ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-202210.00199

Multi-Agent Conflict Resolution Method Based on DDQN: Postprint

Authors: Zhang Yi, Zhao Lingzhong, ZHAI Zhongyi, Zhao Lingzhong

Date: 2022-10-26T00:00:00+00:00

Abstract

To address the problem of agents' ineffective decision-making under partial observability, this paper proposes a conflict resolution method that integrates deep reinforcement learning. Based on the DDQN algorithm, the method leverages the characteristics of the reinforcement learning paradigm to compute agents' cumulative returns, and determines agent priorities according to the magnitude of these return values, thereby achieving conflict resolution. The method is evaluated through simulations of real-world traffic congestion scenarios, and experimental results demonstrate that it can effectively resolve conflicts among agents.

Full Text

Abstract

With the continuous development of artificial intelligence, which is profoundly transforming people' s lifestyles and work patterns, single-agent systems have become inadequate for addressing increasingly complex and dynamic real-world environments. Most challenging problems require extensive multi-agent collaboration. However, during cooperative interactions, conflicts inevitably arise among agents due to resource constraints and other factors. Consider a traffic vehicle conflict scenario where multiple autonomous vehicles must pass through the same intersection simultaneously—if one vehicle is an ambulance, the conflict resolution becomes particularly urgent, as the ambulance must be given priority.

Reinforcement learning technology has advanced rapidly, with various methods being applied to multi-agent domains [1-3]. Since reinforcement learning does not require explicit environment modeling, agents can engage in autonomous interactive learning with their environment, significantly improving computational efficiency. The Google team successfully integrated reinforcement learn-

ing with deep learning technologies, using deep neural networks (DNN) to approximate state-value functions and addressing the dimensionality explosion problem [4]. In single-agent environments, the environment is influenced only by that agent, making the agent' s local observation equivalent to a global observation. However, in multi-agent environments, the environment is affected by multiple agents, and each agent has only partial observability of the entire system.

To address the challenge of effective decision-making under partial observability, this paper proposes a conflict resolution method that combines deep reinforcement learning. Leveraging the characteristics of reinforcement learning paradigms, agent priorities are determined through cumulative return values. The method is evaluated by simulating real-world traffic congestion scenarios, demonstrating its effectiveness in resolving agent conflicts and computing cumulative returns.

1. Introduction

Existing approaches that establish direct communication among agents [15-20] have achieved some success, but in certain environments, direct communication cannot be established or incurs excessive overhead. To overcome the overestimation of real-world decisions and address communication limitations, we propose a multi-agent conflict resolution method based on Deep Double Q-Networks (DDQN). This approach establishes indirect communication among agents through a shared flag mechanism.

The proposed model utilizes flag bits to mark critical information and states. The neural network takes an agent's observation as input and outputs the action value for the next time step. DDQN decouples action selection from action value evaluation using two separate networks. Agents select actions with maximum value with a certain probability. Rather than directly using the target network to obtain Q-values, DDQN first identifies the action with the maximum estimated Q-value in the current network, then obtains the Q-value for that action from the target network.

The mathematical formulation is given by:

$$y_j = R_j + \gamma Q'$$

where s'_i represents the next state of agent i, and the target Q-value calculation follows:

$$y_j = R_j + \gamma Q'(s'_j, \arg\max_a Q(s'_j, a; \theta); \theta')$$

2. Methodology

2.1 Model Architecture

The proposed conflict resolution model computes agent priorities using state and action information over cumulative time periods, then modifies flag bits accordingly. All agents make decisions based on these flag bits and their respective local observations.

Each agent consists of two modules: (1) a priority calculation module that computes cumulative returns, and (2) an action selection module that evaluates all possible actions and selects the one with maximum reward. The internal agent structure processes local observations, priority information, and flag bits to produce optimal actions.

2.2 Decentralized POMDP Formulation

In multi-agent systems, agents often cannot observe the global environment due to various constraints. The entire reinforcement learning process can be modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), defined as a tuple $(N, S, A, P, R, O, \gamma)$:

- N: Number of agents
- S: Global state information for all agents and environment
- $A = \{A_1 \times A_2 \times \cdots \times A_N\}$: Joint action space, where A_i is the set of local actions for agent i
- $P(s'|s, a)$: State transition function mapping to $[0, 1]$
- $R(s, a)$: Shared reward function for all agents
- $O = \{O_1 \times O_2 \times \cdots \times O_N\}$: Joint observation space, where O_i is the observation for agent i
- $\gamma \in [0, 1]$: Discount factor preventing infinite reward accumulation

Each agent makes decisions based solely on its local observation O_i, and the environment transitions to the next state s' according to P, providing reward r. Each agent' s objective is to maximize cumulative return.

2.3 Priority Calculation Module

The priority algorithm computes each agent's priority through cumulative return values. A public information storage flag D serves as a shared memory accessible to all agents for calculation and storage. This flag stores historical experience information from all agents, enabling indirect communication without direct links [15-20].

The priority calculation process: 1. Initialize flag bits 2. Agents compute priorities based on local observations 3. Modify flag information 4. Propagate flag information to other agents 5. Other agents compute new priorities based on received flags and local observations

The priority is calculated as:

$$\theta_e = E[R_t|s_t = s, \theta_p]$$

where θ_p represents the priority network parameters.

