Off-Beat Multi-Agent Reinforcement Learning

Extended Abstract

Wei Qiu  
Nanyang Technological University  
Singapore  
qiuw0008@e.ntu.edu.sg

Weixun Wang  
Tianjin University  
Tianjin, China  
wxxwang@tju.edu.cn

Rundong Wang  
Nanyang Technological University  
Singapore  
rundong001@e.ntu.edu.sg

Bo An  
Nanyang Technological University  
Singapore  
boan@ntu.edu.sg

Yujing Hu  
NetEase Fuxi AI Lab  
Hangzhou, China  
huyujing@corp.netease.com

Svetlana Obraztsova  
Nanyang Technological University  
Singapore  
lana@ntu.edu.sg

Zinovi Rabinovich  
Nanyang Technological University  
Singapore  
zinovi@ntu.edu.sg

Jianye Hao  
Tianjin University  
Tianjin, China  
jianye.hao@tju.edu.cn

Yingfeng Chen  
NetEase Fuxi AI Lab  
Hangzhou, China  
chenyingfeng1@corp.netease.com

Changjie Fan  
NetEase Fuxi AI Lab  
Hangzhou, China  
fanchangjie@corp.netease.com

ABSTRACT

We investigate cooperative multi-agent reinforcement learning in environments with off-beat actions, i.e., all actions have execution durations. During execution durations, the environmental changes are not synchronized with action executions. To learn efficient multi-agent coordination in environments with off-beat actions, we propose a novel reward redistribution method built on our novel graph-based episodic memory. We name our solution method as LeGEM. Empirical results on stag-hunter game show that it significantly boosts multi-agent coordination in environments with off-beat actions and achieves leading performance.

KEYWORDS

multi-agent coordination; multi-agent reinforcement learning

ACM Reference Format:

Wei Qiu, Weixun Wang, Rundong Wang, Bo An, Yujing Hu, Svetlana Obraztsova, Zinovi Rabinovich, Jianye Hao, Yingfeng Chen, and Changjie Fan. 2023. Off-Beat Multi-Agent Reinforcement Learning: Extended Abstract. In Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023, IFAAMAS, 3 pages.

1 INTRODUCTION

Despite the recent successes of multi-agent reinforcement learning (MARL) in autonomous systems [1, 12] and real-time strategy (RTS) video games [10], learning effective multi-agent coordination in environments with off-beat actions remains challenging for MARL. Many cooperative MARL methods [3, 4, 6, 7] fail to learn efficient multi-agent coordination in environments where action durations are caused by off-beat actions. The main reason is TD-learning [8] fails when displaced rewards caused by action durations are used in training. To this end, we propose a novel reward redistribution method built on our novel graph-based episodic memory. We name our method as LeGEM. Empirical results on stag-hunter game show that it significantly boosts multi-agent coordination in environments with off-beat actions and achieves leading performance.

2 PRELIMINARIES

MARL aims to learn optimal policies for all the agents in the team. With TD-learning and a global Q value proxy \(Q_{\text{tot}}\) for the optimal \(Q^*\), \(\{Q_i\}_{i=1}^N\) are optimized via minimizing the loss [2, 11]:

\[\hat{\theta} = \arg \min_{\theta} \mathbb{E}_{D'}[\sum_{t=1}^T ((Q_{\text{tot}}(s_t, a_t)) - r_t + \gamma \max_{a' \in A_s} Q^*_t(s_{t+1}, a'))^2]\]

where \(r_t\) is the parameter of the target \(Q_{\text{tot}}\) and is periodically copied from \(\hat{\theta}\). \(D'\) is a sample from the replay buffer \(D\).

3 METHODOLOGY

3.1 Temporal Recency Reward Redistribution

To learning efficient multi-agent coordination in environment with off-beat actions for MARL methods. We redistribute rewards to agents’ pivot timesteps (we will introduce the method for searching agent’s pivot timesteps in the following subsection). The pivot timestep of each agent is the timestep when the off-beat action was executed and later triggered the reward.

The timestep to which the reward should be distributed is called the final pivot timestep. We denote the final pivot timestep at timestep \(t\) as \(e_t\). For a shared reward at timestep \(t\), each agent’s pivot timestep
Therefore, we can utilize updated transitions with newly replaced $T$ between agents. The length of the trajectory and the triplet $c$ can be different. We can get $e_t$ via proximity:

$$e_t = \arg\min_{t \in [1, N]} \{ t - e_0 \}$$  

(1)

In the example of Fig. 1, the final pivot timestep $e_0$ for timestep 9 is $e_0 = 5$. Then, we can utilize it to update the reward in each transition $(s^\tau, \tilde{u}^\tau, r^\tau, s^{\tau+1})$:

$$\tilde{r}^\tau = \mathbb{I}(e_t \geq t) \cdot r^\tau + \mathbb{I}(e_t < t) \cdot r'$$  

(2)

where $\mathbb{I}(-)$ is the indicator function. $r^\tau$ is replaced with $\tilde{r}^\tau$. This update rule is conducted iteratively from $t = 0$ to $t = T - 1$. To stabilize learning and circumvent the overestimation of the TD target, $r'$ is also updated after Eqn. 2 via $(1 - \gamma (e_t < t) \cdot (1 - \beta)) \cdot r'$. Therefore, we can utilize updated transitions with newly replaced rewards for MARL training. We also present an example in Fig. 2 to illustrate the workflow of our reward redistribution method.

