb{E}_{p(a_t^i|s_t)p(a_t^j|a_t^i, s_t)} \left[\log q(a_t^j|a_t^i, s_t)\right],
\end{aligned} \tag{6}$$

where $H(a_t^j|s_t)$ denotes the entropy of a_t^j given s_t, i.e., the entropy of the following marginal distribution of a_t^j in our case:

$$\tilde{\pi}^j(a_t^j|s_t) := \int_{z_t} \pi^j(a_t^j|S_t = s_t, Z = z_t)p_Z(z_t)dz_t. \tag{7}$$

The last inequality in (6) holds because the KL divergence is always non-negative. For the variational distribution $q(a_t^j|a_t^i, s_t)$ we consider a class of distributions , i.e., $q(a_t^j|a_t^i, s_t) \in \mathcal{Q}$. The lower bound (6) becomes tight when $q(a_t^j|a_t^i, s_t)$ approximates $p(a_t^j|a_t^i, s_t)$ well. Note that in our expansion, the lower bound on the MI $I(a_t^i; a_t^j|s_t)$ is expressed as *the sum of the action entropy* $H(a_t^j|s_t)$ *and the negative of the cross entropy of* $q(a_t^j|a_t^i, s_t)$ *relative to* $p(a_t^j|a_t^i, s_t)$ averaged over $p(a_t^i|s_t)$. Using the symmetry of MI, we can rewrite the lower bound as

$$\begin{aligned}
I(a_t^i; a_t^j|s_t) \geq \frac{1}{2}\Big\{&H(a_t^i|s_t) + H(a_t^j|s_t) + \\
&\mathbb{E}_{p(a_t^i, a_t^j|s_t)} \left[\log q(a_t^j|a_t^i, s_t) + \log q(a_t^i|a_t^j, s_t)\right]\Big\}.
\end{aligned} \tag{8}$$

Then, our goal is to maximize this lower bound of MI by using a tractable approximation $q(a_t^i|a_t^j, s_t) \in \mathcal{Q}$. Our decompsition of MI based on the action entropy and the cross

entropy is effective in our variational formulation for MI-based MARL. Consider one of the cross entropy terms in the right-hand side (RHS) of (8): $\mathbb{E}_{p(a_t^i,a_t^j|s_t)}[\log q(a_t^j|a_t^i,s_t)]$, which can be rewritten as $\mathbb{E}_{p(a_t^i,a_t^j|s_t)}[\log q(a_t^j|a_t^i,s_t)] =$

$$-\mathbb{E}_{p(a_t^i|s_t)}\Big[H(p(a_t^j|a_t^i,s_t)) + D_{KL}(p(a_t^j|a_t^i,s_t)||q(a_t^j|a_t^i,s_t))\Big], \tag{9}$$

based on the well-known decomposition of the cross entropy. Hence, by maximizing the negative of this cross entropy term, we can learn π^i (generating a_t^i) and π^j (generating a_t^j) so that the conditional entropy $H(p(a_t^j|a_t^i,s_t))$ of a_t^j given a_t^i is minimized, i.e., the two actions are more correlated to each other, and learn q that closely approximates the true $p(a_t^j|a_t^i,s_t)$, i.e., the D_{KL} term in (9) is minimized.

4.2. Modified Policy Iteration

Our algorithm construction is based on policy iteration. In order to develop policy iteration for the proposed MI framework, we first replace the original MI-regularized objective function (1) with the following tractable objective function based on the variational lower bound (8):

$$\hat{J}(\boldsymbol{\pi},q) = \mathbb{E}_{\substack{\tau_0\sim\boldsymbol{\pi}\\ z_t\sim p_Z}}\Bigg[\sum_{t=0}^{\infty}\gamma^t\Big(r_t(s_t,\boldsymbol{a_t}) + \alpha N\sum_{i=1}^{N}H(a_t^i|s_t) + \alpha\sum_{i=1}^{N}\sum_{j\neq i}\log q(a_t^j|a_t^i,s_t)\Big)\Bigg], \tag{10}$$

where $\boldsymbol{\pi} = [\pi^1,\cdots,\pi^N]$ and π^i is given by (5) and $\boldsymbol{a}_t = [a_t^1,\cdots,a_t^N]$. Then, we determine the individual objective function $\hat{J}^i(\pi^i,q)$ for Agent i as the sum of the terms in (10) associated with Agent i's policy π^i or action a_t^i, given by

$$\hat{J}^i(\pi^i,q) = \mathbb{E}_{\substack{\tau_0\sim\boldsymbol{\pi}\\ z_t\sim p_Z}}\Bigg[\sum_{t=0}^{\infty}\gamma^t\Big(\underbrace{r_t(s_t,\boldsymbol{a_t}) + \beta\cdot H(a^i|s_t)}_{(a)} + \frac{\beta}{N}\sum_{j\neq i}\Big[\underbrace{\log q(a_t^i|a_t^j,s_t) + \log q(a_t^j|a_t^i,s_t)}_{(b)}\Big]\Big)\Bigg], \tag{11}$$

where $\beta = \alpha N$ is the temperature parameter. Note that maximizing the term (a) in (11) implies that each agent maximizes the weighted sum of the action entropy and the return, which can be interpreted as an extension of maximum entropy RL (Haarnoja et al., 2018) to multi-agent setting. On the other hand, maximizing the term (b) with respect to π^i and q means that we update the policy π^i so that the conditional entropy of a_t^j given a_t^i and the conditional entropy of a_t^i given a_t^j are reduced, as already mentioned below (9). Thus, the objective function (11) can be interpreted as the maximum entropy MARL objective combined with action correlation or coordination. Hence, the proposed objective function (11) can be considered as one implementation of the concept of *correlated exploration* in MARL (Mahajan et al., 2019).

Now, in order to learn policy π^i to maximize the objective function (11), we modify the policy iteration in standard RL. For this, we redefine the state and state-action value functions for Agent i as

$$Q_i^{\boldsymbol{\pi}}(s,a) \triangleq \mathbb{E}_{\substack{\tau_0\sim\boldsymbol{\pi}\\ z_t\sim p_Z}}\Big[r_0 + \gamma V_i^{\boldsymbol{\pi}}(s_1)\Big|s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}\Big], \tag{12}$$

$$V_i^{\boldsymbol{\pi}}(s) \triangleq \mathbb{E}_{\substack{\tau_0\sim\boldsymbol{\pi}\\ z_t\sim p_Z}}\Bigg[\sum_{t=0}^{\infty}\gamma^t\Big(r_t + \beta H(a_t^i|s_t) + \frac{\beta}{N}\sum_{j\neq i}\log q^{(i,j)}(a_t^i,a_t^j|s_t)\Big)\Big|s_0 = s\Bigg], \tag{13}$$

where $q^{(i,j)}(a_t^i,a_t^j|s_t) \triangleq q(a_t^i|a_t^j,s_t)q(a_t^j|a_t^i,s_t)$. Then, the Bellman operator corresponding to $V_i^{\boldsymbol{\pi}}$ and $Q_i^{\boldsymbol{\pi}}$ on the value function estimates $V_i(s)$ and $Q_i(s,\boldsymbol{a})$ is given by

$$\mathcal{T}^{\boldsymbol{\pi}}Q_i(s,\boldsymbol{a}) \triangleq r(s,\boldsymbol{a}) + \gamma\mathbb{E}_{s'\sim p}[V_i(s')], \tag{14}$$

where $V_i(s) = \mathbb{E}_{\substack{\boldsymbol{a}\sim\boldsymbol{\pi}\\ z_t\sim p_Z}}\Big[Q_i(s,\boldsymbol{a}) - \beta\log\tilde{\pi}^i(a^i|s) + \frac{\beta}{N}\sum_{j\neq i}\log q^{(i,j)}(a^i,a^j|s)\Big]$, and $\tilde{\pi}^i$ is the marginal distribution given in (7). In the policy evaluation step, we compute the value functions defined in (12) and (13) by applying the modified Bellman operator $\mathcal{T}^{\boldsymbol{\pi}}$ repeatedly to any initial function $Q_i^{(0)}$.

Proposition 1. *(Variational Policy Evaluation). For fixed $\boldsymbol{\pi}$ and the variational distribution q, consider the modified Bellman operator $\mathcal{T}^{\boldsymbol{\pi}}$ in (14) and an arbitrary initial function $Q_i^{(0)}: \mathcal{S}\times\mathcal{A}\to\mathbb{R}$, and define $Q_i^{(k+1)} = \mathcal{T}^{\boldsymbol{\pi}}Q_i^{(k)}$. Then, $Q_i^{(k)}$ converges to $Q_i^{\boldsymbol{\pi}}$ defined in (12).*

Proof. See Appendix B.

In the policy improvement step, we update the policy and the variational distribution by using the value function evaluated in the policy evaluation step. Here, each agent updates its policy and variational distribution while keeping other agents' policies fixed as follows: $(\pi_{k+1}^i, q_{k+1}) =$

$$\arg\max_{\pi^i,q}\mathbb{E}_{\substack{(a^i,a^{-i})\sim(\pi^i,\pi_k^{-i})\\ z_k\sim P_Z}}\Big[Q_i^{\boldsymbol{\pi}_k}(s,\boldsymbol{a}) - \beta\log\tilde{\pi}^i(a^i|s) + \frac{\beta}{N}\sum_{j\neq i}\log q^{(i,j)}(a^i,a^j|s))\Big], \tag{15}$$

where $a^{-i} \triangleq \{a^1,\cdots,a^N\}\backslash\{a^i\}$ and π_k^{-i} is the collection the policies for all agents except Agent i at the k-th iteration. Then, we have the following lemma regarding the improvement step.

