PAC: Assisted Value Factorisation with Counterfactual Predictions in Multi-Agent Reinforcement Learning

Hanhan Zhou
The George Washington University
hanhan@gwu.edu

Tian Lan
The George Washington University
tlan@gwu.edu

Vaneet Aggarwal
Purdue University
vaneet@purdue.edu

Abstract

Multi-agent reinforcement learning (MARL) has witnessed significant progress with the development of value function factorization methods. It allows optimizing a joint action-value function through the maximization of factorized per-agent utilities. In this paper, we show that in partially observable MARL problems, an agent's ordering over its own actions could impose concurrent constraints (across different states) on the representable function class, causing significant estimation error during training. We tackle this limitation and propose PAC, a new framework leveraging Assistive information generated from Counterfactual Predictions of optimal joint action selection, which enable explicit assistance to value function factorization through a novel counterfactual loss. A variational inference-based information encoding method is developed to collect and encode the counterfactual predictions from an estimated baseline. To enable decentralized execution, we also derive factorized per-agent policies inspired by a maximum-entropy MARL framework. We evaluate the proposed PAC on multi-agent predator-prey and a set of StarCraft II micromanagement tasks. Empirical results demonstrate improved results of PAC over state-of-the-art value-based and policy-based multi-agent reinforcement learning algorithms on all benchmarks.

1 Introduction

Many real-world reinforcement learning (RL) problems, such as autonomous vehicle coordination [1] and network packet delivery [2], often involve coordination among multiple entities and are naturally formulated as multi-agent reinforcement learning (MARL). Factorization-based methods have achieved great progress in dealing with the exponentially growing joint state-action space in MARL. Under the notion of Centralized Training and Decentralized Execution (CTDE), algorithms like VDN [3] and QMIX [4] learn a centralized joint action-value function Q_{tot} through a monotonic factorisation into local per-agent value functions, so that Q_{tot} can be maximized as long as each per-agent value function is maximized by local action selection. Even conditioned on joint state information, a monotonic mixing network for Q_{tot} is shown to restrict the representable function class. Despite efforts to mitigate this, e.g., QTRAN [5] and WQMIX [6], in practice they empirically perform poorly in complex MARL environments with partial observability [7, 8, 9].

In this paper, we show that in partially observable MARL problems (as exemplified by a multi-state matrix game), an agent's ordering over its own actions could impose concurrent constraints on the representable action-value Q_{tot} in different states. This restriction causes large estimation errors of Q_{tot} during training and cannot be addressed by existing methods, e.g., adding a state-value correction

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

term (like QTRAN [5]) or introducing importance weights on dominant state-actions (like WQMIX [6]). It aggravates the relative over-generalization problem [10] – when fully decomposed, per-agent value functions only depend on partial observations and local actions. It renders optimal decentralized policies unlearnable when the employed value function does not have enough representational ability. Solving tasks that require significant coordination in partially observable problems remains as a key challenge.

The key insight of this paper is that an accurate factorization in partially observable MARL problems requires improved representation of the value functions, which is crucial to supporting the learning of optimal decentralized policies. We propose a novel architecture, denoted as PAC, for assisted value factorization with counterfactual predictions. It leverages a counterfactual baseline that marginalizes out an agent's potential optimal action while keeping all other agents' actions fixed. The counterfactual predictions of potential optimal actions enable (i) training assistive information that is generated using the variational inference method to expand the representational ability of value functions, and (ii) directly optimizing the factorization of Q_{tot} through a new counterfactual loss. We note that in contrast to communication-based methods like NDQ [11], the use of assistive information in PAC aims to directly improve per-agent value functions in factorization. Minimizing our proposed counterfactual loss together with an information bottleneck loss ensures that such assistive information is relevant, optimal, and succinct for value function factorization in PAC.

Relying on the accuracy of assisted factorization, we further decouple the decision-making of local value function (through separate policy networks) from value function networks, which allows PAC to maintain full CTDE despite the use of assistive information during training and also enables the maximization of entropy to encourage exploration. The accuracy of assisted factorization makes PAC outperform policy-based methods with factorization like DOP [12] and FOP [13]), especially on difficult tasks that require more coordination among the agents since sub-optimality from one agent's policy might propagate and aggravate the training of other agents through the centralized critic [12, 14]. Our key contributions are summarized as follows:

- We propose a novel method, PAC, the first framework for value function factorization by providing variational-encoded counterfactual predictions as assistive information to facilitate per-agent value function estimation.
- The counterfactual predictions can be efficiently computed from a feed-forward baseline Q^* and based on a local search to facilitate direct optimization of Q_{tot} factorization through a new counterfactual loss.
- PAC decouples individual agents' policy networks from value function networks and thus maintains fully decentralized execution while enjoying the benefits of assisted value function factorization. It also leads to an entropy maximization MARL for more effective exploration.
- We demonstrate the effectiveness of PAC and show that PAC significantly outperforms both state-of-the-art value-based and policy-based multi-agent reinforcement learning algorithms on the StarCraft II micromanagement challenge in terms of better performance and faster convergence.

2 Model and Background

Model: Consider a fully cooperative multi-agent task as decentralized partially observable Markov decision process (DEC-POMDP) [15], given by a tuple $G = \langle I, S, U, P, r, Z, O, n, \gamma \rangle$, where $I \equiv \{1, 2, \cdots, n\}$ is the finite set of agents. The state is given as $s \in S$, from which each agent draws its own observation from the observation function $o_i \in O(s, i) : S \times A \rightarrow O$. At each timestamp t, each agent i choose an action $u_i \in U$, composing a joint action selection $\boldsymbol{u}$. A shared reward is then given as $r{=}R(s, \boldsymbol{a}) : S \times \mathbf{U} \rightarrow \mathbb{R}$, with the next state of each agent is s' with transition probability function $P(s'|s, \mathbf{u}) : S \times U \rightarrow [0, 1]$. Each agent has an action-observation history $\tau_i \in \mathrm{T} \equiv (O \times U)^*$ from its limited local observations. Then the overall objective is to find a joint policy $\boldsymbol{\pi} = \langle \pi_1, ..., \pi_n \rangle$ which corresponds to the joint action-value function $Q(s_t, \boldsymbol{u}_t) = \mathbb{E}[R_t|s_t, \boldsymbol{u}_t]$, that is used to maximize the joint policy function $V(\boldsymbol{\tau}, \boldsymbol{u}) = \mathbb{E}_{s_{0:\infty}, \boldsymbol{u}_{0:\infty}} \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \boldsymbol{u}_0 = \boldsymbol{u}, \boldsymbol{\pi}\right]$, and $\gamma \in [0, 1)$ is the discount factor. Quantities in bold denote a joint quantities across all agents, and quantity with super script i denote a quantity specifically belong to agent i.

Value Decomposition: Following the paradigm of CTDE, VDN [3] and QMIX [4] are popular and representative methods for value function decomposition which learn a centralized action value $Q\mathrm{tot}$ through value decomposition assuming its additivity and monotonicity. In VDN, per-agent sum of the local value is used to calculate the action value $Q^{\mathrm{tot}} = \sum_{a=1}^{n} Q_a(\tau_a, u_a)$. In

QMIX, a monotonic mixing function of each agents' local utilities is proposed as $Q^{\text{tot}}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) = f_\theta\left(\boldsymbol{s}, [Q_1(\tau_1, a_1), ..., Q_n(\tau_n, a_n)]\right)$, where $\frac{\partial Q^{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{u})}{\partial Q_i(\tau_i, u_i)} > 0, \forall i \in \mathcal{N}$. The monotonic mixing function is able to ensure the global optimal Q^{tot} yields the same result as the set of individual optimal Q_i, where f_θ is approximated by the monotonic mixing network parameterized by θ. The weights are generated by a separate hyper-network that conditions on the global state, where its monotonicity is ensured by performing absolution function on generated parameters for non-negative mixing weights. Then QMIX is trained in a way like DQN [3] with goal to minimise the squared TD error on minibatch of b samples from the replay buffer as $\sum_{i=1}^{b}(Q^{\text{tot}}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) - y^{\text{tot}})^2$, where $y^{\text{tot}} = r + \gamma \max_{u'} Q^{\text{tot}}(\boldsymbol{\tau'}, \mathbf{u}', s'; \theta')$, r is the global reward and θ' is the parameters of the target network whose parameters are periodically copied from θ for training stabilization.

