SCIENCE CHINA
Information Sciences

• Supplementary File •

Multi-Agent Reinforcement Learning with Quantified Information-decision Content Measurement

Ershen Wang[1,2], Xiaotong Wu[1], Chen Hong[3,4*], Yanwen Wang[5], Aidong Chen[3,4], Hongyuan Jing[3,4], Song Xu[1] & Pingping Qu[1]

[1]*the School of Electronic and Information Engineering, Shenyang Aerospace University, Shenyang* 110136*, P.R.China*
[2]*the School of Civil and Aviation, Shenyang Aerospace University, Shenyang* 110136*, P.R.China*
[3]*the Multi-Agent Systems Research Centre, Beijing Union University, Beijing 100101, P.R.China*
[4]*the College of Robotics, Beijing Union University, Beijing 100101, P.R.China*
[5]*the School of Computer Science and Engineering, Northeastern University, Shenyang* 110169*, P.R.China*

Appendix A Related Work

Appendix A.1 Multi-agent system

In recent years, research in the field of multi-agent system (MAS) and intelligent decision-making has been thriving. In particular, multi-agent reinforcement learning (MARL) has garnered considerable attention owing to its remarkable effectiveness in various applications [1–4], and has been widely applied in numerous fields, such as unmanned aerial vehicle (UAV) swarms [5–7], autonomous driving [8], traffic signal control [9]. Actually, MARL is not limited to real-world environments, it can also be utilized in virtual environments, including StarCraft II [10], DOTA2 [11] as well as military simulations [12] and so on. These virtual environments often exhibit certain characteristics: high uncertainty, partial observability and dynamic variability, which pose considerable challenges to MARL. Among these virtual environments, StarCraft II is particularly well-suited for MARL research, due to its full support of collaboration, competition, communication, and learning between multiple heterogeneous agents. To better assess the effectiveness of MARL in StarCraft II, researchers have developed the starCraft multi-agent challenge (SMAC) [13]. As a quintessential benchmark in StarCraft II environments, SMAC mainly focuses on addressing micromanagement challenges, where each unit is controlled by an individual agent who makes decisions based on limited and partial observational information.

In general, whether in real or virtual environments, MARL may all face the challenge of partial observability. For single-agent reinforcement learning (SARL), agent can make decisions based on the full state of the environment [14]. However, for MARL, agent can only observe part of the environment and is unable to acquire a global understanding of the environment, which may lead to local optima [15] because agent cannot directly obtain the states and intentions of other agents. Recently, Zhao et. al proposed a novel algorithm named OMPG to address such limitations [16]. Especially, in large-scale MAS, the partial observability may become a notable challenge, when agents process a vast amount of local information, the global information scarcity and tremendous computation complexity will arise inevitably [4, 17, 18].

To address the issue of partial observability, some researchers have adopted partially observable Markov decision process (POMDP) [19] as a mathematical framework to simulate and analyze agents' decision-making process. POMDP not only retains the Markovian property, but also provides a structured paradigm for modeling and solving the partial observability, where the current decision of agents depends solely on their last decisions and observations, and thus there is no need to painfully trace back the entire decision-making history.

However, in MARL scenarios, even there are only several agents, they still cannot fully perceive the overall environmental state, and thus POMDP is extended naturally to a decentralized partially observable Markov decision process (Dec-POMDP) [20]. Considering that cooperation and information sharing are prevalent among multiple agents [21,22], Dec-POMDP allows agents to gain a deeper understanding of each other's actions and intentions. But this will bring about the issue of insufficient information that primarily manifests as a lack of environmental awareness, whereupon agents may struggle to acquire sufficient environmental information, and this will constrain their understanding of the environmental states [17, 23]. In response to this issue, the centralized training and decentralized execution (CTDE) [24] paradigm is put forward to enhance communication and cooperation among agents, enabling them to share information against incomplete observations. During the training process, CTDE allows agents to share the global information to better guide their training [25]. In the case of actual execution process, however, agents make decisions only based on their local observations and strategies. In other words, the decision quality in the execution process is closely related to the amount of information that agent can obtain from the local observation.

* Corresponding author (email: hchchina@sina.com, xxthongchen@buu.edu.cn)

Appendix A.2 MARL

In early research on MARL, single-agent algorithm was directly extended into multi-agent environments [26]. Initially, each agent was treated as an independent learner and trained separately, while the actions of other agents were considered as part of the environment. In this context, Q-learning was applied, and was known as independent Q-learning (IQL) [27]. Subsequently, deep reinforcement learning algorithms, such as deep Q-network (DQN) [28] and deep recurrent Q-network (DRQN) [29], were proposed to handle high-dimensional spaces and adapt to complex MAS. For IQL, one major challenge is the nonstationarity, i.e., the dynamic of environmental changes with the evolution of the strategies of other agents evolve. This will complicate the learning process because the typical assumption of the stationary environment in SARL no longer holds. Consequently, agents struggle to converge to a stable strategy, leading to inefficient action selection and potential conflicts. To address this issue, a variant of IQL [30] was proposed, where one agent is trained at a time step, while the strategies of other agents kept constant; clearly, this approach maintains a stable environment for the currently trained agent so it can better adapt to the nonstationarity of environment.

To deal with the issue of nonstationarity, most MARL algorithms adopt CTDE. One approach assumes that all critics can observe the states and actions of all agents, namely fully observable [31], where critics can learn true states and guide actors to find the optimal strategy. If all agents fully cooperate and share a common reward, then only a single centralized critic is enough. As long as the critic is fully observable, it can alleviate or even eliminate the issue of nonstationarity. COMA [32] uses a single centralized critic, where each agent is trained by its local observation-action history, and introduces a counterfactual baseline to select actions with an advantage function. MAPPDG [33] allows each agent to train a deep deterministic policy gradient (DDPG), where the critic of each agent takes all state-action pairs as input, concatenating them together, and using local rewards to compute the corresponding value function. Wang et al. modeled the dynamics of regret minimization in large agent populations and used forward prediction to get more precise action-making [4]. MAPPO [34] can reduce sampling bias by overlapping calculations with respect to advantage estimation and importance sampling correction. By introducing the idea of value factorisation from QMIX into the actor-critic method, FACMAC [35] combines strategy and value, and uses a centralized policy gradient estimator in the critic to factorize Q-values and calculate the policy gradient for all agents.

As the strategies of each agent change during training, the experience replay buffer stores observations of the different strategies used. One agent cannot effectively use experience replay without addressing nonstationarity. Actually, if experience replay is not used, learning effects may suffer from low sampling efficiency and high sample correlation [36]. In addition, the Bellman equation, which assumes full observability [25], is given by

$$Q_i^*(s, a_i|\boldsymbol{a}_{-i}) = \sum_{\boldsymbol{a}_{-i}} \pi_{-i}\left(\boldsymbol{a}_{-i}, s\right)\left[r(s, a_i, \boldsymbol{a}_{-i}) + \gamma \sum_{\mathbf{s}'} P(\mathbf{s}'|\boldsymbol{s}, a_i, \boldsymbol{a}_{-i}) max Q_i^*(\boldsymbol{s}, a_i')\right], \tag{A1}$$

where $\pi_{-i} = \prod_{j \neq i} \pi_j\left(a_j|s\right)$ represents the joint strategy of other agents. Since the strategies of others change over time, namely π_{-i} changes too, and thus the traditional Bellman equation cannot be directly used to solve for the optimal value in MARL.

Simplifying multi-agent problems to a single-agent setting often makes it difficult for traditional reinforcement learning algorithms to find global optimal solutions. This is primarily because, in such scenarios, agents cannot access the true value of action rewards, resulting in that some agents may become overly conservative over time [37]. Additionally, exploration by agents with relatively poor strategies will exacerbate the overall team reward, impeding agents with already-good strategies from progressing toward the optimal strategy. To address the issue, an idea was proposed to calculate each agent's global share of the rewards, which is formalized as value function factorisation, where the decomposition function learns from the global reward. VDN [38] factorises the global reward into individual local values for each agent via a neural network. With the introduction of certain constraints, the Q-value becomes additive and the action-value function is well shared during training, thus better coordination and cooperation are achieved. QMIX [39] introduces a novel mixing network to weight the local value functions of each agent. Offering better generalization and robustness, QTRAN [40] directly learns a joint action-value function and introduces two conditions to constrain the relation between the joint action-value function and the local action-value functions of each agent. QPLEX [41] employs a duplex dueling architecture as the value decomposition structure and encodes the individual global max (IGM) principle into the neural network architecture. PAC [42] optimizes the joint action-value function by factorisation maximization.