Proposition 2. *(Variational Policy Improvement). Let π^i_{new} and q_{new} be the updated policy and the variational distribution from (15). Then, $Q_i^{\pi^i_{new},\pi^{-i}_{old}}(s,\boldsymbol{a}) \geq Q_i^{\pi^i_{old},\pi^{-i}_{old}}(s,\boldsymbol{a})$ for all $(s,\boldsymbol{a}) \in (\mathcal{S}\times\boldsymbol{\mathcal{A}})$. Here, $Q_i^{\pi^i_{new},\pi^{-i}_{old}}(s,\boldsymbol{a})$ means $Q_i^{\boldsymbol{\pi}}(s,\boldsymbol{a})|_{\boldsymbol{\pi}=(\pi^i_{new},\pi^{-i}_{old})}$.*

Proof. See Appendix B.

The modified policy iteration is defined as applying the variational policy evaluation and variational improvement steps in an alternating manner. Each agent trains its policy, critic and the variational distribution to maximize its objective function (11).

5. Algorithm Construction

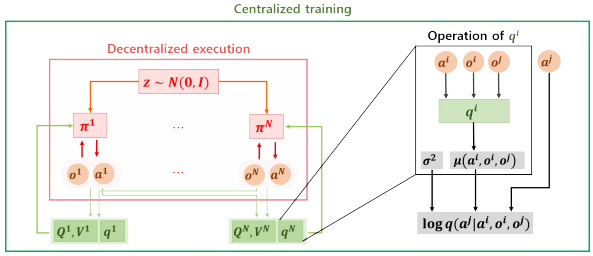


Figure 2. Overall operation of the proposed VM3-AC. We only need the operation in the red box after training.

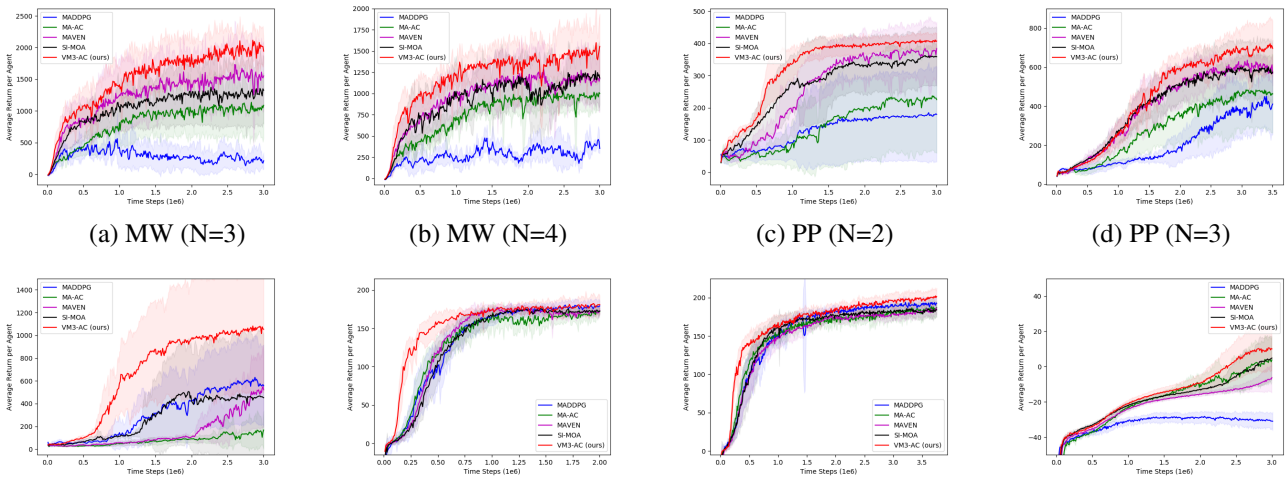
Summarizing the development above, we now propose the variational maximum mutual information multi-agent actor-critic (VM3-AC) algorithm, which can be applied to partially observable multi-agent environments. The overall operation of VM3-AC is shown in Fig. 2. Under CTDE/TI, each agent's policy is conditioned only on local observation o_t^i and the common input z_t, and centralized critics are conditioned on either the environment state or the observations of all agents, depending on the situation (Lowe et al., 2017). Let $\boldsymbol{x}$ denote either the environment state s or the observations of all agents $(o^1,\cdots,o^N)$, whichever is used. In order to deal with the large state-action spaces, we adopt deep neural networks to approximate the required functions. For Agent i, we parameterize the policy as $\pi^i_{\phi^i}(a|o^i,z)$ with parameter ϕ^i, the variational distribution as $q_{\xi^i}(a^j|a^i,(o^i,o^j))$ with parameter ξ^i, the state-value function as $V^i_{\psi_i}(\boldsymbol{x})$ with parameter ψ^i, and two action-value functions as $Q^i_{\theta^{i,1}}(\boldsymbol{x},\boldsymbol{a})$ and $Q^i_{\theta^{i,2}}(\boldsymbol{x},\boldsymbol{a})$ with parameters $\theta^{i,1}$ and $\theta^{i,2}$. Note that in the original variational distribution, a_t^j is conditioned on a_t^i and s_t. In the partially observable case, we replace s_t with (o_i,o_j).

For the prior distribution P_Z of the injection variable z_t, we use zero-mean multivariate Gaussian distribution with identity covariance matrix, i.e., $z_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, where the dimension is a hyperparameter, given in Appendix E. We further assume that the class $\mathcal{Q}$ of the variational distribution is multivariate Gaussian distribution with constant covariance matrix $\sigma^2\mathbf{I}$ with dimension of the action dimension, i.e., $\mathcal{Q} = \{q_{\xi^i}(a^j|a^i,(o^i,o^j)) = \mathcal{N}(\mu_{\xi^i}(a^i,o^i,o^j),\sigma^2\mathbf{I})\}$, where $\mu_{\xi^i}(a^i,o^i,o^j)$ is the mean of the distribution.

Centralized Training The parameterized value functions, the policy, and the variational distribution are trained based on proper loss functions derived from Section 4.2 in a similar way to the training in SAC in a centralized manner. Due to space limitation, training detail and pseudo code are provided in Appendices C and D, respectively.

Decentralized Execution with Timing Information In the centralized training phase, we pick actions $(a_t^1,\cdots,a_t^N)$ according to $\pi^1(a_t^1|s_t,z_t),\cdots,\pi^N(a_t^N|s_t,z_t)$ (or with s_t replaced with $(o_t^1,\cdots,o_t^N)$), where common z_t generated from zero-mean Gaussian distribution is shared under the centralized assumption. In the decentralized execution phase, however, sharing z_t among agents for simultaneous action correlation does not come for free. One simple way is communication among the agents, but this case prevents decentralized execution. We can eliminate the necessity of communication under the assumption of timing information. Note that z_t is not actual message-carrying information but a dummy random sequence. Practically, a random sequence is generated based on a random sequence generator. Hence, we require all agents to have the same Gaussian random sequence generator and distribute the same seed and initiation timing to these random sequence generators before deployment for the execution phase. (Mahajan et al. (2019) also considered that multiple agents share the realization of latent variables in the beginning of the episode.) Such implementation is possible with reference timing information such as global positioning system (GPS) and such synchronization is widely used in cellular communication networks. Note that even with common decentralized execution, time step synchronization is required. Thus, we additionally need reference timing information on top of time step synchronization for CTDE/TI. This is the additional cost for CTDE/TI over CTDE.

An alternative without timing information is to exploit the property of zero-mean Gaussian input variable z_t to the policy network. During the centralized training period, the parameters $\phi^1,\cdots,\phi^N$ of the policy networks $\pi^1_{\phi^1}(a|o^1,z),\cdots,\pi^N_{\phi^N}(a|o^N,z)$ (with input (o^i,z) and output a) are learned so that actions $a_t^1,\cdots,a_t^N$ are coordinated for random perturbation input z_t drawn from P_Z. Note that the coordination behavior is learned and engraved into the parameters $\phi_1,\cdots,\phi_N$ not into the input z_t. So, we only use this stored parameter information during the decentralized execution phase. We apply the common mean value $\mathbb{E}\{z_t\}$ to the z_t input of the trained policy network $\pi^i_{\phi^i}(a_t^i|o_t^i,z_t)$ of Agent i, $\forall i$. In this



(a) MW (N=3) (b) MW (N=4) (c) PP (N=2) (d) PP (N=3)

(e) PP (N=4) (f) CTC (N=4) (g) CTC (N=5) (h) CN (N=3)

Figure 3. Performance of MADDPG (blue), MA-AC (green), MAVEN (purple), SI-MOA (black), and VM3-AC (the proposed method, red) on multi-walker environments (a)-(b), predator-prey (c)-(e), cooperative treasure collection (f)-(g), and cooperative navigation (f). (MW, PP, CTC, and CN denote multi-walker, predator-prey, cooperative treasure collection and cooperative navigation, respectively)

case, actions $a_t^1, \cdots, a_t^N$ are independent conditioned on $s_t \ni (o_t^1, \cdots, o_t^N)$ but a specific joint bias (most representative joint bias) is applied to actions $a_t^1, \cdots, a_t^N$. We expect that this joint bias is helpful and this situation is described in a toy example in Appendix A and the corresponding numerical result is provided in Appendix A.

