MARL Experiment in HOK Environment

Jingfan Qing [*1] **Luyao Wang** [*1] **Bo Zhang** [*1]

Abstract

With the development of reinforcement learning, an increasing number of learning algorithms have emerged. In addition to traditional single-agent reinforcement learning, many approaches based on multi-agent cooperation have also been proposed. One of the key factors affecting overall performance is how to effectively decompose the multi-agent value function. In this report, we present an innovative exploration of two different value function decomposition methods. Both of them are based on VDN algorithm, and one use mlp and another use attention to estimate the weights of q values. We applied both methods in the Honor of Kings environment, and one of them achieved performance exceeding the VDN and QMIX baseline. Code available at: https://github.com/yoimika/MARL-HOK.

1. Introduction

In the rapidly evolving landscape of artificial intelligence, multi-agent algorithms have emerged as a cornerstone of research, playing a pivotal role in diverse applications such as autonomous vehicle coordination, smart grid management, and complex game scenarios. These algorithms empower multiple agents to interact, collaborate, and make decisions collectively, mimicking human teamwork and problem-solving strategies. Choosing the right reinforcement learning environment for different experiments is equally critical, as it provides the virtual "arena" where agents learn and adapt through trial and error. A well-designed environment can accurately simulate real-world complexities, ensuring that the algorithms developed are not only theoretically sound but also practically applicable.

In this context, we introduce the Honor of Kings (HOK) environment (Liu et al., 2024), a rich and dynamic platform that closely mirrors the intricacies of real-time strategy games. HOK is a lightweight multi-agent adversarial environment open sourced by Tencent, specifically designed for multi-agent reinforcement learning research. The environment abstracts and simulates the core mechanics of real MOBA games, such as hero movement, attack, defense and skill release, but greatly simplifies the map size and rules, and reduces the complexity of training and debugging. HOK supports multi-agent cooperative or adversarial scenarios and is suitable for validating classical and cutting-edge multi-agent algorithms. Its complexity lies in the need for agents to coordinate their actions, adapt to opponents' strategies, and optimize their decision-making in real-time, making it an ideal testbed for exploring advanced multi-agent reinforcement learning algorithms.

In this experiment, our goal is to control multiple hero agents to take different actions through a multi-agent reinforcement learning algorithm to reduce the monster's hp as much as possible within a specified time.

This report delves into an in-depth exploration of two novel improvements based on the Value-Decomposition Networks (VDN) algorithm (Sunehag et al., 2017).VDN, a foundational approach in multi-agent reinforcement learning, simplifies the learning process by decomposing the joint value function into individual agent value functions. Our first improvement leverages a Multi-Layer Perceptron (MLP) to estimate the weights of Q-values, enhancing the algorithm's ability to capture complex interactions between agents. The second enhancement incorporates an attention mechanism, enabling agents to focus on relevant information and prioritize actions based on the current state of the environment.

Through extensive experimentation and rigorous analysis in the HOK environment, we evaluate the performance of these improved algorithms against traditional baselines, including VDN(Sunehag et al., 2017) and QMIX (Rashid et al., 2018). Our findings not only shed light on the effectiveness of the proposed methods but also offer valuable insights into the design of more efficient multi-agent reinforcement learning algorithms. The remainder of this report is structured as follows: Section 2 provides a comprehensive review of related work; Section 4 details the proposed algorithms and their implementation; Section 5 presents the experimental

[1]Department of Computer Science, University of Peking, Haidian, Beijing. Correspondence to: Jingfan Qing <2100012656@stu.pku.edu>, Luyao Wang <2200017784@stu.pku.edu>, Bo Zhang <2200013074@stu.pku.edu>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

setup and results; and Section 6 concludes with a summary of our experiment.

2. Related Works

In the realm of multi-agent reinforcement learning (MARL), the development of effective value function decomposition methods used to be a central focus, aiming to address the challenges posed by the exponential growth of the joint action space. Among these approaches, Value-Decomposition Networks (VDN)(Sunehag et al., 2017) marked a significant milestone. VDN operates under the assumption of a linear relationship between individual agent value functions and the joint value function, enabling the decomposition of the global Q-value into a sum of local Q-values for each agent. By doing so, it simplifies the learning process, reducing the complexity of optimizing the joint action-value function. This approach has been particularly effective in cooperative multi-agent scenarios where agents share common goals, providing a straightforward and computationally efficient solution for coordinating actions. However, its reliance on a linear combination limits its expressiveness, making it less suitable for environments with complex, non-linear interactions between agents.

