Improving Multi-agent Reinforcement Learning with Imperfect Human Knowledge

Xiaoxu Han[1,2], Hongyao Tang[1], Yuan Li[3(✉)], Guang Kou[2(✉)], and Leilei Liu[1]

[1] College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
{xiaoxu.han,bluecontra,liuleilei}@tju.edu.cn
[2] Artificial Intelligence Research Center, National Innovation Institute of Defense Technology, Beijing 100072, China
kg5188@163.com
[3] Academy of Military Sciences, Beijing 100091, China
yuan.li@nudt.edu.xn

Abstract. Multi-agent reinforcement learning has gained great success in many decision-making tasks. However, there are still some challenges such as low efficiency of exploration, significant time consumption, which bring great obstacles for it to be applied in the real world. Incorporating human knowledge into the learning process has been regarded as a promising way to ameliorate these problems. This paper proposes a novel approach to utilize imperfect human knowledge to improve the performance of multi-agent reinforcement learning. We leverage logic rules, which can be seen as a popular form of human knowledge, as part of the action space in reinforcement learning. During the trial-and-error, the value of rules and the original action will be estimated. Logic rules, therefore, can be selected flexibly and efficiently to assist the learning. Moreover, we design a new exploration way, in which rules are preferred to be explored at the early training stage. Finally, we make experimental evaluations and analyses of our approach in challenging StarCraftII micromanagement scenarios. The empirical results show that our approach outperforms the state-of-the-art multi-agent reinforcement learning method, not only in the performance but also in the learning speed.

Keywords: Multi-agent reinforcement learning · Logic rules · Exploration · StarCraftII

1 Introduction

Over the past few years, multi-agent reinforcement learning (MARL) has achieved significant progress in various tasks. However, there are still some problems that have not been solved. With the increase of agents, the policy space is dramatically expanded, and the simultaneous learning of multiple agents makes the environment non-stationary, which brings great difficulties to find a converged policy for each agent. Furthermore, due to the nature of reinforcement

I. Farkaš et al. (Eds.): ICANN 2020, LNCS 12397, pp. 369–380, 2020.
https://doi.org/10.1007/978-3-030-61616-8_30

learning (RL), learning from scratch, MARL algorithms learn good control policies only after millions of steps of very poor performance in simulation.

These problems are well-known and there has been much work focusing on them. An effective approach to these problems is leveraging human knowledge to guide learning, which can reduce the inefficient exploration of agents. In many tasks, humans usually have accumulated much useful experience, which should be well utilized. The way of combining human knowledge with RL has attracted much attention. A straightforward method [12] is to utilize logic rules in priority. If a rule is matched in any state, the agent will act following the rule without any judgment. The drawback of this approach resides in the excessive dependence on the quality of rules. In fact, human knowledge may be suboptimal. Therefore, how to improve learning with imperfect human knowledge has become a key problem. A natural idea is to balance the usage of human knowledge and the policy learned by trial-and-error by evaluating the value of the two action sources and reusing knowledge selectively during the learning process. [7,11] designed complicated mechanisms to calculate and update the confidence of knowledge and the learned policy, and the agent chooses actions from the two action sources accordingly. However, such a method increases the computational complexity, and it is inefficient to be applied in problems with large state space. Training additional action selector can also learn to select proper action source under a certain state. [2] trained a DQN model additionally to make decisions from knowledge-based policy or the policy learned by the A3C algorithm. Nevertheless, adding a network means increasing training difficulty and training time, contrary to the original intention of leveraging human knowledge.

