KG-RL: A Knowledge-Guided Reinforcement Learning for Massive Battle Games

Shiyang Zhou[1,2], Weiya Ren[1,2(✉)], Xiaoguang Ren[1,2], Xianya Mi[1,2], and Xiaodong Yi[1,2]

[1] Artificial Intelligence Research Center, Defense Innovation Institute, Beijing 100072, China
[2] Tianjin Artificial Intelligence Innovation Center, Tianjin 300457, China

Abstract. In a multi-agent game, the complexity of the environment increases exponentially as the number of agents increases. Learning becomes difficult when there are so many agents. Mean field multi-agent reinforcement learning (MFRL) uses the average action of the neighbors to increase the input of the value network, which can be applied in the environment with hundreds of agents. However, inefficient exploration and slow convergence speed limit the performance of the algorithm. In this article, we propose a new **K**nowledge-**G**uided **R**einforcement **L**earning (KG-RL) method, which can be divided into *rule-mix* and *plan-extend*. We use the *rule-mix* to encode knowledge into plans which can reduce redundant information and invalid actions in the state. And the *plan-extend* can combine the result of *rule-mix* with reinforcement learning to achieve more efficient joint exploration. Through experiments in Magent environment, we prove that the win rate of our proposed KG-RL is 22% higher than that of knowledge-based decision tree and 39% higher than that of MFRL. Thus, the KG-RL can perform well in massive battle games due to its high exploration efficiency and fast convergence.

Keywords: Knowledge-guided · Reinforcement learning · Multi-agent · Massive battle games

1 Introduction

In recent years, multi-agent reinforcement learning (MARL) has made remarkable progress in various tasks [3,6,18]. However, learning in an environment of multiple agents is still fundamentally difficult, because agents not only interact with the environment, but also with each other [12]. With the increase number of agents, the policy space expands rapidly, and the simultaneous learning of multiple agents makes the environment non-stationary, which brings great difficulties for each agent to find the convergence policy [6], especially when the number of agents is huge.

Due to the limitation of the network size which increases linearly when the number of agents increases [11], the current popular methods, such as MADDPG [12],

D. N. Pham et al. (Eds.): PRICAI 2021, LNAI 13033, pp. 83–94, 2021.
https://doi.org/10.1007/978-3-030-89370-5_7

QMIX [15], etc., are often limited by the number of agents. Thus, the algorithm represented by Independent Q-learning [17] directly treats other agents and the environment as a whole. However, due to the instability caused by the other agents' changing policies, the algorithm cannot converge stably. MFRL [21] proposes a provably-converged mean-field formulation to scale up the actor-critic framework by feeding the state and the average value of nearby agents' actions to the critic, enhancing the stability of learning. However, the average actions of other agents in the execution process will be obtained by communication, which is not easy.

In addition, incorporating human knowledge into reinforcement learning is a good way. Purely knowledge-based approaches, such as decision trees, often require a mass of human labor and expertise knowledge, while reinforcement learning (RL) has a strong ability of sustainable learning. Recently, there has been a lot of work in this area. DARLING [8] proposes a method in which candidate solutions are generated by the planner and then merged and passed to the reinforcement learning module to learn the final approximate policy. However, the effect of this method has a lot to do with the design of the planner. Bougie [1] used additional knowledge as a supervision signal for network learning, and enhanced the information provided to the agent by introducing human expertise, but this supervision signal would interfere with the original training target of the network. Xie [20] proposed to train an additional network to make decisions from knowledge-based policy or the policy learned by the DDPG [9] algorithm. However, adding a network means increasing training difficulty and training time, which runs counter to the original intention of using human knowledge. In addition, these methods all need to provide a complete solution covering all states based on human knowledge, and do not consider about the situation when the number of agents is huge.