Generally, by learning value functions, agents can identify the optimal action strategy in a variety of circumstances. Value-based MARL methods have shown considerable potential to address cooperative problems, where each agent possesses a value function that estimates the expected rewards for taking different actions in the current state.

There are two mainstream approaches to solve value-based MARL problems: independent learning (IL) and CTDE. For IL, agent is considered as an autonomous entity interacting with the environment and updating its value function based on individual observations and rewards, and each agent endeavors to maximize its rewards without considering the impact of other agents' actions. A classical method of IL is independent Q-learning (IQL), where each agent maintains an independent Q-function to estimate the expected cumulative rewards for different actions in a given state; agent update its value function through interactions with the environment, it is then independent of other agents' strategies. A main research direction of IQL is to extend methods such as DQN and DRQN, enabling them to handle high-dimensional and complex settings. Although such method is relatively simple and easy to implement, it may lead to suboptimal and the lack of team cooperation. Furthermore, agents may easily fall into an endless exploration cycle [43] and fail to converge on effective collaborative strategies.

On the other hand, CTDE attempts to mitigate these challenges by centralizing the training process while still allowing for decentralized execution during the operational phase. Such paradigm enables the sharing of extensive information

across agents during the training process, including the policies of other agents, gradients, or even explicit communication protocols. This centralized perspective facilitates the joint optimization of strategy by considering the actions and expected outcomes of all agents, meanwhile, makes selected strategy more coordinated and consistent. Under CTDE, value-based MARL primarily seeks optimal strategy through value function decomposition and adheres to the following relation:

$$\arg\max_{\boldsymbol{u}} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}) = \begin{pmatrix} \operatorname{argmax}_{u_1} Q_1(\tau_1, u_1) \\ \cdots \\ \operatorname{argmax}_{u_N} Q_N(\tau_N, u_N) \end{pmatrix}. \tag{A2}$$

where $Q_{tot}\colon T^N \times U^N \mapsto \mathbb{R}$, defined by the IGM condition, denotes the overall joint action-value function. To satisfy the IGM condition, VDN directly decomposes the value function into an additive form (additivity):

$$Q_{tot}(\tau, \boldsymbol{u}) = \sum_{i=1}^{N} Q_i(\tau_i, u_i). \tag{A3}$$

Taking the partial derivative of the value function from Eq. (A3), we can obtain $\frac{\partial Q_{tot}(\tau,u)}{\partial Q_i(\tau_i,u_i)} \equiv 1, \forall i \in \mathcal{N}$. VDN satisfies the IGM condition, but it does not fully utilize the advantage of centralized training, and it overlooks the fact that under CTDE the global information can be exploited during the training process.

Afterwards, QMIX employs a centralized mixing network to estimate the joint Q-value of agent actions while maintains individual value functions which can used by agents during execution process. In approximating Q_{tot}, QMIX considers the global information, and introduces the condition of (monotonicity):

$$\frac{\partial Q_{tot}(\tau, u)}{\partial Q_i(\tau_i, u_i)} \geqslant 0, \forall i \in \mathcal{N}. \tag{A4}$$

For QMIX, as long as the joint action $\mathbf{u}$, obtained by the argmax function over Q_{tot}, coincides with the actions derived from each Q_i and adheres to the fundamental baseline of the monotonicity condition, the locally optimal actions chosen by each agent will be the precise components of the globally optimal action.

Appendix B RDec-POMDP

Recurrent decentralized partially observable Markov decision process (RDec-POMDP) can be formally represented by a tuple $G = \langle \mathcal{S}, \mathcal{U}, P, \mathcal{R}, \mathcal{X}, \mathcal{Z}, \mathcal{O}, n, \gamma, \mathcal{H} \rangle$, where $\mathcal{S}$ is the real state space, $\mathcal{U}$ is the joint action space, and $P(s'|s, \mathbf{u})$:$\mathcal{S}\times\mathcal{U}^n\times\mathcal{S} \mapsto [0, 1]$ is the state transfer probability. Each agent receives a reward $r = \mathcal{R}(s, \mathbf{u})$ and uses it to judge the decision quality. The information distribution $\mathcal{X}$ uses the probability $X(x \mid s)$ to obtain information $x \in \mathcal{X}$ conditioned on $s \in \mathcal{S}$. The observation distribution $\mathcal{Z}$ uses the probability $Z(o \mid x)$ to get observation $o \in \mathcal{O}$ when information x is given, where $O(o \mid s) = \int_{\chi} Z(o \mid x)X(x \mid s)\mathrm{d}x$ and all agents can sense and respond to relevant information about environment. Here, n is the number of agents, $\gamma \in [0, 1)$ is the discount factor, and $\mathcal{H}$ is the hidden state space. At each time step t, each agent relies on RNNs to learn the action-observation history τ^u with respect to the hidden state h_t, and the joint hidden state for all agents $\mathbf{h}_t \in \mathcal{H} \equiv z(\tau)$, where $\mathcal{H} \to \Delta(\mathcal{U})$ is the set of histories to probability measures over the action space, z is a compression function that maps the action-observation history τ. The hidden state indirectly reflects the dynamic characteristics of the real state by integrating the history of observation and action.

Here, we highlight four main properties in RDec-POMDP: (1) At any time step t, the true state can be written as a function of the true state at time step $t-1$ and the new information updated at time step t. The hidden state can also be updated recursively, specifically, the history information of the hidden state can be updated and accumulated by RNNs; (2) the transition probability P to the real state at the next time step only depends on the current real state and action, and has nothing to do with the hidden state; (3) when the information distribution $\mathcal{X}$ is known, observation o is conditionally independent of action u: $p(o \mid x, u) = p(o \mid x)$, and the relationship between them is: $s \to x \to o$; (4) given the statistic $f(h)$ and action u, the joint probability of reward r and o can be represented by the information variable x, which shows how $f(h)$ effectively captures the information from the hidden state h.

Appendix C The proof

The goal of RDec-POMDP is to maximize the expected return and find an optimal policy. The policy is defined as a mapping from histories to probability measures over the action space, and $\mathcal{H} \to \Delta(\mathcal{U})$ is the set of these mappings. Because the hidden state is cyclic recursive and the next state is predictable, we can draw the following conclusions: [47, 48]:

$$f(h') = \theta(f(h), u, o'), \forall h' = (h, u, o') \tag{C1}$$

$$p(r, o' \mid h, u) = p(r, o' \mid f(h), u), \forall (h, u, r, o') \tag{C2}$$

where θ is the hidden state update function of RNN and f is a statistical function, which is optimal as long as it satisfies Eq. (C1) and Eq. (C2). This means that $f(h)$ captures all important information about h, namely, hidden states contain richer information than real states in the RDec-POMDP framework, and thus the optimal decision can be made by the agent. The specific proof process is as follow:

First, using the total probability formula, we can obtain:

$$p(r, o'|f(h), u) = \int_{\chi} p(r, o', x'|f(h), u)dx' \tag{C3}$$

This means that given $f(h)$, the probability of the joint event (r, o') is summed by the integral over the joint distribution probabilities under all possible information variables x. And then we decompose $p(r, o', x'|f(h), u)$ according to the chain rule:

$$p(r, o', x'|f(h), u) = p(x'|f(h), u)p(r, o'|x', f(h), u) \tag{C4}$$

By decomposing $p(r, o'|x', f(h), u)$, we can get:

$$p(r, o'|x', f(h), u) = p(r|x', f(h), u)p(o'|r, x', f(h), u) \tag{C5}$$

Merging Eq. (C4) and Eq. (C5), we can obtain:

$$p(r, o', x'|f(h), u) = p(o'|r, x', f(h), u)p(r, x'|f(h), u) \tag{C6}$$

Substituting Eq. (C6) into Eq. (C3), we get:

$$p(r, o'|f(h), u) = \int_{\chi} p(o'|r, x', f(h), u)p(r, x'|f(h), u)\mathrm{d}x' \tag{C7}$$

Based on the property 3 of RDec-POMDP in Appendix B, Eq. (C7) can be simplified to:

$$p(r, o'|f(h), u) = \int_{\chi} p(o'|x')p(r, x'|f(h), u)\mathrm{d}x' \tag{C8}$$

According to the chain rule and the total probability formula, we can prove Eq. (C2) as:

$$\begin{aligned} p(r, o'|f(h), u) &= \int_{\chi} p(o'|x')p(r, x'|h, u)\mathrm{d}x' \\ &= \int_{\chi} p(o', r, x'|h, u)\mathrm{d}x' \\ &= p(r, o'|h, u) \end{aligned} \tag{C9}$$

Thus, the hidden state h effectively captures the dynamic characteristics of the environment and integrates historical information through recurrent updates, thereby containing richer information than the real state s corresponding to o' in Eq. (C9) of a single time step.