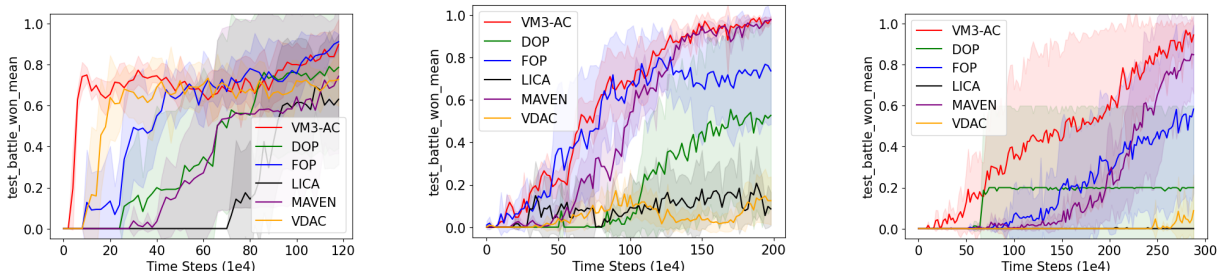
6. Experiment

In this section, we provide numerical results on both continuous and discrete action tasks.

Experiment on continuous action tasks We consider the following continuous action tasks with the varying number of agents: multi-walker (Gupta et al., 2017), predator-prey (Lowe et al., 2017), cooperative treasure collection (Iqbal & Sha, 2019), and cooperative navigation (Lowe et al., 2017). The detailed setting of each tasks is provided in Appendix F. Here, we considered four baselines: 1) MADDPG (Lowe et al., 2017) - an extension of DDPG with a centralized critic to train a decentralized policy for each agent. 2) Multi-agent actor-critic (MA-AC) - a variant of VM3-AC ($\beta = 0$) without the latent variable. 3) Multi-agent variational exploration (MAVEN) (Mahajan et al., 2019). Similarly to VM3-AC, MAVEN introduced latent variable and variational approach for optimizing the mutual information. However, MAVEN does not consider the mutual information between actions but consider the mutual information between the latent variable and trajectories of the agents. 4) Social Influence with MOA (SI-MOA) (Jaques et al., 2018), which is explained in Section 3. Both MAVEN and SI-MOA are implemented on top of MA-AC since we consider continuous action-space environments.

Fig. 3 shows the learning curves for the considered four environments with the different number of agents. The y-axis denotes the average of all agents' rewards averaged over 7 random seeds, and the x-axis denotes time step. The hyperparameters including the temperature parameter β and the dimension of the latent variable are provided in Appendix E. As shown in Fig. 3, VM3-AC outperforms the baselines in the considered environments. Especially, in the case of the multi-walker environment, VM3-AC has large performance gain over existing state-of-the-art algorithms. This is because the agents in the multi-walker environment are strongly required to learn simultaneous coordination in order to obtain high rewards. In addition, the agents in the predator-prey environment, where the number of agents is four, should spread out in groups of two to get more reward. In this environment, VM3-AC also has large performance gain. Thus, it is seen that the proposed MMI framework improves performance in complex multi-agent tasks requiring high-quality coordination. It is observed that both MAVEN and SI-MOA outperform the basic algorithm MA-AC but not VM3-AC. Hence, the numerical results show that the way of using MI by the proposed VM3-AC algorithm has some advantages over those by MAVEN and SI-MOA, especially for MARL tasks requiring coordination of concurrent actions.

Experiment on discrete action task We also considered the StarcraftII micromanagement benchmark (SMAC) environment (Samvelyan et al., 2019). We modified the SMAC environment to be sparse by giving rewards when an ally or an enemy dies and a time penalty. Thus, in the case of *3s vs 3z*, the reward is hardly obtained because it takes a long time to remove a zealot (enemy). We provided



(a) *3m* (b) *2s3z* (c) *3s vs 3z*

Figure 4. Performance of DOP (green), FOP (blue), LICA (black), MAVEN (purple), VDAC (orange) and VM3-AC (red) on three maps in the modified SMAC environment.

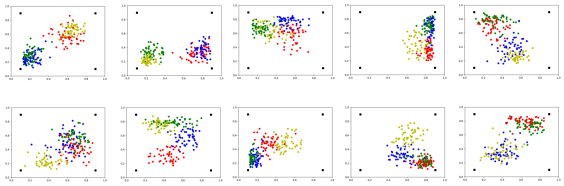


Figure 5. The positions of four agents after five time-steps after the episode begins in the early stage of the training: 1st row - VM3-AC and 2nd row - MA-SAC. The figures in each column correspond to a different seed. The black squares are the preys and each color except black shows the position of each agent.

the detailed setting of the modified SMAC environment in Appendix G. We considered five state-of-the-art baselines: DOP (Wang et al., 2020), FOP (Zhang et al., 2021), LICA (Zhou et al., 2020), MAVEN (Mahajan et al., 2019), and VDAC (Su et al., 2021). We implemented VM3-AC on the top of FOP by introducing the latent variable and replacing the entropy term in (Zhang et al., 2021) with the MI. Fig. 5 shows the performances of VM3-AC and the baselines on three maps in SMAC. It is observed that VM3-AC outperforms the baselines. Especially on 3svs3z, in which reward is higly sparse, VM3-AC outperforms the baselines in terms of both training speed and final performance.

6.1. Ablation Study and Discussion

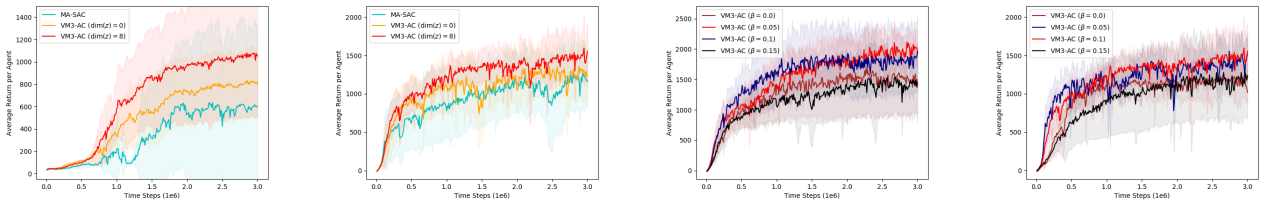
In this subsection, we provide ablation study and discussion on the major techniques and hyperparameters of VM3-AC: 1) mutual information versus entropy 2) the latent variable, 3) the temperature parameter β, and 4) scalability.

Mutual information versus entropy: The proposed MI framework maximizes the sum of the action entropy and the negative of the cross entropy of the variational conditional distribution relative to the true conditional distribution, which provides a lower bound of MI between actions. As aforementioned, maximizing the sum of the action entropy and the negative of the cross entropy of the variational conditional distribution relative to the true conditional distribution enhances exploration and predictability for other agents' actions. Hence, the proposed MI framework enhances correlated exploration among agents.

We compared VM3-AC with multi-agent-SAC (MA-SAC), which is an extension of maximum entropy soft actor-critic (SAC) (Haarnoja et al., 2018) to multi-agent setting. For MA-SAC, we extended SAC to multi-agent setting in the manner of independent learning. Each agent trains its decentralized policy using decentralized critic to maximize the weighted sum of the cumulative return and the entropy of its policy. Adopting the framework of CTDE, we replaced decentralized critic with centralized critic which incorporates observations and actions of all agents.

We performed an experiment in the predator-prey environment with four agents where the number of required agents to catch the prey is two. In this environment, the agents started at the center of the map. Hence, the agents should spread out in groups of two to catch preys efficiently. Fig.5 shows the positions of the four agents at five time-steps after the episode starts. The first and second rows in Fig.5 show the results of VM3-AC and MA-SAC in the early stage of the training, respectively. It is seen that the agents of VM3-AC explore in groups of two while the agents of MA-SAC tend to explore independently. We provided the performance comparisons of VM3-AC with MA-SAC in Fig.6 (a) and (b).

Latent variable: The role of the latent variable is to induce MI among concurrent actions and inject additional degree-of-freedom for action control. We compared VM3-AC and VM3-AC without the latent variable (implemented by setting dimension$(z_t) = 0$) in the multi-walker environment. In both cases, VM3-AC yields better performance that VM3-AC without the latent variable as shown in Fig.6(a) and 6(b). Here, the gain by VM3-AC without the latent variable (i.e., dimension$(z_t) = 0$) over MA-SAC is solely due to passive modeling $p(a_t^j|a_t^i, s_t)$ by using $q(a_t^j|a_t^i, s_t)$, not including active injection of coordination by z_t. Performance results



(a) PP (N=4) (b) MW (N=4) (c) MW (N=3) (d) MW (N=4)

Figure 6. (a) and (b): VM3-AC (red), VM3-AC without latent variable (orange), and MA-SAC (cyan) and (c) and (d): performance with respect to the temperature parameter

on the case of replacing the latent variable $z_t \sim \mathcal{N}(0, \mathbf{I})$ with mean vector $z_t = \mathbb{E}(z_t)$ in the execution phase is provided in Appendix H.

