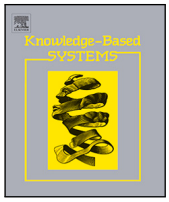
Knowledge-Based Systems 287 (2024) 111430

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



A novel data-driven model-free synchronization protocol for discrete-time multi-agent systems via TD3 based algorithm

Zhongxin Liu [a,b,*], Ye Li [a,b], Ge Lan [c], Zengqiang Chen [a,b]

[a] College of Artificial Intelligence, Nankai University, Tianjin 300350, China
[b] Tianjin Key Laboratory of Interventional Brain-Computer Interface and Intelligent Rehabilitation , Nankai University, Tianjin 300350, China
[c] College of Software, Nankai University, Tianjin 300350, China

ARTICLE INFO

Keywords:
Multi-agent consensus
Optimal control
Reinforcement learning
Twin Delayed Deep Deterministic Policy Gradient Algorithm (TD3)

ABSTRACT

System modeling is a complex and time-consuming task in engineering, and traditional model-based or observer-based methods are unable to deal situations when the system models are unknown or affected by elements such as temperature. As a result, it is critical to build model-free methods capable of iterative learning and real-time updating. In this paper, a data-driven, model-free TD3-based algorithm is proposed. The neural network is used to address the issue of dimensional explosion in the state and action space. Besides, each agent can set a variable number of neighbors via virtual neighbor technology, giving the connection topology more flexibility. What is more, the proposed controller is only constructed based on the consensus error, which can quickly realize synchronization while consuming the least amount of energy. Finally, proofs and some simulation examples are given to verify the efficiency of our algorithm.

1. Introduction

A multi-agent system (MAS) is a computing system composed of multiple agents interacting in the environment. Since many complex tasks, which are difficult for a single agent, can be completed by MASs, the related technologies have received extensive attention in recent years. As an important branch of artificial intelligence, swarm intelligence technologies are being applied to many fields such as unmanned aerial vehicle systems [1,2], smart grids [3,4], transportation systems [5] and robots [6].

The consensus problem is one of the basic problems for various MAS problems, such as formation control [7] and collaborative control [8]. Generally speaking, it can be divided into 2 parts: leaderless MAS problem and leader-following MAS problem. The purpose of the leaderless MAS is to synchronize the agents in a random state through designed control policies [9–12]. In the case of the leader-following MAS, the states of following agents will eventually synchronize with that of the leader [13–16]. In other circumstances, however, simply achieving synchronization is insufficient, specified performance indicators, such as energy consumption, synchronization time and accuracy need to be satisfied. And this kind of problem is called optimal synchronization problem. So far, a series of studies in different aspects like continuous-time MAS [17], discrete-time MAS [18], homogeneous MAS, heterogeneous MAS [19] and formation control have been carried out. Furthermore, optimal synchronization problems in many contexts are also explored, including finite time [20], fixed time [21], time delay [22], fault-tolerant [23–25] and so on. However, the methods listed above are all model-based algorithms, which cannot be employed if the model is unknown. Besides, these issues are complex and time-consuming to solve through system identification or observer design. Furthermore, the model can be affected by factors such as temperature, magnetic field, pressure and component aging, which makes it difficult to solve using model-based methods. Therefore, it is necessary to develop data-driven, model-free algorithms.

Reinforcement Learning (RL) is a subfield of machine learning in which optimal policies are found through interacting with the environment and gaining experience via trial and error [26]. As a result, RL methods produce innovative synchronization solutions. In 1991, WerBos P J [27] proposed an adaptive dynamic programming (ADP) method based on RL, policy iteration (PI) was firstly used to generate the optimal policies. In 2014, integral reinforcement learning (IRL) was constructed by Lewis [28,29] to solve the tracking problem for partially unknown models. Besides, the RL methods were also used to the $H\infty$ problems [30]. So far, the RL methods has been widely applied in a variety of fields, including formation control, zero-sum games, time-delay system and so on. Similarly, the RL methods also have a wide range of applications in the field of MAS. By iteratively fitting the controller parameters, the output-feedback ADP was introduced