3 Existing Limitations in Partially Observable Multi-state Problems

State 1

u_1 \ u_2	**A**	B	C
A	**4**	-2	-2
B	-2	0	0
C	-2	0	0

State 2

u_1 \ u_2	**A**	B	C
A	-2	0	0
B	**4**	-2	-2
C	-2	0	0

(a) Payoff of 2-state matrix game

State 1

Q_1 \ Q_2	**0.3**	-1.2	-2.6
-0.2	0.1	-1.0	-1.0
0.3	**1.1**	-0.9	-1.0
-2.6	-1.0	-1.0	-1.0

State 2

Q_1 \ Q_2	**0.4**	-1.3	-2.2
-0.2	0.4	-1.0	-1.0
0.3	**1.4**	-0.9	-1.0
-2.6	-1.0	-1.0	-1.0

(b) QMIX: Q_1, Q_2, Q_{tot}

State 1

Q_1 \ Q_2	**1.0**	0.2	0.2
1.0	**4.0**	-1.2	-1.2
0.0	-0.2	-1.6	-1.6
0.0	-0.2	-1.6	-1.6

State 2

Q_1 \ Q_2	0.0	0.5	**0.5**
1.0	-1.2	0.1	**0.1**
0.0	-1.6	-1.2	-1.2
0.0	-1.6	-1.2	-1.2

(c) WQMIX: Q_1, Q_2, Q_{tot}

State 1

Q_1 \ Q_2	**0.7**	-2.0	-2.3
0.7	**4.0**	-2.1	-2.1
-2.0	-2.1	-2.4	-2.4
-2.1	-2.1	-2.4	-2.4

State 2

Q_1 \ Q_2	**0.6**	-2.1	-2.0
-1.2	-1.8	-2.5	-2.5
1.6	**4.0**	-2.0	-2.0
-1.8	-2.1	-2.5	-2.5

(d) PAC (ours): Q_1, Q_2, Q_{tot}

Figure 1: In partially observable multi-state games, Q_{tot} is limited to a restricted function class imposing concurrent inequality constraints on Q_{tot} in different states. It causes large factorisation error and thus erroneous computation of $\operatorname{argmax}(Q_{tot})$. PAC successfully addresses this problem by leveraging assistive information trained using counterfactual predictions, with the direct goal of achieving better value function factorisation.

The limitations of monotonic value function factorization have been identified via single-state matrix games, which inspired new algorithms like QTRAN [5] and Weighted-QMIX [6]. In this paper, we show in multi-state problems with partial observability, that one agent's ordering over its own actions could impose concurrent inequality constraints on the joint action-value Q_{tot} in different states, resulting in restrictive function representations of Q_{tot} with large estimate error.

Consider an Markov decision process (MDP) consisting of 2 states with 0.5 transition probabilities between them and two payoff matrices shown in Figure 1(a). Suppose that agent 1 has the same partial observation o_1 in states $s^{(1)}$ and $s^{(2)}$. Then, its per-agent value function $q_1(\cdot, \tau_1)$ computed from partial observation o_1 are also the same in both states. Due to the monotonicity of the mixing network (even though it is provided with complete joint state information), for any u_1 and u_1' with ordering $q_1(u_1, \tau_1) \geq q_1(u_1', \tau_1)$ without loss of generality, we must simultaneously have $Q_{tot}(u_1, u_2, s^{(1)}) \geq Q_{tot}(u_1', u_2, s^{(1)})$ and $Q_{tot}(u_1, u_2, s^{(2)}) \geq Q_{tot}(u_1', u_2, s^{(2)})$ for any action u_2 of agent 2 in both states. Representing Q_{tot} on this restricted function class results in significant error in QMIX as shown in Figure 1(b). Although Weighted-QMIX introduces a importance weighting on the dominant state-actions of this game, it only improves the approximation in state 1 and yet causes even higher error in state 2 (in Figure 1(c)). This is exactly because of the the inequality restrictions simultaneously imposed on both states, limiting the representational ability of the value functions in partially observable problems.

Clearly, additional information is needed to facilitate successful factorization in these partially observable multi-state problems. It is also worth noticing that even with agent-wise communications, NDQ [11] also fails the task since the communication messages and related loss functions are not designed to drive better factorization. It underscores the importance of making effective use of the right information for successful factorization. We put the results from other methods in Appendix A.3. Our proposed PAC (in Figure 1(d)) addresses this issue by leveraging assistive information trained by a novel notion of counterfactual predictions. More precisely, counterfactual predictions of potentially

optimal agent actions are readily computed from an unrestricted, feed-forward baseline Q^*. Training per-agent value functions using a new counterfactual assistance loss leads to substantially improved estimate Q_{tot} and correct $(\operatorname{argmax} Q_{tot})$ in different states. We compare PAC with a number of state-of-the-art value-based and policy-based methods with function factorizations and demonstrate its performance in challenging partially observable tasks that require significant coordination.

4 Methods

To overcome the limitations of existing Value Decomposition methods discussed in Sec. 3, in this section, we introduce the idea of assisted optimal joint policy factorization and propose such a method under the multi-agent soft actor-critic framework.

Framework overview. Fig. 2 shows the architecture of the learning framework. There are three main components in PAC : (1) a weighted Q^{tot} utilizing per-agent local critics $q_i(u_i, \tau_i, m_i)$, where m_i serves as assistive information aiding value factorization, (2) an unrestricted joint action estimator $\hat{Q}^*$, which serves as a baseline estimator of Q^* and allows the computation of counterfactual predictions from a quick local search, and (3) an assisted information generating module, which is able to utilize the deep variational bottleneck method to encode the counterfactual optimal joint action selection. In addition to TD-errors for Q_{tot} and Q^*, the PAC system is trained using three more loss functions: counterfactual assistance loss L_{CA} for optimized value function factorization, information bottleneck loss L_{LB} for succinct and effective assistive information generation, and local policy loss L_{LP} for training factorized agent policy with entropy maximization.

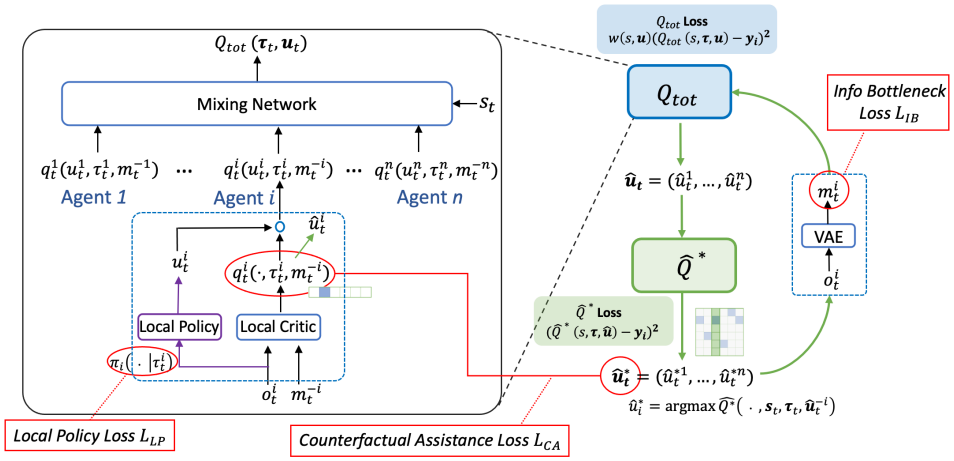


Figure 2: The overall architecture of PAC. With the help of assistive information m_t^{-i}, counterfactual predictions $\hat{u}_t^*$ – which are obtained through an approximation of optimal Q^* and an efficient local search – are used to directly train the per-agent value functions $q_t^i(\cdot, \tau_t^i, m_t^{-i})$ with respect to a new counterfactual assistance loss. It results in significantly improved value function factorisation for partially-observable MARL problems.

Generating Counterfactual Predictions One of the key insights underlying this method is that the optimal joint action selection u^* from the centralised $\hat{Q}^*$ can be used as a direct information assisting Q_{tot} explicitly. Although the complexity of computing $\mathbf{u}^* = \operatorname{argmax}_u \mathbb{E}[Q^*(s, \tau, \cdot)]$ is impractical as it grows in $O(|U|^n)$, however, it is possible to compute an local estimation of $\hat{\mathbf{u}}^*$ as $\hat{u}_i^* = \operatorname{argmax} \hat{Q}^*(s, \boldsymbol{\tau}, \boldsymbol{u}_{-i}, \cdot)$, which reduces the computational complexity from exponential to linear level of $O(n \cdot |U|)$.

However, even with $\mathbf{u}^*$ provided, in what manner it can benefit value function factorization is not researched. Directly feeding this to the policy or critic network is counterproductive as the neural network can easily learn that this specific input is the key to reducing the TD-error while it is not helpful to explore or train the local policy and the training process might stall. Using an extra loss function, e.g. cross-entropy, as an effort to train the network to make decisions u that are close to $\mathbf{u}^*$

sounds promising, however, it only quantifies the error without actually guiding the training since it may trap the policy in local sub-optimum and further limits the exploration process.

Inspired by difference rewards [16] and counterfactual baseline [17] for policy gradients, we propose a counterfactual assistance loss. For each agent a we can use an advantage function that compares the $\hat{u}^*$ from Q^* and u from Q^{tot} to a counterfactual baseline that marginalizes out the agent's potential optimal action which relegates $\hat{u}_a^*$, while keeping all other agents' action $\boldsymbol{u}^{-a}$ fixed, a counterfactual assistance loss is proposed as:

$$\mathcal{L}_{CA}(\mathbf{u}, \pi) = \sum_a \log\pi(u_i^a|\tau_i^a) \sum_{u_i \in [\boldsymbol{u} - u_i^*]} [q(\hat{u}_i^*, o_i, m_i) - \pi(u_i, o_i) q(u_i, o_i, m_i)] \tag{1}$$

This looks similar to calculating the counterfactual advantage as used in COMA, however, in COMA such counterfactual advantage is calculated for centralized critic which is computationally expensive and unstable which limits its performance, while our counterfactual assistance loss is providing direct guidance in training the policy network towards the direction of u_i^*. We later show in ablation studies that this loss is contributing to the performance improvements.

