Comparison of Multi-Agent Reinforcement Learning Algorithms in Cooperative Grid Environments

Abstract

In practical multi-agent reinforcement learning scenarios, it is challenging to sacrifice the immediate reward of a specific agent in order to achieve a higher global reward as a group in the long term. In this paper, we study whether representative multi-agent reinforcement learning algorithms can learn optimal policies in cooperative grid environments that require robust exploration and credit assignment to achieve the global optimum.

1. Introdution

Recent developments in reinforcement learning (RL) algorithms have enabled us to solve various decision-making problems. However, in real-world RL applications such as robot control, where high levels of cooperation among multiple agents are required, solving problems by learning independent policies is often challenging. Therefore, many researchers have studied multi-agent reinforcement learning (MARL), which involves multiple agents interacting with each other and the environment. However, as an agent's learning process can be affected by the actions of other agents and the environment, the MARL requires us to understand more complex learning dynamics and incorporate techniques like game theory and cooperative strategies. For instance, each agent may have its own objective, and there may be a shared objective among agents that requires cooperation between agents. There have been several interesting environments, such as SMAC [4] and GRF [2], employed as successful test beds for MARL algorithms.

There are two notable off-policy MARL algorithms based on *value decomposition* approach named the VDN [7] and QMIX [3]. Value decomposition assigns appropriate credits to each agent by factorizing a joint utility function into individual action-value functions. VDN proposed to factorize the joint action-value function into a sum of individual action-value functions. QMIX extends this additive value factorization to represent the joint action-value function as a monotonic nonlinear function. Off-policy methods such as VDN and QMIX have driven academic progress in the MARL field with superior performance.

Meanwhile, there is an interesting paper [11] that demonstrated the superior performance on MARL benchmarks of variants of the famous on-policy RL algorithm PPO [5] called IPPO and MAPPO. The paper claimed that IPPO and MAPPO could achieve competitive performance to existing off-policy MARL algorithms specifically tailored to address the unique challenge in MARL through sophisticated hyperparameter tuning. However, it remains to be studied whether on-policy algorithms can deal with their intrinsic limitations, such as the lack of exploration. It is well-studied that insufficient exploration may lead algorithms to converge to a local optimum instead of a global optimum.

In this paper, we compare the above MARL algorithms in highly cooperative grid environments such as Switch2 and Checkers[1] to check whether they can learn how to effectively cooperate to achieve the global optimum in Switch2 and Checkers environments.

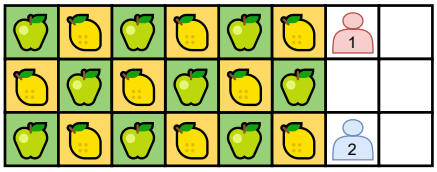
2. Related Work

A Taxonomy of MARL Algorithms MARL algorithms can be classified as on-policy or off-policy depending on how the data is used during training. If an algorithm can only learn using data generated from the current policy π, it is classified as an on-policy method. On-policy methods have the advantage of low bias, but they have low sample efficiency because they can only use data sampled from the current policy. On the other hand, algorithms that are not subject to the above constraints are classified as off-policy methods. They have high sample efficiency because all the collected data can be reused for training, but they have high bias because convergence is not guaranteed.

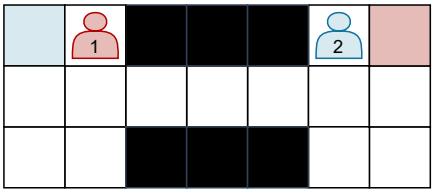
Off-Policy MARL Algorithms VDN and QMIX are powerful but suffer from structural constraints such as additivity and monotonicity. QTRAN [6] transforms the original joint action-value function into a new, easily factorizable function to eliminate the constraints. This allows QTRAN to outperform VDN and QMIX in non-monotonic environments.