* Corresponding author at: College of Artificial Intelligence, Nankai University, Tianjin 300350, China.
E-mail address: lzhx@nankai.edu.cn (Z. Liu).

https://doi.org/10.1016/j.knosys.2024.111430
Received 5 January 2022; Received in revised form 15 March 2023; Accepted 19 January 2024
Available online 23 January 2024

in 2010 to solve the consensus problem for discrete-time linear time-invariant (LTI) MASs [31]. In 2014, a kind of RL-based controller was constructed by Lewis [32] to provide solutions to the graphical games for the discrete-time MASs. Mu [33] recently proposed a Q-learning-based way to achieve consensus for discrete-time MASs, in which the optimal controller is entirely built from the output data. It is worth noting that all the solutions listed above require initial admissible control policies, which still need part of the system's information. So they are difficult to implement in model-free situations. Furthermore, as the state and action spaces expand, optimal policies will become increasingly difficult to get through training, and this kind of problem is known as "dimension explosion".

In 2018, a model-free data-driven RL algorithm called the Twin Delayed Deep Deterministic Policy Gradient Algorithm (TD3) [34] was proposed. Using 2 types of 6 neural networks, this algorithm predicts the value of the state and generates the optimal policies, resulting in fast and excellent control performance. In this paper, the TD3 algorithm is extended to MASs, and a kind of data-driven real-time optimal synchronization method for model-free discrete-time leader-following MASs is proposed. The following are the major contributions:

(1) The TD3 algorithm is extended to MASs. Each agent is equipped with a homomorphic controller, which is constructed through centralized training and decentralized execution. This approach allows for parameter sharing among agents during training, which can accelerate convergence speed. After training, each agent can construct optimal policies based solely on its neighbors' information, enabling distributed control.
(2) The problem of different numbers of neighbors in MASs is solved by virtual neighbor technology. Since the input dimension of a fixed neural network structure is constant, single-agent RL algorithm cannot be directly applied to MASs. In this paper, virtual neighbors are added to agents with fewer neighbors to ensure that each agent has the same input dimension.
(3) The proposed controller is highly flexible and robust due to its numerous adjustable parameters. The proposed controller not only addresses the issue of unstable control caused by system parameter changes due to environmental factors such as temperature and pressure, but also allows for performance adjustments, such as learning speed and robustness, based on specific requirements.

The structure of this paper is as follows: The background knowledge of discrete-time MASs is introduced in Section 2, then the TD3 based algorithm is constructed in Section 3. In order to guarantee the performance of the algorithm, the proofs of convergence and stability are given in Section 4. Finally, some simulation examples are given in Section 5 to show the training process and effect of the algorithm more intuitively.

Notations: In this paper, the maximum eigenvalue and minimum eigenvalue of matrix M are written as $\lambda_{\max}(M)$ and $\lambda_{\min}(M)$ respectively. Matrix I_N is the identity matrix of order N, and matrix I_n is a 2nd order identity matrix. ∇ represents the gradient of the function and $\otimes$ is the Kronecker product. The $L_2 - norm$ o f the vector h is defined as $\|h\|$. And $M = diag\{m_1, m_2, m_3, \dots, m_n\}$ is the diagonal matrix with $m_1, m_2, m_3, \dots, m_n$ as the diagonal elements.