In this article, we define the knowledge that can only give actions in some states as *rules*, and the knowledge that has corresponding actions in all states as *plan*. Then we propose a new **K**nowledge-**G**uided **R**einforcement **L**earning (KG-RL) method for massive battle games, which includes Rule-Mix and Plan-Extend modules. The difference from the previous method is that we only need to design some effective rules instead of manually designing a complete solution. We designed a rule-mix module based on the hypernetwork structure in Qmix [15], which can learn more complex logical relationships than manually designed decision trees and reduce subjective bias of people. Then a plan-extend module is designed by extending the exploration policy of Actor-Critic (AC) [14] algorithm. It uses actor's policy and plan policies for joint exploration through a selector, which increases the exploration efficiency of reinforcement learning and accelerates the convergence speed of the algorithm. Through experiments in the Magent environment, we prove that the win rate of KG-RL is 22% higher than rule-based decision tree and 39% higher than the best-performing MFAC in pure reinforcement learning. The main contributions of this paper can be summarized as follows.

– We innovatively propose the rule-mix module, which uses a hypernetwork structure to learn more complex logical relationships between rules. It improves the win rate by 17% over the rule-based decision tree;

– We innovatively propose the plan-extend module. It combines rule-mix with actor-critic for joint exploration and enhances the exploration efficiency of the algorithm. The model with plan-extend improves the win rate by 22%;
– The KG-RL (rule-mix+plan-extend) method we proposed achieves the highest win rate in the Magent environment, which is 39% higher than the previous best method MFRL in this environment.

2 Related Work

An important cluster of related research is the research on the scalability of MARL. Based on graph convolutional neural network, [13] divided agents into different domains, and the impact of dimensional explosion is reduced by reducing the number of agents in the domain; [10] combine the attention mechanism with reinforcement learning, so that the agent can only consider about a part of the agent which is most relevant to itself, rather than all of them. However, these methods are all learned from scratch, and the number of agents that can support is limited [11], which cannot be applied to the environment where there are hundreds of agents.

Recent years, there has been a lot of research on combining human domain knowledge with RL, such as [2,19] combine knowledge graph with reinforcement learning in recommendation. PROLONETS [16] proposes a new network structure by embedding human knowledge, and realizes the dynamic change of network depth by adding random decision nodes. This method needs to design the structure of decision tree manually, which requires a high level of human technology. Based on QMix, RMLPE [5] expands the action space of RL with the selection of rules, leverages the Q-value in RL as a uniform criterion to judge the value of rules and original actions. The method is convenient to implement, but the expansion of the action space increases the difficulty of learning.

3 Method

In many tasks, the state returned by the environment contains a lot of redundant information, and the original action space also contains many invalid and illegal action options. Therefore, it is necessary to encode some rules from the original state space and action space through people's understanding of the task. By mixing these rules through the rule-mix module, we can quickly get some strategies based on human knowledge, called *plan* 1, *plan* 2.... In order to enhance their exploration capability, we combine them with the Actor-Critic [14] algorithm to reduce the invalid exploration, and accelerate the convergence speed of the algorithm. This method is collectively called KG-RL, and the overall structure is shown in Fig. 1.

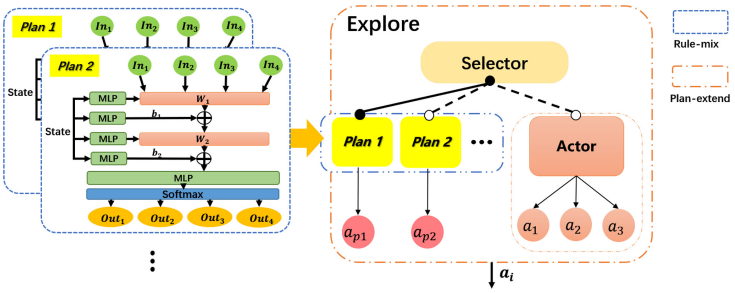

Fig. 1. The overall structure of KG-RL. Rules is combined into *plan* through rule-mix, and then plan-extend combine the *plan* with actor in RL for more efficient joint exploration.