In this paper, we propose a novel approach to improve MARL with high-level human knowledge represented in the form of logic rules. Different from previous methods that require designing complicated mechanisms to calculate confidences, we leverage the Q-value in RL as a uniform criterion to judge the value of rules and original actions. We expand the action space of RL with the selection of rules; once an extended action is chosen, its corresponding rule will be parsed and executed. Therefore, the Q-value of these two types of actions will be estimated and updated during the trial-and-error. In this way, agents have the ability to automatically balance the usage of rules and its own self-learned policy and utilize proper rules under certain states. Further, we find the traditional ϵ-greedy exploration cannot exploit the advantages of rules, so we propose Rule-Prioritized Exploration mechanism to accelerate learning by efficiently using the rules. Our approach can be easily combined with existing MARL algorithms to improve their performance. In this paper, we apply our method to QMIX [8], VDN [9], and IQL [10] and make experiments in challenging micromanagement scenarios of StarCraftII. The empirical results show that our approach achieves significant improvement in the performance of MARL algorithms.

2 Background and Related Work

2.1 Partially Observable Stochastic Games

In this paper we consider a *partially observable stochastic game* which can be defined as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}^1, \cdots, \mathcal{A}^n, \mathcal{T}, \mathcal{R}^1, \cdots, \mathcal{R}^n, \mathcal{O}^1, \cdots, \mathcal{O}^n \rangle$. $\mathcal{S}$ denotes the environmental information and the possible configurations of the environment. $\mathcal{A}_i$ is the set of available actions of agent i. $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function of agent i. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function which defines transition probability between global states and $\mathcal{O}^i$ is the set of observation of agent i.

At each time step, each agent i chooses an action $a_i \in \mathcal{A}^i$, forming a joint action $\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$. Policy $\pi_i : \mathcal{O}^i \times \mathcal{A}^i \to [0, 1]$ specifies the probability distribution over the action space of agent i. The goal of agent i is to learn the optimal policy π_i^* that maximizes the expected return with a discount factor γ: $E_{\pi_i}\{\sum_{t=0}^{\infty} \gamma^t r_t^i\}$. Let $\boldsymbol{\pi} = (\pi_i, \cdots, \pi_n)$ denote the joint policy of all agents. The state-action value function of an agent i under a joint policy $\boldsymbol{\pi}$ can be defined as $Q_i^{\boldsymbol{\pi}} = E_{\boldsymbol{\pi}}\{\sum_{t=0}^{\infty} \gamma^t r_t^i | \mathcal{O}, \mathcal{A}\}$.

2.2 MARL Algorithms

In this paper, we apply our approach to three representative MARL algorithms:

1. **IQL:** In this method, agents are trained independently and simultaneously in a common environment. The action-value function for each agent i is updated following (1). This method has a problem: each agent updates its policy independently, resulting in a non-stationary environment with no convergence guarantees even with infinite exploration.

$$Q_i(o_i, a_i) \leftarrow Q_i(o_i, a_i) + \alpha[r_i + \gamma max_a Q_i(o_i', a_i) - Q_i(o_i, a_i)] \tag{1}$$

2. **VDN:** While each agent learns individual Q_i independently, VDN learns a centralized but factored Q_{tot}. VDN assumes the joint action-value function for the multi-agent system can be additively decomposed into value functions across agents, see Eq. (2).

$$Q_{tot}((o^1, o^2, ..., o^N), (a^1, a^2, ..., a^N)) = \sum_{i=1}^{N} Q_i(o^i, a^i) \tag{2}$$

3. **QMIX:** Assuming that $\frac{\partial Q_{tot}}{\partial Q_i} \geq 0$, QMIX factors the joint action-value Q_{tot} into a monotonic non-linear combination of individual Q_i via a mixing network, as (3):

$$Q_{tot}((o^1, o^2, ..., o^N), (a^1, a^2, ..., a^N)) = M(Q_1(o^1, a^1), ..., Q_N(o^N, a^N)) \tag{3}$$

Where M represents the monotonic function of Q_{tot} and individual value functions $Q_i(o^i, a^i)$. Such monotonic decomposition ensures that a global $argmax$ performed on Q_{tot} yields the same result as a set of individual arg max operations performed on each individual Q_i.

2.3 Related Work

Different approaches to incorporating human knowledge into RL have been proposed. Some works consider improving the state representation in RL through embedding external knowledge. The knowledge graph has been used as a representation of high-dimensional information in RL. [1] expressed the entities and relationships in text-based adventure games as knowledge graphs. The sophisticated models developed for disposing domain knowledge had also been studied. [4] utilized the predication of dynamic models about the environment to improve the state representation for the policy network.