Appendix D QICM method

According to the context of RDec-POMDP, here we propose a MARL method with quantified information-decision content measurement (QICM). The design of the agent networks is crucial to approximating the Q-value of each agent. In our design, each agent not only relies on current partial observations, but also historical information integrated and processed by DRQNs. Here, agents are allowed to output their local Q-value $Q_n(\tau_t^n, u_t^n)$ based on their observation sequence τ_t^n [49] and actions u_t^n taken at the current time step. GRUs are adopted to generate hidden states h_t, which is vital for maintaining the coherence of sequential information because GRUs can effectively capture temporal dependencies, thereby more informed decisions are made by agents. Then hidden states h_t are fed as input into the transformer. Subsequently, hidden states are combined to form a joint hidden state vector $\mathbf{h}_\mathrm{t}$ that contains information of all agents. Since $f(h)$ is the probability distribution over the hidden states corresponding to the entire history of observations and actions, the joint hidden state can be considered as an approximation of the statistic $f(h)$. By integrating the sequential information with GRUs and transformers, the probability distribution is approximated, enabling agents to make decisions based on a comprehensive understanding of both current and historical information.

Although RNNs are good at working with sequential data, they are still limited by things like disappearing gradients and difficulty in capturing long-term relationships among agents. To further enhance cooperation and information integration, we incorporate a transformer and take advantage of two features: the time modeling capabilities of RNN and the self-attention mechanism of transformer, wherein the self-attention mechanism can autonomously learn which time step information is crucial for decision-making at the current time step, and the learning of adaptive weights can help filter out noise and irrelevant information from the hidden states. Notably, because of the independent and parallel decision-making of agents, the sequence order is disregard, namely, omitting position encoding. Concretely, the joint hidden state $\mathbf{h}_\mathrm{t}$ is input into the multi-head attention layer, and the output is defined as [50]

$$att_a(\mathbf{h}_\mathrm{t}) = softmax(\frac{mask(W_q^a(\mathbf{h}_\mathrm{t})W_k^a(\mathbf{h}_\mathrm{t}))^T}{\sqrt{d_{att}}})W_v^a(\mathbf{h}_\mathrm{t}), \tag{D1}$$

where W_q^a, W_k^a, and W_v^a are all MLPs with dimension d_{att} [51]; q, k, and v represent query, key, and value respectively; the $mask$ in Eq. (D1) obscures the observational outcomes of the agent and allows other agents to predict the actions of the agent from limited information [52]. The self-attention mechanism dynamically adjusts the importance of different parts of

the hidden states, enabling effective extraction of key information in partially observable environments. The joint hidden state $\mathbf{h}_t$ reflects the current state of all agents at a specific time step, while $f(h)$ encapsulates the historical information up to that point. The attention mechanism primarily utilizes $\mathbf{h}_t$ for real-time decision-making, and all the outputs are concatenated to form a higher-dimensional representation:

$$att(\mathbf{h}_t) = att_1(\mathbf{h}_t) \oplus ... \oplus att_a(\mathbf{h}_t). \tag{D2}$$

To further process the outputs and to effectively integrate them together, a linear transformation based on a fully connected layer is applied to the output. Such solution simplifies the information presentation and ensures that the output maintains the same dimension with the joint hidden state $\mathbf{h}_t$, which is crucial to subsequent monotonic value function factorisation. A decoder is used to produce the final output of the transformer, where the decoder uses multiple linear transformations and softmax activation functions to map the high-dimensional representations into probability distributions in the target space, ensuring that the transformer outputs the final results $att(\mathbf{h}_t)$ for value functions directly. The detailed internal structure of the transformer is plotted in Figure D1.

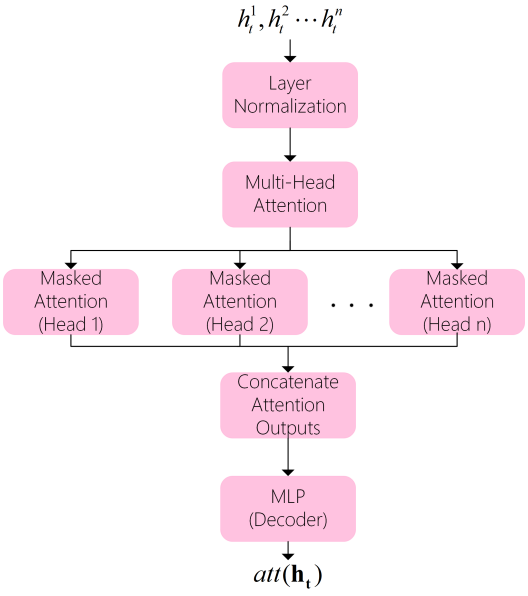

Figure D1 The structure of the transformer.

Afterwards, to accurately factorise the contribution of each agent to the total action value, the output of the fully connected layer enters the mixing network. The process is implemented by a monotonic function, which ensures that any increase in an agent's Q-value correspondingly enhances the overall value of the joint action, reflecting the added value of cooperative actions. The process also transforms the joint hidden state into a set of action strategies, and unfolds in an autoregressive manner by encoding, a scaled dot-product attention mechanism and decoding. During the process, based on the current environmental information, the interactions and objectives are shared across agents, and the real environmental state s will be replaced by a highly abstracted and informational representation h.

The mixing network optimizes the overall collaborative behaviors by using a linear combination. The local Q-value outputs from agent networks are combined to produce Q_{tot}, which reflects the best action selected by agents. Moreover, the mixing network embodies collective intelligence and efficacy, and thus provides a clear path toward general goals.

Here, two tactics are performed for model training. One is sharing network parameters among all agents, which reduces the total number of parameters, accelerates training process, and promotes consistency; the other is individually training parameters for each agent, which will increase model's flexibility because each agent can learn behavioral strategies with more customization.

The output of the transformer is a sequence of hidden states generated by the multi-head attention mechanism. Each hidden state is transformed by a feedforward neural network to obtain the local Q value. The local Q values are fed into the mixing network, which generates the global Q value according to the monotonicity constraint described above. This process ensures that the features extracted by the Transformer are effectively integrated through the mixing network while preserving the IGM condition.

In previous work, agents' experiences are usually stored in experience replay memory, and agents are commonly trained by random uniform sampling, often resulting in high variance. Meanwhile, due to the introduction of hidden states, the uncertainty level increases. To address this issue, the experience replay is replaced with the prioritized experience replay, where the uncertainty of samples is performed by dynamically adjusting the probability of experience sampling. Concretely, by employing techniques derived from Rainbow's prioritized experience replay [53,54] and TD learning [55,56], the temporal

correlations between experience samples are broken, thereby the variance is reduced and the training stability is ultimately enhanced.