3.1 Rule-Mix

The original input often contains a lot of information and is not easy to process. For example, for image input, we often only care about the content of the picture, and it is quite time-consuming to directly input the entire image into the network for end-to-end training. Besides we can use existing algorithms or common sense to get some information that is helpful for decision-making. We first form the method of extracting these more valuable information into decision modules, which is represented by the function $In_i(s)$, as shown in Table 1.

Similarly, the original action space directly uses the most direct actions like up, down, left, right and attack, which does not contain any knowledge. Therefore, in some cases, many invalid or illegal actions are often selected. Especially in some environments, the cost of illegal actions is high. However, these invalid and illegal actions are intuitive and easy to judge for humans in many problems, such as suicide operations in games. Thus, we can embed human knowledge into simple actions and form rules. We call these nodes which finally determine the output action as action modules. They are represented by the function $Out_i(s)$, as shown in Table 1.

Inspired by QMix, we also design a hypernetwork in rule-mix, which uses state s to generate weight W_i and bias B_i through MLP, as shown in the left part of Fig. 1. We combine W_i and B_i with the decision module, which can be calculated as Eq. (1). This hypernetwork structure correlates the state s with the decision module In_i through multiplication, by which an additional representation of the current state is integrated into the gradient of In_i, so that more information can be provided [23].

$$Hypernet\_out = Relu((W_i * in_i(s)) + B_i), \tag{1}$$

Then, the output of the hypernetwork is processed through a Softmax layer, and the probability of each action module is output of the latter. Finally, the AC algorithm is used to train the network, and the goal is to maximize the long-term return. The role of the hypernetwork is to generate a logical structure similar to

a decision tree. With the powerful representation ability of deep neural network, the hypernetwork designed by us can represent a more complex logical structure than decision tree. This is an important reason why rule-mix performs better than regular manual rules.

3.2 Plan-Extend

Actor-critic algorithm uses a stochastic policy for discrete actions, and actor output the probability of each action. It is explored by sampling distribution of the actor. In the early stage of training, the actor network contains less information. The exploration based on the actor is almost equivalent to random exploration, which is inefficient. Due to the low probability of success, rewards are often sparse. This seriously affects the learning speed of the network. Therefore, we can combine a actor policy with the rule-based policies for joint exploration. Furthermore, we design a *selector* to choose the exploration policy.

We call the rule-based strategies obtained by rule-mix as $Planj$. *Selector* decides to choose $Planj$ or the current actor's policy to interact with the environment. Specifically, the *selector* determines the final action interaction with the environment, as shown in Fig. 1. We can make a selection at each step or at the end of each episode. The goal of selection is to choose the current better policy, so that better samples can be generated and the convergence speed of the algorithm can be accelerated.

Our goal is to use $Planj$ to explore when the actor's policy is not as good as $Planj$. Thus *selector* can be a predefined function, or an adaptive variable. In this article, we define it as a variable, which is selected by evaluation after each round of training. When the win rate of the actor's policy is lower than that of the $Planj$ in the last 30 evaluations, the $Planj$ is selected for exploration. Otherwise, the current actor's policy is selected.

It is important to note that the *selector* and $Planj$ here are only used to interact with the environment to generate trajectories $T(s_t, a_t, r_t, s_{t+1}, done)$, which is not included in the final reinforcement learning training model. Secondly, the $Planj$ uses deterministic rules, which means $P(a_t^p|s_t) = 1.0, \forall s_t$. So that the importance sampling rate c can be represented as $c = \frac{\pi(a_t|s_t)}{P(a_t|s_t)} = \pi(a_t|s_t)$. Therefore, the trajectory got by $Planj$ can directly be used for Actor-Critic algorithm updates. We call this method plan-extend which is shown in the right part of Fig. 1.

