Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/99/20251759

Multi-Agent System Anti-Interference Algorithm Based on Deep Reinforcement Learning

Fangbo Zhang[1,a,*,†], Yongkang Zhang[2,b,†], Yongqing Zhao[1,c], Jiahui Hou[1,d]

[1]Taizhou College, Nanjing Normal University, Nanjing, Jiangsu, 225300, China
[2]Pingdingshan University, Pingdingshan, Henan, 467000, China
a. 2052836606@qq.com, b. 2357383824@qq.com, c. 19515922173@139.com, d. 2286715378@qq.com
**corresponding author*
[†]These authors contributed equally to this work

Abstract: As multi-agent systems are increasingly applied in various environments, achieving stable control under communication interference has become a key challenge. This paper proposes a stability control algorithm based on a Markov model and reinforcement learning, which enables the multi-agent system to achieve its goals through a reward and punishment mechanism under interference. The results demonstrate that the algorithm effectively handles communication interference, converges well, and ultimately completes control tasks, showing promising application prospects for the proposed solution.

Keywords: Reinforcement Learning, Multi-Agent System, Stable control, Anti-Interference, Communication.

1. Introduction

With the development of Multi-Agent Systems (MAS), improving their robustness in complex environments has become a research focus[1]. Traditional methods enhance system stability and anti-interference capabilities but rely on precise mathematical models and high-quality parameters. However, these methods often face limitations when dealing with uncertainties or dynamic environments. Recently, Deep Reinforcement Learning (DRL)[2], based on the Markov Decision Process (MDP), has shown great potential in addressing these challenges. Unlike traditional approaches, DRL does not require precise mathematical models. Instead, it uses a reward-punishment mechanism, allowing agents to learn and adapt through trial and error to achieve control objectives, even in the presence of interference[3,4]. This makes DRL especially suitable for handling complex disturbances and uncertainties, thereby enhancing system robustness and stability.

In MAS, agents not only interact with the environment but also collaborate or compete with each other. The Multi-Agent Soft Actor-Critic (SAC) algorithm is noted for its ability to handle continuous action spaces and multi-agent collaboration[5]. By introducing an entropy regularization term, SAC balances exploration and exploitation, improving system robustness and avoiding local optima. Given the complexity and uncertainties present in MAS environments, this study selects reinforcement learning as a solution to stabilize the multi-agent model under interference. MAS has broad applications in areas such as autonomous driving, drone swarming, intelligent manufacturing, and robot control. This study provides a new approach to MAS stability and demonstrates the adaptability

DOI: 10.54254/2755-2721/99/20251759

of DRL in achieving control objectives without relying on precise models and high-quality parameters but only based on MDPs, even in the face of interference.

2. Methodology

This section mainly explains the model establishment of the multi-agent system with communication in the absence of interference (2.1), the control law used to achieve the goals of the multi-agent system (2.2), and the model establishment under the condition of communication interference. Finally, it elaborates on the Markov process of the anti-interference multi-agent system (2.3) and the principles of the multi-agent SAC algorithm (2.4).

2.1. Establishment of Multi-Agent System with communication

The multi-agent system considered in this paper consists of n agents, where there is a designated leader agent. The leader agent can communicate information to other agents. In the designed multi-agent system, the leader agent communicates with the other agents, providing them with relevant information. The follower agents use the information from the leader to control their movements and reach the target position.

The communication information provided by the leader agent includes the position of the leader relative to the target. The other agents determine their own positions relative to the target by measuring their relative position to the leader agent and using the communication information provided by the leader. As a result, the follower agents are able to move towards and enter the target region. First, there are some variables defined: n represent the total number of agents in the system; $p_L(t)$ be the position of the leader agent at time; p_T be the target position.

Base these variables, there is the mathematical formulation of the Multi-Agent System with communication:

Leader Agent's Information

The leader agent communicates its position relative to the target as follows:

$$\Delta p_L(t) = p_L(t) - p_T \tag{1}$$

(1) Follower Agents' Control

Each follower agent measures its relative position to the leader as:

$$\Delta p_i(t) = p_i(t) - p_T \tag{2}$$

Using the leader's communication, each follower calculates its position relative to the target:

$$p_i(t) - p_T = \Delta p_i(t) + \Delta p_L(t) \tag{3}$$

Thus, each follower agent knows its position relative to the target and can design a control strategy to minimize the distance to the target, expressed as:

$$u_i(t) = -k(p_i(t) - p_T) \tag{4}$$

where k is a positive constant dictating the rate of convergence towards the target.