Based on the size of TD error, prioritized experience replay can adjust the sampling probability of each sample, and greater weights will be allocated to samples with larger TD error. Note that TD error reflects the difference between the prediction of current policy and the actual reward, and thus the TD error of each experience sample can be calculated by

$$p_i \propto |R + \gamma \max Q_{\theta^-}(s', u') - Q_\theta(s, u)|, \tag{D3}$$

where θ^- represents the parameters of the target network. The TD error is applied to compute the priority of each experience sample, and larger TD errors correspond to samples with higher priorities that are more valuable for network training. Here, the priority is denoted by

$$P(i) = \frac{(p_i^\alpha)}{\Sigma_k p_k^\alpha}, \tag{D4}$$

where $P(i)$ is the priority of the sample i, and p_i is the TD error of sample i; α is a hyperparameter with positive value, which controls the balance of the priority distribution. Obviously, the sample priority is directly proportional to the α-th power of the TD error; when $\alpha = 0$, all samples own equal priority, which equals to uniform sampling from the experience replay pool; when $\alpha = 1$, the sample priority is just proportional to the TD error. Since the sample with larger TD error is more likely to be selected for training, the network can easily adapt to high-error situations.

To further facilitate sampling efficiency, the priority of all samples is normalized, and the total weight of all samples is set as 1. During the sampling, the selection probability of each sample is determined by its priority with respect to the weight. The selected samples are used for network training, where back propagation is performed and the loss function is defined as

$$\mathcal{L}(\theta) = \sum_i^p [(r_i + \gamma max_{\mathbf{u}'} Q_{tot}(\tau', \mathbf{u}', s'; \theta^-) - Q_{tot}(\tau, \mathbf{u}, s; \theta))^2],$$

where p is the size of the replay buffer. When the TD error is extremely high, agents have to struggle to accurately estimate long-term dependencies, leading to the unstable update of the value function. Such instability may manifest as considerable fluctuations in the value function across different time steps, affecting the reliability of the training process accordingly. Here, we use a parameterized TD(λ) method to address this issue, where the value of λ ranges between 0 and 1. The λ value determines the trade-off between the rewards of the current time step and the predictions of multiple future time steps; when λ is 0, only the reward of the current time step is considered; when $\lambda = 1$, however, the rewards of all future time steps will be taken into account.

Obviously, TD(λ) can be viewed as a sliding window over time steps, and can effectively estimate the value function of the current state by considering reward signals from multiple time steps within a certain range, in another word, TD(λ) allows agents to flexibly obtain reward signals from multiple time steps during training.

Appendix E Simulation

Appendix E.1 Experimental setup

Based on the real-time strategy game StarCraft II, StarCraft Multi-Agent Challenge (SMAC) is a comprehensive and popular benchmark for evaluating MARL methods. To evaluate the performance of QICM, all simulation experiments are conducted on SMAC. And all simulations are performed using default hyperparameters and run on Windows 11 with a 3.20-GHz Intel® Core™ i9-12900KF CPU, 32-GB RAM, an NVIDIA GeForce GTX 3090Ti GPU and StarCraft II with version SC2.4.6.2.69232. All the results represent an average of 10 training runs under different seeds with a 95% confidence interval. Here, the confidence interval is calculated using the mean and standard deviation (SD) of the win rates from all the training samples, where the confidence interval = $mean \pm (1.96 \times SD/sqrt(10))$.

QICM will be compared with state-of-the-art value-based MARL algorithms, including QPLEX, QMIX and QTRAN as well as policy-based MARL algorithms such as MAPPO and COMA. All simulation experiments are conducted on a set of StarCraft II micromanagement scenarios. Considering the characteristics of maps, we select nine maps and categorize them as easy (3m, 2s3z, and 3s5z), hard (1c3s5z, 2c_vs_64zg, and 3s_vs_5z), and super hard (MMM2, 3s5z_vs_3s6z, and 6h_vs_8z). A complete map list is presented in Table E1.

Each algorithm will be trained until convergence or a maximum time step is reached. For easy and hard maps, the maximum time step is 2 million (2M); for super hard maps, the maximum time step is 10 million (10M). An episode will be finished when all agents on either side are destroyed or when the episode reaches the predetermined time limit T. Each agent selects action based on a greedy strategy after each 2000 steps of network training.

The key hyperparameters we use in our experiments, such as λ in TD(λ) and α in PER, are based on common settings for the vast majority of relevant work, not on experimental tuning. We choose the value of λ to be 0.8 [54, 57] and the value of α to be 0.6 [58, 59].

The objective of all MARL methods is to find an optimal decision strategy with limited resources and time, and maximize team's gain. Here, the specific reward scheme is set as follows: the reward is 10 points when an enemy unit is killed, and 200 points when all enemy units are destroyed. And all cumulative rewards are normalized to within 20 points.

Table E1 Map List

Map Name	Category	Ally Units	Enemy Units
3m	Easy	3 Marines	3 Marines
2s3z		2 Stalkers and 3 Zealots	2 Stalkers and 3 Zealots
3s5z		3 Stalkers and 5 Zealots	3 Stalkers and 5 Zealots
1c3s5z	Hard	1 Colossus, 3 Stalkers, and 5 Zealots	1 Colossus, 3 Stalkers, and 5 Zealots
2c_vs_64zg		2 Colossi	64 Zerglings
3s_vs_5z		3 Stalkers	5 Zealots
MMM2	Super hard	1 Medivac, 2 Marauders, and 7 Marines	1 Medivac, 3 Marauders, and 8 Marines
3s5z_vs_3s6z		3 Stalkers and 5 Zealots	3 Stalkers and 5 Zealots
6h_vs_8z		6 Hydralisks	8 Zealots

Table E2 The influence of λ on the win rate

λ	win rate
0.6	0.813
0.8	0.844
0.9	0.826

Appendix E.2 Experimental results and analysis

The performance of the MARL methods are evaluated by the win rate, which refers to the proportion of defeating runs to the total runs. Figure E3 illustrates the win rate on nine maps, where shaded regions are plotted with a 95% confidence interval. One can see that QICM effectively explores and consistently performs as the best method on almost all maps, especially in those complex maps (Fig. E1 (e)-(i)), such as map 3s_vs_5z which is characterized by a relatively high complexity and requires superior strategic planning. Due to the adaptability and complex policy formulation of QICM, resulting in the distinct advantages of QICM over other baselines on map 3s_vs_5z. In contrast, QTRAN and COMA are constrained by their value function approximation strategies and critic structures, leading to their relatively poor performances.

On the other hand, MAPPO, QPLEX, QMIX, QTRAN, and COMA also exhibit certain advantages on specific maps. For instance, MAPPO, renowned for its stability and scalability, exhibits high win rate values on maps including 3m and 2s3z, which suggests that MAPPO is efficacious on easy or less strategically demanding maps. The fluctuating performance across other maps also indicates MAPPO's limitations in coping with environments that present complex agent interactions or that require sophisticated temporal coordination.

QPLEX, with a novel value decomposition tactic, shows proficiency on maps including 3m, 2s3z, 3s5z, 1c3s5z, and 3s5z_vs_3s6z. Nevertheless, the inconsistent performances across other maps suggest that QPLEX will encounter potential challenges in value function estimation of complex strategic settings or when agents' role need to be precisely delineated.

The performance of QMIX is notable, particularly on map 3s5z, where the win rate value expeditiously achieves a commendable 90%. Generally, QMIX maintains a strong performing position as the second-best, reflecting the robust value function approximation of QMIX in cooperative scenes. However, QMIX's lower performance on maps 3s_vs_5z and 6h_vs_8z highlights its potential limitations on asymmetric maps.

The performance of QTRAN is generally moderate, with the exception of better outcomes on maps 2c_vs_64zg and MMM2. Due to the relaxed constraint conditions, resulting in a suboptimal value function approximation of QTRAN, particularly on complex maps that strategic depth and tight coordination are all paramount.

COMA's reasonable performance on map 3m contrasts with its underwhelming performances on other maps. This discrepancy is traced back to COMA's critical structure, leading to nonideal policy updates due to the sequential nature of these updates. Such a mechanism is less effective on maps that demand concurrent learning and adaptation among multiple agents.

In short, the convergence speed of QICM may not be the fastest because of the high volume of information which always leads to increased uncertainty and extended training time. But the introduction of experience prioritization and TD(λ) mitigates the amount of time spending on uncertainty.

