RIIT: Rethinking the Importance of Implementation Tricks in Multi-Agent Reinforcement Learning

Jian Hu[1] **Haibin Wu**[*2] **Seth Austin Harding**[*1] **Siyang Jiang**[*3] **Shih-wei Liao**[1]

[1]Department of Computer Science, National Taiwan University, Taipei
[2]Graduate Institute of Communication Engineering, National Taiwan University, Taipei
[3]Graduate Institute of Electrical Engineering, National Taiwan University, Taipei

Abstract

In recent years, Multi-Agent Deep Reinforcement Learning (MADRL) has been successfully applied to various complex scenarios such as computer games and robot swarms. We investigate the impact of "implementation tricks" of state-of-the-art (SOTA) cooperative QMIX-based algorithms. Firstly, we find that such tricks described as auxiliary details to the core algorithm, seemingly of secondary importance, have a major impact. This finding demonstrates that, after modest tuning, the QMIX attains extraordinarily high win rates and achieves SOTA in the StarCraft Multi-Agent Challenge (SMAC). Furthermore, we find that the consideration of QMIX's monotonicity condition is critical for cooperative tasks. Based on the above findings, we propose a new algorithm called: RIIT, which achieves SOTA among policy-based algorithms (allowing for convenient complex action space modeling). We open-sourced the code at https://github.com/hijkzzz/pymarl2.

1 INTRODUCTION

MADRL has seen increased interest in recent years due to its capability in allowing neural-network-based agents to learn to act in multi-agent environments through interactions. For discrete cooperative control tasks, value-based algorithms such as QMIX Rashid et al. [2018] which serves as the baseline, Qatten Yang et al. [2020], WQMIX Rashid et al. [2020], and QPLEX Wang et al. [2020a], and policy-based algorithms such as LICA Zhou et al. [2020] and DOP Wang et al. [2020b] have achieved SOTA in the SMAC Samvelyan et al. [2019], a standard benchmark for evaluating SOTA MADRL algorithms. These papers claim to significantly improve the performance of the baseline QMIX through enhanced or redesigned methods. However, they fail to acknowledge that they are using tricks in their implementation to achieve SOTA performance in their experiments. While one may assume that such "choices" are unimportant, there is some evidence that they are, in fact, vital for good performance.

These algorithms are indeed comparable through open-source code; however, due to inconsistent comparison standards and a lack of a reliable baseline, it remains very difficult for researchers to judge whether they provide performance improvements because of a genuinely improved architecture or because of implementation tricks. Their experiment results may be inaccurate or misleading and stand as a hindrance to future MADRL research.

The main contributions in the paper are: (1) We study how implementation tricks affect the test performance of QMIX and provide some suggestions for tuning. After modest tuning, QMIX can attain extraordinarily high win rates in all hard scenarios in SMAC and achieve SOTA. (2) In investigating implementation tricks and how they affect fairness in MADRL experiments, we find that consideration of QMIX's monotonicity condition is critical for cooperative tasks, which is contrary to the claims of previous SOTA works. (3) Based on the above findings, we propose the new policy-based algorithm (allowing for convenient modeling of complex action spaces) called: RIIT (Sec. 5.3) that achieves SOTA in policy-based algorithms.

This paper is organized as follows: Sec. 2 discusses the main MADRL algorithms for the discrete cooperative task. Sec. 3 illustrates the benchmark and evaluation metric. Sec. 4 demonstrate how the tricks affect the performance of QMIX. Sec 5 shows our analysis and proposed RIIT algorithm. Sec. 6 and Sec. 7 are related works and conclusion, respectively.

2 BACKGROUND

Dec-POMDP. A fully cooperative multi-agent task can be described as a decentralized partially observable Markov de-

*Seth, Haibin and Siyang contributed equally to this work.

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

cision process (Dec-POMDP) Ong et al. [2009] composed of a tuple $G=\langle\mathscr{S},\mathscr{U},P,r,\mathscr{Z},O,N,\gamma\rangle$. $s\in S$ describes the true state of the environment. At each time step, each agent $i\in\mathscr{N}:=\{1,\ldots,N\}$ chooses an action $u_i\in\mathscr{U}$, forming a joint action $u\in\mathscr{U}^N$. All state transition dynamics are defined by function $P(s'\mid s,u):\mathscr{S}\times\mathscr{U}^N\times\mathscr{S}\mapsto[0,1]$. Each agent has independent observation $z\in\mathscr{Z}$, determined by observation function $O(s,i):\mathscr{S}\times\mathscr{N}\mapsto\mathscr{Z}$. All agents share the same reward function $r(s,u):\mathscr{S}\times\mathscr{U}^N\to\mathbb{R}$ and $\gamma\in[0,1)$ is the discount factor. Given that π_i is the policy of agent i, the objective of the joint agent is to maximize:

$$J(\pi)=\mathbb{E}_{u_1\sim\pi^1,\ldots,u^N\sim\pi^N,s\sim T}\left[\sum_{t=0}^{\infty}\gamma_t r_t\left(s_t,u_t^1,\ldots,u_t^N\right)\right] \quad (1)$$