Algorithms that simply use value decomposition may be less efficient for tasks that require multiple roles. ROMA [9] proposes two additional loss functions for agents to learn the concept of roles from trajectories. The greater the difference in trajectory, the more

1) Switch2 refers to the Switch2-v0 and Checker refers to the Checkers-v0 environments (note: https://github.com/koulanurag/ma-gym/wiki/Environments)



(a) Checker



(b) Switch2

Figure 1: Two simple cooperative grid environments

they will learn to fulfill different roles. RODE [8], on the other hand, does not utilize trajectory, but instead learns an action representation through a forward model to understand the kinds of roles an agent can take. The estimated roles determine the actions that can be performed. Actually, ROMA and RODE has strong performance compared to VDN and QMIX in SMAC environments.

On-Policy MARL Algorithms COMA [1] attempts to perform centralized training decentralized execution (CTDE) using single centralized critic and decentralized actors. Since it assumes a single reward function, it can perform counterfactual operations in a single forward pass, enabling robust credit assignment. COMA demonstrated superior performance compared to Independent Actor-Critic (IAC) in SMAC.

MAT [10] introduces a novel architecture that effectively transforms MARL into a sequence modeling problem. The algorithm offers only linear time complexity for multi-agent problems using an encoder-decoder architecture that leverages the multi-agent advantage decomposition theorem to transform the joint policy discovery problem into a sequential decision-making process. The authors claim that this allows MAT to achieve superior performance and sample efficiency compared to robust baselines such as MAPPO.

3. Experiment Setting

MARL Environments We consider the following two grid-based cooperative MARL environments.

- Checkers: There are apples and lemons in Figure 1a. The first agent is reward-sensitive and gives 10 points to the team for an apple and -10 points for a lemon. The second agent, which is insensitive to rewards, receives 1 point from the team for apples and -1 point for lemons. Apples and lemons disappear when collected and the environment is reset when all the apples are eaten. To maximize the team's reward (theoretically 86.18), the reward-insensitive agent should eat the lemons in the center aisle to clear the way for the reward-sensitive agent.

- Switch2: There are two agents and one narrow corridor in Figure 1b. Each agents trying to get to a location painted with the same color. They must cooperate to avoid blocking the other agent's path as they must travel through a corridor that can only be passed by one agent at a time. Agents can only observe their own location. The environment is initialized when both agents have moved to the specified location. The Switch2 has two optimal policies that achieve the maximum reward of 8.3. It is important that one of the two agents passes through the corridor first while not being interfered with by the other agent.

Note that all agents receive the same reward, the sum of individual rewards from all agents.

Table 1: Hyperparameter settings in Checkers and Switch2. The commonly available hyperparameters are unified and are in Checkers/Switch2 format when different parameters are used.

Hyperparameters	**VDN**	**QMIX**	**IPPO**	**MAPPO**
Use shared obs.	✗	✓	✗	✓
Data chunk length	10	10	10	10
PPO epoch	N/A	N/A	5 / 2	5 / 2
ϵ-annealing time (%)	80 / 50	80 / 50	N/A	N/A

Hyperparameter Settings We implement representative off-policy methods, VDN and QMIX, and on-policy methods, IPPO and MAPPO, in two grid environments where two agents can achieve optimal rewards through close cooperation. All algorithms use the shared policy and the agent ID by appending the one-hot encoding of the agent ID to each agent's observations. Through experiments, we analyze the effect of the interaction between the algorithm's characteristics and an environment on learning. The notable hyperparameter settings are listed in Table 1.

4. Experiments

Figure 2 demonstrates the comparison of on-policy and off-policy algorithms on Checkers and Switch2 environments. We analyze the experimental results as follows:

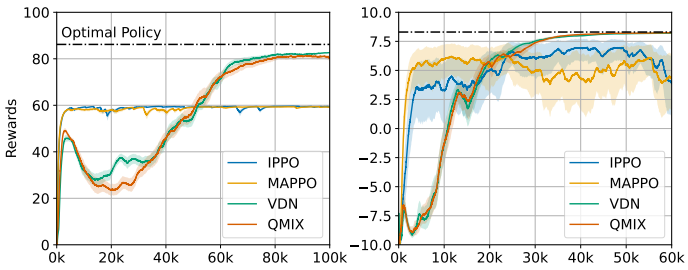


Figure 2: Comparison of on-policy (MAPPO and IPPO) and off-policy (QMIX and VDN) algorithms on Checkers (left) and Switch2 (right) environments with standard error.