In order to better verify the sensitivity analysis, we chose the super hard map 3s5z_vs_3s6z as the map of sensitivity analysis, and used the win rate as the main evaluation index. In the sensitivity analysis, we fixed other hyperparameters and only adjusted the target hyperparameters respectively. The experimental results are shown in Table E2 and Table E3 respectively.

It can be seen that with the increase of λ the performance of QICM first improves and then decreases (Table E2).

Table E3 The influence of α on the win rate

α	win rate
0.4	0.75
0.6	0.844
0.8	0.743

The best performance is achieved at λ =0.8. The value of sensitivity about λ: $S(\lambda_1) = \frac{\Delta W_r}{\Delta \lambda} = \frac{0.844-0.813}{0.8-0.6} = 0.155$, $S(\lambda_2) = \frac{\Delta W_r}{\Delta \lambda} = \frac{0.826-0.844}{0.9-0.8} = -0.18$.

The value of sensitivity about α: $S(\alpha_1) = \frac{\Delta W_r}{\Delta \alpha} = \frac{0.844-0.75}{0.6-0.4} = 0.47$, $S(\alpha_2) = \frac{\Delta W_r}{\Delta \alpha} = \frac{0.743-0.844}{0.8-0.6} = -0.505$.

Table E2 and Table E3 show that QICM is more robust to λ change and more sensitive to α change near the optimal hyperparameter value.

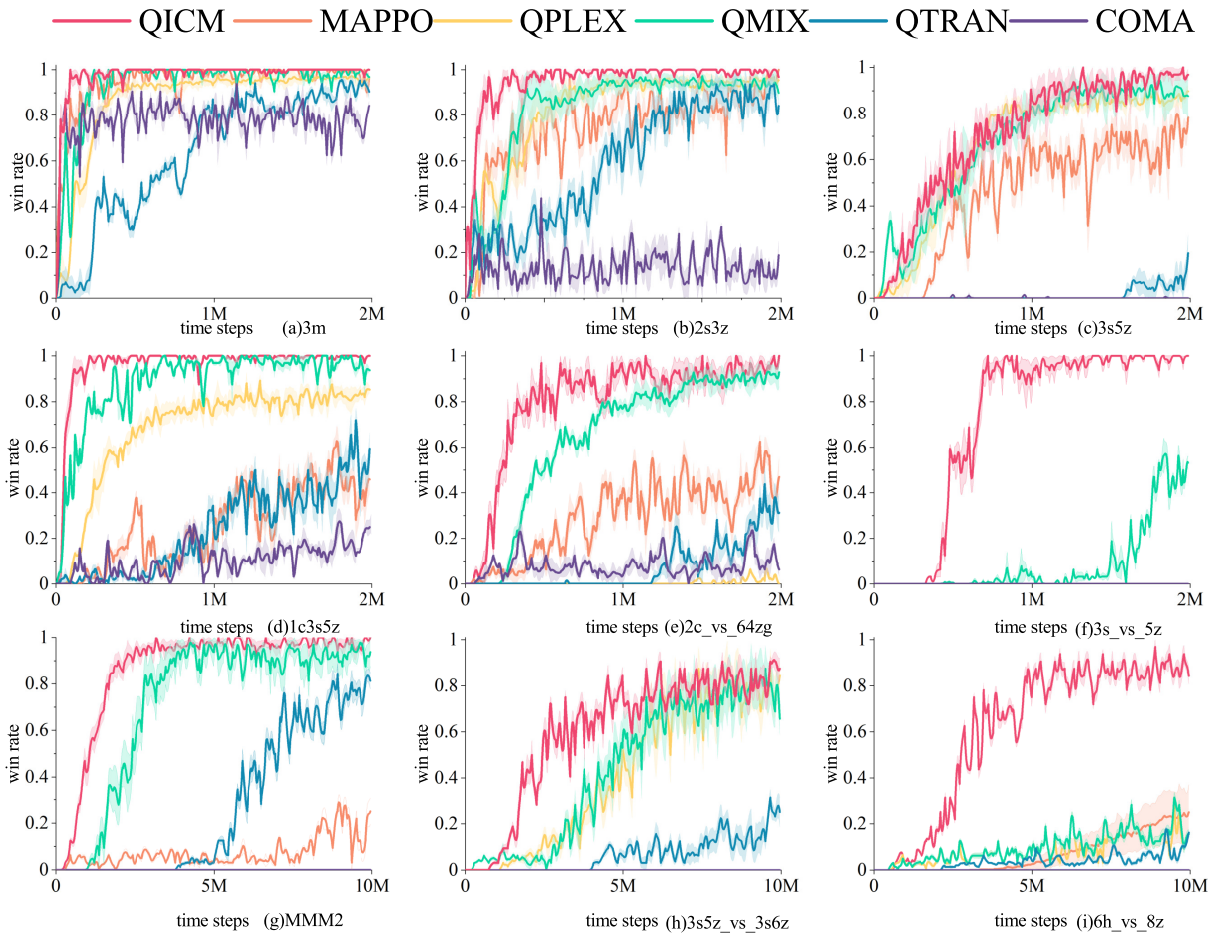

Figure E1 The win rates under QICM, MAPPO, QPLEX, QMIX, QTRAN, and COMA on nine maps over 10 runs, where shaded areas are displayed with a 95% confidence interval.

However, through multiple experiments and observations, we noticed that the win rate curve has limitations in some cases. Particularly, at certain time steps, the win rates of different algorithms may overlap, making it difficult to intuitively show the subtle differences and actual effects of the algorithms. To validate its practicality and effectiveness, we introduced the friendly survival rate and enemy mortality rate, all of which are inherently normalized with values in the range of [0,1]. The friendly survival rate is the number of surviving friendly units divided by the initial number of friendly units; the enemy mortality rate is the number of dead enemy units divided by the initial number of enemy units. We also compared QICM with MAPPO, QPLEX, QMIX, QTRAN and COMA, and the friendly survival rate and the enemy mortality rate of different algorithms at different time steps are displayed, using the average of five training runs with different seeds (Figure E2 and E3).

It can be seen that both the friendly survival rate and enemy mortality rate increase with the number of time steps in all experiments. This indicates that longer time steps help improve the friendly survival rate regardless of the method used. Specifically, QICM's friendly survival rate and enemy mortality rate are significantly superior to that of other algorithms, with enemy mortality rate reaching 0.99436, and friendly survival rate reaching 0.78246.

Appendix E.3 Ablation study

Firstly, in QICM, component TD(λ) places more emphasis on immediate returns and reduces long-term dependencies. When component TD(λ) is not included, only the reward of the current time step is considered while future rewards will be disregarded.

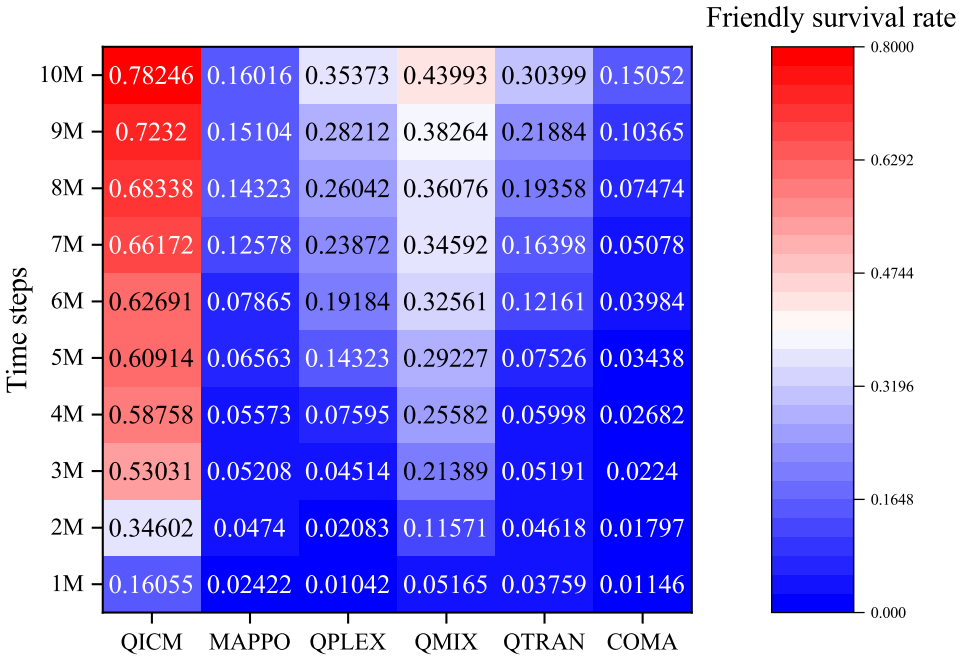

Figure E2 The friendly survival rate at different time steps.