Generating Assistive Information Additionally, we show $\hat{\boldsymbol{u}}^*$ can be encoded and then used as input for local critics $q_i(u_i, \tau_i, m_i)$, as assistive information aiding value factorization. As previous works suggests [11, 18], we consider the information bottleneck method [19], with the Markov chain **o-m-$\hat{\mathbf{u}}^*$** during encoding. To be specific, we regard the internal representation of intermediate layer as stochastic encoding m_i of input source o_i, with the goal to learn an encoding that is maximally informative about the result $\hat{u}_i^*$. Formally, the objective for each agent i can be written as:

$$J_{IB}(\boldsymbol{\theta}_m) = \sum_{j=1}^{n} \left[I_{\theta_m}\left(\hat{u}_j^*; m_i | \mathrm{o}_j, m_{-j}\right) - \beta I_{\boldsymbol{\theta}_m}\left(m_i; o_i\right) \right] \tag{2}$$

m_i is an instance of random variable of assitive information that is drawn from multivariate Gaussian distribution $\boldsymbol{N}(f_m(o_i; \theta_m), \boldsymbol{I}))$, θ_m is the parameters of encoder f_m and $\boldsymbol{I}$ is an identity matrix, a Lagrange multiplier parameter $\beta \geq 0$ is used to control the trade-off between the encoding the necessary information and reaching maximal compression. Yet this does not lead to a learnable model, since this mutual information I is intractable. With the help of variational approximation, specifically, deep variational information bottleneck [20], we are able to parameterize this model using a neural network. We then derive and optimize a variational lower bound of such objective as

$$\mathcal{L}_{IB}(\boldsymbol{\theta}_m) = \mathbb{E}_{o_i \sim \mathcal{D}, m_j \sim f_m}[-\mathcal{H}[p(\hat{u}_j^*|\mathbf{o}), q_\psi(\hat{u}_j^*|\mathrm{o}_j, \boldsymbol{m})] + \beta D_{\mathrm{KL}}(p(m_i|\mathrm{o}_i) \| q_\phi(m_i))] \tag{3}$$

where $\mathcal{H}$ is the entropy operator, D_{KL} denotes Kullback-Leibler divergence operator and $q_\phi(m_i)$ is a variational posterior estimator of $p(m_i)$ with parameters ϕ. Using the loss above a message encoder f_m with parameters θ_m is trained to generate information $m_i \sim \boldsymbol{N}(f_m(o_i; \theta_m), \boldsymbol{I}))$ that is useful for decision making. Compared to [11, 18] that encodes the general state information and other agents' action selections as communication messages, which can not reduce the uncertainties in action-value functions; using the encoded $\hat{u}^*$ as assistance information can provide an explicit direction toward better individual value estimation and thus a joint value factorization. Detailed derivations and proofs can be found in Appendix A.1.

Factorized Policy Iteration with Entropy Maximization We leverage factorized policies to maintain decentralized execution in PAC despite the use of assistive information for training. Recent works have shown that Boltzmann exploration policy iteration is guaranteed to improve the policy and converge to optimal with unlimited iterations and full policy evaluation[21], within MARL domain it can be defined as: $J(\pi) = \sum_t \mathbb{E}\left[r\left(\mathbf{s}_t, \mathbf{u}_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot|\mathbf{s}_t\right)\right)\right]$ where α denotes the temperature parameter that is used to adjust the balance between maximizing the entropy for a better exploration and maximizing the expected reward. We present one possible method of expanding this to MARL problems, to achieve decentralized policies in PAC and to encourage efficient exploration.

Several recent works are proposed to expand actor-critic or soft-actor-critic in to factorization based MARL methods [13, 12, 14], they all follow a centralized critic with decentralized actors (CCDA) framework. In this work, we train the actors in a centralized but factorized way. Unlike [14] that reuses the local utility network for both actor or critic or [13] which consists of a soft V-network and a soft Q-network for local policy net, we use a separate network as policy networks and propose a

centralized but factorized soft policy iteration with factorized Maximum-Entropy trained as:

$$\begin{aligned}\mathcal{L}_{LP}(\pi) &= \mathbb{E}_{\mathcal{D}}\left[\boldsymbol{\alpha}\log\boldsymbol{\pi}\left(\boldsymbol{u}_t|\boldsymbol{\tau}_t\right) - Q^{\pi}_{tot}\left(\boldsymbol{s_t},\boldsymbol{\tau_t},\boldsymbol{u}_t,\boldsymbol{m}_t\right)\right] \\ &= -q^{\text{mix}}\left(\boldsymbol{s}_t, \mathbb{E}_{\pi^i}\left[q^i\left(\tau^i_t, u^i_t, m^i_t\right) - \alpha^i\log\pi^i\left(u^i_t|\tau^i_t\right)\right]\right)\end{aligned} \tag{4}$$

where q^{mix} is the monotonic value decomposition network with $u_i \sim \pi_i(o_i)$ and $\mathcal{D}$ is the replay buffer used to sample training data.

As noted in previous research, choosing the temperature term in soft-actor-critic is non-trivial as it can vary in an unpredictable way when the policy becomes better as the training continues [13]. Instead of finding each individual temperature using approximation functions, since we use a centralized but factored policy, we consider one global temperature that is automatically adjusted by considering it as a constrained optimization problem as a parameter that is trained independently with loss:

$$\mathcal{J}(\alpha) = \mathbb{E}_{u_t\sim\pi_t}[-\alpha^i\log\pi_i(u_t|s_t) - \boldsymbol{\alpha}\mathcal{H}_0], \tag{5}$$

where $\mathcal{H}_0$ is a fixed entropy term so the temperature term α is generally decreasing such that the degree of exploration is reduced as the training proceeds[21].

Training Process So far we have discussed the components of our method, to formulate the new scalable RL algorithm, we now explain the implementation and the centralized training process for deep RL under DEC-POMDP. In the decentralized execution phase, only the policy (agent) network (marked as local policy in Fig.1) is enabled so that full CTDE is maintained.

Despite that monotonicity constraints limitations to the expressive power of the mixing network, recent research [14, 22] demonstrated that the IGM principle [5] as equivalent to joint greed actions and individual greedy actions is crucial, as it greatly promotes the sample efficiency; meanwhile most designs with only one unrestricted joint action-value function show poor empirical performance[5, 23], and thus we follow the design of WQMIX and keep both Q_{tot} network and $\hat{Q}^*$ network.

The $\hat{Q}^*$ architecture (Green part in Fig. 1) is used as the estimator for Q^* from unrestricted functions, where its mixing network is a feed-forward network that takes its local utilities. While the Q_{tot} module (blue part in Fig.1) looks similar to QMIX and many other factorized methods, there is a significant difference, for its local estimator is provided with assistive information and the local value feeding to the network $q(u, \tau_i, m_i)$ is selected based on local policy, rather than taking $\operatorname{argmax} q_i(\cdot, \tau_i, m_i)$. Then Q_{tot} and Q^* are trained with the objective to reduce their respective loss as:

$$\mathcal{L}_{\hat{Q}^*}(\theta) = \sum_i (\hat{Q}^*(s, \boldsymbol{\tau}, \hat{\boldsymbol{u}}) - y_i)^2. \tag{6}$$

$$\mathcal{L}_{Q_{tot}}(\theta) = \sum_i w(s, \mathbf{u})(Q_{tot}(s, \boldsymbol{\tau}, \mathbf{u}, \boldsymbol{m}) - y_i)^2 \tag{7}$$

where $\hat{u}_i = \operatorname{argmax} q_i(\cdot, \tau_i, m_i)$, $y_i = r + \gamma\hat{Q}^*(s', \boldsymbol{\tau}', \operatorname{argmax}_{\hat{\mathbf{u}}'} Q_{tot}(\boldsymbol{\tau}', \hat{\mathbf{u}}', s'; \boldsymbol{\theta}))$ with $\boldsymbol{\theta}^-$ being the parameters of the target network that are periodically updated to stabilize the training. $u_i \sim \pi_i(o_i)$ and $w(s, \mathbf{u})$ is the weighting function[1] to ensure the weighted projection can recover the correct maximal joint action value function for any Q[6]. Note the action selection for loss function in our method is different from the original design of QMIX and WQMIX. Apart from the independently updated entropy term α, all other components (including the message encoder) are trained in a end-to-end manner with objective to minimize the weighted sum of all losses proposed above, including the counterfactual loss and information bottleneck loss as

$$\mathcal{L}(\theta) = \mathcal{L}_{LP} + \mathcal{L}_{CA} + \mathcal{L}_{IB} + \mathcal{L}_{\hat{Q}^*} + \mathcal{L}_{Q_{tot}} \tag{8}$$

Detailed derivations can be found in Appendix A.1

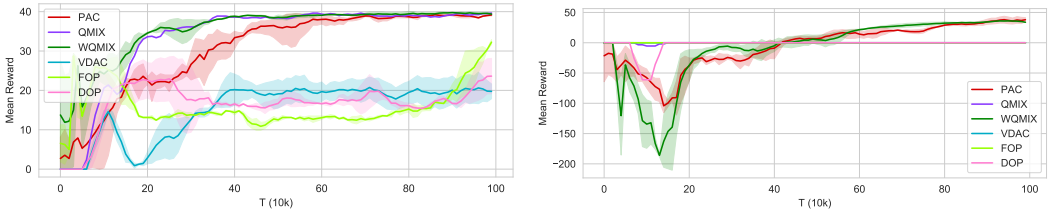
5 Experiments

In this section, we compare the results with several state-of-the-art MARL methods on Predator-Prey [24] and selected StarCraft Multi-Agent Challenge (SMAC) [8] scenarios as benchmarks. More details on implementation, experiment settings, and hyperparameters are included in Appendix A.3. Code is available at github.com/hanhanAnderson/PAC-MARL.