2. Preliminaries

2.1. Algebraic graph theory

For discrete-time multi-agent systems, the topological structure can be represented as $G = (V, E, \mathcal{A})$, where $V = \{v_1, v_2, v_3, \dots, v_N\}$ represents the agent set, which contains N agents. $E = \{(i, j) \in V \times V\}$ represents the edge set and $\mathcal{N} = \{1, 2, 3, \dots, N\}$ is the subscript set of agents. Matrix $\mathcal{A} = \{a_{ij} \in \mathbb{R}^{n\times n} \mid i, j \in \mathcal{N}\}$ is an nonnegative matrix, storing the adjacency information of graph G, the element a_{ij} is the connection weight between agent i and j. If agent i can receive the message from agent j, then $a_{ij} > 0$, otherwise $a_{ij} = 0$. Note that the topologies discussed in this paper are directed weighted simple graphs, meaning that $a_{ii} = 0, \forall i, j \in \mathcal{N}$. For each agent i, the neighbor set is defined as $N_i = \{j \mid v_j : (v_j, v_i) \in E, a_{ij} \neq 0\}$, and the number of the neighbors is $|N_i|$. The in-degree and out-degree of agent i are defined as $d_{in}(v_i) = \sum_{j=1}^{n} a_{ij}$ and $d_{out}(v_i) = \sum_{j=1}^{n} a_{ji}$ respectively. The degree matrix in this paper refers to the in-degree matrix, which is defined as $D = diag\{d_i \mid i = 1, 2, 3, \dots, N\}$. If the in-degree and out-degree of all agents in graph G are equal, the graph G is a balanced graph. The laplacian matrix of graph G is defined as $L = D - \mathcal{A}$, which stores all the information of graph G. For any two agents i and j, if a set of directed sequence $\{a_{k_1,i} > 0, a_{k_2,k_1} > 0, a_{k_3,k_2} > 0, \dots, a_{k_j,k_m} > 0\}$ exists, then the directed path exists between these two agents. And if agent i can reach any other agent in the graph G, the agent i is called the root node, and the graph G is said owing a spanning tree.

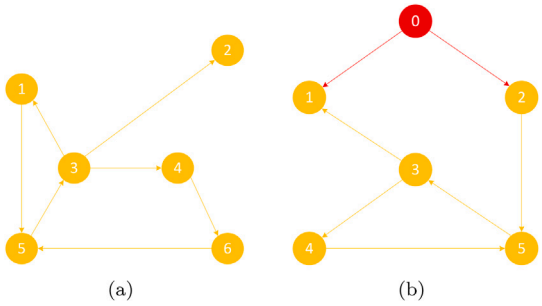


Fig. 1. Examples of communication topologies for leaderless MAS and leader-following MAS.

The MAS synchronization problem is generally divided into two parts: the leaderless MAS synchronization problem and the leader-following MAS synchronization problem. Two examples are shown in Fig. 1. As the special case of leader-following MASs, the agents in a leaderless MAS generally synchronize to an uncertain state. For leader-following MASs, the states of the following agents will be synchronized with that of the leader. The connection matrix $\Gamma = diag\{g_i \mid i = 1, 2, 3, \dots, N\}$ represents the connection weight between leader and following agents.

2.2. Optimal synchronization problem

In this paper, the leaderless MASs are considered as a special case of the leader-following MASs, and the synchronization problem is the foundation of all other MAS problems.

2.2.1. The synchronization of leaderless MASs

For a leaderless MAS with N agents, the dynamic equation is as follows:

$$x_i(k+1) = Ax_i(k) + Bu_i(k),\ i \in \mathcal{N}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are the inherent matrices of the system, $u_i \in \mathbb{R}^m$ is the control policies, $x_i \in \mathbb{R}^n$ denotes the states of the agent. The natural number in parentheses, such as k, $k+1$, represents the iteration number.

Remark 1. In this paper, the matrix pair (A, B) is stable and the inherent matrix A can have stable, marginally stable, or unstable eigenvalues. In fact, the complex state trajectory can better reflect the performance of the control policies than the state trajectory that tends to zero.