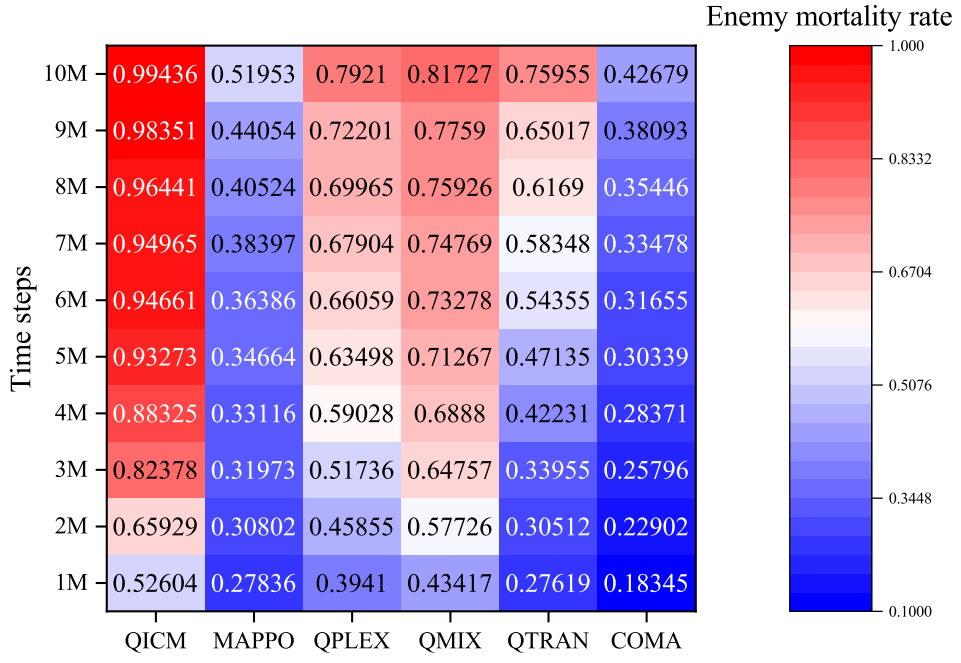

Figure E3 The enemy mortality rate at different time steps.

Secondly, when the prioritized experience replay component is not adopted to allocate sampling probabilities, all experience samples will be randomly sampled for training with equal probability, namely, uniform sampling is employed.

Thirdly, the self-attention mechanism can reduce some of the limitations with respect to observability and achieve better performances in the presence of large uncertainties. To investigate whether a combined approach that adopts RNN generated hidden states followed by a transformer is superior to an approach using hidden states alone, the self-attention mechanism in QICM is removed.

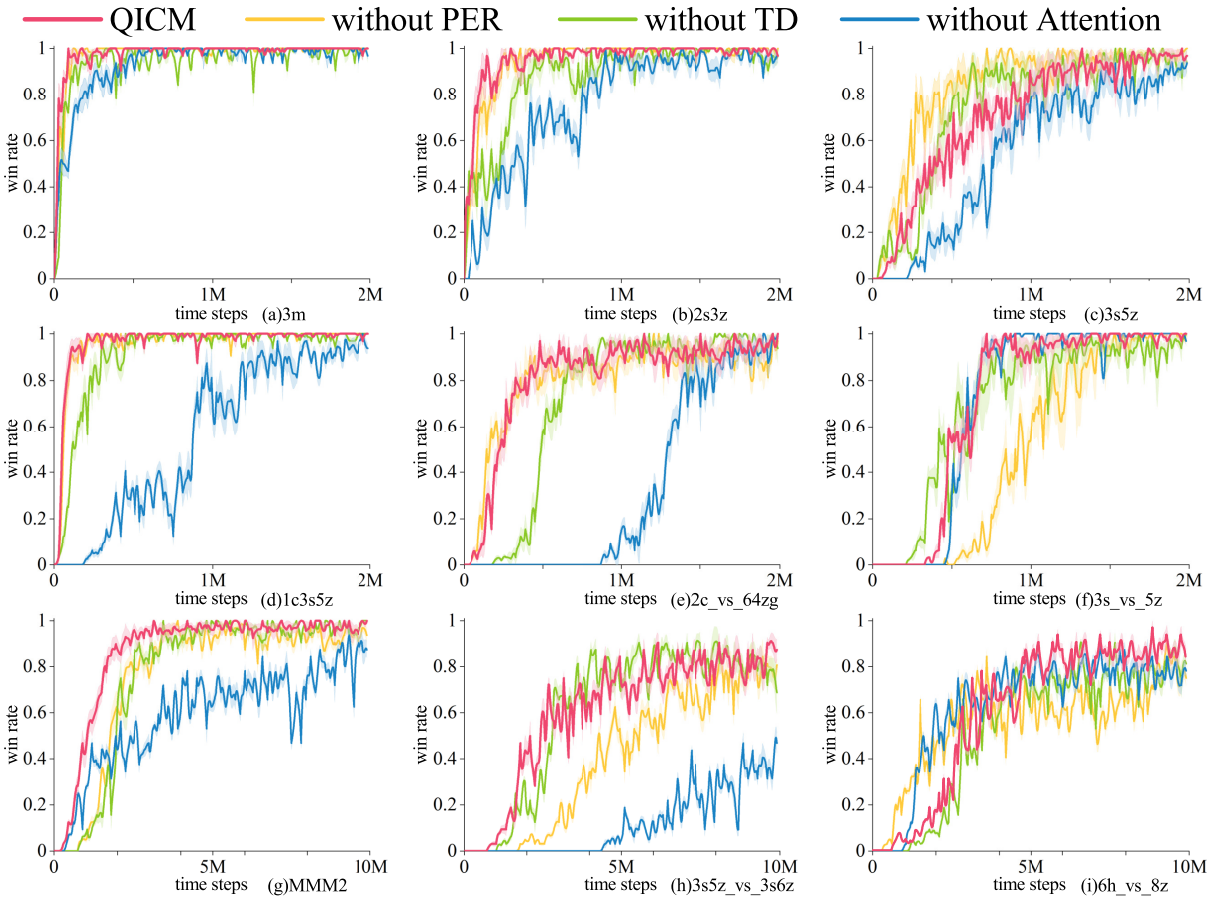

Figure E4 The win rates under QICM, with and without key components with respect to different maps.

To clearly demonstrate the connotation of QICM and highlight the impact of each component on QICM's performance, we conduct ablation experiments by considering three methods: QICM without prioritized experience replay (without PER), QICM without TD(λ) (without TD), and QICM without self-attention (without Attention). Similarly, each method will be verified on nine maps.

The win rates under QICM, with and without these key components are plotted in Figure E4. We can see that QICM demonstrates a high win rate and a short average convergence time, which suggests that QICM possesses strong generalization capabilities and adapts well to various maps and opponent strategies. Particularly, QICM's advantages are pronounced on three maps: 3s_vs_5z, MMM2, and 6h_vs_8z, indicating that QICM can better access environmental information, plan long-term strategies, and learn value functions via integrating the three components.

The complexity of different scenarios significantly affects the effectiveness of key QICM modules, such as the attention mechanism and TD learning. In 2c_vs_64zg, this map is categorized as Hard and involves 2 Colossi fighting against 64 Zerglings. The key challenge of this map lies in the overwhelming number of enemy units, which requires precise positioning and efficient area-of-effect attacks from the Colossi. The attention mechanism in QICM may not provide a significant advantage in this map. The primary requirement is not complex coordination but rather effective area-of-effect (AoE) damage control, which can be achieved by simpler mechanisms. This explains why the performance of without attention is worse than that of QICM. In 3s5z_vs_3s6z, this map is categorized as Super Hard and involves 3 Stalkers and 5 Zealots fighting against 3 Stalkers and 6 Zealots. The key challenge here is the numerical disadvantage and the need for highly precise micro-management to maximize unit survivability and damage output. In this map, the state-space complexity is high due to the need for precise individual unit control. However, without attention, the hidden states lose expressiveness, limiting the effectiveness of both PER and TD methods and leading to significantly worse performance compared to QMIX and QICM.