Temperature parameter β: The role of temperature parameter β is to control the relative importance between the reward and the MI. We evaluated VM3-AC by varying $\beta = [0, 0.05, 0.1, 0.15]$ in the multi-walker environment with $N = 3$ and $N = 4$. Fig. 6(c) and 6(d) show that VM3-AC with the temperature value around $[0.05, 0.1]$ yields good performance.

Scalability: Many MARL algorithms which use a centralized critic such as MADDPG (Lowe et al., 2017) can suffer from the problem of scalability due to increasing joint state-action space as the number of agents increases. VM3-AC can also suffer from the same issue but we can address the problem by adopting an attention mechanism as in MAAC (Iqbal & Sha, 2018). Additionally, VM3-AC needs more variational approximation networks as the number of agents increases. As many MARL algorithms share the parameters among agents, we can share the parameters for the variational approximation networks. We expect that parameter sharing can handle the scalability of the proposed method.

7. Conclusion

In this paper, we have proposed a new approach to MI-based coordinated MARL to induce coordination of concurrent actions under CTDE/TI. In the proposed approach, a common correlation-inducing random variable is injected into each policy network, and the MI between actions induced by this variable is expressed as a tractable form by using a variational distribution in order to enable construction of a practical algorithm based on policy iteration. We evaluated the derived algorithm named VM3-AC on both continuous and discrete action tasks and the numerical results show that VM3-AC outperforms other state-of-the-art baselines, especially in multi-agent tasks requiring high-quality coordination among agents.

References

Achiam, J. Spinning Up in Deep Reinforcement Learning. 2018.

Agarwal, P., Jleli, M., and Samet, B. Fixed point theory in metric spaces. *Recent Advances and Applications*, 2018.

Andriotis, C. and Papakonstantinou, K. Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety*, 191:106483, 2019.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley, 2006.

de Witt, C. S., Foerster, J., Farquhar, G., Torr, P., Böhmer, W., and Whiteson, S. Multi-agent common knowledge reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 9924–9935, 2019.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Folland, G. B. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Gupta, J. K., Egorov, M., and Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83. Springer, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Iqbal, S. and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912*, 2018.

Iqbal, S. and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 2961–2970. PMLR, 2019.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P. A., Strouse, D., Leibo, J. Z., and De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *arXiv preprint arXiv:1810.08647*, 2018.

Kim, W., Cho, M., and Sung, Y. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6079–6086, 2019.

Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., and Ye, J. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pp. 983–994, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Liu, M., Zhou, M., Zhang, W., Zhuang, Y., Wang, J., Liu, W., and Yu, Y. Multi-agent interactions modeling with correlated policies. *arXiv preprint arXiv:2001.03415*, 2020.

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

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pp. 7611–7622, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2125–2133, 2015.

OroojlooyJadid, A. and Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Su, J., Adams, S., and Beling, P. A. Value-decomposition multi-agent actor-critics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11352–11360, 2021.

Wang, T., Wang, J., Wu, Y., and Zhang, C. Influence-based multi-agent exploration. *arXiv preprint arXiv:1910.05512*, 2019.

Wang, Y., Han, B., Wang, T., Dong, H., and Zhang, C. Dop: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*, 2020.

Wen, Y., Yang, Y., Luo, R., Wang, J., and Pan, W. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

Zhang, T., Li, Y., Wang, C., Xie, G., and Lu, Z. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 12491–12500. PMLR, 2021.

Zheng, S. and Yue, Y. Structured exploration via hierarchical variational policy networks. 2018.

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

Appendix A: Correlation Based on Common Z_t and Basic Idea

Here, we provide a toy example explaining our idea. The example is as follows. We have two agents: Agents 1 and 2 in a 2-dimensional half-plane (x, y) with $y > 0$. The state is the locations of the two agents, i.e., $s_t = ((x_t^1, y_t^1), (x_t^2, y_t^2))$, where (x_t^i, y_t^i) is the location of Agent i. The action of each agent is the displacement, i.e., the action of Agent i is $a_t^i = (\Delta x_t^i, \Delta y_t^i)$, $i = 1, 2$. The location of Agent i at time $t + 1$ is determined as a function of the state and action at current time t:

$$(x_{t+1}^i, y_{t+1}^i) = (x_t^i, y_t^i) + (\Delta x_t^i, \Delta y_t^i).$$

Suppose that Agent i can only observe its own location $o^i = (x_t^i, y_t^i)$ and suppose that 22 the policies $\pi^1(a_t^1|o_t^1, z_t)$ and $\pi^2(a_t^2|o_t^2, z_t)$ of the two agents are functions of the observation and an additional common random variable z_t, and given by the following simple linear stochastic model:

$$\begin{aligned} a_t^1 &= \begin{bmatrix} a_t^{x,1} \\ a_t^{y,1} \end{bmatrix} = \begin{bmatrix} \Delta x_t^1 \\ \Delta y_t^1 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix}}_{parameter\ for\ o_t^1} \underbrace{\begin{bmatrix} x_t^1 \\ y_t^1 \end{bmatrix}}_{o_t^1} + \underbrace{\begin{bmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{bmatrix}}_{parameter\ for\ z_t\ at\ Agent\ 1} \underbrace{\begin{bmatrix} z_t^x \\ z_t^y \end{bmatrix}}_{z_t} + \underbrace{\begin{bmatrix} n_t^{x,1} \\ n_t^{y,1} \end{bmatrix}}_{noise\ \mathbf{n}_t^1\ at\ Agent\ 1} \end{aligned} \tag{16}$$

$$\begin{aligned} a_t^2 &= \begin{bmatrix} a_t^{x,2} \\ a_t^{y,2} \end{bmatrix} = \begin{bmatrix} \Delta x_t^2 \\ \Delta y_t^2 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix}}_{parameter\ for\ o_t^2} \underbrace{\begin{bmatrix} x_t^2 \\ y_t^2 \end{bmatrix}}_{o_t^2} + \underbrace{\begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \end{bmatrix}}_{parameter\ for\ z_t\ at\ Agent\ 2} \underbrace{\begin{bmatrix} z_t^x \\ z_t^y \end{bmatrix}}_{z_t} + \underbrace{\begin{bmatrix} n_t^{x,2} \\ n_t^{y,2} \end{bmatrix}}_{noise\ \mathbf{n}_t^2\ at\ Agent\ 2}, \end{aligned} \tag{17}$$

where the two random noise terms $\mathbf{n}_t^1$ and $\mathbf{n}_t^2$ at Agents 1 and 2 are independent random variables; $z_t = \begin{bmatrix} z_t^x \\ z_t^y \end{bmatrix}$ is a random variable (precisely speaking, random vector) drawn from $P_Z(z)$; and the notation of two consecutive brackets $[\cdot][\cdot]$ means matrix multiplication. Fig. 7 describes the policy function of Agent 1 given by (16) in a graphical form. ((17) can be described in a similar graphical form.)

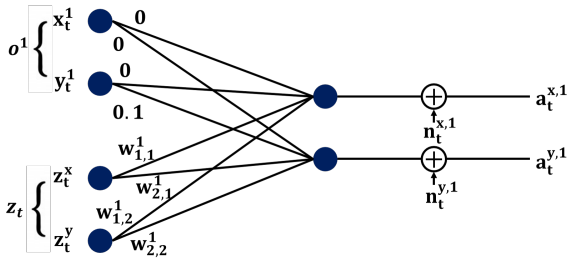


Figure 7. Graphical representation of the policy function of Agent 1, eq. (16)

In (16), the noise term $\mathbf{n}_t^1$ is added to perturb the action of Agent 1 for exploration around the given term $\begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix} \underbrace{\begin{bmatrix} x_t^1 \\ y_t^1 \end{bmatrix}}_{o_t^1}$ for given s_t. In (17), the noise term $\mathbf{n}_t^2$ is added to perturb the action of Agent 2 for exploration around the given term $\begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix} \underbrace{\begin{bmatrix} x_t^2 \\ y_t^2 \end{bmatrix}}_{o_t^2}$ for given s_t. Note that two perturbation terms $\mathbf{n}_t^1$ and $\mathbf{n}_t^2$ are independent. Hence, these two terms induce independent exploration for Agents 1 and 2. That is, without the z_t-induced terms in (16) and (17), a_t^1 and a_t^2 given $s_t \ni (o_t^1, o_t^2)$ are independent since in this case only the noise terms $\mathbf{n}_t^1$ and $\mathbf{n}_t^2$ remain and the noise terms are independent

random variables by assumption. However, with the z_t-induced terms in (16) and (17), a_t^1 and a_t^2 are correlated and the corresponding covariance matrix is given by

$$\mathbb{E}[(a_t^1 - \mathbb{E}[a_t^1])(a_t^2 - \mathbb{E}[a_t^2])^T | s_t] = \mathbf{W}^1 \mathbf{C}_z (\mathbf{W}^2)^T, \tag{18}$$

where a_t^1 and a_t^2 are column vectors as shown in (16) and (17); $(\cdot)^T$ denotes matrix transpose; $\mathbf{C}_z$ is the covariance matrix of $z_t = \begin{bmatrix} z_t^x \\ z_t^y \end{bmatrix}$ determined by p_Z;

$$\mathbf{W}^1 = \begin{bmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{bmatrix} \quad \text{and} \quad \mathbf{W}^2 = \begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \end{bmatrix}.$$

Note that the perturbation structure of $\mathbf{W}^1 z_t$ and $\mathbf{W}^2 z_t$ is different from that of $\mathbf{n}_t^1$ and $\mathbf{n}_t^2$. Indeed, we are injecting correlated random perturbation into a_t^1 and a_t^2 to promote correlation exploration to better explore the joint state-action space. By properly designing $\mathbf{C}_z$ (i.e., properly designing P_Z), $\mathbf{W}^1$ and $\mathbf{W}^2$, we can impose an arbitrary correlation structure between a_t^1 and a_t^2 conditioned on s_t.