For leaderless MASs, the synchronization problem aims to bring all agents to a common uncertain state, which can be defined as:

$$\lim_{k\to\infty} \|x_i(k) - x_j(k)\| = 0,\ \forall i, j \in \mathcal{N}. \tag{2}$$

Due to the lack of access to global information in distributed MASs, agents rely on local communication to achieve synchronization. The consensus among neighbors is the only way to achieve synchronization. The consensus error for agent i can be defined as:

$$\varepsilon_i(k)=\sum_{j\in N_i} a_{ij}(x_j(k)-x_i(k)),\ \forall i\in\mathcal{N}, \tag{3}$$

where a_{ij} is the connection weight between the ith agent and its neighbors, and ε_i is defined as the consensus error for each agent i. The global consensus error vector is defined as follows:

$$\varepsilon=[\varepsilon_1^T,\varepsilon_2^T,\varepsilon_3^T,\ldots,\varepsilon_N^T]^T. \tag{4}$$

Thus we have:

$$\varepsilon(k)=-(L\otimes I_n)x_{ll}(k), \tag{5}$$

where $x_{ll}(k)=[x_1^T(k),x_2^T(k),x_3^T(k),\ldots,x_N^T(k)]^T$ is the state vector.

For model-based leaderless MASs, the control policies can be designed based on the idea of optimal control [35]:

$$u_i(k)=c(1+d_i)^{-1}K\varepsilon_i(k), \tag{6}$$

where c is the coupling gain and K is the feedback matrix, both of them can be calculated based on the model information and topology structure. d_i is the in-degree of agent i. The Laplace form of control policies is:

$$u(k)=c(I_N+D)^{-1}K\varepsilon(k). \tag{7}$$

The control policy (7) permits the lowest energy consumption while achieving synchronization, and the following is the energy consumption function:

$$J_i=\frac{1}{2}\sum_{k=0}^{\infty}\left(\varepsilon_i^T(k)Q\varepsilon_i(k)+u_i^T(k)Ru_i(k)\right), \tag{8}$$

where the weight matrices Q and R are both symmetric positive definite matrices. The communication energy consumption and enable energy consumption are both considered in the paper.

2.2.2. The synchronization of leader-following MASs

For a leader-following MAS with $N+1$ agents, the dynamic equation is as follows:

$$\begin{cases} x_i(k+1)=Ax_i(k)+Bu_i(k),\ i\in\mathcal{N} \\ x_0(k+1)=Ax_0(k) \end{cases}, \tag{9}$$

where x_0 is the leader agent. The goal of this kind of system is that the state of each following agent is ultimately the same as that of the leader:

$$\lim_{k\to\infty}\|x_i(k)-x_0(k)\|=0,\ \forall i\in\mathcal{N}. \tag{10}$$

The tracking error is defined as:

$$\eta=x_{lf}(k)-\bar{x}_0(k), \tag{11}$$

where $x_{lf}(k)=[x_1^T(k),x_2^T(k),x_3^T(k),\ldots,x_N^T(k)]^T$ is the states of all the following agents at step k, and $\bar{x}_0(k)=1_N\otimes x_0(k)$ is the leader state vector at step k.

Similar to leaderless MASs, not every following agent has access to the leader's information. The system synchronization can only be achieved indirectly by synchronizing each agent's state with its neighbors, since the tracking error cannot be used to construct control policies. The consensus error can be defined as follows:

$$\varepsilon_i(k)=\sum_{j\in N_i} a_{ij}(x_j(k)-x_i(k))+g_i(x_0(k)-x_i(k)),\ \forall i\in\mathcal{N}, \tag{12}$$

where g_i is the connection weight between the following agent i and the leader, if the following agent can receive the information from the leader, $g_i\neq 0$, otherwise $g_i=0$. Then we have:

$$\varepsilon(k)=-\left((L+\Gamma)\otimes I_n\right)x_{lf}(k)+\left((L+\Gamma)\otimes I_n\right)\bar{x}_0(k), \tag{13}$$

where Γ is the communication matrix between the following agents and the leader.