4 Experiment Setup

4.1 Environment

We conduct experiments in the Magent [22] environment, which is a confrontation environment of large-scale agents. In the experiment, we use a 40×40 map. At the beginning, there are 64 agents on each side, and they will not be supplemented after death in battle. The termination condition is that one party is completely wiped out or the maximum step of episode is reached. When the terminal state is reached, the one with more surviving agents wins.

There are two parts in the agent's observation. One is a 7-channel 13×13 matrix representing obstacles, teammates, teammates' HP, own mini map, opponent location, opponent's HP, opponent's mini map, the other is a 34 dimensional feature vector including ID embedding, last action, last reward and position. The 21 actions of the agent include 13 movable positions and 8 attackable positions around the agent. The reward setting in learning is the same as in MFRL[1].

4.2 Human Knowledge Based Module Design

In the Magent environment, we use the rules of human experience in the battle problem: attack only when there is an enemy within the attack range; give priority to attacking the enemy with the least HP or the nearest distance; in order to strengthen cooperation, people will approach teammates; for better survive, humans will pay attention to their blood volume in time. Based on the above human knowledge, we abstract the reules into the decision module In_i and action module Out_i as shown in Table 1.

Table 1. The decision modules and action modules we used in the experiment.

i	$In_i(s)$	$Out_i(s)$
1	Are there any opponents in attack range	Attack the enemy with the least health
2	Are there any opponents in observation	Move to the nearest opponent
3	Are there any teammates in view	Move to the teammate with the least health
4	Is my current health more than half	Move to the opponent with the least health
5	Whether the number of our agents is greater than that of the opponent	Move to the opponent with the least health
6	Whether the last action is an attack action	Attack any one within range

4.3 Experiment Settings

In the experiment, agents in the same team share parameters. We use Adam optimizer with a learning rate of 1×10^{-4}. The discount factor γ is 0.95. For value-based methods (MFQ, DQN, RMLPE [5]), the batch size is set to 64, and the buffer contains the most recent 80000 transitions. All models are trained for 2000 rounds of self-plays, and then are used for battles.

Through the combination of human knowledge modules (rules), we can manually design a decision tree based on human knowledge as shown in Fig. 3 as **Baseline**. Experiments have proved that even this simple rule-base decision tree performs better than other algorithms which are trained from scratch. In addition, we also selected MFRL and RMLPE [5] for comparison. The MFRL is the current state-of-the-art method in this environment and the RMLPE is another practical method based on human domain knowledge.

[1] https://github.com/mlii/mfrl.

5 Experimental Results

Winning or losing is a good condition for judging in the battle environment, and it is also the most valuable indicator. Therefore, we first use the models trained by each algorithm to play against each other. Then, with the Baseline as the opponent, the model improvement speed of each model was compared in the training phase. In addition, we also compared the output policies of rule-mix and the Baseline to show the differences between them. Furthermore, we study the influence of different decisions and action modules. Finally, we analyzed the behavior of the model combined with the battle playback.

5.1 Battle Game

In this experiment, we directly use the models trained by each algorithm for comparison. In order to reduce random errors, we trained three models for each algorithm, named Algo 1, Algo 2, and Algo 3. In each round we randomly choose two from all the models to have a battle. At the end of the experimentation, we get a total of 200,000 duels. In addition to the number of wins for each model, we also record the number of kills and be killed by each model.