[1] In this work we follow the weighting function design of OW-QMIX in WQMIX [6], as w = 1 if $Q_{tot}(s, \boldsymbol{\tau}, \mathbf{u}, \boldsymbol{m}) - y_i$ <0, w = α otherwise.

5.1 Cooperative Predator-Prey

We first consider Predator-Prey [24], which is a partially-observable multi-agent environment that involves 8 agents in cooperation as predators to catch 8 built-in-AI controlled units as prey on a 10x10 grid. Only when two or more predator agents surround and capture the same prey at the same time, it is considered a successful capture. We consider two settings: a simpler task without failed capture punishment and a harder task with a punishment reward of -2 when a capture attempt fails. This is to test the baseline algorithms on relative over-generalization and monotonicity constraints limitations. More details about this environment can be found at Appendix A.3.2.



(a) PP without mis-capture penalty

(b) PP with mis-capture penalty

Figure 3: Results on Predator-Prey benchmark

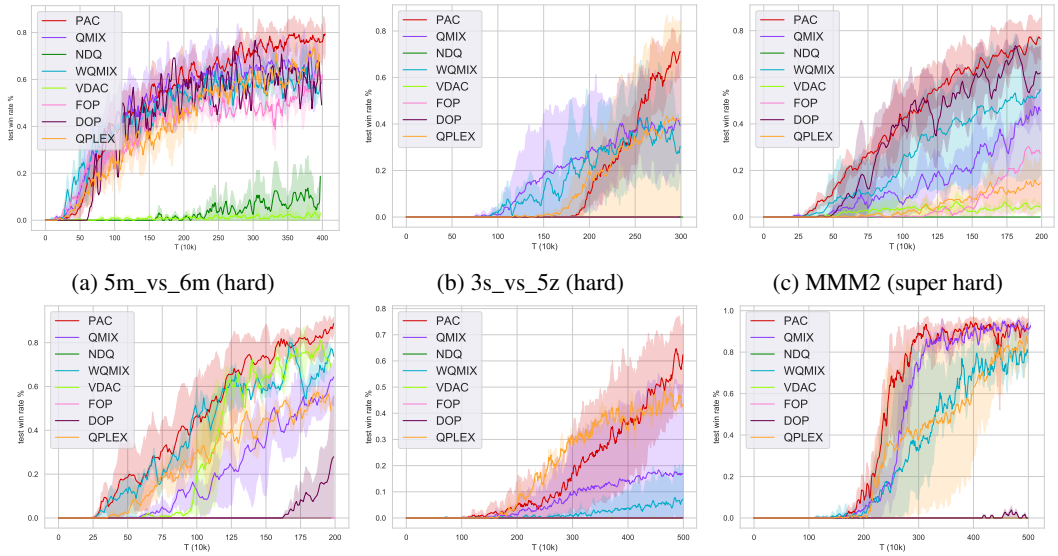
As shown in Fig. 3a, for the easy task value-based method like QMIX, WQMIX can learn a policy that reaches the highest reward. However, in this environment, we see the performance gap between state-of-the-art value-based and policy-based methods since policy-based methods might suffer from relative over-generalization problem [9, 25, 26] or making a poor trade-off in joint policy representation. Our work as an actor-critic method can match the highest performance, although it takes a slightly longer time to converge, potentially due to the training of the variational inference models.

On the other hand, for the harder task that requires significant coordination among agents as shown in Fig. 3b, increased exploration and joint representation capabilities are required to finish the task which makes WQMIX and PAC the only two methods capable of learning a usable policy. Intuitively, when joint actions from uncoordinated decision-making occur more than coordinated ones, the penalty term will then dominate the average return from the environment and further the value estimation of each agent's local utility. We attribute these performance improvements to the use of entropy maximization which promotes more exploration attempts and the higher coordination abilities from assistive information.

5.2 SMAC

We then consider the SMAC [2] as the second benchmark, wherein each agent controls a unit cooperating with other friendly units in combat against the game's built-in AI-controlled units. The combat can be symmetric (same units for both parties) or asymmetric. Since it is shown that most state-of-the-art algorithms perform really well on easy and medium maps, which limits the demonstration of clear comparison and the potential improvements, we begin our test in six maps including two hard maps (`5m_vs_6m, 3s_vs_5z`) and four super-hard maps (`MMM2, 27m_vs_30m, 6h_vs_8z, corridor`). Selected maps are classified as hard or super hard due to (i) very large action space like `27m-vs-30m`, (ii) require advanced exploration strategies like corridor (iii) require a higher level of coordination between the agents like `6h_vs_8z` etc. We use the same default environment setting for all benchmark algorithms throughout the test. Each baseline algorithm is trained with 4 random seeds and evaluated every 10k training steps with 32 testing episodes. Details of the environment setup and hyper-parameter settings are listed in Appendix A.3.3. We choose state-of-the-art MARL algorithms as baseline including decomposed actor critic method: FOP [13] and DOP [12], decomposed policy gradient method: VDAC [14], decomposed value based method: QMIX [4], QPLEX [9] and WQMIX

[2] In this paper all SMAC experiments are carried out utilizing the latest `SC2.4.10`, performance is always not comparable across versions. We implemented our algorithm based on a open-sourced codebase [22] and acquired the results of QMIX and WQMIX from it.



(a) 5m_vs_6m (hard) (b) 3s_vs_5z (hard) (c) MMM2 (super hard)

(d) 27m_vs_30m (super hard) (e) 6h_vs_8z (super hard) (f) corridor (super hard)

Figure 4: Results on SMAC benchmark

[6] [3] and a communication enabled value-based method: NDQ [11]. Results are shown in Fig.4. Overall, our method achieves the highest win rate compared to the baseline algorithms in terms of higher performance or faster convergence, especially on those maps that require more exploration and agents' coordination. As previous research demonstrated, there is a gap between SOTA value-based method and policy gradients methods, as the performance for most of them is limited on those maps that require extensive exploration techniques. Specifically, on `3s_vs_5z` and `6h_vs_8z`, PAC is able to train a usable policy that outperforms all baseline algorithms. On `corridor` PAC and the selected two value-based methods are able to learn a model, with our method converging faster with slightly better performance, while policy-based methods suffer from this map as it requires more exploration to find the specific trick in winning this challenging scenario. Finally on relatively easier maps, although most baseline algorithms hold a relatively close performance, the value-based and policy-based method performance gap still exists. Although FOP recently claimed to be the first multi-agent actor-critic method that outperforms state-of-the-art value-based methods on SMAC, we empirically found its limits when the chosen environment is substantially complicated and harder. We especially observe that our method as an actor-critic method has over-performed SOTA value-based MARL methods and brought significant improvements for actor-critic MARL.

5.3 Ablation Studies

We conduct ablation experiments to validate the effectiveness and contribution of each core components introduced in PAC on `MMM2` scenario in SMAC. Namely, in Fig.5 we consider to verify the effect of (1) optimization of temperature term in policy $\mathcal{J}(\alpha)$ by fixing $\alpha = 0.5$ as *PAC_fixed*α, (2) assisted information loss $\mathcal{L}_{IB}$ by disable it as *PAC_no_info*, (3) counterfactual assistance loss $\mathcal{L}_{CA}$ by replace it with a simple cross-entropy loss as *PAC_CE_Loss*, (4) disable both $\mathcal{L}_{IB}$ and $\mathcal{L}_{CA}$ while substituting all $\hat{u}^*$ with $\hat{u}$ as *PAC_disabled* and (5) further remove the Q^* structure from *PAC_disabled* as *PAC_No_Q**. In this way, overall we can observe PAC outperforms ablated versions, especially by a large margin compared to *PAC_disabled* which indicates that a general encoding of the state infor-

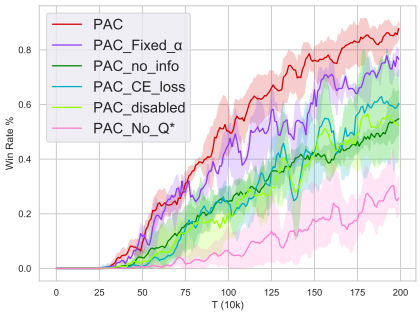


Figure 5: Ablations results comparing PAC and its hyperparameter-tuned ablated versions on SMAC map `MMM2`

[3] In this section we refer WQMIX to ow-qmix as it shows a general better performance than cw-qmix.

mation without explicit direction for training is not helpful
and may also harm the training.Although *PAC_CE_ Loss* later acquires competitive results, it takes a longer time, potentially due to the cross-entropy loss in this case trapped the training in a local sub-optimum. By fixing the entropy term α at 0.5 seems to be a balance point for maximizing the reward while promoting the exploration, yet it brings a performance drop, indicating the importance of updating the entropy term with its loss previously shown. *PAC_No_Q** suffers from most significant performance drop after most core components removed from the original design. Such results validate how each component is crucial for achieving a performance through experiments.

5.4 Limitations

Although the empirical results of PAC demonstrated improvements over both SOTA value-based and policy-based methods, at this stage, no strict convergence guarantee to the optimum is provided due to the scope of research. We also observe a higher variance in terms of the performance of PAC than value-based methods, this might be because the entropy-maximization penalty. Also, we follow the tradition of parameter sharing among the agents, thus the role assignment of formulating distinctive behaving agents, which is another important topic was not considered.