CTDE. The centralized training and decentralized execution paradigm (CTDE) Kraemer and Banerjee [2016] allows the learning process to utilize additional state information. CTDE allows the learning algorithm to access all local action observation histograms and global states and share gradients and parameters. However, in the execution stage, each individual agent can only access its local action observation history τ^i. Next, we introduce the CTDE MADRL algorithms for the discrete control tasks.

2.1 IQL

Independent Q-learning(IQL) Tan [1993] breaks down a multi-agent task into a series of simultaneous single-agent tasks that share the same environment, just like multi-agent Deep Q-networks (DQN) Mnih et al. [2013]. DQN represents the action-value function with a deep neural network parameterized by θ. DQN uses a replay buffer to store transition tuple $\langle s,u,r,s'\rangle$, where state s' is observed after taking action u in state s and obtaining reward r. DQN learns by sampling a multitude of transitions from the replay buffer and minimizing the mean squared temporal-difference (TD) error loss:

$$\mathscr{L}(\theta)=\frac{1}{2}\sum_{i=1}^{b}\left[\left(y_i-Q(s,u;\theta)\right)^2\right] \quad (2)$$

where the TD target value $y=r+\gamma\max_{u'}Q\left(s',u';\theta^-\right)$ and θ^- are the target network parameters copied periodically from the current network and kept constant for a number of iterations. However, IQL does not resolve non-stationarity Hernandez-Leal et al. [2017] in the multi-agent setting. Thus, unlike single-agent DQN, there is no guarantee of convergence even at the limit of infinite exploration.

2.2 QMIX

By contrast, QMIX [1] Rashid et al. [2018] aims to learn a joint action-value function Q_{tot} by minimizing the squared TD error. It decomposes the joint Q_{tot} value to each agent Q_i through a monotonic Q value mixing network. The relationship between Q_{tot} of the joint agent and Q_i of the invidual agents can be expressed as:

$$\begin{aligned}&Q_{tot}(s,u;\theta,\phi)\\=&g_\phi\left(s,Q_1\left(\tau^1,u^1;\theta^1\right),\ldots,Q_N\left(\tau^N,u^N;\theta^N\right)\right)\end{aligned} \quad (3)$$

$$\frac{\partial Q_{tot}(s,u;\theta,\phi)}{\partial Q_i\left(\tau^i,u^i;\theta^i\right)}\geq 0,\quad \forall i\in\mathscr{N} \quad (4)$$

Where ϕ is the trainable parameter of the monotonic mixing network, maximizing joint Q is precisely equivalent to maximizing individual Q_i, and therefore, the optimal individual action maintains consistency with the optimal joint action.

2.2.1 Shortage of QMIX.

However, the **monotonicity condition (Eq. 4)** limits the mixing network's expressiveness, which may fail to learn in the non-monotonic case Mahajan et al. [2020], as shown in Table 1 and Table 2.

12	-12	-12
-12	0	0
-12	0	0

(a) Payoff matrix

-12	-12	-12
-12	0	0
-12	0	0

(b) QMIX: Q_{tot}

Table 1: A non-monotonic matrix game which violates the monotonicity condition. QMIX may learn an incorrect Q_{tot} which has an incorrect argmax.

1	0
0	1

(a) Payoff matrix

1	1/3
1/3	1/3

(b) QMIX: Q_{tot}

1/3	1/3
1/3	1

(c) QMIX: Q_{tot}

Table 2: A non-monotonic matrix game (from Rashid et al. [2020]). The convergence of QMIX cannot be guaranteed . QMIX may converge to (b) or (c).

Therefore, the goal of Qatten, QPLEX, WQMIX, and LICA is to improve the expressiveness of QMIX and claimed to significantly improve performance.

2.3 QPLEX

QPLEX[2] Wang et al. [2020a] decomposes Q values into advantages and values based on Qatten, similar to Dueling-

[1]Code from: https://github.com/oxwhirl/pymarl
[2]Code from: https://github.com/wjh720/QPLEX

DQN Wang et al. [2016]:

$$\begin{aligned}\text{(Joint Dueling) } Q_{tot}(\tau,u) &= V_{tot}(\tau)+A_{tot}(\tau,u)\\ V_{tot}(\tau) &= \max_{u'} Q_{tot}\left(\tau,u'\right)\end{aligned} \tag{5}$$

$$\begin{aligned}\text{(Individual Dueling) } Q_i(\tau_i,u_i) &= V_i(\tau_i)+A_i(\tau_i,u_i)\\ V_i(\tau_i) &= \max_{u'} Q_i\left(\tau_i,u_i'\right)\end{aligned} \tag{6}$$

$$\frac{\partial A_{tot}(s,u;\theta,\phi)}{\partial A_i(\tau^i,u^i;\theta^i)} \geq 0, \quad \forall i \in \mathcal{N} \tag{7}$$

In other words, Eq. 7 (advantage-based monotonicity) transfers the monotonicity condition from Q values to advantage values; QPLEX thereby reduces limitation on the mixing network's expressiveness.