An important cluster of related research is Imitation Learning (IL), which aims to imitate expert behaviors from demonstration data. [6] introduced an adversarial training mechanism into IL, where the generator attempted to imitate expert and the discriminator distinguished between the fake sample and the expert sample. To address the 'cold start' problem, [5] leveraged a combinatorial loss function to pre-train neural network based on demonstration data and then updated the model with RL methods.

The form of human knowledge is not limited to the demonstrations but can be extended to logic rules. A wide range of expert systems making use of such rules have been developed, but many of them turn out to be ineffective. The main reason is that expert systems do not have the ability to learn. Several studies have been conducted in combining rules with RL. [3] leveraged rules to reduce the size of the state space by dividing state space into several patterns. Nevertheless, it is inefficient to classify all states, and the approach is only suitable for problems with extremely small state space. [13] interposed the training of DQN with a decaying probability to follow the rules. However, this approach cannot filter suboptimal rules so that it works only with high-quality rules.

3 Methodology

3.1 Extending Action Space with Rule Selection

Let $\Gamma = \{\Gamma_1, ...\Gamma_K\}$ represents the rule set consisting of K rules, which is shared among n agents. A mapping function set $F = \{f_1, ..., f_k, ..., f_K\}$, where the f_k represents a parsing map for rule Γ_k. The rule parsing map takes the current state as input and outputs the action to be performed under rule policy. For agent i with the length J of action space, $\dot{a}_i^j, 1 \leq j \leq J$ denotes agent's original actions. We extend the rule actions which are denoted with $\ddot{a}_i^k, 1 \leq k \leq K$ with the original action space. Thus, the action space of agent i becomes $a_i = \{\dot{a}_i^1, ...\dot{a}_i^J, \ddot{a}_i^1, ..., \ddot{a}_i^K\}$. Once a rule action $\ddot{a}_i^k$ is selected, the corresponding mapping function f_k will be triggered to find the corresponding action $\dot{a}_i^j \leftarrow f_k(\ddot{a}_i^k)$ under the rule policy Γ_k.

The rule actions and the original actions under certain states are both evaluated by the Q-value, which represents the expected cumulated reward from the current state to the final state by performing the current policy. Whether to use the original action or the rule action can be determined by Q_i, which is updated

following Eq. (4), where o_i is the current local observation state and o_i' is the next observation state for agent i.

$$\begin{aligned} Q_i(o_i, \dot{a}_i) &\leftarrow Q_i(o_i, \dot{a}_i) + \alpha[r_i + \gamma max_{\dot{a}_i} Q_i(o_i', \dot{a}_i) - Q_i(o_i, \dot{a}_i)], 1 \leq i \leq N \\ Q_i(o_i, \ddot{a}_i) &\leftarrow Q_i(o_i, \ddot{a}_i) + \alpha[r_i + \gamma max_{\ddot{a}_i} Q_i(o_i', \ddot{a}_i) - Q_i(o_i, \ddot{a}_i)], 1 \leq i \leq N \end{aligned} \tag{4}$$

3.2 Rule-Prioritized Exploration

A traditional exploration strategy is ϵ-greedy. In this method, exploration and exploitation divide the probability of choosing actions into two sections, and the probability of exploration ϵ is decaying during learning. During exploration, ϵ-greedy does not distinguish between actions, and the probability of each action being explored is uniform.

However, treating rule actions and original actions indiscriminately cannot exploit the advantages of rules. Therefore, we design a new mechanism, Rule-Prioritized Exploration, to distinguish the exploration of the rule action and the original action. To encourage agents to explore from rule actions, we define a parameter δ to balance the probability of choosing the rule actions and original actions. The probability interval of choosing action in our method is shown in Fig. 1. Notice that the probability for selecting rule actions is also decaying with the decline of ϵ, which means the agent explores rules preferentially at the early training process.