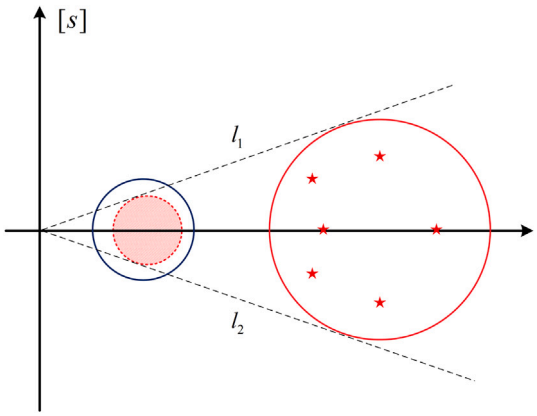


Fig. 2. The Riccati design circle and covering circle of the weighted graph matrix eigenvalues.

Lemma 1 ([32]). *If the topology matrix $L+\Gamma$ is non-singular, then the consensus error and the tracking error satisfy the following inequality:*

$$|\eta(k)\ \|\leq(\lambda_{\min}(L+\Gamma))\|\varepsilon(k)\|, \tag{14}$$

Remark 2. The convergence of tracking error can be guaranteed if the consensus error converges to 0, which is a sufficient condition based on Lemma 1.

If the model information is known, the control policies can be designed as follows [35]:

$$u_i(k)=c(1+d_i+g_i)^{-1}K\varepsilon_i(k), \tag{15}$$

and it can be written as:

$$u(k)=c\left((I_N+D+G)^{-1}\otimes K\right)\varepsilon(k), \tag{16}$$

where the coupling gain c can be calculated by the topology structure, the feedback matrix K can be calculated by model information.

Remark 3. Define the weighted graph matrix as $\Xi=(I_N+D+\Gamma)^{-1}(L+\Gamma)$, with $\{\Lambda_k|k=1,2,3,\ldots,N\}$ being the eigenvalues of it. The covering circle $C(c_0,r_0)$ of the eigenvalues should be radially scaled and projected by the origin to a circle C_s, which is concentric with the Riccati design circle $C(1,r)$ and contained in it. The mapping relationship is shown in Fig. 2.

There are still some assumptions that need to be put forward, since not only they are the basic assumptions of the above work, but also the work in this paper obeys them.

Assumption 1. The inherent matrix A and B of MASs (1) and (9) are time-invariant and unknown matrices.

Assumption 2. The matrix pair (A,B) is controllable.

Assumption 3. In leader-following MASs, the system topology has at least a directed spanning tree, and at least one following agent can communicate with the leader.

Remark 4. Assumption 3 is a sufficient and necessary condition for the matrix $L+\Gamma$ to be non-negative.

2.2.3. The methods based on the Reinforcement Learning

As we can see, all the methods mentioned above are solutions to model-based situations. However, the system information is not available in all conditions. On the one hand, modeling and identifying MASs are complex and difficult tasks that require a significant amount

of time and effort, and the resulting models may not be accurate. On the other hand, factors such as pressure, temperature, humidity, magnetic field, and component aging caused by long-term operation will lead to the change of the inherent properties. Then the control accuracy may decrease, and the system may even be out of control. Therefore, it is essential to develop data-driven, model-free adaptive algorithms.

Recent works on the synchronization of MASs or formation based on RL have partially addressed the aforementioned issues. Based on the idea of Q-learning [36], the Q function is constructed to estimate the value of error–action pair:

$$Q(\varepsilon_k, A_k) = Q(\varepsilon_k, A_k) + \alpha(R + \gamma \max_{A}(\varepsilon'_{k+1}, A_{k+1}) - Q(\varepsilon_k, A_k)), \tag{17}$$

where k is the iteration step, ε is the consensus error, A is the control policies, α and γ are the discount factors, R is the step reward, $Q(\varepsilon_k, A_k)$ is the value of the error–action pair at step k. And the control policies are those that can maximize the Q function.