Table 2. Result of battle

	Elo score	Wins	Draws	Totalls	Win rate	Killed	Be_killed	Kill_ratio
KG-RL 1	**2521**	15104	79	16792	90.42%	**1039159**	634900	**1.64**
KG-RL 2	2498	15070	72	16715	90.59%	1034676	636241	1.63
KG-RL 3	2409	**15208**	66	16766	**91.10%**	1038573	637606	1.63
Rule_Mix 3	2351	14539	71	16807	86.93%	1031148	653521	1.58
Rule_Mix 2	2322	14348	85	16677	86.54%	1022675	650434	1.57
Rule_Mix 1	2242	14430	85	16732	86.75%	1025922	654432	1.57
Baseline 1	1848	11257	101	16496	68.85%	909402	620709	1.47
Baseline 3	1769	11454	106	16809	68.77%	926807	635147	1.46
Baseline 2	1749	11413	114	16558	69.62%	913718	621215	1.47
MFAC 3	1344	8407	131	16596	51.45%	869221	755577	1.15
MFAC 1	1299	8570	132	16638	52.30%	874899	748140	1.17
AC 3	1288	7770	159	16615	47.72%	835245	781361	1.07
MFAC 2	1268	8688	160	16857	52.49%	886280	760472	1.17
AC 1	1227	7866	161	16751	47.92%	843229	785245	1.07
AC 2	1169	7670	**170**	16554	47.36%	830664	778978	1.07
MFQ 3	842	3824	44	16414	23.57%	623389	931098	0.67
MFQ 1	804	3849	46	16470	23.65%	625485	934899	0.67
MFQ 2	802	4078	53	16528	24.99%	632251	931803	0.68
RMLPE 1	774	2910	6	**16942**	17.21%	380365	939619	0.40
RMLPE 2	655	2786	3	16715	16.69%	368833	929389	0.40
DQN 1	633	2269	96	16622	14.23%	480286	863041	0.56
DQN 3	617	2293	121	16715	14.44%	485276	867214	0.56
DQN 2	614	2266	111	16543	14.37%	481625	856605	0.56
RMLPE 3	552	2842	6	16688	17.07%	374305	925787	0.40

In addition, learning from [4,7], we also use ELO ratings to describe the performance of each agent, as commonly used in both traditional games like chess and in competitive video game ranking and matchmaking services. Assuming that the current grade scores of agent A and agent B are R_A and R_B respectively, then the expected win rate of agent A to B, according to Logistic distribution, should be:

$$E_A = \frac{1}{1 + 10^{(R_A - R_B)/400}}. \tag{2}$$

If agent's grade is adjusted accordingly, the specific mathematical formula is $R_A^{'} = R_A + K(S_A - E_A)$. At the masters level K is usually 16. In order to create a gap between agents, we set it to 32 here.

The result of the battle is shown in Table 2. It shows that the KG-RL (rule-mix+plan-extend) we proposed is better than other methods in terms of ELO scoring, win rate, or KD ratio. In particular, KG-RL and rule-mix are better than the Baseline. On the contrary, other methods starting from scratch is not as good as the Baseline. This illustrates the huge potential of embedding human knowledge into RL.

5.2 Comparison of Training Process

Because each algorithm is trained by self-play. The model itself is constantly changing when it is updated. In order to evaluate the convergence speed of each algorithm, we let the model play a round against the Baseline after each round of training. Then we calculate the win rate of the last 30 rounds. It can be seen from Fig. 2 that KG-RL converges the fastest, and stably reaching a win rate of 1.0 at 500 steps. In the end, only rule-mix and KG-RL have a winning percentage of 1.0. This shows the good performance of reinforcement learning based on human knowledge.

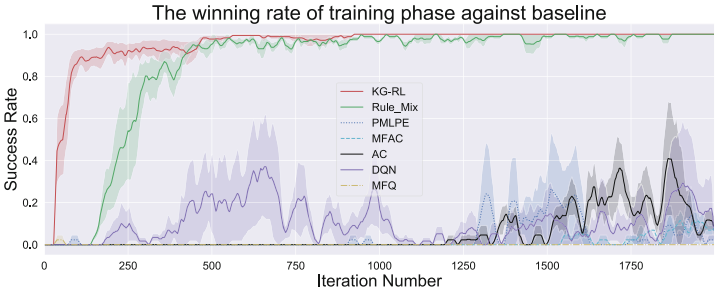

Fig. 2. The win rate of each algorithm when Baseline is used as opponent. We calculate the win rate of the last 30 rounds after each round of training.