Fig. 1. The probability interval of Rule-Prioritized Exploration

The procedure of Rule-Prioritized Exploration is shown in Algorithm 1. Function $rand()$ is used to generate a random number. If it is in the range of 0 and $\epsilon * \delta$, the rule action is selected. If the number is in the range of $\epsilon * \delta$ and ϵ, an

Algorithm 1. Rule-Prioritized Exploration

Input: Exploration probability δ, ϵ, original action set $\{\dot{a}_i^j\}_{j=1}^J$, and rule action set $\{\ddot{a}_i^k\}_{k=1}^K$
Output: a

```
if 0 ≤ rand() ≤ ϵ * δ then
    a is sampled from {ȧ_i^j}_{j=1}^J
else if ϵ * δ ≤ rand() ≤ ϵ then
    a is sampled from {ä_i^k}_{k=1}^K
else if rand() ≥ ϵ then
    a = argmax_{a'} Q(o, a')
end if
```

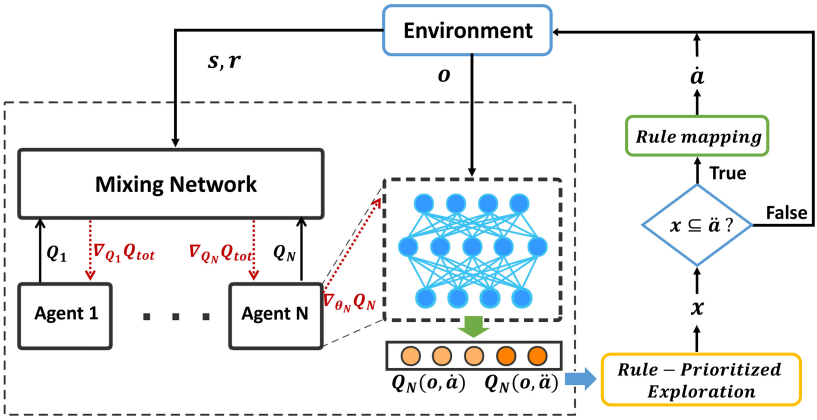


Fig. 2. The overall architecture of RMLPE-QMIX. Red dashed lines indicate the gradient flow. (Color figure online)

action (either a rule action or an original action) is randomly selected. In this way, the rule action is well used during the exploration. At last, if the number is bigger than ϵ, the action that maximizes Q_i will be chosen.

3.3 Rule Mixing Learning with Prioritized Exploration

We define the integration of the above two methods as Rule Mixing Learning with Prioritized Exploration method (RMLPE). Our approach can be easily applied to various MARL algorithms: the agents share a predefined rule set Γ, and rule actions are added in the action space of each agent. The Q-values for rule actions and original actions will be used to make decisions following at each time step, and updated under the network of agent i based on the feedback of the environment. To illustrate the whole procedure of RMLPE, we take QMIX as an example and show the architecture of the method combining RMLPE with QMIX (RMLPE-QMIX) in Fig. 2.

The network of agent i takes its local observation information o_i as input and outputs Q_i, which contains $Q_i(o_i, \dot{a}_i)$ and $Q_i(o_i, \ddot{a}_i)$. According to the rule prioritized exploration, agent i will select an action x. Then $x \in \{\dot{a}_i^1, ...\dot{a}_i^J\}$ or $x \in \{\ddot{a}_i^1, ..., \ddot{a}_i^K\}$ will be determined as presented in Algorithm 1. If x represents one of rule actions $\ddot{a}_i^k$, the mapping f_k will be triggered to figure out the corresponding executable primitive action $\dot{a}_i^j$ and execute. Otherwise, x will be executed directly. No matter x belongs to which kind of action source, the transition $\langle o_t, x, r, o_{t+1} \rangle$ will be stored in the replay buffer. The mixing network and decentralized network of each agent will be updated through sample transitions. The pseudocode of RMLPE is described in Algorithm 2.