2.4 WQMIX

WQMIX[3] Rashid et al. [2020], just like Optimistically-Weighted QMIX (OW-QMIX), uses different weights for each sample to calculate the squared TD error of QMIX:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} w(s,\mathbf{u})\left(Q_{tot}(\tau,\mathbf{u},s)-y_i\right)^2 \tag{8}$$

$$w(s,\mathbf{u}) = \begin{cases} 1 & Q_{tot}(\tau,\mathbf{u},s) < y_i \\ \alpha & \text{otherwise.}\end{cases} \tag{9}$$

Where α is a hyperparameter, the authors prove that this way can solve QMIX's estimation error in non-monotonic cases.

2.5 LICA

Policy-based algorithms allow for convenient modeling of complex action spaces such as AlphaStar Vinyals et al. [2019] and continuous-discrete hybird control tasks Neunert et al. [2020]. **LICA**[4] Zhou et al. [2020] is an on-policy policy-based algorithm that removes the monotonicity condition through a policy mixing critic (Figure 1):

LICA's mixing critic is trained using squared TD error. With a trained critic estimate, the decentralized policy networks can then be optimized end-to-end simultaneously by maximizing $Q^{\pi}_{\theta_c}$ with the stochastic policies $\pi^i_{\theta_i}$ as inputs:

$$\begin{aligned}\max_{\theta} \mathbb{E}_{t,s_t,u_t^1,\ldots,\tau_t^n}\Big[&Q^{\pi}_{\theta_c}\left(s_t,\pi^1_{\theta_1}\left(\cdot \mid \tau_t^1\right),\ldots,\pi^n_{\theta_n}\left(\cdot \mid \tau_t^n\right)\right)\\ &+\mathbb{E}_i\left[\mathcal{H}\left(\pi^i_{\theta_i}\left(\cdot \mid \tau_t^i\right)\right)\right]\Big]\end{aligned} \tag{10}$$

[3]Code from: https://github.com/oxwhirl/wqmix

[4]Code from: https://github.com/mzho7212/LICA

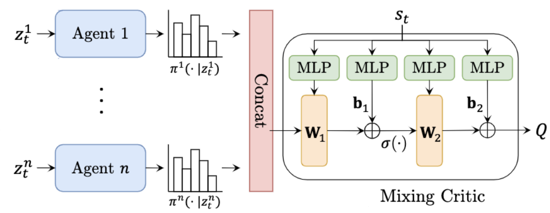

Figure 1: Architecture for LICA. LICA's mixing critic maps policy distribution to the Q value directly, in effect obviating the monotonicity condition. z denotes local action observation history.

where the gradient of entropy item $\mathbb{E}_i\left[\mathcal{H}\left(\pi^i_{\theta_i}\left(\cdot \mid z_t^i\right)\right)\right]$ is normalized by taking the quotient of its own modulus length: **Adaptive Entropy (Adapt Ent)**. Adaptive Entropy automatically adjusts the coefficient of entropy loss in different scenarios.

3 STUDY DESIGN

In this paper, we consider the setting of CTDE MADRL for cooperative discrete control. We investigate the SOTA MADRL algorithm tricks of QMIX and its improved variants: Qatten, QPLEX, WQMIX, LICA, and DOP.

3.1 BENCHMARK ENVIRONMENT

For our benchmark testing environment, we leverage SMAC, a ubiquitously-used multi-agent cooperative discrete control environment. SMAC consists of a set of StarCraft II micro scenarios used for evaluating how effectively independent agents can coordinate to solve complicated tasks. SMAC classifies micro scenarios into three difficulty levels: Easy, Hard, and Super Hard. The Super Hard scenario $6h\_vs\_8z$ is also hard to explore. Easy scenarios are usually simple to solve using the uncomplicated Value-Decomposition Networks (VDN) Sunehag et al. [2017]. Therefore, we opt to test using Hard and Super Hard scenarios. In past SMAC experiments Samvelyan et al. [2019], QMIX achieves a 0% win rate in three Super Hard scenarios: *corridor*, $3s5z\_vs\_3s5z$, and $6h\_vs\_8z$, and so it is meaningful to test on them.