To overcome the limitations of VDN, the QMIX algorithm (Rashid et al., 2018) was proposed, introducing a more sophisticated approach to value function factorization. QMIX introduces a monotonic mixing network that allows for non-linear combinations of individual agent Q-values, enabling it to capture more complex relationships within the joint action space. The key innovation of QMIX lies in its use of a hypernetwork to generate the mixing weights, which are conditioned on the global state. This mechanism ensures that the overall value function remains monotonic with respect to the individual Q-values, maintaining the stability of the learning process while enhancing the algorithm's representational power. As a result, QMIX has demonstrated superior performance in a variety of challenging multi-agent tasks, outperforming VDN in scenarios with intricate coordination requirements. Nevertheless, QMIX still struggles in highly dynamic and complex environments where the relationships between agents and the environment are constantly changing.

Building on the advancements of VDN and QMIX, QATTEN (Yang et al., 2020) presents a general framework for cooperative multi-agent reinforcement learning that incorporates an attention mechanism. QATTEN leverages attention to dynamically assign weights to different agents' Q-values based on the current state, allowing agents to focus on the most relevant information and prioritize their actions accordingly. This approach enables QATTEN to adapt more effectively to varying environmental conditions and agent interactions, providing a more flexible and powerful solution for multi-agent learning. By integrating attention, QATTEN can handle complex scenarios with diverse agent roles and changing task requirements, offering a significant improvement over previous methods in terms of both performance and adaptability. However, we have to notice that though the idea to use attention is the same, ATTENVDN and QATTEN have something different. First, ATTENVDN do not use multi-head mechanism to estimate high order scale $\lambda_{i,h}$, but instead use attention block to directly predict the scale for each Q_i. Second, ATTENVDN utilize action value to calculate the scale while qatten in sample code do not.

3. Preliminaries

3.1. Q-learning

Q-learning (Watkins & Dayan, 1992) is a kind of reinforcement learning. It use TD-method to estimate the Q-values with given transitions.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a))$$

Owning to the maximum operation which ensure the sampled action a' has nothing to do with sampling distribution, the Q-learning is an off-policy method that is easy to implement. Based on that, the DQN (Mnih et al., 2013) utilizes the approximation ability of deep neural network to estimate the Q-value with two important modifications, replay buffer and target network.

3.2. Individual-Global-Max (IGM)

The overall idea behind value decomposition method is separating the global Q-value into multi local Q-values.

$$Q(s,a) = f(Q_1(s_1,a_1),\dots,Q_n(s_n,a_n),i)$$

where s and a refer to the joint state and joint action, and s_i, a_i is the state and action of the ith agent. f refers to a certain function that perfectly matches the global Q-value of the ground truth value with the local Q-values and some global information i given.

Individual Global Max (IGM), which asserts that $\exists Q_i$, such that,

$$\arg\max_a Q(s,a) = (\arg\max_{a_1} Q_i(s_i,a_i),\dots,\arg\max_{a_n} Q_n(s_n,a_n))$$

This is the main assumption used by multi-agent value decomposition methods such as VDN and QMIX. IGM make sure that if we update the global Q-value to optimal point, we can gain optimal return by following each agent action generated by each local Q-value.

4. Implementation

According to QATTEN Theorem1 (Yang et al., 2020), the global Q-value can be expanded such that,

$$Q(s,a) \approx c(s) + \sum_{i,h} \lambda_{i,h} Q_i(s,a_i)$$
$$\approx c(s) + \sum_{i} \sigma_i^{\theta} Q_i(o_i,a_i)$$

where o_i is the observation of ith agent.We utilize two different networks to approximate σ_i^{θ}. Because the weights mentioned below is based on VDN, so the sum of each agents weight is 5.

4.1. Adaptive VDN

By contrast, VDN naively sums over all Q-values which may loss the different contribution from different agents. So we use a simple MLP to estimate the weights according to global states.

$$Q(s,a) \approx \sum_{i} MLP(s)_i Q_i(o_i,a_i)$$

4.2. Attention VDN

Intuitively, the weights should be related to the current global state and the actions taken currently. Moreover, in order to obtain the global Q-value, it's natural to consider the relationship between agents.