Algorithm 2. Rule Mixing Learning with Prioritized Exploration

Input: Rule set Γ, MARL model, rule mapping function set F, the exploration probability δ and ϵ

Output: MARL model with RMLPE method

Initialize MARL model with random weights θ and the replay buffer D
while not done **do**
  for agent i **do**
    Get current observation o_i for each agent i
    Compute Q_i according to MARL models based on o_i and select an action x_i following *Rule-Prioritized Exploration*
    if $x_i \in \{\dot{a}_i^j\}_{j=1}^{J}$ **then**
      Execute x_i in the environment
    end if
    if $x_i \in \{\ddot{a}_i^k\}_{k=1}^{K}$ **then**
      Execute $f_k(x_i) \in \Gamma_i$ in the environment
    end if
    Get reward r_t and next observation o_{next}
    Store transition (o_t, x_t, r_t, o_{t+1}) in D
    Train MARL models (e.g., IQL, QMIX) with mini-batch samples from replay buffer D
  end for
end while

4 Experimental Setup

4.1 Environments

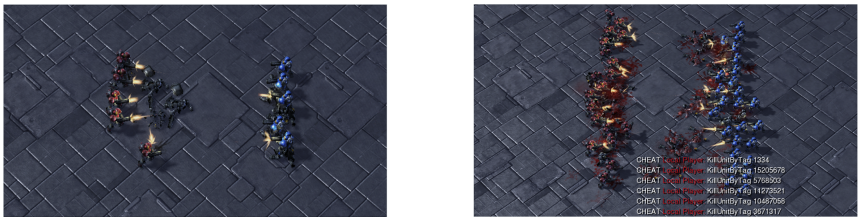
Our experiments are carried out on StarCraft Multi-Agent Challenge(SMAC), an open-source environment[1] for evaluating MARL approaches. SMAC contains various micromanagement combat scenarios in which each of the learning agents controls an individual army unit based only on local observations, and the opponent's units are controlled by the handcrafted heuristics.

We adopt two challenging combat scenarios: 5 Marines and 6 Marines ($5m$ v $6m$), 27 Marines and 30 Marines ($27m$ v $30m$). Both scenarios are asymmetric, in which the enemy army outnumbers the allied army by one or more units. The environment is partially observable: each unit has a certain sight range based on its local observation, and the information about allied or enemy units out of range cannot be received. The action space consists of the following set of discrete actions: *noop*, *stop*, *move*[*direction*], *attack*[*enemy id*]. The *attack*[*enemy id*] action is available only if the enemy is within shoot range. Figure 3 shows the screenshots of the scenarios.

4.2 Imperfect Rules

In our experiments, we leverage rules derived from human experience in firing strategy. The premise of firing is that there are enemies within the agent's shoot

[1] https://github.com/oxwhirl/smac.



(a) $5m$ v $6m$ (b) $27m$ v $30m$

Fig. 3. Screenshots of the combat scenrios in StarCraft II

range. The firing micromanagement of human players in combat games is: attack the nearest enemy or the enemy with the least hit points; attack an enemy with several companions to concentrate fire. Therefore, we incorporate the following rules into learning. For the third rule, we work out the number of allied units focusing fire on one enemy is 3, based on the firepower and hit points per unit.

1. IF *there are enemies within shoot range* THEN *attack the nearest enemy.*
2. IF *there are enemies within shoot range* THEN *attack the least hit points enemy.*
3. IF *there are enemies within shoot range* THEN *attack the enemy which is aimed by less than three allied units.*

These rules are imperfect for two reasons. Firstly, in a specific state, at most one of the three rules can take effect, which means these rules are mutually exclusive. Secondly, even if the condition of a rule is satisfied, it is merely an option as whether the rule is beneficial to the final performance is doubtful.

4.3 Experimental Settings

For the adopted scenarios in StarCraftII, the difficulty level of the opponent is set to be 7, *very difficult*. Besides, our proposed approaches share the same learning parameters with the original MARL algorithms. The shared learning setting is as follows: we use the RMSprop optimizer with a learning rate of 5×10^{-4}, and the discounting factor γ is set to 0.99. The ϵ is annealed linearly from 1.0 to 0.05 over 50k time steps and is kept constant for the rest of the learning. The replay buffer contains the most recent 5000 episodes and a batch of 32 episodes will be sampled for training. Due to the partially observable settings, the architecture of each agent is a *Deep Recurrent Q-networks* (DRQN) with a recurrent layer comprised of a GRU with a 64-dimensional hidden state.