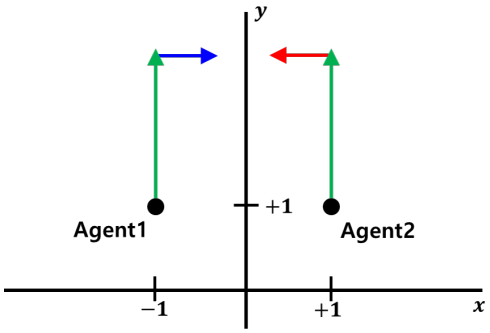


Figure 8. An example

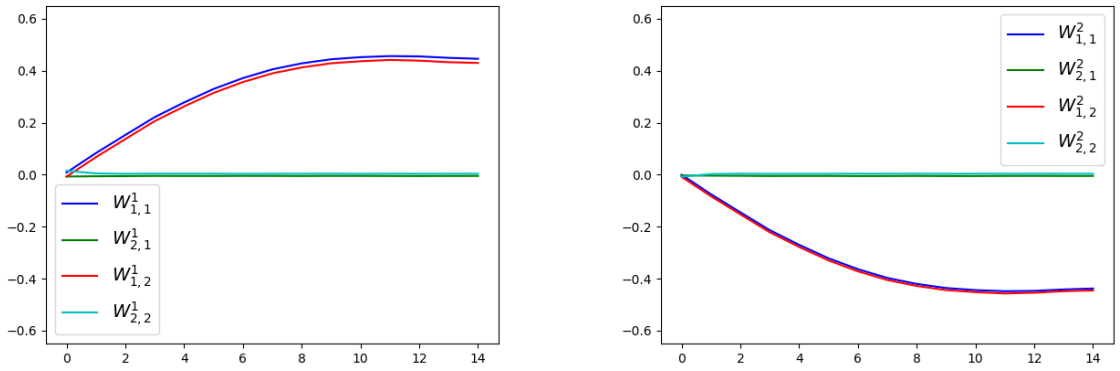
Now, consider the following joint task. The initial location of Agent 1 is (-1,1) and the initial location of Agent 2 is (1,1). The joint goal is that the two agents meet while going upward, and an episode ends when the two agents meet, as described in Fig. 8. Suppose that we pick the prior distribution $P_Z(z)$ for z_t as

$$z_t^x \sim \text{Unif}[0, 0.1] \tag{19}$$
$$z_t^y \sim \text{Unif}[0, 0.1], \tag{20}$$

where $\text{Unif}[a, b]$ means the uniform distribution over interval $[a, b]$. Now, we design the reward $r_t(s_t, a_t^1, a_t^2)$ as the distance between the two agents' locations. We use the policies of the two agents given by (16) and (17).

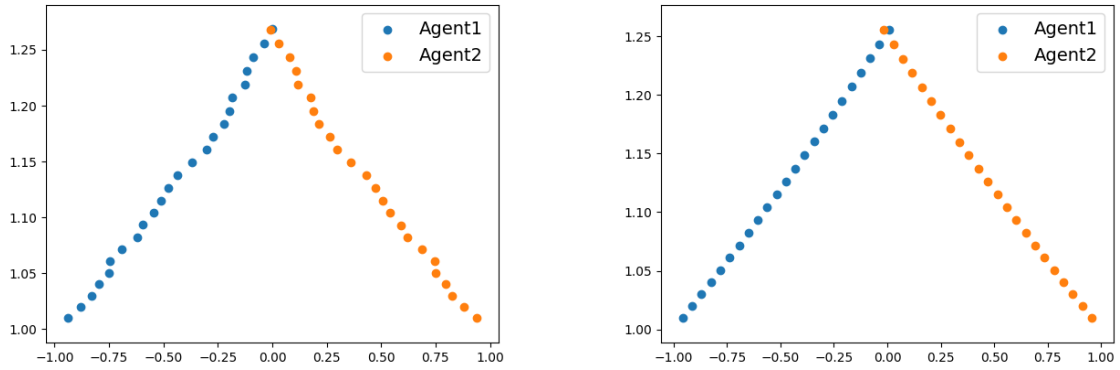
With this setup, we simply learned the policy parameters $\mathbf{W}^1$ and $\mathbf{W}^2$ associated with z_t by greedily maximizing the instantaneous reward r_t by stochastic gradient descent with Adam optimizer with learning rate 3×10^{-4}. The parameter learning curves of $\mathbf{W}^1$ and $\mathbf{W}^2$ are shown in Fig. 9. It is observed that w_{11}^1 and w_{12}^1 of Agent 1 converge to positive values, whereas w_{21}^1 and w_{22}^1 of Agent 1 converge to zero. This setting of parameters w_{11}^1, w_{12}^1, w_{21}^1 and w_{22}^1 of Agent 1 generates movement of Agent 1 to the right side since $z_t^x \geq 0$ and $z_t^y \geq 0$ due to (19) and (20). (Please see (16).) On the other hand, w_{11}^2 and w_{12}^2 of Agent 2 converge to negative values, whereas w_{21}^2 and w_{22}^2 of Agent 2 converge to zero. This setting of parameters w_{11}^2, w_{12}^2, w_{21}^2 and w_{22}^2 of Agent 2 generates movement of Agent 2 to the left side since $z_t^x \geq 0$ and $z_t^y \geq 0$ due to (19) and (20). (Please see (17).) Hence, the two agents meet. Note that the desired coordination between Agents 1 and 2 can be achieved by injecting common random variable z_t and learning the set of parameters $\mathbf{W}^1$ and $\mathbf{W}^2$ associated with z_t properly.

Fig. 10 shows the trajectories of Agents 1 and 2 for an episode in the execution phase after training. Fig. 10(a) shows the trajectory when we input the random variable z_t with distribution (19) and (20) to the policy network just as we did in the training phase. Fig. 10(b) shows the trajectory when we input $\mathbb{E}[z_t] = (0.05, 0.05)$ to the policy network for all t in the execution phase after training. The desired action is still obtained in the case of Fig. 10(b). This is because the



(a) (b)

Figure 9. (a) the parameter values of $\mathbf{W}^1$ associated with z_t for Agent 1 and (b) the parameter values of $\mathbf{W}^2$ associated with z_t for Agent 2 (during the training phase)



(a) (b)

Figure 10. The trajectory of two agents in the execution phase after training: (a) $z_t^x \sim \text{Unif}[0, 0.1]$, $z_t^y \sim \text{Unif}[0, 0.1]$) and (b) $z_t^x = 0.05$, $z_t^y = 0.05$

parameters $\mathbf{W}^1$ and $\mathbf{W}^2$ associated with z_t are properly learned during the training phase by enhanced exploration of the joint state-action space based on correlated exploration due to $\mathbf{W}^1 z_t$ and $\mathbf{W}^2 z_t$ with z_t random. This learned parameters are used in the execution phase. We can view that by setting $z_t = \mathbb{E}[z_t]$, we pick and apply the representative joint bias on actions. Note that the desired joint bias in the case of Fig. 10(b) is obtained because of the fact that z_t is distributed over $[0, 0.1]$ by (19) and (20). Hence, the choice of P_Z is important in this method. However, at least the shift of the support of z_t is not a big concern when a general neural network is used as the policy function. In the case of a general neural network as the policy function, shift of z_t is automatically done by the node bias of the neural network and this node bias is also learned as parameter.

In this example, we observe that coordination of actions and coordinated exploration are feasible by injecting a common random variable z_t to the input of every policy function and learning the parameters associated with z_t. In this example, we fixed the weights associated with the observation o_t^i to show the exploration and control capability of the z_t part. In general cases, we have the freedom to design the weights associated with the observation o_t^i too. Designing the conventional policy parameters associated with the observation together with additional degree-of-freedom for exploration and design generated by injecting z_t combined with nonlinear deep neural network can lead to learning of complicated coordinated behavior via correlated exploration. This paper fully develops this idea.