3.2 EVALUATION METRIC

Our primary evaluation metric is the function that maps the steps for the environment observed throughout the training to the median winning percentage of the evaluation. We repeat each experiment with many independent training runs; results include median performance and percentiles ranging from 25% to 75%. We run the experiment 5 times independently in SMAC.

Our goal is to obtain as many samples from SMAC as possible in a short time so that we can accurately evaluate the convergence of each algorithm. We use **eight rollout processes** for parallel sampling [5].

4 IMPORTANCE OF TRICKS

An implementation trick is an optimization in code that is unaccounted for in the experimental design, but that may significantly affect the result. Our overarching goal is to understand the isolated performance of these MADRL algorithms more deeply. Therefore, we perform analysis on critically important settings and provide some suggestions for tuning.

4.1 OPTIMIZATION

Study description. QMIX and most of its variant algorithms use RMSProp to optimize neural networks because it proves stable in SMAC. However, lack of momentum may slow convergence of the network. We try to use Adam to optimize QMIX's neural network:

Figure 2: **Eight rollout processes are used for sampling**; samples are updated quickly. Adam significantly improves performance.

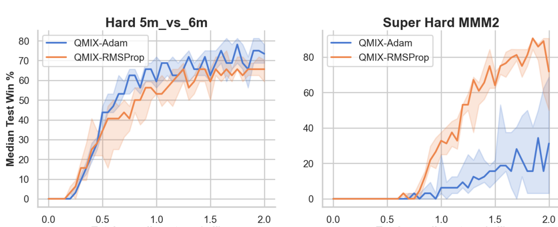

Figure 3: **Only one rollout process is used for sampling**; samples are updated slowly. The neural network optimized by Adam is prone to overfitting.

Interpretation. Figure 2 shows that Adam Kingma and Ba [2014] increases win rate by 100% on the Super Hard map *corridor*. This is because Adam boosts the network's convergence, allowing for full utilization of the large quantity of samples sampled in parallel. However, Figure 3 shows that when we only use one sampling process, the samples are updated slower than with eight processes (the replay buffer size is fixed), and the neural network becomes prone to overfitting.

Recommendation. Use Adam and quickly update the samples.

4.2 ELIGIBILITY TRACES

Study description. Return-based algorithms (where return refers to the sum of discounted rewards $\sum_t \gamma^t r_t$) offer some advantages over value bootstrap algorithms (where return refers to $r_t + V(s_{t+1})$): they are better behaved when combined with function approximation due to having a lower bias, and quickly propagating the fruits of exploration Sutton [1996]. However, they have a higher variance that can lead to instability in learning. Eligibility traces such as TD(λ) Sutton and Barto [2018], Q(λ) Peng and Williams [1994], and TB(λ) Precup [2000] achieve a balance between return-based algorithms and bootstrap algorithms, speeding up the convergence of reinforcement learning algorithms. Therefore, we study the application of Q(λ) in QMIX:

Q(λ) can be expressed as Eq. 11:

$$\begin{aligned} G_s^{\lambda} &\doteq (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{s:s+n} \\ G_{s:s+n} &\doteq \sum_{t=s}^{s+n}\gamma^{t-s}r_t + \gamma^{n+1}\max_u Q(x_{s+n+1}, u) \end{aligned} \tag{11}$$

where λ is the discount factor of the traces and $(\prod_{s=1}^{t}\lambda) = 1$ when $t = 0$. When λ is set to 0, it is equivalent to 1-step bootstrap returns. When λ is set to 1, it is equivalent to Monte Carlo Sutton and Barto [2018] returns.

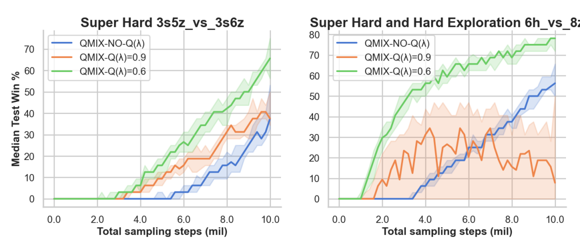

Figure 4: Q(λ) significantly improves performance of QMIX, but large values of λ lead to instability in the algorithm.

Interpretation. Value networks without sufficient training usually have a large bias that impacts bootstrap returns. This is particularly the case in QMIX because its monotonicity condition limits the model's expressiveness. Figure 4 shows that Q(λ) allows for faster convergence in our experiments by reducing this bias. However, large values of λ may lead to failed convergence due to variance and off-policy bias.

[5] Our experiments can collect 10 million samples within 9 hours with a Core i7-7820X CPU and a GTX 1080 Ti GPU.

Figure 4 shows that when λ is set to 0.9, it has a detrimental impact on the performance of QMIX.