Although the existing methods can iteratively obtain optimal control policies in model-free conditions, they still require initial admissible policies. This assumption severely limits the applicability of the mentioned methods since it assumes that the system model is actually partially known in advance and can be synchronized. What is more, due to the fact that the information is not independent and identically distributed, the training process is slow and sometimes the policies may diverge the system. In order to solve the existed problems, a synchronization algorithm based on TD3 is put forward in this work.

2.3. Nash equilibrium solutions

In distributed discrete-time MASs, each agent can only obtain information from its neighbors, so the global optimal policies need to game between individual energy consumption and global energy consumption to obtain the Nash equilibrium solution.

Lemma 2 ([37,38]). *Nash equilibrium: If the MAS, which is driven by a set of policies, reach the target state with the lowest global energy consumption, then the system has reached the Nash equilibrium. The policies are called the optimal policies under the Nash equilibrium, and the inequality is as follows:*

$$\begin{aligned} V_i^* &= V_i(a_1^*, a_2^*, a_3^*, \dots, a_i^*, \dots, a_N^*) \\ &\le V_i(a_1^*, a_2^*, a_3^*, \dots, a_i, \dots, a_N^*). \end{aligned} \tag{18}$$

Lemma 3 ([37]). *Admissible control: If a set of policies $u(k)$ can not only synchronize the leaderless MAS (1) and the leader-following MAS (9), but also bound the loss function (8), then they are called admissible control policies.*

In fact, all control problems in MASs aim to find Nash equilibrium through different kinds of control policies, while ensuring specific performance indexes under certain situations.

3. Synchronization policies based on TD3

3.1. Virtual neighbors and network construction

Based on the Q-learning method [36], the proposed solution, which can be seen as a type of value-based method, is going to estimate the state value and indirectly generate the optimal policies. However, the state space and action space are both infinite for discrete-time MASs, which will lead to the explosion of dimensionality. The neural network can be used to overcome it. Before constructing them, the virtual neighbor is introduced first to solve the problem of inconsistent input dimensions.

Based on traditional methods, the optimal policies can be constructed through consensus error. However, each agent in the MAS has a variable number of neighbors, which results in different input

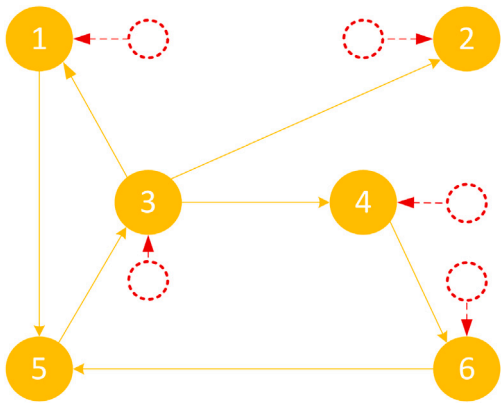


Fig. 3. Schematic diagram of virtual neighbors (The red dashed circles are the virtual neighbors replenished for each agent).

dimensions. Besides, the neural network requires a fixed structure for training. Therefore, the input data should be preprocessed first. As shown in Fig. 3, the virtual neighbor technology is introduced to accomplish it. Selecting the agent with the largest number of neighbors as the benchmark, then virtual neighbors will be added to agents with fewer neighbors. Unlike the agents in real system, the virtual neighbors' states are always synchronized with agent i, and no control policies are required. During the data collection process, the state data of virtual neighbors will be randomly inserted into the normal data. In fact, the consensus error and control policies generated by the virtual neighbors are all 0, which is the least impactful way for the value estimation and policies generation.