Appendix B: Proofs

In the main paper, we defined the state and state-action value functions for Agent i as follows:

$$Q_i^{\boldsymbol{\pi}}(s,a) \triangleq \mathbb{E}_{\substack{\tau_0 \sim \boldsymbol{\pi} \\ z_t \sim p_Z}} \left[r_0 + \gamma V_i^{\boldsymbol{\pi}}(s_1) \middle| s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a} \right], \tag{21}$$

$$V_i^{\boldsymbol{\pi}}(s) \triangleq \mathbb{E}_{\substack{\tau_0 \sim \boldsymbol{\pi} \\ z_t \sim p_Z}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_t + \beta H(a_t^i|s_t) + \frac{\beta}{N} \sum_{j \neq i} \log q^{(i,j)}(a_t^i, a_t^j|s_t) \Big) \middle| s_0 = s \right], \tag{22}$$

Then, the Bellman operator corresponding to $V_i^{\boldsymbol{\pi}}$ and $Q_i^{\boldsymbol{\pi}}$ on the value function estimates $V_i(s)$ and $Q_i(s,\boldsymbol{a})$ is given by

$$\mathcal{T}^{\boldsymbol{\pi}} Q_i(s,\boldsymbol{a}) \triangleq r(s,\boldsymbol{a}) + \gamma \mathbb{E}_{s' \sim p}[V_i(s')], \tag{23}$$

where

$$V_i(s) = \mathbb{E}_{\substack{\boldsymbol{a} \sim \boldsymbol{\pi} \\ z_t \sim p_Z}} \left[Q_i(s,\boldsymbol{a}) - \beta \log \tilde{\pi}^i(a^i|s) + \frac{\beta}{N} \sum_{j \neq i} \log q^{(i,j)}(a^i, a^j|s) \right]. \tag{24}$$

(21), (21), (23) and (24) are the rewritings of equations Equations (12), (13), (14) and (15) in the main paper.

Proposition 1 (Variational Policy Evaluation). For fixed $\boldsymbol{\pi}$ and the variational distribution q, consider the modified Bellman operator $\mathcal{T}^{\boldsymbol{\pi}}$ in (23) and an arbitrary initial function $Q_i^{(0)} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and define $Q_i^{(k+1)} = \mathcal{T}^{\boldsymbol{\pi}} Q_i^{(k)}$. Then, $Q_i^{(k)}$ converges to $Q_i^{\boldsymbol{\pi}}$ defined in (21).

Proof. From (23), we have

$$\begin{aligned}
\mathcal{T}^{\pi} Q_i(s_t, \boldsymbol{a_t}) &= r(s_t, \boldsymbol{a_t}) + \gamma \mathbb{E}_{\substack{s_{t+1} \sim p, \boldsymbol{a_{t+1}} \sim \boldsymbol{\pi} \\ z_{t+1} \sim p_Z}} \Big[Q_i(s_{t+1}, \boldsymbol{a_{t+1}}) - \beta \log \tilde{\pi}^i(a_t^i|s_t) \\
&\qquad + \frac{\beta}{N} \sum_{j \neq i} \log q^{(i,j)}(a_t^i, a_t^j|s_t) \Big] \qquad (25) \\
&= \underbrace{r(s_t, \boldsymbol{a_t}) + \gamma \mathbb{E}_{\substack{s_{t+1} \sim p, \boldsymbol{a_{t+1}} \sim \boldsymbol{\pi} \\ z_{t+1} \sim p_Z}} \left[- \beta \log \tilde{\pi}^i(a_t^i|s_t) + \frac{\beta}{N} \sum_{j \neq i} \log q^{(i,j)}(a_t^i, a_t^j|s_t) \right]}_{=: r_\pi(s_t, \boldsymbol{a_t})} \\
&\quad + \gamma \mathbb{E}_{\substack{s_{t+1} \sim p, \boldsymbol{a_{t+1}} \sim \boldsymbol{\pi} \\ z_{t+1} \sim p_Z}} \left[Q_i(s_{t+1}, \boldsymbol{a_{t+1}}) \right] \qquad (26) \\
&= r_\pi(s_t, \boldsymbol{a_t}) + \gamma \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t, a_t), z_{t+1} \sim p_Z, \boldsymbol{a_{t+1}} \sim \boldsymbol{\pi}(\cdot|s_{t+1}, z_{t+1})} \left[Q_i(s_{t+1}, \boldsymbol{a_{t+1}}) \right], \qquad (27)
\end{aligned}$$

where in the last line the expectation arguments are explicitly shown without abbreviation for clarity. Then, we can apply the standard convergence results for policy evaluation. Define

$$\mathcal{T}^{\pi}(v) = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v \tag{28}$$

for $v = [Q(s,\boldsymbol{a})]_{s \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}}$. Then, the operator $\mathcal{T}^{\pi}$ is a γ-contraction.

$$\begin{aligned}
\|\mathcal{T}^{\pi}(v) - \mathcal{T}^{\pi}(u)\|_\infty &= \|(\mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v) - (\mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} u)\|_\infty &(29)\\
&= \|\gamma \mathcal{P}^{\pi}(v - u)\|_\infty &(30)\\
&\leq \|\gamma \mathcal{P}^{\pi}\|_\infty \|v - u\|_\infty &(31)\\
&\leq \gamma \|u - v\|_\infty &(32)
\end{aligned}$$

since $||\mathcal{P}^{\boldsymbol{\pi}}||_\infty \leq 1$ Therefore, the operator $\mathcal{T}^\pi$ has a unique fixed point by the contraction mapping theorem. Let $Q_i^\pi(s, \boldsymbol{a})$ be this fixed point. Since

$$\|Q_i^{(k)}(s, \boldsymbol{a}) - Q_i^\pi(s, \boldsymbol{a})\|_\infty \leq \gamma \|Q_i^{(k-1)}(s, \boldsymbol{a}) - Q_i^\pi(s, \boldsymbol{a})\|_\infty \leq \cdots \leq \gamma^k \|Q_i^{(0)}(s, \boldsymbol{a}) - Q_i^\pi(s, \boldsymbol{a})\|_\infty, \tag{33}$$

we have

$$\lim_{k\to\infty} \|Q_i^{(k)}(s, \boldsymbol{a}) - Q_i^\pi(s, \boldsymbol{a})\|_\infty = 0 \tag{34}$$

and this implies

$$\lim_{k\to\infty} Q_i^{(k)}(s, \boldsymbol{a}) = Q_i^\pi(s, \boldsymbol{a}), \quad \forall (s, \boldsymbol{a}) \in (\mathcal{S} \times \boldsymbol{\mathcal{A}}). \tag{35}$$

□

We proved the variational policy evaluation in a finite state-action space. We can expand the result to the case of an infinite state-action space by assuming the followings:

- Assume that Q functions for π are in L infinity
- From (Folland, 1999), L infinity is a Banach space
- From (Agarwal et al., 2018), by the Banach fixed point theorem, Q function should converge to a unique point in L infinity space and that is the Q function of given π

Proposition 2 (Variational Policy Improvement). Let π_{new}^i and q_{new} be the updated policy and the variational distribution from (36). Then, $Q_i^{\pi_{new}^i, \pi_{old}^{-i}}(s, \boldsymbol{a}) \geq Q_i^{\pi_{old}^i, \pi_{old}^{-i}}(s, \boldsymbol{a})$ for all $(s, \boldsymbol{a}) \in (\mathcal{S} \times \boldsymbol{\mathcal{A}})$. $(\pi_{k+1}^i, q_{k+1}) =$

$$\arg\max_{\pi^i, q} \mathbb{E}_{\substack{(a^i, a^{-i}) \sim (\pi^i, \pi_k^{-i}) \\ z_k \sim p_Z}} \left[Q_i^{\boldsymbol{\pi}_k}(s, \boldsymbol{a}) - \beta \log \tilde{\pi}^i(a^i|s) + \frac{\beta}{N} \sum_{j \neq i} \log q^{(i,j)}(a^i, a^j|s) \right]. \tag{36}$$

Proof. Let us rewrite (36) to clarify that which terms are given and which terms are the optimization arguments. We use the subscript "old" for the given terms. Then, π_{new} is updated as $(\pi_{new}^i, q_{new}) =$

$$\arg\max_{\pi^i, q} \mathbb{E}_{\substack{(a^i, a^{-i}) \sim (\pi^i, \pi_{old}^{-i}) \\ z_k \sim p_Z}} \left[Q_i^{\boldsymbol{\pi}_{old}}(s_t, \boldsymbol{a}_t) - \beta \log \tilde{\pi}^i(a_t^i|s_t) + \frac{\beta}{N} \sum_{j \neq i} \log q^{(i,j)}(a_t^i, a_t^j|s_t)) \right]. \tag{37}$$

Then, the following inequality is hold

$$\mathbb{E}_{\substack{(a_t^i, a_t^{-i}) \sim (\pi_{new}^i, \pi_{old}^{-i}) \\ z_k \sim P_Z}} \left[Q_i^{\boldsymbol{\pi}_{old}}(s_t, \boldsymbol{a}_t) - \beta \log \tilde{\pi}_{new}^i(a_t^i|s_t) + \frac{\beta}{N} \sum_{j \neq i} \log q_{new}^{(i,j)}(a_t^i, a_t^j|s_t)) \right] \tag{38}$$

$$\geq \mathbb{E}_{\substack{(a_t^i, a_t^{-i}) \sim (\pi_{old}^i, \pi_{old}^{-i}) \\ z_k \sim p_Z}} \left[Q_i^{\boldsymbol{\pi}_{old}}(s_t, \boldsymbol{a}_t) - \beta \log \tilde{\pi}_{old}^i(a_t^i|s_t) + \frac{\beta}{N} \sum_{j \neq i} \log q_{old}^{(i,j)}(a_t^i, a_t^j|s_t)) \right] \tag{39}$$

$$= V_i^{\boldsymbol{\pi}_{old}}(s_t). \tag{40}$$

From the definition of the Bellman operator,

$$Q_i^{\boldsymbol{\pi}_{old}}(s_t, \boldsymbol{a_t}) = r(s_t, \boldsymbol{a_t}) + \gamma \mathbb{E}_{s_{t+1}\sim p}[V_i^{\boldsymbol{\pi}_{old}}(s_{t+1})] \tag{41}$$