Recommendation. Use Q(λ) with a small value of λ.

4.3 REPLAY BUFFER SIZE

Study description. In Atari games and DQN, the experience replay buffer size is usually set to a large value. However, in multi-agent tasks, as the action space becomes larger than that of single-agent tasks, the distribution of samples changes more quickly. We study the impact of the replay buffer size on the performance of QMIX.

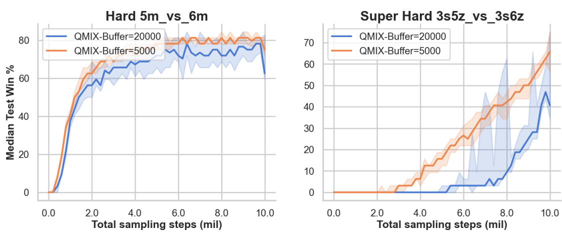

Figure 5: Setting the replay buffer size to 5000 episodes allows for QMIX's learning to be more stable than by setting it to 20000 episodes.

Interpretation. Figure 5 shows that a large replay buffer size makes QMIX's learning unstable. The causes of this phenomenon are as follows:

1. In multi-agent tasks, samples become obsolete faster than in single-agent tasks.
2. As we discussed in Sec. 4.1, Adam performs better with samples with fast updates.
3. Q(λ) converges more effectively with new samples than with old samples.

Recommendation. Use a small replay buffer size.

4.4 ROLLOUT PROCESS NUMBER

Study description. When we collect samples in parallel as is done in Advantage Actor-Critic (A2C), Stooke and Abbeel [2018] shows that when there is a defined total number of samples and either an inconsistent or an unspecified number of rollout processes, the median test performance becomes inconsistent. The goal of this study is to understand the impact of the number of processes on the final performance.

Interpretation. Under the A2C Mnih et al. [2016] training paradigm, the total number of samples can be calculated as follows:

$$S = E \cdot P \cdot I \tag{12}$$

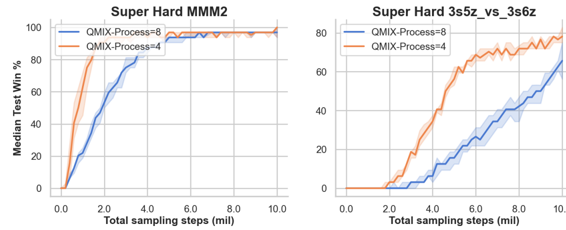

Figure 6: The results show that, given the total number of samples, fewer processes achieve better performance. We set the **replay buffer size to be proportional to the number of processes** to ensure that the novelty of the samples is consistent.

S is the total number of samples, E is the number of samples in each episode, P is the number of rollout processes, and I is the number of policy iterations. Figure 6 shows that we are given both S and E; the fewer the number of rollout processes, the greater the number of policy iterations Sutton and Barto [2018]; a higher number of policy iterations leads to an increase in performance. However, this also causes both longer training time and decreased stability.

Recommendation. Use fewer rollout processes when samples are difficult to obtain; otherwise, use more rollout processes.

4.5 EXPLORATION STEPS

Study description. In SMAC, some scenarios are hard to explore, such as $6h\_vs\_8z$, so the settings of ε-greedy become critically important. In this study, we analyze the effect of ε anneal period on performance:

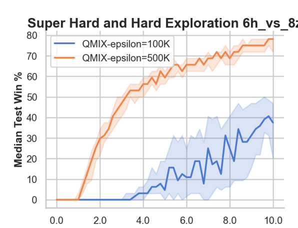

Figure 7: On the hard-to-explore scenario $6h\_vs\_8z$, a longer ε anneal period significantly improves performance.

Interpretation. As shown in Figure 7, increasing the length of the ε anneal period from 100K steps to 500K steps allows for a 38% increase in win rate in the Super Hard Exploration scenario $6h\_vs\_8z$.

Recommendation. Increase the length of the ε anneal period on hard-to-explore scenarios.

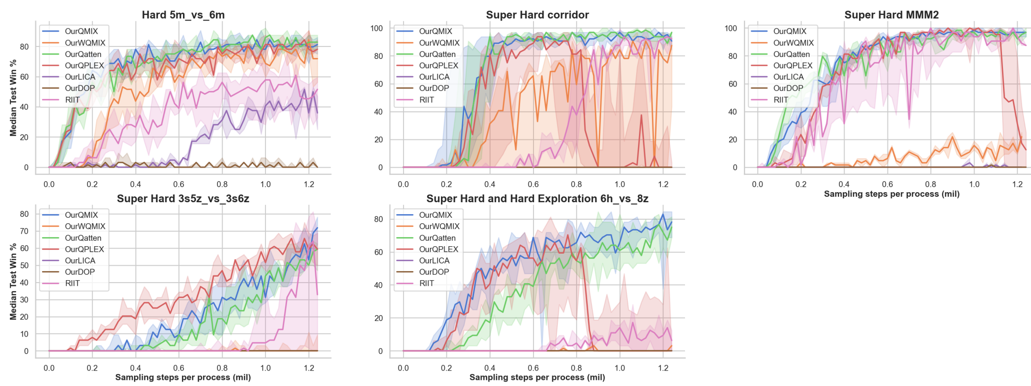

Figure 8: Median test win rate of SOTA MADRL algorithms. After unifying settings to leverage each of these tricks, QMIX achieves the best performance among all algorithms.