The system flow chart is shown in Fig. 4 to visually depict the data processing, and various parts of the controller will be introduced in order. The critic net, which is actually a BP net, is used to accurately estimated the value of the error–action pair. $c\_in = [\varepsilon_1^T, \varepsilon_2^T, \dots, \varepsilon_m^T, a_1^T, a_2^T, \dots, a_m^T]^T, m \in \mathcal{I}_i$ is the input, the net will output the estimated value. The structure of it is:

$$\begin{cases} L_1 = r(W_1 c\_in + B_1) \\ L_2 = r(W_2 L_1 + B_2) \\ \quad\vdots \\ O = r(W_{nc} L_{nc-1} + B_{nc}) \end{cases}, \tag{19}$$

where W_i is the coefficient matrix, B_i is the bias matrix, L_{i-1} is the hidden layer, i is the index of the layer. O is the output, nc is the number of net layers, and $r(\cdot)$ is the $relu$ activation function. In the following part, the critic net and actor net will be represented as $Q(\varepsilon, a|\theta^Q)$ and $\pi(\varepsilon|\theta^\pi)$ respectively, and θ is the net parameter. Then the Temporal-Difference target($TD-target$) is:

$$y_k = r_k + \gamma Q(\varepsilon_{k+1}, a_{k+1}|\theta^Q), \tag{20}$$

where r_k is the reward at step k, γ is the discount factor, k and $k+1$ is the step index. The $TD-target$ is the updating target of the critic net $Q(\varepsilon, a|\theta^Q)$, which plays the same role as labels in supervised learning. So RL is actually a kind of in-label supervised learning. Due to the fact that the nets are updated in batches, the Mean Square Error (MSE) is used to update it:

$$L_c = \frac{1}{N}\sum_{k=1}^{N}\left(\left(r_k + \gamma Q(\varepsilon_{k+1}, a_{k+1}|\theta^Q)\right) - Q(\varepsilon_k, a_k|\theta^Q)\right)^2, \tag{21}$$

where N is the batch size.

The actor net $\pi(\varepsilon|\theta^\pi)$, whose aim is to find optimal policies that can maximize the critic net $Q(\varepsilon, \pi|\theta^Q)$, is also a BP net. Its structure is as