Based on the consideration mentioned above, we consider using attention (Vaswani et al., 2023) to estimate weights. The detail of ATTEN block will be mentioned in section 5.2.

$$Q(s,a) \approx MLP'(s) + \sum_{i} ATTEN(s,a) Q_i(o_i,a_i)$$

5. Experiment

The results of adaptive and attention vdn compared with baseline can be seen in Figure 1.

The hyper parameters used in each networks will be mentioned in Appendix A.

5.1. Adaptive VDN Results

We use the following MLP structure. We use softmax to ensure weights is in a valid range, multiply 5 to satisfy the detail mentioned in Section4.

$$MLP(s) = 5 * Softmax(Linear(ReLU(Linear(s))))$$

The experiment results can be seen from Figure 1 which seems performs worse even than VDN algorithm. We assume that the reason for this poor performance is some

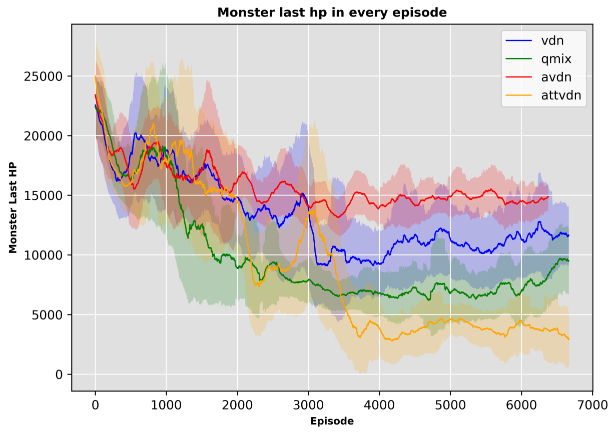

Figure 1. Experiment results for different multi agent methods. The solid line shows the average monster last hp across nearest 200 episodes, and shaded area reflect the std error of the monster last hp.

agents' weights is too low, which makes the update on the corresponding Q-function slow. As show in the following equation, with the low value of w_i, the change $\partial L/\partial \theta_i$ become low.

$$\frac{\partial L}{\partial \theta_i} = w_i \frac{\partial Q_i}{\partial \theta_i}$$

We also sampled the weight change for each agent at training time in Figure 2. We set 0.25 as the minimum tolerant weight value. It's obvious that the weight of agent ZhuanZhou is always below the tolerant value in training time. Notice that the shaded area is 0.3std error, meaning that lots of agent's weight reach the tolerant value which will greatly slow down the training process.

5.2. ATTEN Details

We conclude two reasons for the very unsatisfactory weights estimation in Section 5.1. One is the MLP structure is poor at extracting the mutual information between each agent which is crucial in getting a proper weights. Another is only state do not provide enough information to find weights. If agents at the same state but do different action, the weights should be different. In order to solve the problem we mentioned above, we use ATTEN block.

In ATTEN block, we use global state and action as input, the details for each part are as follows.

State in HOK environment is a concatenate for each agent state. So, we first reshape them into (N, ds), the N is the number of agents, ds is the state dimension for each agent.As mentioned in HOK doc, the state is not easy to learn directly, so we use a mlp to map it into (N, dq). To

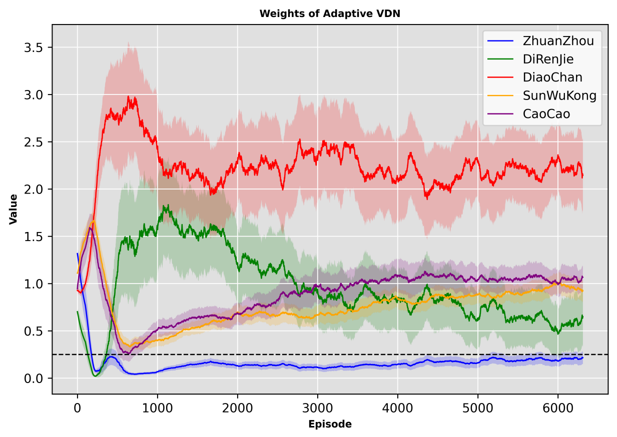

Figure 2. Weights of adaptive vdn sampled in every episodes. The solid line shows the average weights across nearest 200 episodes, and shaded area is 0.3std error. The dash line is a minimum tolerant value (0.25).