(2) Objective:

The goal of the multi-agent system is for all agents to reach the target region, mathematically described as:

$$\lim_{t\to\infty} \| p_i(t) - p_T \| = 0, \forall i = 1,2,\dots,n \tag{5}$$

This formulation ensures that all agents eventually reach the target, using information provided by the leader and their relative positions to the leader

2.2. Proportional Navigation Guidance (PNG) Control Strategy

To ensure the followers reach the target area, we use Proportional Navigation Guidance (PNG) as the control strategy. PNG is commonly used in control systems, especially for guiding objects like

DOI: 10.54254/2755-2721/99/20251759

missiles, where control action is proportional to the rate of change of the Line-of-Sight (LOS) to the target. Here, PNG helps followers converge to the target based on their relative position to the leader and the target.

Mathematical Formulation of the Proportional Navigation Guidance (PNG) Control Strategy:

(1) Relative Velocity and LOS Rate

We define the $v_i(t)$ be the velocity of the i-th follower agent, $\lambda_i(t)$ represent the Line-of-Sight (LOS) angle between the i-th follower agent and the target. In proportional navigation, the control input is proportional to the rate of change of the LOS angle $\lambda_i(t)$, which is influenced by the follower agent's relative position to the leader and the target.

(2) Control Law

The control input for each follower agent is defined based on the PNG law, which is typically proportional to the closing velocity $v_{\text{closing}}(t)$ (the relative speed between the agent and the target) and the LOS rate $\dot{\lambda}_i(t)$.

The control law for the can be expressed as:

$$u_i(t) = N \cdot v_{\text{closing}}(t) \cdot \dot{\lambda}_i(t) \tag{6}$$

where N is The Proportional Navigation Guidance (PNG) coefficient for the i-th agent.

(3) Determining the LOS Rate

The LOS Angle between the i-th follower agent and the target is calculated as:

$$\lambda_i(t) = \arctan\left(\frac{y_i(t) - y_{\text{T}}}{x_i(t) - x_{\text{T}}}\right) \tag{7}$$

Where $(x_i(t), y_i(t))$ is the position of the i-th follower agent, and $(x_{\text{T}}, y_{\text{T}})$ is the position of the target. The LOS rate $\dot{\lambda}_i(t)$ is the time derivate of the LOS angle:

$$\dot{\lambda_t}(t) = \frac{\frac{d}{dt}(y_i(t)-y_{\text{T}})(x_i(t)-x_{\text{T}}) - \frac{d}{dt}(x_i(t)-x_{\text{T}})(y_i(t)-y_{\text{T}})}{(x_i(t)-x_{\text{T}})^2+(y_i(t)-y_{\text{T}})} \tag{8}$$

(4) Convergence to the Target Region

The control law ensures that the closing velocity decreases as the agents approach the target, while proportional navigation adjusts the agents' trajectories based on changes in the LOS rate. This allows the follower agents to steer towards the target while minimizing deviations from the desired path. By continuously adjusting their velocities using PNG control, the follower agents can converge to the target region. The system's ultimate objective is to satisfy equation (5).

2.3. Establishment of a Multi-Agent Model with communication interference

If there is communication interference between the leader agent and the follower agents, then the final tracking information of the follower agents to the target region will also have an error, as given by the following formula:

$$\Delta p_i(t) = (p_i(t) - p_T) + \varepsilon(t) \tag{9}$$

where: $\Delta p_i(t)$ represents the tracking error of the i-th follower agent at time t, $p_i(t)$ is the position of the i-th follower agent at time t. $\varepsilon(t)$ is the communication error between the leader agent and the follower agents at time *t*.

2.4. Markov Model of Anti-Interference Multi-Agent System

In order to train the anti-Interference multi-agent system using reinforcement learning algorithms, a Markov model needs to be established. The Markov model includes elements such as states, actions, and reward functions:

State: The positions of all agents at time t, denoted as:

$$s_t = [p_1(t), p_2(t), \dots, p_n(t)] \tag{10}$$

DOI: 10.54254/2755-2721/99/20251759

Actions: The Proportional Navigation Guidance (PNG) coefficient for the i-th agent at time t, denoted as $a_i(t)$ in the equation (6), it can be summarized as:

$$u_i(t) = a_r(t) \cdot v_{\text{closing}}(t) \cdot \dot{\lambda}_i(t) \tag{11}$$

The reward function: The reward function is defined based on the change in distance of all agents from the target region between time t and time t−1. The formula for the reward at time t is:

$$r(t) = \begin{cases} -1, \text{if} \|p_i(t) - p_T\| > \|p_i(t-1) - p_T\| \\ +1, \text{if} \|p_i(t) - p_T\| \leq \|p_i(t-1) - p_T\| \end{cases} \tag{12}$$

In summary, by constructing the Markov model of the above anti-interference multi-agent system, the multi-agent system can still achieve its goals through adaptive decision-making, even in the case of communication interference.