6 Related Works

Cooperative multi-agent decision-making often suffers from exponential joint state and action spaces. Multiple approaches including independent Q-learning and mean-field games have been considered in the literature, while they do not perform well in challenging tasks or require homogeneous agents [14]. A paradigm of centralized training and decentralized execution (CTDE) has been proposed for scalable decision making [27]. QPLEX [9] takes a duplex dueling network architecture to factorize the joint value function. Some of the key CTDE approaches include value function decomposition and multi-agent policy gradient methods.

Policy Gradient methods are considered to have more stable convergence compared to value-based methods [28, 29, 13] and can be extended to continuous action problems easily. A representative multi-agent policy gradient method is COMA [17], which utilizes a centralized critic module for estimating the counterfactual advantage of an individual agent. DOP [12] uses a factorized policy gradients with archteture similar to Qatten[30]. However, as pointed out in [31, 23], multi-agent policy-based methods like MADDPG[32] are still outperformed by value-based methods StarCraft multi-agent challenge (SMAC) [8].

Decomposed actor-critic methods, which combine value function decomposition and policy gradient methods with the use of the decomposed critics rather than centralized critics, are introduced to guide policy gradients. VDAC [14] combined the structure of actor-critic and QMIX for the joint state-value function estimation, while DOP [33] directly uses a network similar to Qatten [30] for policy gradients with off-policy tree backup and on-policy TD. The authors of [33] pointed out that decomposed critics are limited by restricted expressive capability and thus cannot guarantee the convergence of global optima; even though the individual policies may converge to local optima [13]. Extensions of the monotonic mixing function have also been considered, e.g., QTRAN [5] and weighted QMIX [6]. But solving tasks that require significant coordination remains a key challenge.

Another related topic is representational learning in reinforcement learning. A VAE-based forward model is proposed in [34] to learn the state representations in the environment. [35] considers a model to learn Gaussian embedding representations of different tasks during meta-testing. The authors in [36] proposed a recurrent VAE model which encodes the observation and action history and learns a variational distribution of the task. NDQ [11] encodes the state information as communication messages between agents. [37] use an inference model to represent the decision-making of the opponents. RODE [38] uses an action encoder to learn the action representations in restricting the role action spaces for a reduced policy search space. MAR [39] learns the metarepresentation for generalization problems. Unlike previous work, our method focuses on learning information from counterfactual predictions that are explicitly assistive for local estimation and efficient factorization.

7 Conclusions

In this paper, we propose PAC, a multi-agent framework utilizing extra state information as assistance for a better value function factorization, under a centralized but factored soft-actor critic setting. With the newly proposed counterfactual optimal joint action selection used for training and encoding assistance information, empirically we show our method not only matches or outperforms both state-of-the-art policy-based and value-based MARL algorithms on selected benchmarks but also bridges the gap between the two. Future work will also explore more effective ways to formulate and utilize extra state information to accelerate the training and tackle tasks in more complicated environments.

References

[1] Yeping Hu, Alireza Nakhaei, Masayoshi Tomizuka, and Kikuo Fujimura. Interaction-aware decision making with adaptive strategies under merging scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–158. IEEE, 2019.

[2] Dayong Ye, Minjie Zhang, and Yun Yang. A multi-agent framework for packet routing in wireless sensor networks. *sensors*, 15(5):10026–10047, 2015.

[3] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[4] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[5] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896. PMLR, 2019.

[6] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning, 2020.

[7] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *arXiv preprint arXiv:1910.07483*, 2019.

[8] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[9] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[10] Jacopo Castellini, Frans A Oliehoek, Rahul Savani, and Shimon Whiteson. The representational capacity of action-value networks for multi-agent reinforcement learning. *arXiv preprint arXiv:1902.07497*, 2019.

[11] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. *arXiv preprint arXiv:1910.05366*, 2019.

[12] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. Dop: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*, 2020.

[13] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 12491–12500. PMLR, 2021.

[14] Jianyu Su, Stephen Adams, and Peter Beling. Value-decomposition multi-agent actor-critics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11352–11360, 2021.

[15] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[16] Scott Proper and Kagan Tumer. Modeling difference rewards for multiagent learning. In *AAMAS*, pages 1397–1398, 2012.

[17] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[18] Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. Value functions factorization with latent state information sharing in decentralized multi-agent policy gradients. *arXiv preprint arXiv:2201.01247*, 2022.

[19] Naftali Tishby, Fernando Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[20] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[22] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. Riit: Rethinking the importance of implementation tricks in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*, 2021.

[23] Kyunghwan Son, Sungsoo Ahn, Roben Delos Reyes, Jinwoo Shin, and Yung Yi. Qtran++: Improved value transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2006.12010*, 2020.

[24] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pages 980–991. PMLR, 2020.

[25] Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Böhmer, and Shimon Whiteson. Uneven: Universal value exploration for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 3930–3941. PMLR, 2021.

[26] Liviu Panait, Sean Luke, and R Paul Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, 2006.

[27] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[28] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[29] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34, 2021.

[30] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.

[31] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Comparative evaluation of multi-agent deep reinforcement learning algorithms. *arXiv preprint arXiv:2006.07869*, 2020.

[32] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[33] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. Off-policy multi-agent decomposed policy gradients. *arXiv preprint arXiv:2007.12322*, 2020.

[34] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[35] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR, 2018.

[36] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pages 2117–2126. PMLR, 2018.

[37] Georgios Papoudakis and Stefano V Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*, 2020.

[38] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020.

[39] Shenao Zhang, Li Shen, and Lei Han. Learning meta representations for agents in multi-agent reinforcement learning. *arXiv preprint arXiv:2108.12988*, 2021.

[40] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In appendix
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All code, sample data and running instructions are provided in supplementary files for reproducing the results
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All training details are provided in Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We use random seed and all algorithms are evaluated in 4 different runs
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The details of our experiment settings are provided in Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We use Apache License for our code
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We use and modify open-sourced code under Apache License
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Our data does not contain such contents.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Mathematical Details

A.1.1 Boundaries for counterfactual-prediction information

Following the introduction of the variational information bottleneck, we explain it as follows. Consider a Markov chain of $o - \hat{u*} - m$, (substituting the $X - Y - Z$ in the original IB and VIB), regarding the hidden representation of encoding $\hat{u*}$ of the input o, the goal of learning an encoding is to maximize the information about target m measured by the mutual information between encoding and the target $I(\hat{u*}; m)$.

To prevent the encoding of data from being $m = \hat{u*}$, which is not a useful representation, a constraint on the complexity can be applied to the mutual information as $I(\hat{u*}; m) \leq I_c$, where I_c is the information constraint. This is equivalent to using the Lagrange multiplier β to maximize the objective function $I(\hat{u*}; m) - \beta I(\hat{u*}; o)$. Intuitively, as the first term is to encourage m to be predictive of $\hat{u*}$ while the second term is to encourage m to forget o. Essentially m is to act like a minimal sufficient statistic of o for predicting $\hat{u*}$ [19].

Then specifically for each agent i, we intend to encourage assistive informationm_{-j} from other agents to agent j to memorize its $\hat{u_j*}$ when assistive information from agent i is conditioned on observation o_j, while we encourage assistive information m_i from agent i to not depend directly on its own observation o_i.

To efficiently and effectively encode extra state information for individual value estimation, we consider this information encoding problem as an information bottleneck problem [19], the objective for each agent i can be written as:

$$J_{IB}\left(\boldsymbol{\theta}_m\right) = \sum_{j=1}^{n} \left[I_{\theta_m}\left(\hat{u}_j^*; m_i | \mathrm{o}_j, m_{-j}\right) - \beta I_{\boldsymbol{\theta}_m}\left(m_i; o_i\right) \right] \tag{9}$$

This object is appealing because it defines what is a good representation in terms of trade-off between a succinct representation and inferencing ability. The main shortcoming is that the computation of the mutual information is computationally challenging. Inspired by the recent advancement in Bayesian inference and variational auto-encoder [40, 37, 11], we propose a novel way of representing it by utilizing latent vectors from variational inference models using information theoretical regularization method, and then derive the evidence lower bound (ELBO) of its objective.

Lemma 1. *A lower bound of mutual information* $I_{\theta_m}\left(\hat{u}_j^*; m_i | \mathrm{o}_j, m_{-j}\right)$ *is*

$$\mathbb{E}_{o_i \sim \mathcal{D}, m_j \sim f_m}[-\mathcal{H}(p(\hat{u}_j^* | \mathbf{o}), q_\psi(\hat{u}_j^* | \mathrm{o}_j, \boldsymbol{m}))]$$

where q_ψ *is a variational Gaussian distribution with parameters* ψ *to approximate the unknown posterior* $p(\hat{u}_j^* | \mathrm{o}_j, m_j)$, $\mathbf{o} = \{o_1, o_2, \cdots, o_n\}$, $\mathbf{m} = \{m_1, m_2, \cdots, m_n\}$.