$$\leq r(s_t, \boldsymbol{a_t}) + \gamma \mathbb{E}_{s_{t+1}\sim p} \mathbb{E}_{\substack{(a^i_{t+1}, a^{-i}_{t+1}) \sim (\pi^i_{new}, \pi^{-i}_{old}) \\ z_{t+1} \sim p_Z}} \Bigg[Q_i^{\boldsymbol{\pi}_{old}}(s_{t+1}, \boldsymbol{a}_{t+1}) - \beta \log \tilde{\pi}^i_{new}(a^i_{t+1}|s_{t+1}) + \frac{\beta}{N} \sum_{j\neq i} \log q^{(i,j)}_{new}(a^i_{t+1}, a^j_{t+1}|s_{t+1}) \Bigg] \tag{42}$$

$$\leq r(s_t, \boldsymbol{a_t}) + \gamma \mathbb{E}_{s_{t+1}\sim p} \mathbb{E}_{\substack{(a^i_{t+1}, a^{-i}_{t+1}) \sim (\pi^i_{new}, \pi^{-i}_{old}) \\ z_{t+1} \sim p_Z}} \Bigg[r(s_{t+1}, \boldsymbol{a_{t+1}}) - \beta \log \tilde{\pi}^i_{new}(a^i_{t+1}|s_{t+1}) + \frac{\beta}{N} \sum_{j\neq i} \log q^{(i,j)}_{new}(a^i_{t+1}, a^j_{t+1}|s_{t+1}) + \gamma \mathbb{E}_{s_{t+2}\sim p} [V_i^{\boldsymbol{\pi}_{old}}(s_{t+2})] \Bigg] \tag{43}$$

$$\vdots$$

$$\leq Q_i^{\pi^i_{new}, \pi^{-i}_{old}}(s_t, a_t). \tag{44}$$

□

Appendix C: Details of Centralized Training

The value functions $V^i_{\psi_i}(\boldsymbol{x})$, $Q^i_{\theta_i}(\boldsymbol{x}, \boldsymbol{a})$ are updated based on the modified Bellman operator defined in (13) and (14). The state-value function $V^i_{\psi_i}(\boldsymbol{x})$ is trained to minimize the following loss function:

$$\mathcal{L}_V(\psi^i) = \mathbb{E}_{s_t \sim D}\left[\frac{1}{2}(V^i_{\psi^i}(\boldsymbol{x}_t) - \hat{V}^i_{\psi^i}(\boldsymbol{x}_t))^2\right] \tag{45}$$

where D is the replay buffer that stores the transitions $(\boldsymbol{x}_t, \boldsymbol{a}_t, r_t, \boldsymbol{x}_{t+1})$; $Q^i_{min}(\boldsymbol{x}_t, a^i_t) = \min[Q^i_{\theta^{i,1}}(\boldsymbol{x}_t, a^i_t), Q^i_{\theta^{i,2}}(\boldsymbol{x}_t, a^i_t)]$ is the minimum of the two action-value functions to prevent the overestimation problem (Fujimoto et al., 2018); and

$$\hat{V}^i_{\psi^i}(\boldsymbol{x}_t) = \mathbb{E}_{z_t \sim N(0,\boldsymbol{I}), \{a^k \sim \pi^k(\cdot|o^k_t, z_t)\}_{k=1}^N}\Bigg[Q^i_{min}(\boldsymbol{x}_t, \boldsymbol{a}_t) - \beta \log \pi^i_{\phi^i}(a^i_t|o^i_t, z_t) + \frac{\beta}{N}\sum_{j\neq i} \log q^{(i,j)}_{\xi^i}(a^i_t, a^j_t|o^i_t, o^j_t)\Bigg]. \tag{46}$$

Note that in the second term of the RHS of (46), originally we should have used the marginalized version, $-\beta \log \tilde{\pi}^i_{\phi^i}(a^i_t|o^i_t) = -\beta \log \mathbb{E}_{z_t \sim N(0,\boldsymbol{I})}[\pi^i_{\phi^i}(a^i_t|o^i_t, z_t)]$. However, for simplicity of computation, we took the expectation $\mathbb{E}_{z_t \sim N(0,\boldsymbol{I})}$ outside the logarithm. Hence, there exists Jensen's inequality type approximation error. We observe that this approximation works well.

The two action-value functions are updated by minimizing the loss

$$\mathcal{L}_Q(\theta^i) = \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{a}_t) \sim D}\left[\frac{1}{2}(Q_{\theta^i}(\boldsymbol{x}_t, \boldsymbol{a}_t) - \hat{Q}(\boldsymbol{x}_t, \boldsymbol{a}_t))^2\right] \tag{47}$$

where

$$\hat{Q}(\boldsymbol{x}_t, \boldsymbol{a}_t) = r_t(x_t, \boldsymbol{a}_t) + \gamma \mathbb{E}_{\boldsymbol{x}_{t+1}}[V_{\overline{\psi}^i}(x_{t+1})] \tag{48}$$

and $V_{\overline{\psi}^i}$ is the target value network, which is updated by the exponential moving average method. We implement the reparameterization trick to estimate the stochastic gradient of policy loss. Then, the action of agent i is given by $a^i = f_{\phi^i}(s; \epsilon^i, z)$, where $\epsilon^i \sim \mathcal{N}(0, \boldsymbol{I})$ and $z \sim \mathcal{N}(0, \boldsymbol{I})$. The policy for Agent i and the variational distribution are trained to minimize the following policy improvement loss,

$$\mathcal{L}_{\pi^i,q}(\phi^i, \xi) = \mathbb{E}_{\substack{s_t \sim D, \\ \epsilon^i \sim \mathcal{N}, \\ z \sim \mathcal{N}}}\Bigg[-Q^i_{\theta^{i,1}}(\boldsymbol{x}_t, \boldsymbol{a}) + \beta \log \pi^i_{\phi^i}(a^i|o^i_t, z) - \frac{\beta}{N}\sum_{j\neq i} \log q^{(i,j)}_{\xi^i}(\pi^i_{\phi^i}(a^i|o^i_t, z), \pi^j_{\phi^j}(a^j|o^j_t, z)|o^i_t, o^j_t)\Bigg] \tag{49}$$

where $q^{(i,j)}_{\xi^i}(\pi^i_{\phi^i}(a^i|o^i_t, z), \pi^j_{\phi^j}(a^j|o^j_t, z)|o^i_t, o^j_t)$

$$= \underbrace{q_{\xi^i}(\pi^i_{\phi^i}(a^i|o^i_t, z)|\pi^j_{\phi^j}(a^j|o^j_t, z)|o^i_t, o^j_t)}_{(a)}\underbrace{q_{\xi^i}(\pi^j_{\phi^j}(a^j|o^j_t, z)|\pi^i_{\phi^i}(a^i|o^i_t, z)|o^i_t, o^j_t)}_{(b)}. \tag{50}$$

Again, for simplicity of computation, we took the expectation $\mathbb{E}_{z_t \sim N(0,\boldsymbol{I})}$ outside the logarithm for the second term in the RHS in (49). Since approximation of the variational distribution is not accurate in the early stage of training and the learning via the term (a) in (50) is more susceptible to approximation error, we propagate the gradient only through the term (b) in (50) to make learning stable. Note that minimizing $-\log q_{\xi^i}(a^j|a^i, s_t)$ is equivalent to minimizing the mean-squared error between a^j and $\mu_{\xi^i}(a^i, o^i, o^j)$ due to our Gaussian assumption on the variational distribution.

Appendix D: Pseudo Code

Algorithm 1 VM3-AC (L=1)

Centralized training phase
Initialize parameter $\phi^i, \theta^i, \psi^i, \overline{\psi}^i, \xi^i, \ \forall i \in \{1, \cdots, N\}$
for $episode = 1, 2, \cdots$ **do**
  Initialize state s_0 and each agent observes o_0^i
  for $t < T$ and $s_t \neq$ terminal **do**
    Generate $z_t \sim \mathcal{N}(0, I)$ and select action $a_t^i \sim \pi^i(\cdot|o_t^i, z_t)$ for each agent i
    Execute $\boldsymbol{a_t}$ and each agent i receives r_t and o_{t+1}^i
    Store transitions in D
  end for
  for each gradient step **do**
    Sample a minibatch from D and generate $z_l \sim \mathcal{N}(0, I)$ for each transition.
    Update θ^i, ψ^i by minimizing the loss (47) and (48)
    Update ϕ^i, ξ^i by minimizing the loss (49)
  end for
  Update $\overline{\psi}^i$ using the moving average method
end for

Decentralized execution phase
Initialize state s_0 and each agent observes o_0^i
for each environment step **do**
  Select action $a_t^i \sim \pi^i(\cdot|o_t^i, z_t)$ where $z_t = \overrightarrow{0}$ (or sample from the Gaussian random sequence generator with the same seed)
  Execute $\boldsymbol{a_t}$ and each agent i receives o_{t+1}^i
end for

Appendix E: Hyperparameter and Training Detail

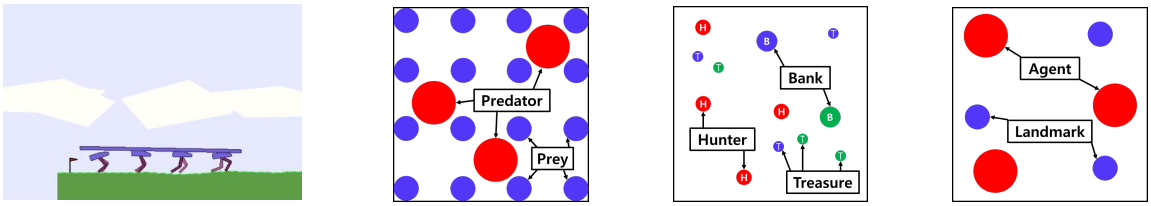
The hyperparameters for MA-AC, MA-SAC, MADDPG, and VM3-AC are summarized in Table 1.