2.5. Multi-Agent Soft Actor-Critic (SAC) algorithm

Through sections 2.1 to 2.4, the multi-agent system with communication interference and the corresponding Markov process are established. This section introduces the main principles of the MASAC algorithm used in this paper.

The Multi-Agent Soft Actor-Critic (SAC) algorithm is an extension of the Soft Actor-Critic algorithm designed for multi-agent systems. SAC is a model-free, off-policy algorithm based on the maximum entropy framework, which encourages exploration by maximizing the expected return while also maximizing entropy. This results in more stable and efficient learning, making it suitable for complex multi-agent environments.

(1) Entropy Regularization

The SAC algorithm uses an entropy term to encourage exploration. The objective is to maximize both the expected cumulative reward and the entropy of the policy. The entropy term is denoted as $H(\pi(\cdot\,|s))$.

(2) Loss Function for Policy (Actor)

The policy update rule for the actor maximizes the expected Q-value and the entropy term, given by the following loss function:

$$J_x(\theta) = E_{s_t \sim D, a_t \sim \pi_\theta}\left[\alpha \log\left(\pi_\theta(\alpha_t \mid s_t)\right) - Q_\phi(s_t, a_t)\right] \tag{13}$$

Where D is the replay buffer, α is the temperature parameter controlling the trade-off between reward maximization and entropy maximization, $Q_\phi(s_t, a_t)$ is the estimated Q-value of taking action a_t in state s_t.

(3) Loss Function for Critic (Q-function)

The critic is updated by minimizing the temporal difference (TD) error, defined as:

$$J_Q(\phi) = E_{(s_t, a_t, s_{t+1}, a_{t+1}) \sim D}\left[\left(Q_\phi(s_t, a_t) - \left(r_t + \gamma E_{a_{t+1} \sim \pi_\theta}\left[Q_{\phi'}(s_{t+1}, a_{t+1}) - \alpha \log\left(\pi_\theta(a_{t+1}|s_{t+1})\right)\right]\right)\right)\right] \tag{14}$$

(4) Temperature Parameter Update

The temperature parameter, which controls the balance between exploration and exploitation, is also updated to ensure that the policy maintains a desired level of entropy. The loss function for updating

α is given by:

$$J(\alpha) = E_{a_t \sim \pi_\theta}\left[-\alpha \log\left(\pi_\theta(a_r|s_t)\right) - \alpha \widehat{H}\right] \tag{15}$$

Where $\widehat{H}$ is the target entropy.

3. Experiment

This section mainly conducts three experiments: the collaborative control simulation of the multi-agent system without interference, the collaborative control simulation of the multi-agent system under interference, and finally, the collaborative control training of the multi-agent system based on

DOI: 10.54254/2755-2721/99/20251759

the deep reinforcement learning algorithm. The anti-interference capabilities of the multi-agent system in these three environments are then compared. The experiment is simulated on the device of the i5 with 12cores, the deep learning environment is Pytorch.

3.1. Experimental parameter settings

For the experiment, this section provides the physical parameters of the multi-agent system, as well as the initial states and other parameters corresponding to the interference model, as shown in Table 1. Finally, the network and training parameters of the multi-agent reinforcement learning algorithm are presented, as shown in Table 2.

Table 1: The Parameter for the physical parameters of the multi-agent system

Parameter names	Value
Number of the agents	5
Action space	[2,8]
Interference amplitude	10

Table 2: Network and training parameters of the multi-agent reinforcement learning

Parameter names	Value
Learning Rate for network (actor/critic)	0.0001/0.0001
Number of Steps per Episode	100
Policy Update Frequency	Every 100 episodes
Num of all Episodes	1500

3.2. The result of the experiment

Based on the environmental parameters shown in Table 1, we conducted simulations for the multi-agent system in two environments: one without communication interference and one with communication interference, following the principles outlined in Chapter 2. The simulation results of the system are shown in Figures 1 and 2.