Lazy Agent Problem in PPO While IPPO and MAPPO have better sample efficiency over traditional on-policy methods, exploration is still limited as data sampling still relies on the behavioral network of the previous phase. It causes the model to overestimate the samples obtained early in training. The results in Figure 2 show that IPPO and MAPPO only exploit the reward-sensitive agents in Checker by moving them to earn rewards and rarely exploit the reward-insensitive agents. This is because the algorithms converge rapidly without performing enough exploration and do not find a way to exploit the reward-insensitive agent. On the other hand, VDN and QMIX find ways to exploit reward-insensitive agents through sufficient exploration and attain the optimal policy relatively more quickly. The experimental results in the Checker indicate that IPPO and MAPPO face the lazy agent problem due to insufficient exploration in an environment where agents receive unequal rewards, which prevents them from performing high-level cooperation.

Policy Oscillation in Switch2 Environment In the Switch2 environment, there are two optimal policies depending on which agent enters the corridor first. Since each agent utilizes only its location information, it can only estimate the location of other agents if they meet. It makes learning unstable for IPPO and MAPPO, which choose actions stochastically, as the optimal policy they learn constantly becomes different. The results in Figure 2 show that MAPPO and IPPO do not converge stably and have a consistently high variance. However, PPO is sensitive to hyperparameters; IPPO and MAPPO can achieve maximum reward in Switch2 when additional learning heuristics such as linear LR decay are applied. However, the same settings do not reach the maximum reward for Switch4, a more complex environment than Switch2. On the other hand, VDN and QMIX learn deterministic policies as the ϵ decreases, so they learn a single optimal policy stably in Switch2, including Switch4.

5. Conclusion

Although IPPO and MAPPO overcome the limitations of on-policy methods by increasing the sample efficiency compared to traditional on-policy methods, they fail to learn high-level cooperative policies due to their weak exploration capabilities. In addition, the stochastic behavior selection of policy-based methods makes it difficult to achieve stable convergence in situations where the optimal policy changes frequently. These results highlight that IPPO and MAPPO still have limitations in exploration or convergence compared to representative off-policy methods.

References

[1] J. N. Foerster et al. “Counterfactual Multi-Agent Policy Gradients”. In: *AAAI 2018*. 2018.

[2] K. Kurach et al. “Google Research Football: A Novel Reinforcement Learning Environment”. In: *AAAI 2020*. 2020.

[3] T. Rashid et al. “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning”. In: *ICML 2018*. 2018.

[4] M. Samvelyan et al. “The StarCraft Multi-Agent Challenge”. In: *AAMAS 2019*. 2019.

[5] J. Schulman et al. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017).

[6] K. Son et al. “QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning”. In: *ICML 2019*. 2019.

[7] P. Sunehag et al. “Value-Decomposition Networks For Cooperative Multi-Agent Learning”. In: *CoRR* abs/1706.05296 (2017).

[8] T. Wang et al. “RODE: Learning Roles to Decompose Multi-Agent Tasks”. In: *CoRR* abs/2010.01523 (2020).

[9] T. Wang et al. “ROMA: Multi-Agent Reinforcement Learning with Emergent Roles”. In: *ICML 2020*. 2020.

[10] M. Wen et al. “Multi-Agent Reinforcement Learning is a Sequence Modeling Problem”. In: *NeurIPS 2022*. 2022.

[11] C. Yu et al. “The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games”. In: *NeurIPS 2022*. 2022.