5.3 Model Differences

We count the output frequency of rule-mix and the Baseline on different action modules. Through the different selection actions of each model in Table 3, we can

find that the network after learning is quite different from the decision tree we manually designed. This is because the neural network can learn more complex logical relationships, corresponding to better strategies.

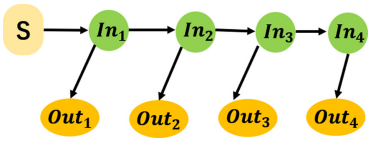

Fig. 3. Decision tree built from human knowledge, which we used as Baseline in experiment.

Table 3. Selection ratio of each action module

	Baseline		Rule_Mix	
	Times	Rate	Times	Rate
Out_1	13925	32.60%	13561	34.16%
Out_2	17726	41.50%	14316	36.06%
Out_3	840	1.96%	10659	26.85%
Out_4	10214	23.91%	1165	2.93%

5.4 The Influence of Different Decisions and Action Modules

The different choices of the decision module and the action module have different effects on the algorithm. We named the models that used the decision modules (In_1, In_2, In_3, In_4), (In_1, In_2) and the model without the decision module as $rule\_mix\_in4$, $rule\_mix\_in2$ and $rule\_mix\_in0$ respectively. Figure 4 shows their training curve. It can be seen that the more input modules used, the faster the convergence the algorithm will be. The added decision-making module reduces the redundant information in the original state, providing more concise and more valuable information for the network, which accelerates the learning speed of the network.

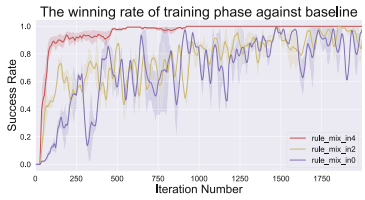

Fig. 4. Comparison of models using different decision modules.

Table 4. Comparison of models using different action modules.

Models	ELO score	Win rate
rule_mix_out 1	1586	82%
rule_mix_out 2	1268	21%
rule_mix_out 3	1369	43%
rule_mix_out 4	1479	51%

We named the models that used action modules $(Out_1, Out_2, Out_3, Out_4)$, $(Out_1, Out_2, Out_3, Out_5)$, $(Out_2, Out_3, Out_4, Out_5)$, $(Out_2, Out_3, Out_4, Out_6)$ as $rule\_mix\_out1$, $rule\_mix\_out2$, $rule\_mix\_out3$, $rule\_mix\_out4$, respectively. Table 4 shows a comparison of the win rate of the trained models. It can be seen that choosing different output modules will have different effects on the final models. In this article, $rule\_mix\_out1$ performs best, so its corresponding action modules are also used in other experiments.

5.5 Discussion

As shown in Fig. 5, these are screenshots of a battle between KG-RL and MFAC. It can be seen that KG-RL (in red) formed a semi-encircled state of the opponent from the beginning. This semi-encirclement is a more advantageous position for agents to concentrate their firepower and strengthen cooperation. Although each agent is trained separately, it shows the intelligence of group as a whole, which is meaningful.

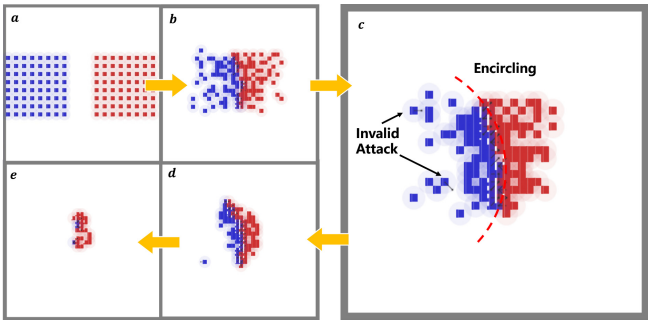

Fig. 5. These are screenshots of a battle between KG-RL (red) and MFAC (blue). At the beginning of battle, two group are initialized in a symmetrical position on the left and right. In order to show the details, *c*) enlarges one of the screenshots. (Color figure online)

Secondly, by observing the local actions of agents, we can see that MFAC (in blue) has more ineffective attacks (attacks no-one's areas) than KG-RL. Taking advantage of human rules, KG-RL directly shields those invalid and illegal actions through rule-mix, which reduces the exploration space of the algorithm.