5 Experimental Results

Combined with various famous MARL algorithms, including QMIX, VDN, and IQL, RMLPE is evaluated in several experiments. We firstly illustrate the

improvements brought by RMLPE to three baseline algorithms. Then, we show RMLPE is capable of leveraging rules with different qualities to improve learning. Finally, we investigate the impact of the exploration ratio: δ.

The *test win rate* is leveraged as an indicator to evaluate the performance of methods. For each run of a method, we run 24 test episodes every 5000 training steps in which agents performing action selection greedily in a decentralized fashion. The percentage of episodes where the agents defeat all enemy units within the permitted time limit is referred to as the test win rate. Besides, all the experimental results are performed at least 5 runs with different seeds.

5.1 Improvement over Baselines

Figure 4 illustrates comparisons between RMLPE enabled and non-RMLPE enabled MARL algorithms in $5m$ v $6m$ and $27m$ v $30m$ scenarios. It is obvious that RMLPE improves the learning process of all the baseline MARL methods in two scenarios, at both the win rate and the learning speed. For the baseline algorithms, QMIX performs best, whereas IQL performs worst due to the non-stationary problem. To investigate the performance brought by pure rules, we design experiments where agents act following rules only. When the pre-condition of rules is satisfied, i.e., the conditions for firing are met, the agent will randomly select one rule to execute. In our design, the agents move toward the location of enemies. The performance of pure rules is presented in the dashed line in Fig. 4.

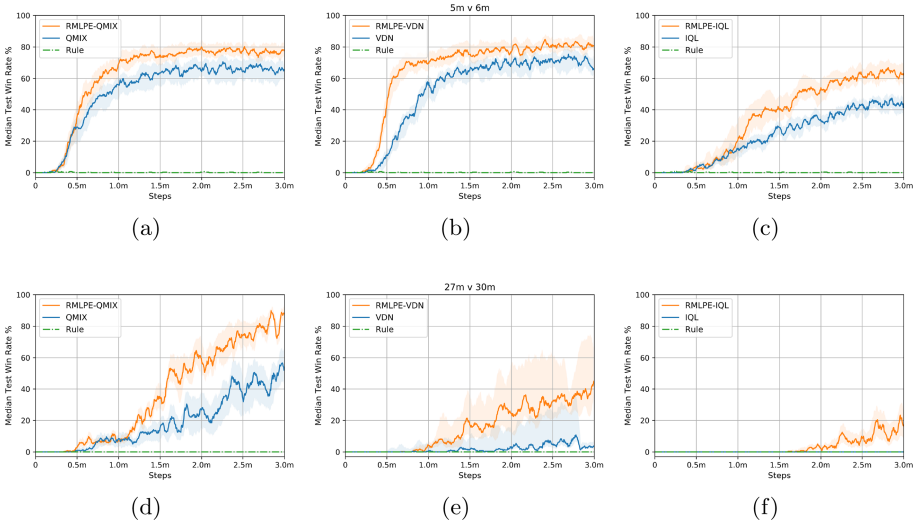


Fig. 4. Median win rate of different algorithms on two scenarios, [25%, 75%] percentile interval is shaded. (a), (b), (c) represent the comparisons of RMLPE-QMIX/VDN/IQL method and their baseline algorithms on the *hard* scenario: 5m v 6m. (d), (e), (f) show the results on the *super hard* scenario: 27m v 30m.

The results demonstrate that RMLPE can dramatically improve the learning process even with such poor performance rules.