The complexity of different scenarios significantly affects the effectiveness of key QICM modules, such as TD learning. In 2c_vs_64zg and 3s5z_vs_3s6z, the performance of without TD learning surpasses that of QICM during the middle stages of training. This indicates that TD learning may occasionally hinder faster convergence in highly complex environments due to its reliance on bootstrapped value estimates, which can propagate inaccuracies over time.

In map like 3s_vs_5z, we observe that QICM experiences a temporary performance drop or slower improvement in the middle stages of training compared to without attention. However, it is important to note that QICM ultimately converges to the highest performance level, outperforming all other conditions.

On the other hand, one can see that the performance of without PER is clearly poorer than that of QICM on most maps, which indicates that the prioritized experience replay is vital for QICM's learning. Its absence will slow down the convergence rate, resulting in a lower win rate, particularly noticeable on complex map 6h_vs_8z. This highlights the role of the prioritized experience replay in handling complex decision-making environments. Actually, the prioritized experience replay accelerates the learning process by focusing on more surprising or less frequently occurred experiences, leading to a more informative condition for agents.

Furthermore, one can see that the results of without TD exhibit an obvious performance decrease, which suggests that TD(λ), a λ-return learning method, is vital for future reward estimation and long-term planning. In particular, the performance decline is much obvious on maps that require more sophisticated planning and strategy, such as map MMM2, where the long-term benefits must be well balanced with the immediate rewards.

From the performance of without Attention, one can see that this variant performs poorly on all maps, particularly on those challenging maps. Actually, the attention mechanism allows QICM to dynamically focus on different aspects of the input, and it is crucial for QICM to process the partial observability of the environment. When the attention module is removed, the effectiveness of PER and TD methods may be limited, resulting in performance degradation. Therefore, its absence will lead to QICM's inability to changing environment and opponent strategies.

Comparing the above results, we can obtain that the three components all contribute to QICM's improved performance. To be specific, the attention mechanism appears to be the most critical component, followed by the prioritized experience replay and TD(λ). This hierarchy suggests that the attention mechanism provides a foundational processing advantage, however, learning efficiency and long-term planning, respectively enabled by the prioritized experience replay and TD(λ), are also indispensable to QICM's success.

Similarly, we evaluated the friendly survival rate and enemy mortality rate in the ablation experiments. At all time steps, the QICM method's friendly survival rate and enemy mortality rate are always higher than that of the other three methods. The without Attention method performs the worst at all time steps, indicating that the self-attention mechanism significantly contributes to the friendly survival rate and enemy mortality rate, and its absence significantly reduces model performance. When the PER and TD modules were removed, both friendly survival rate and enemy mortality rate decreased significantly, indicating that the modules played an important role in improving sampling efficiency and training effectiveness. The specific results are shown in Figures E5 and E6, respectively.

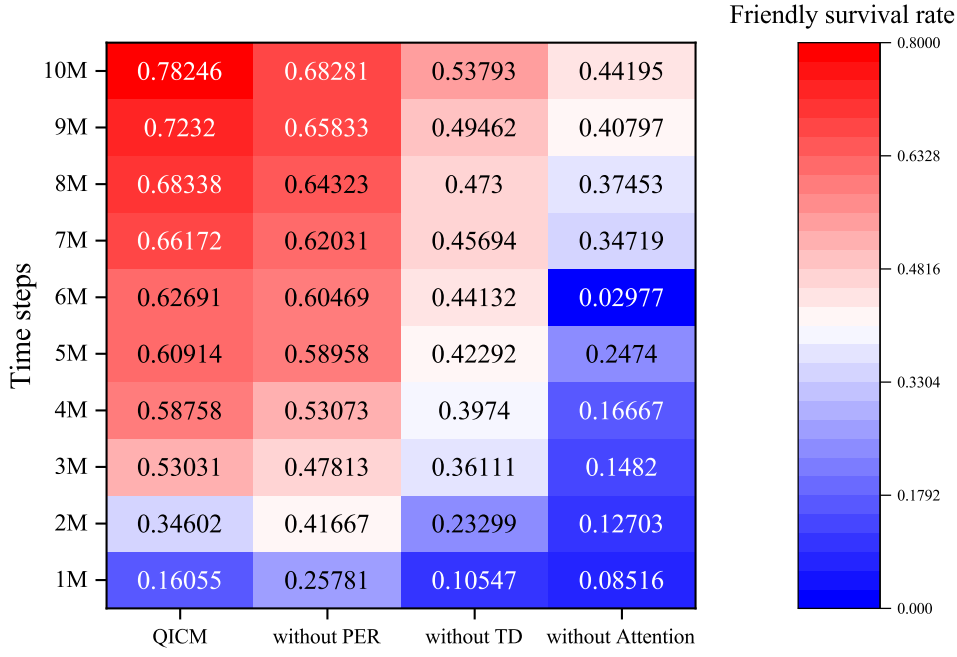

Figure E5 The friendly survival rate of ablation experiments at different time steps.

In a word, the self-attention mechanism, the prioritized experience replay, and TD(λ) play distinct and complementary roles in enhancing the performance of QICM. These crucial components can make QICM to efficiently learn in complex environments, strategically plan over long term time steps, and robustly adapt to variable maps and opponent strategies.

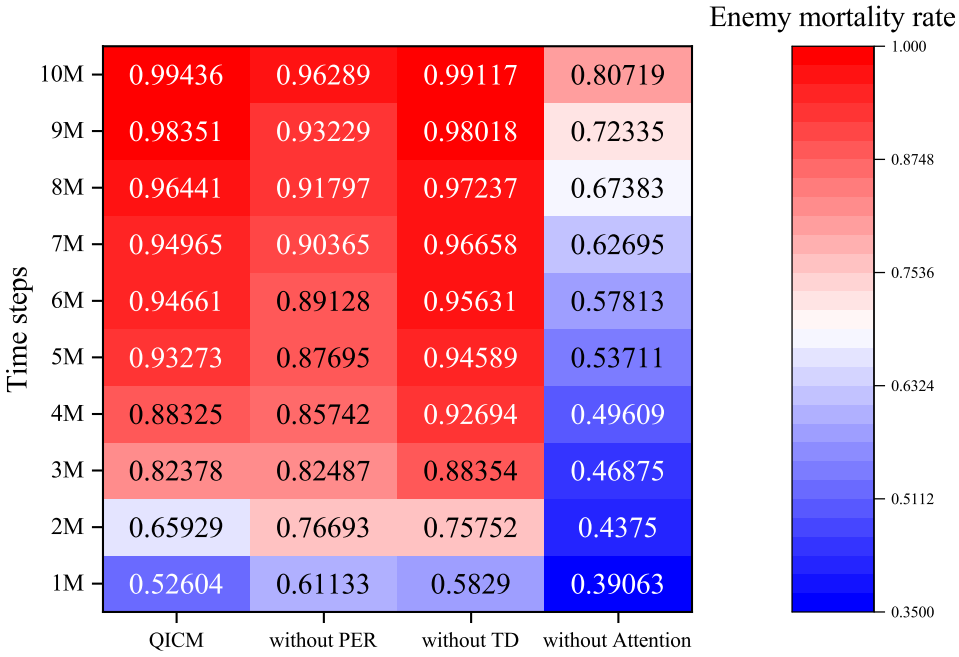

Figure E6 The enemy mortality rate of ablation experiments at different time steps.