6 Conclusion

In this article, we propose a knowledge-guided reinforcement learning method for massive agent battle games, named KG-RL, which can be divided into rule-mix and plan-extend. The hypernetwork structure in the rule-mix can obtain more complex logical relationships than manually designed decision trees. The plan-extend can combine the result of rule-mix with reinforcement learning to achieve more efficient joint exploration. In fact, the experimental results has proved it. In the Magent environment, it shows that the win rate of KG-RL is 22% higher than rule-based decision tree and 39% higher than the best-performing MFAC in pure reinforcement learning. In the future, it will be meaningful to study the design and evaluation of rule modules. In addition, we will continue to conduct research on the automation of rule module design.

Acknowledgment. This work was partly supported by both National Key Research and Development Program of China (Grant No.2019AAA0104800) and NSFC under grant No. 91648204.

References

1. 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)
2. Chen, X., Huang, C., Yao, L., Wang, X., Zhang, W., et al.: Knowledge-guided deep reinforcement learning for interactive recommendation. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
3. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: First return, then explore. Nature **590**(7847), 580–586 (2021)
4. Elo, A.E.: The rating of chessplayers, past and present. Arco Pub. (1978)
5. Han, X., Tang, H., Li, Y., Kou, G., Liu, L.: Improving multi-agent reinforcement learning with imperfect human knowledge. In: International Conference on Artificial Neural Networks, pp. 369–380. Springer (2020). https://doi.org/10.1007/978-3-030-61616-8_30
6. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning
7. Jaderberg, M., Czarnecki, W., Dunning, I., Marris, L., Lever, G., Castaneda, A., et al.: Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. arXiv:1807.01281 (2018)
8. Leonetti, M., Iocchi, L., Stone, P.: A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. Artif. Intell. **241**, 103–130 (2016)
9. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv:1509.02971 (2015)
10. Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., Gao, Y.: Multi-agent game abstraction via graph attention neural network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 7211–7218 (2020)
11. Long, Q., Zhou, Z., Gupta, A., Fang, F., Wu, Y., Wang, X.: Evolutionary population curriculum for scaling multi-agent reinforcement learning. arXiv:2003.10423 (2020)
12. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. NIPS (2017)
13. Mao, H., et al.: Neighborhood cognition consistent multi-agent reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 7219–7226 (2020)
14. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937. PMLR (2016)
15. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 4295–4304. PMLR (2018)
16. Silva, A., Gombolay, M.: Neural-encoding human experts' domain knowledge to warm start reinforcement learning. arXiv:1902.06007 (2019)

17. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning. pp. 330–337 (1993)
18. Tang, Z., et al.: Discovering diverse multi-agent strategic behavior via reward randomization. arXiv:2103.04564 (2021)
19. Wang, P., Fan, Y., Xia, L., Zhao, W.X., Niu, S., Huang, J.: Kerl: a knowledge-guided reinforcement learning model for sequential recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 209–218 (2020)
20. Xie, L., et al.: Learning with stochastic guidance for navigation. arXiv:1811.10756 (2018)
21. Yang, Y., Rui, L., Li, M., Ming, Z., Wang, J.: Mean field multi-agent reinforcement learning (2018)
22. Zheng, L., et al.: Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
23. Zhou, M., Liu, Z., Sui, P., Li, Y., Chung, Y.Y.: Learning implicit credit assignment for cooperative multi-agent reinforcement learning (2020)