5.2 Ability of Dealing with Imperfect Rules

To investigate the effectiveness of each rule in learning and to show RMLPE can optionally leverage uneven quality rules, we present the usage rate of rules during training. We record the number of times each rule is used for every 5000 train steps. The mean using frequency of rules in final 50k train steps is shown in Table 1, with the peak in bold. As we can see, most methods tend to use the second rule more frequently, indicating that RMLPE picks out the second rule which appears to be more effective in learning. As for the total rule usage, rules are used much less frequently in $27m$ v $30m$ than $5m$ v $6m$. Different dimensions of the action space between two scenarios cause this phenomenon. The action space in $27m$ v $30m$ is three times the size of $5m$ v $6m$, there are fewer times to choose rules as the action space gets bigger.

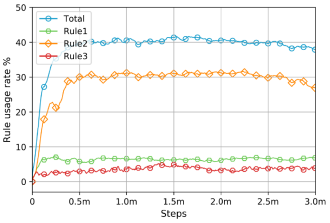
Table 1. Mean usage frequency of rules

Methods	5m v 6m				27m v 30m			
	Rule1	Rule2	Rule3	Total	Rule1	Rule2	Rule3	Total
RMLPE-QMIX	0.0667	**0.2830**	0.0392	0.3889	0.0537	**0.0730**	0.0536	0.1803
RMLPE-VDN	0.0762	**0.2850**	0.0432	0.4044	0.0158	**0.0270**	0.0899	0.1327
RMLPE-IQL	0.0464	**0.2800**	0.03852	0.3649	0.0045	0.0243	**0.0268**	0.0556

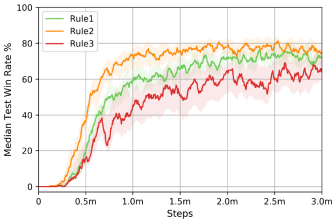
The curves in Fig. 5(a) show the mean using frequency of each rule during the training process of RMLPE-QMIX method in $5m$ v $6m$ scenario. In the beginning, all using frequencies increase significantly in both scenarios, due to the Rule-Prioritized Exploration. Then the using frequencies of *rule* 1 and *rule* 3 maintain a low level while that of *rule* 2 continues to increase quickly and converges at a high level. From the results of using frequency curve, we can find that *rule* 2 appears to the most useful rule for the training in $5m$ v $6m$. To further investigate the impact of each rule on training, we perform an experiment in which the rules in the RMLPE method are set to be *rule* 1, *rule* 2, *rule* 3 respectively and the results are shown in Fig. 5(b). It is obvious that in terms of contribution to the performance of RMLPE, *rule* 2 ranks top, *rule* 1 s and *rule* 3 third. These results illustrate that RMLPE can identify the useful rule from rule set and decide which rule should be applied under a different state.

5.3 Effect of the Exploration Ratio

The exploration parameter δ plays an important role in balancing the exploration between the rule action and the original action. In our experiments on combat tasks (see Fig. 4), we choose the best hyperparameters δ for each method independently. In this section, we make comparisons of the impact of different values



(a) Mean rule usage rate curve



(b) Median test win rate of only using one rule

Fig. 5. The mean rule usage rate curve and the median test win rate during the training of RMLPE-QMIX only with *rule* 1/2/3 respectively in $5m$ v $6m$ scenario.

of δ (varying from 0.1 to 0.5) on the performance of RMLPE. Figure 6 depicts the final mean win rate of RMLPE method combined with three baseline algorithms for each exploration ratio δ. Comparing to the baseline algorithms (dashed line in Fig. 6), we find that RMLPE enabled algorithms beat all the baselines, with any exploration ratio. The proper value of δ varies with the MARL methods, and it should not be too big or too small for most algorithms. Therefore a suitable value interval of exploring rules exists for the different methods.