References

1 Yang Y, Wang J. An overview of multi-agent reinforcement learning from game theoretical perspective. arXiv preprint arXiv:2011.00583, 2020
2 Hernandez-Leal P, Kartal B, Taylor M E. A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems, 2019, 33(6): 750-797
3 Yu D, Zhai J, Jin X, et al. Cooperative Game-based Intelligent Actions Making for Constrained Multi-agent System. IEEE Transactions on Artificial Intelligence, 2024.3522866
4 Wang Z, Mu C, Hu S, Chu C, et al. Modelling the Dynamics of Regret Minimization in Large Agent Populations: a Master Equation Approach. Proceedings of international joint conference on artificial intelligence, 2022: 534-540
5 Wang E S, Guo J, HONG C, et al. Cooperative confrontation model of UAV swarm with random spatial networks. Journal of Beijing University of Aeronautics and Astronautics, 2023, 49(1): 10-16
6 Wang E S, Liu F, Hong C, et al. MADRL-based UAV swarm non-cooperative game under incomplete information. Chinese Journal of Aeronautics, 2024, 37(6): 293-306
7 Wang E S, Liu F, Hong C, et al. MASAC-based confrontation game method of UAV clusters. China Science: Information Sciences, 2022, 52: 2254-2269
8 Kiran B R, Sobh I, Talpaert V, et al. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(6): 4909-4926
9 Mannion P, Duggan J, Howley E. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. Autonomic road transport support systems, 2016: 47-66
10 Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. nature, 2019, 575(7782): 350-354
11 Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680, 2019
12 Kasenberg D, Scheutz M. Interpretable apprenticeship learning with temporal logic specifications. 2017 IEEE 56th Annual Conference on Decision and Control (CDC), 2017: 4914-4921
13 Samvelyan M, Rashid T, De Witt C S, et al. The starcraft multi-agent challenge. arXiv preprint arXiv:1902.04043, 2019
14 Berrueta T A, Pinosky A, Murphey T D. Maximum diffusion reinforcement learning. Nature Machine Intelligence, 2024: 1-11
15 Nair R, Tambe M, Roth M, et al. Communications for improving policy computation in distributed POMDPs. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004: 1098-1105
16 Zhao W, Zhao Y, Li Z, et al. Optimistic multi-agent policy gradient. arXiv preprint arXiv:2311.01953, 2023
17 Bernstein D S, Givan R, Immerman N, et al. The complexity of decentralized control of Markov decision processes. Mathematics of operations research, 2002, 27(4): 819-840
18 Tuyls K, Verbeeck K, Lenaerts T. A selection-mutation model for q-learning in multi-agent systems. Proceedings of the second international joint conference on Autonomous agents and multiagent systems, 2003: 693-700
19 Puterman M L. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014
20 Oliehoek F A, Amato C. A concise introduction to decentralized POMDPs. Cham, Switzerland: Springer International Publishing, 2016
21 Wang Z, Jusup M, Guo H, et al. Communicating sentiment and outlook reverses inaction against collective risks. Proceedings of the National Academy of Sciences, 2020, 117(30): 17650-17655
22 Wang Z, Jusup M, Shi L, et al. Exploiting a cognitive bias promotes cooperation in social dilemma experiments. Nature communications, 2018, 9(1): 2954
23 Jia D, Guo H, Song Z, et al. Local and global stimuli in reinforcement learning. New Journal of Physics, 2021, 23(8): 083020
24 Oliehoek F A, Spaan M T J, Vlassis N. Optimal and approximate Q-value functions for decentralized POMDPs. Journal of Artificial Intelligence Research, 2008, 32: 289-353
25 Foerster J, Nardelli N, Farquhar G, et al. Stabilising experience replay for deep multi-agent reinforcement learning. International conference on machine learning. PMLR, 2017: 1146-1155

26 Oroojlooy A, Hajinezhad D. A review of cooperative multi-agent deep reinforcement learning. Applied Intelligence, 2023, 53(11): 13677-13722
27 Tan M. Multi-agent reinforcement learning: Independent vs. cooperative agents. Proceedings of the tenth international conference on machine learning, 1993: 330-337
28 Tampuu A, Matiisen T, Kodelja D, et al. Multi-agent cooperation and competition with deep reinforcement learning. PloS one, 2017, 12(4): e0172395
29 Hausknecht M, Stone P. Deep recurrent q-learning for partially observable mdps. 2015 aaai fall symposium series, 2015
30 Fuji T, Ito K, Matsumoto K, et al. Deep multi-agent reinforcement learning using dnn-weight evolution to optimize supply chain performance. 51th Hawaii International Conference on System Sciences, 2018
31 Chua K, Calandra R, McAllister R, et al. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. Advances in neural information processing systems, 2018, 31
32 Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients. Proceedings of the AAAI conference on artificial intelligence, 2018, 32(1)
33 Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 2017, 30
34 Yu C, Velu A, Vinitsky E, et al. The surprising effectiveness of ppo in cooperative multi-agent games. Advances in Neural Information Processing Systems, 2022, 35: 24611-24624
35 Peng B, Rashid T, Schroeder de Witt C, et al. Facmac: Factored multi-agent centralised policy gradients. Advances in Neural Information Processing Systems, 2021, 34: 12208-12221
36 Liu S, Lever G, Merel J, et al. Emergent coordination through competition. arXiv preprint arXiv:1902.07151, 2019
37 Li W, Jin B, Wang X, et al. F2a2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning. Journal of Machine Learning Research, 2023, 24(178): 1-75
38 Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296, 2017
39 Rashid T, Samvelyan M, De Witt C S, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 2020, 21(178): 1-51
40 Son K, Kim D, Kang W J, et al. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. International conference on machine learning. PMLR, 2019: 5887-5896
41 Wang J, Ren Z, Liu T, et al. Qplex: Duplex dueling multi-agent q-learning. arXiv preprint arXiv:2008.01062, 2020
42 Zhou H, Lan T, Aggarwal V. Pac: Assisted value factorization with counterfactual predictions in multi-agent reinforcement learning. Advances in Neural Information Processing Systems, 2022, 35: 15757-15769
43 Zhang K, Yang Z, Basar T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of reinforcement learning and control, 2021: 321-384
44 Amato C, Oliehoek F. Scalable planning and learning for multiagent POMDPs. Proceedings of the AAAI Conference on Artificial Intelligence, 2015, 29(1)
45 Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation, 1997, 9(8): 1735-1780
46 Chung J, Gulcehre C, Cho K, et al. Gated feedback recurrent neural networks. International conference on machine learning. PMLR, 2015: 2067-2075
47 Lambrechts G, Bolland A, Ernst D. Informed POMDP: Leveraging additional information in model-based RL. arXiv preprint arXiv:2306.11488, 2023
48 Subramanian J, Mahajan A. Approximate information state for partially observed systems. 2019 IEEE 58th Conference on Decision and Control (CDC), 2019: 1629-1636
49 Hu H, Foerster J N. Simplified action decoder for deep multi-agent reinforcement learning. arXiv preprint arXiv:1912.02288, 2019
50 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems, 2017, 30
51 Phan T, Ritz F, Altmann P, et al. Attention-based recurrence for multi-agent reinforcement learning under stochastic partial observability. International Conference on Machine Learning. PMLR, 2023: 27840-27853
52 Zhang S, Cao J, Yuan L, et al. Self-motivated multi-agent exploration. arXiv preprint arXiv:2301.02083, 2023
53 Hessel M, Modayil J, Van Hasselt H, et al. Rainbow: Combining improvements in deep reinforcement learning. Proceedings of the AAAI conference on artificial intelligence, 2018, 32(1)
54 Hu J, Jiang S, Harding S A, et al. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. arXiv preprint arXiv:2102.03479, 2021
55 Lim H D, Lee D. Temporal Difference Learning with Experience Replay. arXiv preprint arXiv:2306.09746, 2023
56 Barfuss W, Donges J F, Kurths J. Deterministic limit of temporal difference reinforcement learning for stochastic games. Physical Review E, 2019, 99(4): 043305
57 Modayil J, White A, Sutton R S. Multi-timescale nexting in a reinforcement learning robot. Adaptive Behavior, 2014, 22(2): 146-160
58 Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015
59 Liu L, Xu L, Liu J, et al. Prioritized Experience Replay-Based DDQN for Unmanned Vehicle Path Planning. 2024 5th International Conference on Information Science, Parallel and Distributed Systems (ISPDS), 2024: 718-721