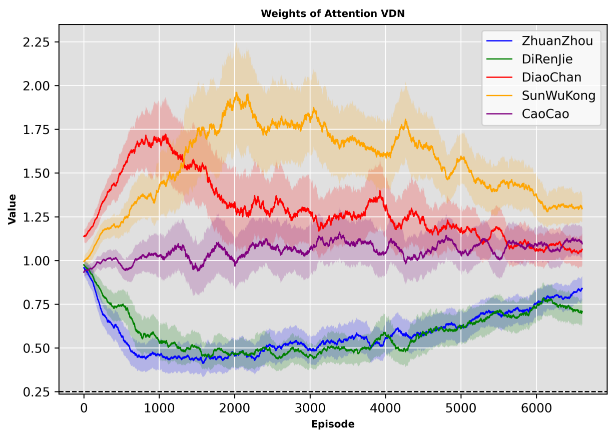

Figure 3. Weights of attention vdn sampled in every episodes. The solid line shows the average weights across nearest 200 episodes, and shaded area is 0.3std error. The dash line is a minimum tolerant value (0.25).

distinguish different agent, we add sin/cos position encoding (Vaswani et al., 2023).

There are 13 type of **Action** in HOK environment represented in integer. The first four and the last four is move action which contains the similar information, while the action represented in $[4, 8]$ is skills and attack action which have much more information. We first map all the move action to 0, and $[4, 8]$ to $[1, 5]$. Then, because the number is integer which is not easy to train, so we use embedding to embed them into shape (N, dq).

Finally, we use the transformed action and state to gain a attention matrix with shape (N, N).

$$M = A \times S.T$$

Each column of M represent different the state. Because the Q_i is about agent state not about action. So we should use softmax along each row.

$$tmp_q = Softmax(M) \cdot Q$$

tmp_q has shape of $(N, 1)$, we then directly sum them and add $MLP(s)$ to estimate the final global Q-values.

$$Q(s, a) = sum(tmp_q) + MLP(s)$$

Beside ATTEN block, in order to avoid slow training, we clamp all weights which lower than 0.25 to 0.25.

5.3. Attention VDN Results

The result in Figure 1 is much better than QMIX, showing the power of attention. We also sample the weights in training times (Figure 3). It shows the importance of clamp operation.

5.4. Interesting Findings

- **Monster HP** will somehow recover at training time. We assume is the monster out of battle range which trigger the game mechanism of auto recovery.

- **ATTVDN Training** result within $[2000, 3000]$ range occurs a rapid bump. The direct reason is the auto recovery mechanism, because in training period monster hp will first down to 1k around at 100 steps, but after that the monster hp will randomly top up. So actually in episode 2000 model already learn how to beat the monster, but do not learn how to prevent trigger the auto recovery mechanism.

- **Beat down monster**. In ATTVDN training period, it's interesting that there're 71 times that the agent beat down the monster hp to 0!!

- **Weights** of each agents is very intuitive. ZhuanZhou is a support which do little damage so usually the weights is low, and SunWoKong and DiaoChan can do much damage so the weights is high.

6. Conclusion

In this report, we make MARL experiment in HOK environment with AVDN and ATTVDN methods, and compare them with VDN and QMIX. The result shows AVDN is not very good, but ATTVDN performs much better than VDN and QMIX in this environment.

References

Liu, L., Zhao, J., Hu, C., Cao, Z., Zhao, Y., Ye, Z., Meng, M., Wang, W., He, Z., Li, H., Lin, X., and Huang, L. Mini honor of kings: A lightweight environment for multi-agent reinforcement learning, 2024. URL https://arxiv.org/abs/2406.03978.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, 2018. URL https://arxiv.org/abs/1803.11485.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning, 2017. URL https://arxiv.org/abs/1706.05296.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Yang, Y., Hao, J., Liao, B., Shao, K., Chen, G., Liu, W., and Tang, H. Qatten: A general framework for cooperative multiagent reinforcement learning, 2020. URL https://arxiv.org/abs/2002.03939.

A. Training Parameters

Hyper parameters	VDN	Adaptive VDN	Attention VDN
buffer size	5000	5000	5000
batch size	64	64	64
optimizer	Adam	AdamW	AdamW
max t	1e+6	1e+6	1e+6
target update interval	200	200	200
mixer	vdn	avdn	attvdn
mixing embed dim	32	32	32
hypernet embed	64	64	64
lr	1e-3	1e-3	1e-3
td lambda	0.6	0.6	0.6