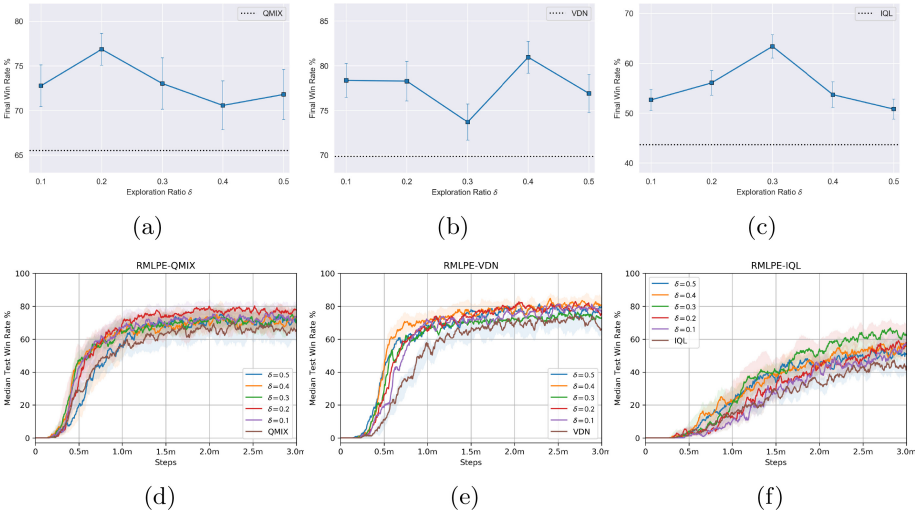


Fig. 6. The above figures: the median win rate in final 50k train steps for RMLPE-QMIX/VDN/IQL methods in $5m$ v $6m$ map, as a function of exploration ratio δ. The square markers for each ratio denote the mean win rate, and the error bars show [25%, 75%] percentile interval for the mean win rate. The dashed line represents the final win rate of the baseline MARL algorithms. (d), (e), (f) show the learning curves of RMLPE methods with different δ.

6 Conclusion

In this paper, we propose a novel approach called Rule Mixing Learning with Prioritized Exploration, RMLPE, to improve MARL by incorporating logic rules into the learning. The incorporated rules can be imperfect, that is, rules are not necessarily guaranteed to be effective under all circumstances. RMLPE can efficiently select useful rules and exploit them to facilitate learning by extending the action space with rules. A new exploration method is also proposed to accelerate learning in the early training stage. The evaluation of our methods is conducted on challenging StarCraftII micro scenarios and the results show RMLPE can greatly improve and accelerate the learning process of MARL methods even with suboptimal rules. In the future, it is worthwhile to investigate combining large-scale human knowledge with RL to solve more challenging problems.

References

1. Ammanabrolu, P., Riedl, M.O.: Playing text-adventure games with graph-based deep reinforcement learning. arXiv preprint arXiv:1812.01628 (2018)
2. Bougie, N., Ichise, R.: Deep reinforcement learning boosted by external knowledge. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, pp. 331–338 (2018)
3. Bougie12, N., Ichise, R.: Rule-based reinforcement learning augmented by external knowledge
4. Du, Y., Narasimhan, K.: Task-agnostic dynamics priors for deep reinforcement learning. arXiv preprint arXiv:1905.04819 (2019)
5. Hester, T., Vecerik, M., et al.: Deep q-learning from demonstrations. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
6. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems, pp. 4565–4573 (2016)
7. Moreno, D.L., Regueiro, C.V., et al.: Using prior knowledge to improve reinforcement learning in mobile robotics. In: Proceedings of the Towards Autonomous Robotics Systems, University of Essex, UK (2004)
8. Rashid, T., Samvelyan, M., et al.: QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, pp. 4292–4301 (2018)
9. Sunehag, P., Lever, G., et al.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, pp. 2085–2087 (2018)
10. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337 (1993)
11. Wang, Z., Taylor, M.E.: Interactive reinforcement learning with dynamic reuse of prior knowledge from human and agent demonstrations. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 3820–3827. ijcai.org (2019)
12. Zhang, G., Li, Y., et al.: Efficient training techniques for multi-agent reinforcement learning in combat tasks. IEEE Access **7**, 109301–109310 (2019)
13. Zhang, H., Gao, Z., et al.: Faster and safer training by embedding high-level knowledge into deep reinforcement learning. arXiv preprint arXiv:1910.09986 (2019)