4.6 TRICKS OVERALL IMPACT

As Table 3 shows, after adding the tricks we mentioned before, OurQMIX ('Our' as in our customized settings) attains extraordinarily high win rates in all hard and super hard SMAC scenarios, far exceeding the QMIX-no-tricks.

Senarios	Difficulty	QMIX-no-tricks	OurQMIX
5m_vs_6m	Hard	84%	**90%**
3s_vs_5z	Hard	96%	**100%**
bane_vs_bane	Hard	**100%**	**100%**
2c_vs_64zg	Hard	**100%**	**100%**
corridor	Super Hard	0%	**100%**
MMM2	Super Hard	98%	**100%**
3s5z_vs_3s6z	Super Hard	3%	**75%**
27m_vs_30m	Super Hard	56%	**100%**
6h_vs_8z	Super Hard	0%	**84%**

Table 3: Test results of OurQMIX and QMIX-no-tricks in all hard scenarios. These tricks tremendously improve upon QMIX's performance.

5 RETHINKING AND ANALYSIS

This section is organized into three parts: (1) Retest these SOTAs based on fair experimental criteria, which ensures the same tricks we mentioned before. (2) Rethinking and answering the following question: Does the monotonicity condition always limit the performance of QMIX? (3) Then we propose a simple algorithm: RIIT, which can achieve the SOTA in policy-based algorithms, and also proves the importance of the monotonicity condition. (4) We try to propose a hypothesis to explain the monotonicity condition.

5.1 NEW BENCHMARKS

Due to these algorithms use different standards and implementation tricks in experiments (as shown in A), making it difficult for researchers to evaluate the algorithms' isolated performance, we firstly define and unify the settings used across the following experiments:

1. We adjust the network size of each algorithm to a reasonable size.
2. We use Adam with a large batch size to optimize the neural networks of each algorithm.
3. To accelerate the convergence of the model, we use eligibility traces to estimate the TD target value for each algorithm.
4. We show all other changes made to the settings of these algorithms in Appendix A (Table 5 and 6).

We then retest every SOTA algorithm based on the basic setting; as shown in Table 4, the test results actually indicate that WQMIX and Qatten in isolated form lead to lower performance than QMIX. Figure 8 shows that QPLEX's policy collapses in the test of Super Hard *6h_vs_8z* and *corridor*. Excitingly, the convergence of QPLEX is faster than QMIX in *3s5z_vs_3s6z*, which we believe comes from its complex attention mechanism and a large number of parameters. Figure 8 and Table 4 also show that when sample size and network size are reduced, LICA has terrible performance; Surprisingly, the baseline QMIX achieves SOTA.

5.2 MONOTONICITY CONDITION

Based on the following findings:

1. Table 4 shows these new algorithms have weaker monotonicity condition and do not have better performance than QMIX.
2. de Witt et al. [2020] show that the monotonicity condition attains performance boosts in other cooperative tasks (the famous Multi-agent Particle World Lowe et al. [2020] and Multi-agent Mujoco de Witt et al. [2020]).

Algorithms	Critic(Mixing) Net Size	*5m_vs_6m*	*3s5z_vs_3s6z*	*corridor*	*6h_vs_8z*	*MMM2*
OurQMIX (VB)	41K	**90%**	**75%**	**100%**	**84%**	**100%**
OurQatten (VB)	58K	**90%**	62%	**100%**	68%	**100%**
OurQPLEX (VB)	152K	**90%**	68%	96%	78%	**100%**
OurWQMIX (VB)	247K	**90%**	6%	96%	78%	23%
OurLICA (PG)	208K	53%	0%	0%	3%	0%
OurDOP (PG)	122K	9%	0%	0%	0%	0%
RIIT (PG)	69K	62%	68%	**100%**	19%	**100%**

Table 4: Median test win rate of SOTA MADRL algorithms. PG denotes Policy-gradient; VB denotes Value-based. The Critic(Mixing) Net size is calculated under $6h\_vs\_8z$.