2.4 Action Selection Module

The action selection module determines the next action based on computed priorities. Agents typically select actions with maximum return values. The process involves:

1. Calculate cumulative return at time t: $y_t = \max_a Q'(s_t, a)$
2. Compute individual agent priority from cumulative returns
3. Input priority and local observation into the action selection network
4. Select action with maximum Q-value: $a_t = \arg\max_a Q(s_t, a; \theta)$

The model parameters are updated periodically to optimize performance.

3. Experiments and Analysis

3.1 Experimental Setup

All experiments were conducted on a computer with an Intel E5-1630 CPU@3.10 GHz and 8GB RAM, using PyCharm as the development environment with PyTorch. The traffic conflict scenario was constructed as a simulation environment where different colored squares represent target locations that agents must reach to complete tasks.

Environment Rules: - Different colored circles represent different agent types - Agents must reach corresponding colored target positions - Four actions available per time step: up, down, left, right - Black regions are prohibited areas - One action per time step

Parameters: Learning rate $\alpha = 0.005$, discount factor $\gamma = 0.99$, with the discount coefficient gradually decreasing during training. The neural network updates every 100 steps.

3.2 Results and Analysis

The simulation conducted 10,000 training episodes. The evaluation metric was the average joint reward across agents. Results demonstrate rapid growth until convergence, with the proposed method achieving higher return levels compared to traditional approaches.

Key Findings: 1. **Performance Advantage**: The method enables agents to obtain greater returns within the same time frame, making better decisions through learned priorities. 2. **Computational Efficiency**: Unlike traditional methods requiring complete environment modeling and state value storage, this

approach allows agents to learn through autonomous environmental interaction using only flag bits and local observations. 3. **Conflict Resolution**: The model effectively resolves inter-agent conflicts by calculating priorities from cumulative returns, providing a novel solution to the partial observability problem.

The experimental results validate that the proposed DDQN-based conflict resolution method can effectively address multi-agent conflicts in partially observable environments.

References

[1] TESAURO G. Temporal difference learning and TD-Gammon[J]. Communications of the ACM, 1995, 38(3): 58-68.

[2] KOHL N, STONE P. Policy gradient reinforcement learning for fast quadrupedal locomotion[C]//IEEE International Conference on Robotics and Automation. Piscataway, NJ: IEEE Press, 2004, 3: 2619-2624.

[3] ARULKUMARAN K, DEISENROTH M P, BRUNDAGE M, et al. Deep reinforcement learning: a brief survey[J]. IEEE Signal Processing Magazine, 2017, 34(6): 26-38.

[4] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[5] ZHENG L, YANG J, CAI H, et al. MAgent: a many-agent reinforcement learning platform for artificial intelligence[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press, 2018: 2-7.

[6] SCHWARTZ H. Multi-agent machine learning: a reinforcement approach[M]. New York: John Wiley & Sons, 2014: 978-1002.

[7] LOWE R, WU Y, TAMAR A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[J]. Proceedings of the International Conference on Neural Information Processing Systems. New York, NY: ACM, 2017: 6379-6390.

[8] KAI Z, ZHENG H, ZHONG H, et al. Conflict resolution in multi-agent systems[J]. Neural Information Processing, 2016.

[9] GOLPAYEGANI F, DUSPARIC I, TAYLOR A. Multi-agent collaboration and conflict management in residential demand response[J]. Computer Communications, 2016, 96: 63-72.

[10] XIANG L, TAO Y. Learning cooperation in multi-agent systems[J]. Neural Information Processing, 2015.

[11] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[EB/OL]. (2013-12-19)[2021-08-20]. https://doi.org/10.4853/arxiv.13125602.

[12] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press, 2016: 12-19.

[13] RICHARD S, DAVID M, AJIT S. Policy gradient methods for reinforcement learning with function approximation[C]//Proceedings of the International Conference on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2000: 1057-1063.

[14] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous methods for deep reinforcement learning[C]//International Conference on Machine Learning. New York, NY: ACM, 2016: 1928-1937.

[15] HAARNOJA T, ZHOU A, ABBEEL P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with stochastic actor[C]//International Conference on Machine Learning. New York, NY: ACM, 2018: 1861-1870.

[16] TAMPUU A, MATIISEN T, KODELJA D, et al. Multiagent cooperation and competition with deep reinforcement learning[J]. Plos One, 2017, 12(4): 17-23.

[17] FOERSTER J, ASSAEL I A, DE FREITAS N, et al. Learning to communicate with deep multi-agent reinforcement learning[C]//Advances in Neural Information Processing Systems. New York, NY: ACM, 2016.

[18] PENG P, WEN Y, YANG Y, et al. Multi-agent bidirectionally-coordinated nets for learning to play StarCraft combat games[J]. Arxiv Preprint, 2017: 1703-10069.

[19] SAINBAYAR S, ARTHUR S, FERGUS R. Learning multiagent communication with backpropagation[C]//International Conference on Neural Information Processing Systems, 2016: 5-10.

[20] JIANG H, ZHAI Y. Dynamic coalition formation based on multi-sided negotiation in multi-agent systems[J]. International Journal of Database Theory and Application, 2015, 8(1): 7265-7275.

Editor: ZHANG Suobin

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv —Machine translation. Verify with original.