Proof.

$$\begin{aligned}
& I_{\theta_m}\left(\hat{u}_j^*; m_i | \mathrm{o}_j, m_{-j}\right) \\
= & \int d\hat{u}_j^* do_j dm_{-j} p\left(\hat{u}_j^*, o_j, m_{-j}\right) \log \frac{p\left(\hat{u}_j^*, m_i | o_j, m_{-j}\right)}{p\left(\hat{u}_j^* | o_j, m_{-j}\right) p\left(m_i | o_j, m_{-j}\right)} \\
= & \int d\hat{u}_j^* do_j dm_{-j} p\left(\hat{u}_j^*, o_j, m_{-j}\right) \log \frac{p\left(\hat{u}_j^* | o_j, m_{-j}\right)}{p\left(\hat{u}_j^* | o_j, m_{-j}\right)}
\end{aligned}$$

where $p(\hat{u}_j^* | o_j, m_{-j})$ is fully defined by our encoder and Markov Chain. Since this is intractable in our case, let $q_\psi(\hat{u}_j^* | \tau_j, m_{-j})$ be a variational approximation to $p(\hat{u}_j^* | o_j, m_{-j})$, where this is our decoder which we will take to another neural network with its own set of parameters ψ. Using the fact that Kullback Leibler divergence is always positive, we have

$$KL[p(\hat{u}_j^* | o_j, m_{-j}), q_\psi(\hat{u}_j^* | \tau_j, m_{-j})] \geq 0$$

$$\int d\hat{u}_j^* do_j dm_{-j} p(\hat{u}_j^*, o_j, m_{-j}) \log p(\hat{u}_j^* | o_j, m_{-j}) \geq \int d\hat{u}_j^* do_j dm_{-j} p(\hat{u}_j^*, o_j, m_{-j}) \log q_\psi(\hat{u}_j^* | o_j, m_{-j})$$

and hence

$$
\begin{aligned}
& I_{\theta_c}\left(\hat{u}_j^*; m_i | o_j, m_j\right) \\
\geq & \int d\hat{u}_j^* do_j dm_j p\left(\hat{u}_j^*, o_j, m_j\right) \log \frac{q_\psi\left(\hat{u}_j^* | o_j, m_j\right)}{p\left(\hat{u}_j^* | o_j, m_{-j}\right)} \\
= & \int d\hat{u}_j^* do_j dm_j p\left(\hat{u}_j^*, o_j, m_j\right) \log q_\psi\left(\hat{u}_j^* | o_j, m_j\right) - \int d\hat{u}_j^* do_j dm_j p\left(\hat{u}_j^*, o_j, m_j\right) \log p\left(\hat{u}_j^* | o_j, m_{-j}\right) \\
= & \int d\hat{u}_j^* do_j dm_j p(o_j) p(m_j | o_j) p(\hat{u}_j^* | o_j) \log q_\psi(\hat{u}_j^* | o_j, m_j) + \mathcal{H}(\hat{u}_j^* | o_j, m_j) \\
= & \mathbb{E}_{\mathbf{o} \sim \mathcal{D}, m_j \sim f_m}\left(\int d\hat{u}_j^* p(\hat{u}_j^* | \mathbf{o}) \log q_\psi(\hat{u}_j^* | o_j, m_j)\right) + \mathcal{H}(\hat{u}_j^* | o_j, m_j) \\
= & \mathbb{E}_{\mathbf{o} \sim \mathcal{D}, m_j \sim f_m}[-\mathcal{H}[p(\hat{u}_j^* | \mathbf{o}), q_\psi(\hat{u}_j^* | \mathrm{o}_j, \boldsymbol{m})]] + \mathcal{H}(\hat{u}_j^* | o_j, m_j)
\end{aligned}
$$

Notice that the entropy of labels $\mathcal{H}(\hat{u}_j^* | o_j, m_j)$ is an positive term that is independent of our optimization procedure and thus can be ignored. Then we have

$$
I_{\theta_m}(\hat{u}_j^*; m_i | \mathrm{o}_j, m_j) \geq \mathbb{E}_{\mathbf{o} \sim \mathcal{D}, m_j \sim f_m}[-\mathcal{H}[p(\hat{u}_j^* | \mathbf{o}), q_\psi(\hat{u}_j^* | \mathrm{o}_j, \boldsymbol{m})]]
$$

which is the lower bound of the first term in Eq.(2) □

Lemma 2. *A lower bound of mutual information $I_{\boldsymbol{\theta}_m}(m_i; o_i)$ is*

$$
\mathbb{E}_{\mathrm{T}_i \sim D, m_j \sim f_m}[\beta D_{\mathrm{KL}}(p(m_i | \mathrm{o}_i) \| q_\phi(m_i))]
$$

where D_{KL} denotes Kullback-Leibler divergence operator and $q_\phi(m_i)$ is a variational posterior estimator of $p(m_i)$ with parameters ϕ.

Proof.

$$
\begin{aligned}
& I_{\boldsymbol{\theta}_m}(M_i; T_i) \\
& = \int dm_i do_i p(m_i | o_i) p(o_i) \log \frac{p(m_i | o_i)}{p(m_i)} \\
& = \int dm_i do_i p(m_i | o_i) p(o_i) \log p(m_i | o_i) - \int dm_i do_i p(m_i | o_i) p(o_i) \log p(m_i)
\end{aligned}
$$

Again, $p(m_i)$ is fully defined by our encoder and Markov Chain, and when it is fully defined, computing the marginal distribution $\int do_i p(m_i | o_i) P(o_i)$ might be difficult. So we use $q_\phi(m_i)$ as a variational approximation to this marginal. Since $KL[p(m_i), q_\phi(m_i)] \geq 0$,

We have

$$
\int dm_i p(m_i) \log p(m_i) \geq \int dm_i p(m_i) \log q_\phi(m_i)
$$

Then

$$
\begin{aligned}
& I_{\boldsymbol{\theta}_m}(M_i; T_i) \\
& \leq \int dm_i do_i p(m_i | o_i) p(o_i) \log p(m_i | o_i) - \int dm_i do_i p(m_i | o_i) p(o_i) \log q_\phi(m_i) \\
& = \int dm_i do_i p(m_i | o_i) p(o_i) \log \frac{p(m_i | o_i)}{q_\phi(m_i)} \\
& = \mathbb{E}_{\mathrm{o}_i \sim D, m_j \sim f_m}\left[D_{\mathrm{KL}}\left(p\left(m_i | \mathrm{o}_i\right) \| q_\phi\left(m_i\right)\right)\right]
\end{aligned}
$$

□

Combining **Lemma 1** and **Lemma 2**, we have the ELBO for the message encoding objective, which is to minimize

$$
\mathcal{L}_{IB}\left(\boldsymbol{\theta}_m\right) = \mathbb{E}_{o_i \sim \mathcal{D}, m_j \sim f_m}[-\mathcal{H}[p(\hat{u}_j^* | \mathbf{o}), q_\psi(\hat{u}_j^* | \mathrm{o}_j, \boldsymbol{m})] + \beta D_{\mathrm{KL}}(p(m_i | \mathrm{o}_i) \| q_\phi(m_i))] \tag{10}
$$

A.1.2 Factorized soft policy iteration

Recent works have shown that Boltzmann exploration policy iteration is guaranteed to improve the policy and converge to optimal with unlimited iterations and full policy evaluation, within MARL domain it can be defined as:

$$J(\pi) = \sum_t \mathbb{E}\left[r\left(\mathbf{s}_t, \mathbf{u}_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot|\mathbf{s}_t\right)\right)\right]$$

In section 4 we give the factorized soft policy gradients as:

$$\begin{aligned} \mathcal{L}_{LP}(\pi) &= \mathbb{E}_{\mathcal{D}}\left[\alpha \log \boldsymbol{\pi}\left(\boldsymbol{u}_t|\boldsymbol{\tau}_t\right) - Q_{tot}^{\pi}\left(\boldsymbol{s_t}, \boldsymbol{\tau_t}, \boldsymbol{u}_t, \boldsymbol{m}_t\right)\right] \\ &= -q^{\text{mix}}\left(\boldsymbol{s}_t, \mathbb{E}_{\pi^i}\left[q^i\left(\tau_t^i, u_t^i, m_t^i\right) - \alpha \log \pi^i\left(u_t^i|\tau_t^i\right)\right]\right) \end{aligned} \tag{11}$$

We now derive it in detail, we use the aristocrat utility to perform credit assignment:

Let q^{mix} be the operator of a one-layer mixing network with no activation functions in the end whose parameters are generated from the hyper-network with input $\boldsymbol{s}_t$, then

$q^{\text{mix}}(\boldsymbol{s}_t, \boldsymbol{q}(\tau_t, a_t, m_t) - \boldsymbol{\alpha} \log \boldsymbol{\pi}(\boldsymbol{a_t}|\boldsymbol{\tau_t}))$

$$= \sum_i [k^i(\boldsymbol{s})\mathbb{E}_\pi[q^i(\tau_t^i, a_t^i, m_t^i)] - \sum_i [k^i(\boldsymbol{s})\alpha^i \log \pi^i(a_t|\tau_t)] + b(\boldsymbol{s})$$

($k^i(s)$ and $b^i(s)$ are the corresponding weights and biases of q^{mix} conditioned on $\boldsymbol{s}$)

$$= \mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})] - \sum_i [k^i(\boldsymbol{s})\mathbb{E}_\pi[\alpha^i \log \pi^i(a_t|\tau_t)]$$

$$(q^{\text{mix}}(\boldsymbol{s}_t, \boldsymbol{q}(\tau_t, a_t, m_t)) = \sum_i [k^i(\boldsymbol{s})\mathbb{E}_\pi[q^i(\tau_t^i, a_t^i, m_t^i)] + b(\boldsymbol{s}) = \mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})]\ )$$

$$(\ \mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})] = \sum_{\boldsymbol{a}} \pi^i(a_t^i|\tau_t^i)\mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})]\ )$$

$$= \mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})] - \sum_i \mathbb{E}_\pi[\boldsymbol{\alpha} \log \pi^i(a_t^i|\tau_t^i)]$$

$$(\text{let } \alpha^i = \frac{\boldsymbol{\alpha}}{k^i(\boldsymbol{s})})$$

$$= \mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})] - \sum_i \sum_\pi [\boldsymbol{\alpha}\pi^i(a_t^i|\tau_t^i) \log \pi^i(a_t^i|\tau_t^i)]$$

$$= \mathbb{E}_\pi[Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{m}; \boldsymbol{\theta})] - \sum_\pi \boldsymbol{\alpha} \log \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{\tau}_t)$$

$$(\text{Assume } \boldsymbol{\pi} = \prod \pi^i, \text{ then } \sum_i \sum_\pi [\boldsymbol{\alpha}\pi^i(a_t^i|\tau_t^i) \log \pi^i(a_t^i|\tau_t^i)] = \sum_\pi \boldsymbol{\alpha} \log \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{\tau}_t))$$

$$= \mathbb{E}_\pi[Q_{tot}^{\pi}(\boldsymbol{s_t}, \boldsymbol{\tau_t}, \boldsymbol{a}_t) - \boldsymbol{\alpha} \log \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{\tau}_t)]$$

Which then complies to the original soft-actor-critic policy update policy.