Table 1. Hyperparameters of all algorithms

	MA-AC	SI-MOA	MAVEN	MADDPG	VM3-AC
REPLAY BUFFER SIZE	5×10^5	5×10^5	5×10^5	5×10^5	5×10^5
DISCOUNT FACTOR	0.99	0.99	0.99	0.99	0.99
MINI-BATCH SIZE	128	128	128	128	128
OPTIMIZER	ADAM	ADAM	ADAM	ADAM	ADAM
LEARNING RATE	0.0003	0.0003	0.0003	0.0003	0.0003
TARGET SMOOTHING COEFFICIENT	0.005	0.005	0.005	0.005	0.005
NUMBER OF HIDDEN LAYERS (ALL NETWORKS)	2	2	2	2	2
NUMBER OF HIDDEN UNITS PER LAYER	128	128	128	128	128
ACTIVATION FUNCTION FOR HIDDEN LAYER	RELU	RELU	RELU	RELU	RELU
ACTIVATION FUNCTION FOR FINAL LAYER	TANH	TANH	TANH	TANH	TANH

Table 2. The temperature parameter β and the dimension of the latent variable z for VM3-AC on the considered environments. Note that the temperature parameter β in I-SAC and MA-SAC controls the relative importance between the reward and the entropy, whereas the temperature parameter β in VM3-AC controls the relative importance between the reward and the mutual information.

VM3-AC	β	DIM(Z)
MW (N=3)	0.05	8
MW (N=4)	0.1	8
PP (N=2)	0.15	8
PP (N=3)	0.1	8
PP (N=4)	0.2	8
CTC (N=4)	0.05	10
CTC (N=5)	0.05	10
CN (N=3)	0.1	8



(a) (b) (c) (d)

Figure 11. Considered environments: (a) Multi-Walker, (b) Predator-Prey, (c) Cooperative Treasure Collection, and (d) Cooperative Navigation

Appendix F: Environment Detail

We implemented our algorithm based on OpenAI Spinning Up (Achiam, 2018) and conduct the experiments on a server with Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz. Each experiment took about 12 to 24 hours. We illustrate the considered environments in Fig. 11.

Multi-walker The multi-walker environment, which was introduced in (Gupta et al., 2017), is a modified version of the BipedalWalker environment in OpenAI gym to multi-agent setting. The environment consists of N bipedal walkers and a large package. The goal of the environment is to move forward together while holding the large package on top of the walkers. The observation of each agent consists of the joint angular speed, the position of joints. Each agent has 4-dimensional continuous actions that control the torque of their legs. Each agent receives shared reward R_1 depending on the distance over which the package has moved and receives negative local compensation R_2 if the agent drops the package or falls to the ground. An episode ends when one of the agents falls, the package is dropped or T time steps elapse. To obtain higher rewards, the agents should learn coordinated behavior. For example, if one agent only tries to learn to move forward, ignoring other agents, then other agents may fall. In addition, the different coordinated behavior is required as the number of agents changes. We set $T = 500$, $R_2 = -10$ and $R_1 = 10d$, where d is the distance over which the package has moved. We simulated this environment in three cases by changing the number of agents ($N = 2$, $N = 3$, and $N = 4$).

All algorithms used neural networks to approximate the required functions. We used the neural network architecture proposed in (Kim et al., 2019) to emphasize the agent's own observation and action for centralized critics. For Agent i, we used the shared neural network for the variational distribution $q_{\xi^i}(a_t^j|a_t^i, o_t^i, o_t^j)$ for $j \in \{1, \cdots, N\}\backslash\{i\}$, and the network takes the one-hot vector which indicates j as input.

Predator-prey The predator-prey environment, which is a standard task for MARL, consists of N predators and M preys. We used a variant of the predator-prey environment into the continuous domain. The initial positions on the predators are randomly determined, and those of the preys are in the shape of a square lattice. The goal of the environment is to capture as many preys as possible during a given time T. A prey is captured when C predators catch the prey simultaneously. The predators get team reward R_1 when they catch a prey. After all of the preys are captured and removed, we set the preys to respawn in the same position and increase the value of R_1. Thus, the different coordinated behavior is needed as N and C change. The observation of each agent consists of relative positions between agents and other agents and those between agents and the preys. Thus, each agent can access to all information of the environment state. The action of each agent is two-dimensional physical action. We set $R_1 = 10$ and $T = 100$. We simulated the environment with three cases: $(N = 2, M = 16, C = 1)$, $(N = 3, M = 16, C = 1)$ and $(N = 4, M = 16, C = 2)$.

Cooperative treasure collection The cooperative treasure collection environment, which was introduced in (Iqbal & Sha, 2019), consists of 2 banks, $N - 2$ collectors, and 6 hunters. Each bank has a different color and each treasure has one of the banks' colors. The goal of this environment is to deposit the treasures by controlling the banks and hunters. The hunters collect the treasure and then give it to the corresponding bank. Both hunters and banks receive shared reward R_1 if a treasure is deposited. The hunters receive a positive reward R_2 when a treasure is collected and a negative reward $-R_3$ if colliding with other agents. The observation of each agent consists of the locations of all other agents and landmarks, and action is two-dimensional physical action. We set $R_1 = 5$, $R_2 = 5$, $R_3 = 5$. We simulated the environment with two cases: $(N = 4)$ and $(N = 5)$.

Cooperative navigation Cooperative navigation, which was proposed in (Lowe et al., 2017), consists of N agents and L landmarks. The goal of this environment is to occupy all landmarks while avoiding collision with other agents. The agent

receives shared reward R_1 which is the sum of the minimum distance of the landmarks from any agents, and the agents who collide each other receive negative reward $-R_2$. In addition, all agents receive R_3 if all landmarks are occupied. The observation of each agent consists of the locations of all other agents and landmarks, and action is two-dimensional physical action. We set $R_2 = 10$, $R_3 = 1$, and $T = 50$. We simulated the environment in the cases of ($N = 3$, $L = 3$).

Appendix G: SMAC environment

We modified the SMAC environment to be sparse to make the problem more difficult. The considered sparse reward setting consists of a time-penalty reward which is obtained -0.1 every time step and a dead reward which is obtained $+10$ and -1 when one enemy dies and one ally dies, respectively. If all enemies die, the dead reward is given $+200$.

We implemented VM3-AC by modifying the code provided by (Zhang et al., 2021). We replace the entropy term in (Zhang et al., 2021) with the sum of entropy and variational approximation. We used the categorical distribution with the dimension of 3 for the latent variable. We used the deep neural network which consists of a 64-dimensional MLP with ReLU activation function, GRU, and an MLP to parameterize the policies. In addition, we use an MLP with 2 hidden layers which have 64 hidden units, and a ReLU activation function for both the critic networks. For the variational approximation, $q(a^j|a^i, s)$, we use the deep neural network which takes Agent i's action and outputs Agent j's action. The variational approximation is a feed-forward network whose weight is the output of a hyper-network which is a deep neural network taking the global state as input. The hyper-network is implemented similar to the mixing network in QMIX (Rashid et al., 2018).

As in (Zhang et al., 2021), we annealed the temperature parameter from 0.5 to 0.05 over 2×10^5 steps. We provided source code in the supplementary material.

Appendix H: Replacing the latent variable with mean vector

Injecting mean vector $\mathbb{E}\{z_t\}$ to the z_t-input of policy network $\pi^i_{\phi^i}(\cdot|o^i_t, z_t)$ during the execution phase: As mentioned in the main paper, we applied the mean vector of z_t, i.e., $\mathbb{E}\{z_t\}$ to the z_t-input of the policy deep neural network $\pi^i_{\phi^i}(\cdot|o^i_t, z_t)$ during the execution phase so as to execute actions without communication in the execution phase. We compared the performance of decentralized policies that use the mean vector $\mathbb{E}\{z_t\}$ and decentralized policies which use the latent variable z_t assuming communication. We used deterministic evaluation based on 20 episodes generated by the corresponding deterministic policy, i.e., each agent selects action using the mean network of Gaussian policy $\pi^i_{\phi^i}$. We averaged the return over 7 seeds, and the result is shown in Table 3. It is seen that the mean vector replacement method yields almost the same performance and enables fully decentralized execution without noticeable performance loss. Please see Appendix A for intuition.

Table 3. Impact of replacing the latent variable $z_t \sim \mathcal{N}(0, \mathbf{I})$ with mean vector $z_t = \mathbb{E}(z_t)$ in the execution phase

	PP (N=2)	PP (N=3)	PP (N=4)
$z_t \sim \mathcal{N}(0, \mathbf{I})$	413	734	1123
$z_t = \mathbb{E}(z_t)$	409	743	1147