We speculate that the monotonicity condition is significant for cooperative tasks; maybe the majority of cooperative tasks already satisfy the monotonicity condition. Surprisingly, this conclusion is contrary to that of WQMIX, QPLEX, and LICA. In other words, the conclusions of previous works are derived from inconsistent experimental settings.

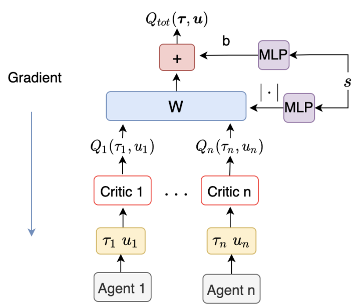

Figure 9: Architecture for RIIT. $|\cdot|$ **denotes absolute value operation**, implementing the monotonicity condition of QMIX. W denotes the nonnegative mixing weights.

5.3 RIIT

To further verify the importance of monotonicity and improve the performance of policy-based algorithms. We propose a new end-to-end off-policy Actor-Critic algorithm combine LICA and monotonicity network. We make the following optimizations based on LICA:

1. Replace the policy mixing critic in LICA with **monotonic mixing critic**, as shown in Figure 9.
2. First, use **offline samples** to only train the critic network with 1-step TD error loss.
3. Then, use **online samples** to train actor (by Eq.10) and critic with TD(λ).

This algorithm decomposes training into offline and online phases; it is named RIIT. Training actors with online samples improves learning stability (Cobbe et al. [2020] shows that actor-networks generally have a lower tolerance for sample reuse than the critic network). Furthermore, the offline training phase improves sample efficiency for the critic network. For the completely non-monotonic tasks, it can also be solved by removing the absolute value operation in the mixing network. Table 4 shows that RIIT performs best among all policy-based multi-agent algorithms. Figure 10 demonstrate that monotonicity condition significantly improve the performance of RIIT on the Super Hard scenarios $MMM2$ and $3s\_5z\_vs\_6z$.

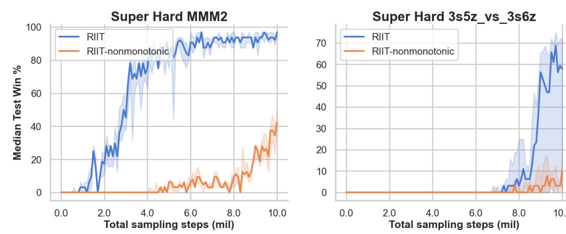

Figure 10: Comparing RIIT and RIIT without monotonicity condition (remove absolute value operation) on the mixing network of the critic.

5.4 EXPLANATION AND HYPOTHESIS

We propose a hypothesis to explain why the monotonicity condition will improve performance. As shown in Figure 11, (1) the outermost circle indicates the parameter space without any constraints; (2) the optimal solutions of most cooperative tasks and the parameter space with the monotonicity condition are highly overlapping. The monotonicity condition avoids searching for the optimal solution in invalid parameter space and, therefore, significantly improves the sample efficiency.

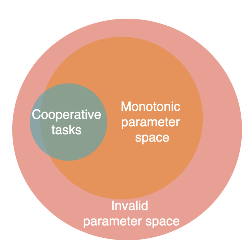

Figure 11: Our hypothesis on the relationship between monotonic parameter space and cooperative tasks.

Algorithms	LICA	**OurLICA**	DOP	**OurDOP**	RIIT
Optimizer	Adam	**Adam**	RMSProp	**Adam**	Adam
Batch Size(episodes)	32	32	Off=32, On=32	**Off=64, On=32**	Off=64, On=32
Eligibility traces	TD(λ=0.8)	**TD(λ=0.6)**	TD(λ=0.8), TB(λ=0.93)	**TD(λ=0.6), TB(λ=0.9)**	TD(λ=0.6)
Exploration	Adapt Ent=0.06	Adapt Ent=0.06	Anneal Noise = 500K steps	**Adapt Ent=0.0005**	Adapt Ent=0.03
Critic-Net Size	29696K	**208K**	122K	122K	69K
Rollout Processes	32	**8**	4	**8**	8

Table 5: Setting of Policy-based algorithms; Adaptive Entropy (Adapt Ent) is proposed by LICA.

Algorithms	QMIX-no-tricks	**OurQMIX**	Qatten	**OurQatten**	QPLEX	**OurQPLEX**	WQMIX	**OurWQMIX**
Optimizer	RMSProp	**Adam**	RMSProp	**Adam**	RMSProp	**Adam**	RMSProp	**Adam**
Batch Size	128	**128**	32	**128**	32	**128**	32	**128**
Q(λ)	0	**0.6**	0	**0.6**	0	**0.6**	0	**0.6**
Attention Heads	-	-	4	4	10	**4**	-	-
Mixing-Net Size	41K	41K	58K	58K	476K	**152K**	247K	247K
ε Anneal Steps			50K $\rightarrow$ **500K for** $6h\_vs\_8z$**, 100 K for others**					
Rollout Processes	8	**8**	1	**8**	1	**8**	1	**8**

Table 6: Setting of Value-based algorithm; Batch Size is calculated by episodes.