We use the derivation above to show that directly using $\boldsymbol{q}(\tau_t, a_t, m_t) - \boldsymbol{\alpha} \log \boldsymbol{\pi}(\boldsymbol{a_t}|\boldsymbol{\tau_t})$ as input to feed in the mixing network to serve as soft-actor-critic policy update policy in a value decomposition method. It holds when using a single-layer mixing network without activation function, but nevertheless it offers insights of the proposed design, and when using relu activation function, it can be served as a lower bound object for optimization.

A.2 Environment Details

We use more recent baselines (i.e., FOP and DOP) that are known to outperform QTRAN [5] and QPLEX [9] in the evaluation. In general, we tend to choose baselines that are more closely related to our work and most recent. This motivated the choice of QMIX (baseline for value-based factorization

Algorithm 1 pseudocode for training PAC

for $k = 0$ to max_train_steps **do**
  Initiate environment, critic network q, mixing network Q^*, Q_{tot}, policy network π, message encoder m
  Initiate Replay buffer $\mathcal{D}$
  for $t = 0$ to $max_episode_limits$ **do**
    For each agent i, take action $a_i \sim \pi_i$
    Execute joint action $\mathbf{a}$, observe reward r,
    and observation $\boldsymbol{\tau}$, next state s_{t+1}
    Store $(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau}')$ in replay buffer $\mathcal{D}$
  end for
  for t = 1 to T **do**
    Sample trajectory minibatch $\mathcal{B}$ from $\mathcal{D}$
    Generate peer-assisted information
    $m_i \sim \boldsymbol{N}(f_m(o_i; \theta_m), \boldsymbol{I}))$, for $i = 0$ to n
    Calculate Loss
    $\mathcal{L}(\theta) = \mathcal{L}_{LP} + \mathcal{L}_{CA} + \mathcal{L}_{IB} + \mathcal{L}_{\hat{Q}^*} + \mathcal{L}_{Q_{tot}}$
    Update critic network and mixing network
    $\boldsymbol{\theta_{nn}(q, Q^*, Q_{tot})} \leftarrow \eta\hat{\nabla}\mathcal{L}(\theta)$
    Update policy network
    $\boldsymbol{\theta}(\pi) \leftarrow \eta\hat{\nabla}\mathcal{L}(\pi)$
    Update encoding network
    $\boldsymbol{\theta}_m(m) \leftarrow \eta\hat{\nabla}\mathcal{L}(\theta)$
    Update temperature parameter
    $\alpha \leftarrow \eta\hat{\nabla}\alpha$
    if t mod d = 0 **then**
      Update target networks: $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
    end if
  end for
end for
Return $\boldsymbol{\pi}$

methods), WQMIX (close to our work that uses weighted projections so better joint actions can be emphasized), NDQ [11] (which similarly uses common information to assist decision making but as generating messages for agent-wise communication), VDAC [14], FOP [13], DOP [12] (SOTA actor-critic based methods). Our code implementation is available at Github.

A.2.1 Multi-State Matrix Game

To highlight the importance of the extra state information for an assisted value function factorization, we present a multi-state matrix game as inspired by the single-state matrix game proposed in [5] and present how our method performs compared with the existing works.

The multi-state matrix game and the detailed mlearning results for more algorithms as shown in table 1 and table 2.

The multi-state matrix game can be considered the single state matrix game with the same goal of factorizing the global value, consider an Markov decision process (MDP) consisting of 2 states with 0.5 transition probabilities between them and two payoff matrices shown in table 1(a). Suppose that agent 1 has the same partial observation o_1 in states $s^{(1)}$ and $s^{(2)}$. Then, its per-agent value function $q_1(\cdot, \tau_1)$ computed from partial observation o_1 are also the same in both states. Due to the monotonicity of the mixing network (even though it is provided with complete joint state information), for any u_1 and u_1' with ordering $q_1(u_1, \tau_1) \geq q_1(u_1', \tau_1)$ without loss of generality, we must simultaneously have $Q_{tot}(u_1, u_2, s^{(1)}) \geq Q_{tot}(u_1', u_2, s^{(1)})$ and $Q_{tot}(u_1, u_2, s^{(2)}) \geq Q_{tot}(u_1', u_2, s^{(2)})$ for any action u_2 of agent 2 in both states.

x	$\mathbf{a}_2^1$	a_2^2	a_1^1
$\mathbf{a}_1^1$	**4**	-2	-2
a_1^2	-2	0	0
a_1^3	-2	0	0

x	a_2^1	a_2^2	$\mathbf{a}_1^1$
a_1^1	-2	0	0
a_1^2	**4**	-2	-2
$\mathbf{a}_1^3$	-2	0	0

(a) Payoff matrix for state s_1 and s_2

x	**0.3**	-1.2	-2.6
-0.2	0.1	-1.0	-1.0
0.3	**1.1**	-0.9	-1.0
-2.6	-1.0	-1.0	-1.0

x	**0.4**	-1.3	-2.2
-0.2	0.4	-1.0	-1.0
0.3	**1.4**	-0.9	-1.0
-2.6	-1.0	-1.0	-1.0

(b) QMIX: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

x	1.0	0.2	0.2
1.0	**4.0**	-1.2	-1.2
0.0	-0.2	-1.6	-1.6
0.0	-0.2	-1.6	-1.6

x	0.0	0.5	**0.5**
1.0	-1.2	0.1	**0.1**
0.0	-1.6	-1.2	-1.2
0.0	-1.6	-1.2	-1.2

(c) WQMIX: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

x	1.0	0.2	0.2
1.0	**4.0**	-1.8	-1.9
0.0	-1.9	0.0	0.1
0.0	-2.1	-0.1	0.0

x	0.0	0.5	**0.5**
1.0	-2.0	0.0	**0.0**
0.0	0.1	-2.0	-2.0
0.0	-2.2	-0.0	-0.0

(d) WQMIX: $\hat{Q}^*(s_1)$, $\hat{Q}^*(s_2)$

x	**0.7**	-2.0	-2.1
0.7	**4.0**	-2.1	-2.4
-2.0	-2.1	-2.4	-2.4
-2.1	-2.1	-2.4	-2.4

x	**0.6**	-2.0	-2.0
-1.2	-1.8	-2.5	-2.5
1.6	**4.0**	-2.0	-2.0
-1.8	-2.1	-2.5	-2.5

(e) OURS: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

x	**0.7**	-2.0	-2.1
0.7	4.0	-2.1	-2.1
-2.0	-2.1	-0.1	-0.1
-2.1	-2.1	-0.1	-0.1

x	**0.7**	-2.0	-2.1
-1.2	-2.1	-0.0	-0.0
1.6	**4.0**	-2.0	-2.0
-1.8	-2.1	-0.0	-0.0

(f) OURS: $\hat{Q}^*(s_1)$, $\hat{Q}^*(s_2)$

Table 1: Payoff matrix of the one-step multi-state non-monotonic cooperative matrix game and reconstructed results from corresponding baselines. State s_1 and s_2 are selectet randomly on equal probability. Boldface indicates the local and joint optimal actions from local utilities and action-state value function.