### 3.2 Episodic Memory and Searching Method

The reward redistribution method introduced in Sec. 3.1 relies on certain structures to searching the pivot timestep $e_t$ for agent $i$. We propose our novel episodic memory (EM). During training, each agent $i$ collects its own individual trajectories $T_i$. We then define $T_i$ of agent $i$ as $T_i = \{(s_0^i, \tilde{u}_0^i, r_0^i), \ldots, (s_{T-1}^i, \tilde{u}_{T-1}^i, r_{T-1}^i)\}$, where $T$ is the length of the trajectory and the triplet $(s_0^i, \tilde{u}_0^i, r_0^i)$ represents the observation, action and reward of timestep $t$. $r^\tau$ is globally shared between agents.

**Graphs and Sub-Graph.** Each agent’s episodic memory (EM) has $T$ graphs categorized by the length of the episode. Each graph consists of many sub-graphs that are categorized by the episode return. We define the graph of agent $i$’s EM as a directed graph $\phi_i^k \in \Phi_i$ where $\Phi_i$ is the set of graphs of agent $i$ and $\phi_i^k$ is the $t$-th graph of $\Phi_i$, $t \in \{0, \ldots, T - 1\}$. To model an agent’s behaviour explicitly and make the trajectories easy to represent, we create $T$ graphs for each agent and let $\Phi_i = \{\phi_i^0, \ldots, \phi_i^{T-1}\}$ where $T$ is the maximum depth of all graphs and the maximum length of the episode as well. The maximum level of $\phi_i^k$ is $t + 1$. The graph consists of nodes that are connected by edges. Each node contains visit count and pointers connecting the precursors (nodes at the previous level) and the successors (nodes at the next level). Besides the $\phi_i^t$, we define the sub-graph set of $\phi_i^k$ as $\Psi_i^{\Omega} = \{\phi_i^{t, \omega}\}_{\omega \in \Omega}$ by using the discretized episode return and there are $O$ sub-graphs. $\phi_i^{t, \omega}$ is the $\omega$-th sub-graph whose episode return is the $\omega$-th item in $[0, \ldots, r^T]$ where $r^T$ is the maximum discretized episode return of $\phi_i^k$. We define the key $(\phi_i^t, \tilde{u}_i^k)$ using agent $i$’s $\phi_i^k$ and $\tilde{u}_i^k$ at timestep $t$. The visit count of the node indicates the total number of visits made by agent $i$ to the node. The initial value of the visit count is 1.

**Updating Graph.** Given $\tau_i$ of length $T_i$, if the node is already in the graph at length $t$, we then increase the visit count by 1. Otherwise, we create a new node for level $t$ of the graph and update its pointers. Meanwhile, sub-graphs will be also created and updated.

**Searching method.** We propose our search schemes for our reward redistribution method. For agent $i$, given $\tau_i$, the corresponding graph is $\phi_i^l = \Phi_i[\ell]$ ($\ell = \text{length}(\tau_i) - 1$) and $\phi_i^{(l, \omega)} = \Phi_i^{(l, \omega)}$, and episode return is $r^{l, \omega}$. Agent $i$ searches from the node (the key is $(\phi_i^l, \tilde{u}_i^k)$) and $(\phi_i^l, \tilde{u}_i^k) \in \tau_i$, $(\phi_i^l, \tilde{u}_i^k) \in \tau_i$ at level $t$ in sub-graph $\phi_i^{(l, \omega)}$ to find the pivot timestep $e_t$ for $r^T$. Searching ends when the pattern of decreasing or increasing visit count ends and the corresponding level is the candidate pivot timestep.

### 4 EXPERIMENTS AND CONCLUSION

We conduct experiments on stag-hunt game as shown in Fig. 1. We select QMIX [4], VDN [7], IQL [9] and Qtran [6] as baselines. We implement our method on PyMAML [5] and use 10 random seeds to train each method on the testbed. Results show that with our method LeGEM, MARL methods gain enhanced performance.
REFERENCES

[1] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2012. An overview of recent progress in the study of distributed multi-agent coordination. IEEE Transactions on Industrial Informatics 9, 1 (2012), 427–438.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. Nature 518, 7540 (2015), 529–533.

[3] Frans A Oliehoek, Christopher Amato, et al. 2016. A Concise Introduction to Decentralized POMDPs. Vol. 1. Springer.

[4] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In International Conference on Machine Learning. 4295–4304.

[5] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Min Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. CoRR abs/1902.04843 (2019).

[6] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In International Conference on Machine Learning. 5887–5896.

[7] Peter Sunehag, Guy Lever, Andruzas Grudys, Wojciech Marian Czarnecki, Vincius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296 (2017).

[8] Richard Stuart Sutton. 1984. Temporal credit assignment in reinforcement learning. Ph.D. Dissertation. University of Massachusetts Amherst.

[9] Ardi Tampuu, Tambet Mattiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. PLoS ONE 12, 4 (2017).

[10] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michael Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575, 7782 (2019), 350–354.

[11] Christopher JCH Watkins and Peter Dayan. 1992. Q-Learning. Machine Learning 8, 3-4 (1992), 279–292.

[12] Ming Zhou, Jun Luo, Julian Villalba, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Albin, Iman Fadakar, Zheng Chen, et al. 2020. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. arXiv preprint arXiv:2010.09776 (2020).