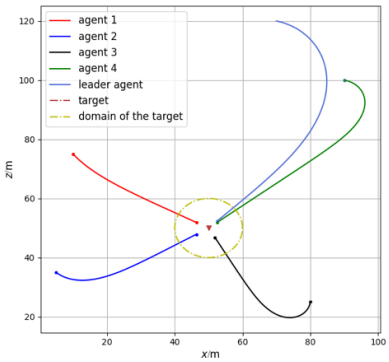

Figure 1: Multi-agent system without communication interference

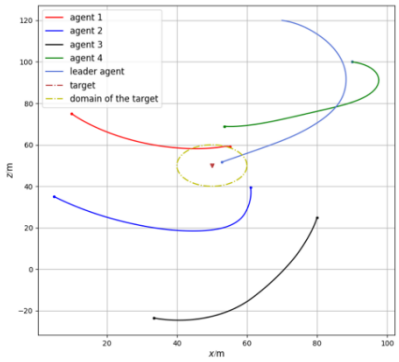

Figure 2: Multi-agent system with communication interference

Using the reinforcement learning parameters provided in Table 2, we trained the multi-agent system with communication error interference. The reward curve of the reinforcement learning algorithm is shown in Figure 3. Applying the trained network to the environment with communication

DOI: 10.54254/2755-2721/99/20251759

interference produced the simulation results shown in Figure 4. Additionally, the curves of the actions of each agent changing with the steps are shown in Figure 5.

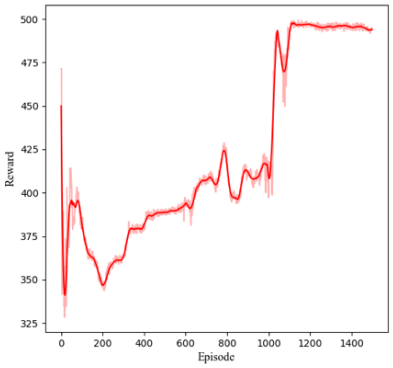

Figure 3: The MASAC reward with the episode

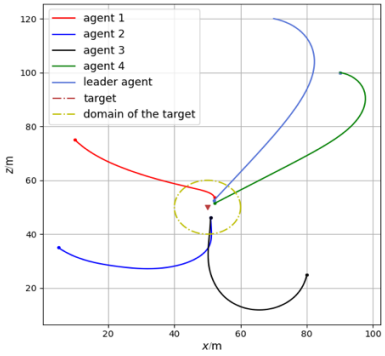

Figure 4: The Multi-agent system without communication interference controlled by MASAC

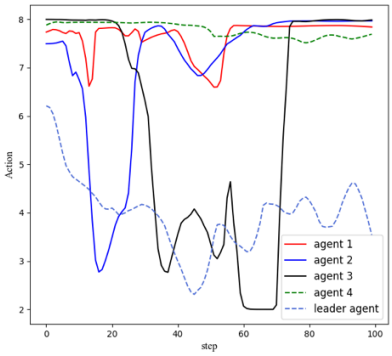

Figure 5: The actions for the 5agents controlled by the MASAC

3.3. Results Analysist

From Figures 1 and 2, it can be seen that the multi-agent system without interference can achieve its goal, while the system with interference cannot, indicating that a multi-agent system without anti-interference algorithms cannot succeed under interference. From Figures 3 and 4, it can be observed that the reinforcement learning algorithm converges after 1500 training rounds, and the trained network effectively enables the system to achieve its goal even with interference. Combining Figure 5 with Figures 3 and 4, it can be concluded that the stable control algorithm based on reinforcement learning proposed in this paper can adaptively adjust control parameters, allowing the multi-agent system to handle communication errors and complete the task effectively.

4. Conclusion

To address the issue where a multi-agent system fails to complete control tasks under communication interference, a Markov model of the multi-agent system was established, and reinforcement learning was applied to train the model. The results show that reinforcement learning, through a reward and punishment mechanism, enables the interference-affected multi-agent system to achieve its goals. The algorithm can converge, and the trained network can effectively handle communication interference, ultimately completing the control tasks.

References

[1] *Gao, Y., Chen, S., & Lu, X. (2004). Research on reinforcement learning technology: a review. ACTA AUTOMATICA SINICA, 30(1), 86-100.*

[2] *LandersMatthew, & DoryabAfsaneh. (2023). Deep reinforcement learning verification: a survey. ACM Computing Surveys.*

[3] *Yang, Z. , Merrick, K. , Abbass, H. , & Jin, L. . (2017). Multi-Task Deep Reinforcement Learning for Continuous Action Control. Twenty-Sixth International Joint Conference on Artificial Intelligence.*

[4] *Buoniu, L. , Bruin, T. D. , Toli, D. , Kober, J. , & Palunko, I. . (2018). Reinforcement learning for control: performance, stability, and deep approximators. Annual review in control.*

[5] *Wu, L. , Wu, Y. , & Tian, Q. Y. (2023). Multiagent soft actor-critic for traffic light timing. Journal of Transportation Engineering, Part A. Systems, 149(2), 4022133.1-4022133.11.*