x	**0.7**	0.2	0.2
0.2	0.4	0.2	-0.2
0.7	**0.7**	0.4	-0.4
0.2	0.4	0.2	0.2

x	**0.7**	0.2	0.2
0.2	0.4	0.2	-0.2
0.7	**0.7**	0.4	-0.4
0.2	0.4	0.2	0.2

(a) DOP: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

x	**0.7**	0.6	-0.2
1.1	**1.7**	-0.9	-0.1
0.0	-0.6	0.5	-0.5
0.0	-0.6	0.5	-0.5

x	**0.7**	0.6	-0.2
1.1	**1.7**	-0.9	-0.1
0.0	-0.6	0.5	-0.5
0.0	-0.6	0.5	-0.5

(b) FOP: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

x	**0.7**	0.2	0.2
0.7	**2.3**	0.9	0.9
0.2	2.3	0.9	0.9
0.2	2.2	0.9	0.9

x	**0.7**	0.2	0.2
0.7	**3.2**	2.2	2.2
0.2	3.1	2.2	2.2
0.2	3.1	2.2	2.2

(c) VDAC: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

x	**0.5**	0.3	0.2
0.3	1.2	-0.9	-1.2
1.2	**1.2**	-1.0	-0.5
0.2	-0.7	-0.9	-0.1

x	**0.6**	0.2	0.1
0.5	**0.7**	-1.2	-0.0
0.3	0.6	-1.1	-0.8
0.2	-2.1	-0.1	-1.0

(d) QTRAN: $Q_{tot}(s_1)$, $Q_{tot}(s_2)$

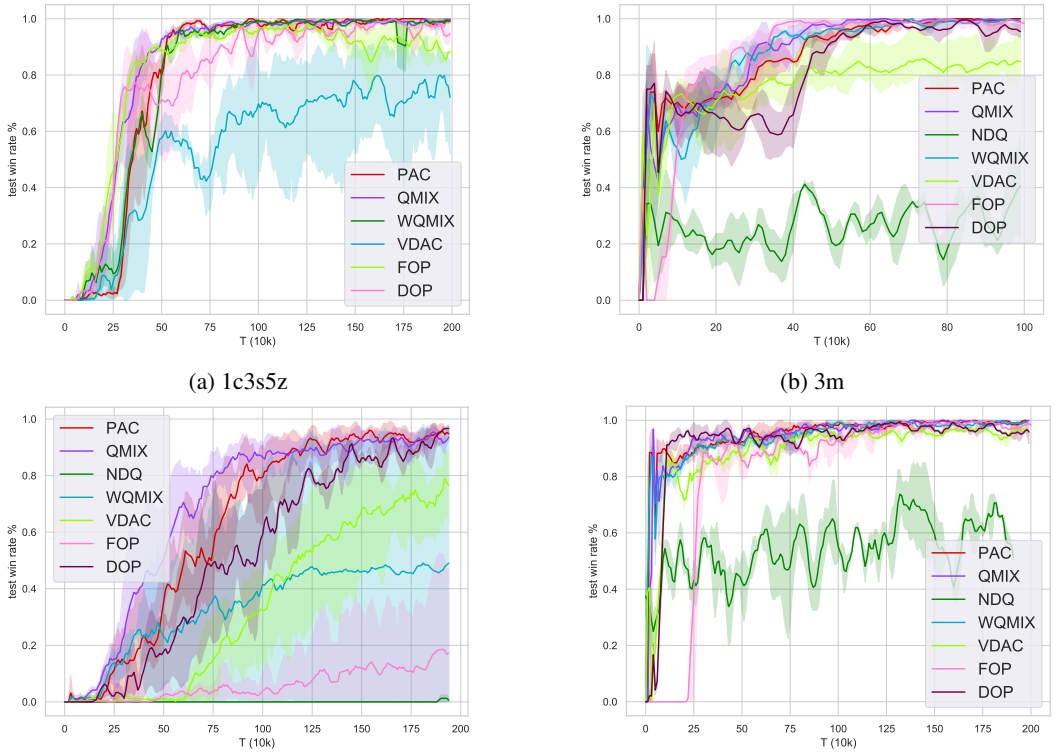
Table 2: Matrix results for other Benchmarks.

A.2.2 Predator-Prey

A partially observable environment on a grid-world predator-prey task is used to model relative overgeneralization problem [24] where 8 agents have to catch 8 prey in a 10 × 10 grid. Each agent can either move in one of the 4 compass directions, remain still, or try to catch any adjacent prey. Impossible actions, i.e., moves into an occupied target position or catching when there is no adjacent prey, are treated as unavailable. If two adjacent agents execute the catch action, a prey is caught and both the prey and the catching agents are removed from the grid. An agent's observation is a 5 × 5 sub-grid centered around it, with one channel showing agents and another indicating prey. An episode ends if all agents have been removed or after 200 time steps. Capturing a prey is rewarded with r = 10, but unsuccessful attempts by single agents are punished by a negative reward p. In this paper we consider two sets of experiments with p = 0 and p = -2. The task is similart to matrix game proposed by [5] but significantly more complex, both in terms of the optimal policy and in the number of agents.

map	Ally Units	Enemy Units
1c3s5z	1 Colossus, 3 Stalkers & 5 Zealots	1 Colossus, 3 Stalkers & 5 Zealots
3m	3 Marines	3 Marines
3s5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots
8m	8 Marines	8 Marines
3s_vs_5z	3 Stalkers	5 Zealots
5m_vs_6m	5 Marines	6 Marines
MMM2	1 Medivac, 2 Marauders & 7 Marines	1 Medivac, 3 Marauders & 8 Marines
27m_vs_30m	27 Marines	30 Marines
6h_vs_8z	6 Hydralisks	8 Zealots
corridor	6 Zealots	24 Zerglings

Table 3: Brief Introduction of SMAC map scenarios used in experiments



(a) 1c3s5z (b) 3m

(c) 3s5z (d) 8m

Figure 6: Additional results on SMAC benchmark

A.2.3 SMAC

For the experiments on StarCraft II micromanagement, we follow the setup of SMAC [8] with open-source implementation including QMIX [4], WQMIX [6], NDQ [11], FOP [13], DOP [12] and VDAC [14]. We consider combat scenarios where the enemy units are controlled by the StarCraft II built-in AI and the friendly units are controlled by the algorithm-trained agent. The possible options for built-in AI difficulties are Very Easy, Easy, Medium, Hard, Very Hard, and Insane, ranging from 0 to 7. We carry out the experiments with ally units controlled by a learning agent while built-in AI controls the enemy units with difficulty = 7 (Insane). Depending on the specific scenarios(maps), the units of the enemy and friendly can be symmetric or asymmetric. At each time step each agent chooses one action from discrete action space, including noop, move[direction], attack[enemy_id] and stop. Dead units can only choose noop action. Killing an enemy unit will result in a reward

of 10 while winning by eliminating all enemy units will result in a reward of 200. The global state information are only available in the centralized critic. Each baseline algorithm is trained with 4 random seeds and evaluated every 10k training steps with 32 testing episodes for main results, and with 3 random seeds for ablation results and additional results. We carried out our experiment on a Nvidia GeForce RTX 2080 Ti workstation, on average it takes 3.5 hours to finish 3s5z map on SMAC environment for one run.

Additional Results We present additional results on easier maps including `1c3s5z`, `3m`, `3s5z`, and `8m` in Fig. 1.

	self-updating alpha	Q * Network	L_ca	L_ib
PAC	Y	Y	Y	Y
PAC_no_info	Y	Y	Y	
PAC_fixed_alpha	alpha = 1.0	Y	Y	Y
PAC_CE_Loss	Y	Y	replace with CE	Y
PAC_disabled*	Y	Y		
PAC_No_Q	Y			

Table 4: Comparison of our method and its ablated versions

A.2.4 Implementation details and Hyper-parameters

In this section we introduce the implementation details and hyper-parameters we used in the experiment. Recently [22] demonstrated that MARL algorithms are significantly influenced by code-level optimization and other tricks, e.g. using TD-lambda, Adam optimizer and grid-searched hyper-parameters (where many state-of-the-art are already adopted), and proposed fine-tuned QMIX and WQMIX, which is demonstrated with significant improvements from their original implementation. We implemented our algorithm based on its open-sourced codebase and acquired the results of QMIX and WQMIX from it. We use one set of hyper-parameters for each environment, i.e., no tuned hyper-para for individual maps. Unless otherwise mentioned, we keep the same setting for common hyper-parameters shared by all algorithms, e.g. learning rate, and keep their unique hyper-parameters to their default settings.

We use epsilon greedy for action selection with annealing from ϵ = 0.995 decreasing to ϵ = 0.05 in 100000 training steps in a linear way.

Batch size bs = 128, replay buffer size = 10000

Target network update interval: every 200 episodes

β = 0.001, since o_i and $\hat{u}_i^*$ are within similar dimensions and thus does not require very high compression.

Weights w =0.5 in weighting functions.

learning rate lr = 0.001

td lambda λ = 0.6

initial entropy term logα = -0.07, with its learning rate lr_α = 0.0003

performance for each algorithm is evaluated for 32 episodes every 1000 training steps.

A.2.5 Hyperparameter-tuning for ablation studies

To fully demonstrate the effectiveness of each components, we performed hyperparameter-tuning for each ablated settings. Specifically, we choose exploration steps (epsilon anneal time = [50k, 100k], where ϵ decays from 0.095 to 0.05 in ϵ- greedy), eligibility traces (λ = [0.3, 0.5, 0.6] in TD-λ), and replay-buffer size (buffer = [5000, 10000]) and use the results with best performance to serve as the ablated results. We show the optimal hyperparameter setting for each ablated versions and their corresponding final winning rates on SMAC environment as in Table 4.

Setting	Optimal Hyperparameter Setting	Average Winning Rates
PAC	λ=0.5 buffer_size=10000, episilon_anneal_time=100000	0.87
PAC_Fixed_alpha	λ=0.6 buffer_size=5000, episilon_anneal_time=80000	0.80
PAC_no_info	λ=0.6 buffer_size=10000, episilon_anneal_time=100000	0.51
PAC_CE_loss	λ=0.5 buffer_size=10000, episilon_anneal_time=100000	0.61
PAC_disabled	λ=0.5 buffer_size=5000, episilon_anneal_time=80000	0.49
PAC_No_Q*	λ=0.6 buffer_size=10000, episilon_anneal_time=100000	0.25

Table 5: Optimal Hyperparameter Setting for ablated versions