6 RELATED WORKS

The work most closely related to our paper is probably Engstrom et al. [2020], which investigates code-level optimizations based on PPO Schulman et al. [2017] code and concludes that the majority of performance differences between PPO and TRPO originate from code tricks. Andrychowicz et al. [2020] investigates the influence of tricks on the performance of PPO and provides tuning optimizations.

7 CONCLUSION

In this study, we investigate the influence of tricks on the performance of QMIX and provides tuning optimizations. We find that implementation tricks have a significant impact on MADRL experimental results and even lead to inaccurate conclusions: we find that the monotonicity condition is critical for the major cooperative tasks. Moreover, we find that policy-based algorithms perform poorly in discrete cooperative tasks. Therefore, we propose a new policy-based algorithm, RIIT, that achieves SOTA among policy-based algorithms, which also supports the monotonicity condition is significant.

A OMITTED SETTINGS

A.1 EXPERIMENTAL FAIRNESS

The implementation tricks and comparison standards of these SOTA MADRL algorithms are inconsistent; some of these tricks are shown below:

1. DOP sets its number of processes to half that of QMIX to get more policy iterations (DOP's comparison standard is defined as the test win rate given the total number of samples).
2. LICA sets the number of processes to 32 times that of QMIX and obtains a huge sample size (LICA's comparison standard is defined as test win rate given the number of samples per rollout process)
3. The enormous critic network of LICA is thousands of times larger than the mixing network of QMIX. Also, the mixing network of WQMIX and QPLEX is five times larger than that of QMIX.
4. LICA and DOP use eligibility traces to estimate TD target values but unfairly make a direct comparison with the performance of QMIX, which does not use eligibility traces.
5. These SOTA algorithms select test scenarios that specifically give them starting advantages or avoid the scenarios in which they fail.

We unify these settings for experimental fairness. **Table 5 and 6 shows our settings for the these algorithms**. The network size is calculated under $6h\_vs\_8z$, and 'Our' denotes the new settings.

A.2 LEARNING RATE AND OTHER SETTINGS

For LICA, we set the learning rate of the agent network to 0.0025 and the critic network's learning rate to 0.0005. For DOP, we set the agent network's learning rate to 0.0005 and the learning rate of the critic network to 0.0001. For RIIT, we set the learning rates of neural networks to 0.001.

For all value-based algorithms, the neural networks are trained with 0.001 learning rate, and we use ε-greedy action selection, decreasing ε from 1 to 0.05 over n-time steps (n can be found in Table 6) for exploration. Also, we update the target network every 200 episodes. Specifically, we use OW-QMIX in WQMIX as the baseline.

All mixing networks and agent networks are the same as in QMIX Rashid et al. [2018]. All the replay buffer size is set to 5000 episodes. For the discount factor, we set $\gamma = 0.99$. In the tuning stage, we experiment on more than five sets of settings for each algorithm.

References

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. *arXiv:2006.05990*, 2020.

Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. *arXiv preprint arXiv:2009.04416*, 2020.

Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Deep Multi-Agent Reinforcement Learning for Decentralized Continuous Cooperative Control. *arXiv preprint arXiv:2003.06709*, 2020.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. *arXiv:2005.12729*, 2020.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016. ISSN 09252312. doi: 10.1016/j.neucom.2016.01.031.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *arXiv preprint arXiv:1706.02275*, 2020.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-Agent Variational Exploration. *arXiv preprint arXiv:1910.07483*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783*, 2016.

Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, pages 735–751. PMLR, 2020.

Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Pomdps for robotic tasks with mixed observability. 5:4, 2009.

Jing Peng and Ronald J Williams. Incremental multi-step q-learning. In *Machine Learning Proceedings 1994*, pages 226–232. Elsevier, 1994.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1803.11485*, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding Monotonic Value Function Factorisation. *arXiv preprint arXiv:2006.10800*, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *arXiv preprint arXiv:1902.04043*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Adam Stooke and Pieter Abbeel. Accelerated methods for deep reinforcement learning. *arXiv preprint arXiv:1803.02811*, 2018.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *arXiv preprint arXiv:1706.05296*, 2017.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044, 1996.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex Dueling Multi-Agent Q-Learning. *arXiv:2008.01062*, 2020a.

Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. Off-Policy Multi-Agent Decomposed Policy Gradients. *arXiv:2007.12322*, 2020b.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

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.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning Implicit Credit Assignment for Multi-Agent Actor-Critic. *arXiv preprint arXiv:2007.02529*, 2020.