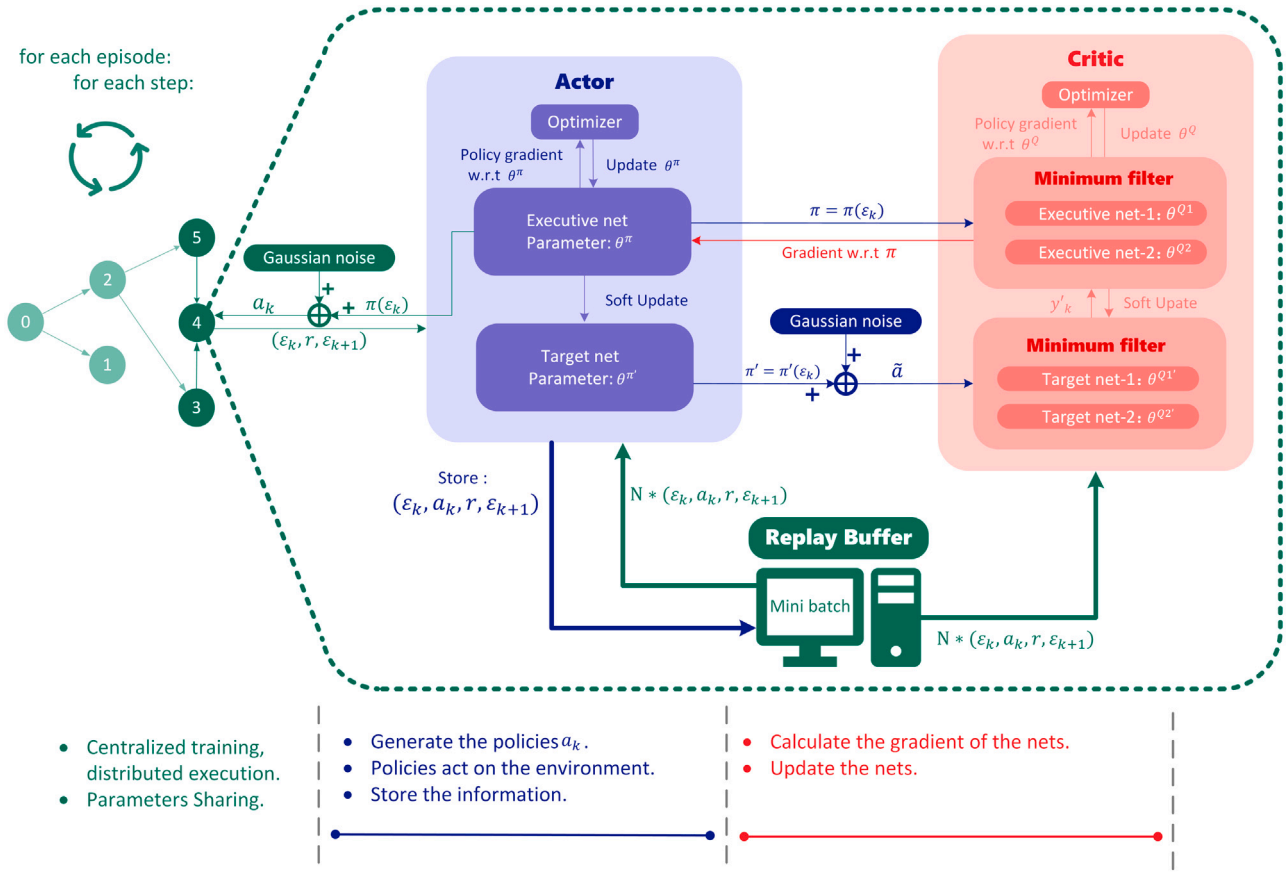


Fig. 4. The flow chart of synchronization algorithm based on TD3 for discrete-time MASs.

follows:

$$\begin{cases} L_1 = r(W_1 a\_in + B_1) \\ L_2 = r(W_2 L_1 + B_2) \\ \vdots \\ L_{na} = \tanh(W_{na} L_{na-1} + B_{na}) \\ O = L_{na} \times action\_bound \end{cases}, \tag{22}$$

where $a\_in = [\varepsilon_1^T, \varepsilon_2^T, \ldots, \varepsilon_m^T]^T, m \in \mathcal{I}_i$ is the input of the net, and $\mathcal{I}_i = \{1, 2, 3, \ldots, m\}$ is the subscript set of the neighbor set. $\tanh(\cdot)$ is the *tanh* activation function, *action_bound* is the upper bound of the policies. Different from the critic net $Q(\varepsilon, a|\theta^Q)$, the output values will be normalized first and then be zoomed in to the desired sizes, which ensures that the generated policies are within the executable and safe range.

Remark 5. The output value of most actuators is bounded, which ensures the appropriateness of setting the *action_bound* [34]. Moreover, the TD3-based solution learns how to stabilize the system from divergent system data and then learns how to minimize energy consumption. Thus, setting a reasonable action boundary is crucial.

Similar to $\varepsilon-greedy$ strategy, a normal distributed noise will be added to the deterministic strategy of $\pi(\varepsilon|\theta^{\pi})$:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{23}$$

where σ is the noise variance, which controls the exploration rate.

Remark 6. Actually, the variance will decay with the iteration step in the training process. A large exploration rate in the early stage helps to explore the action space effectively, and a small exploration rate in the later stage is more helpful for system stability.

Unlike policy-based methods, which generate optimal policies directly, the TD3-based method will develop policies aimed at maximizing the $Q(\varepsilon, a|\theta^Q)$ and hence achieving optimal control indirectly. The value function is as follows:

$$J_{\beta}(\pi_{\theta}) = \int_S \rho^{\beta}(\varepsilon) Q(\varepsilon, \pi(\varepsilon)) d\varepsilon, \tag{24}$$

where S is the set of all possible situations, $\rho^{\beta}(\varepsilon)$ is the probability of each event. Taking the derivative of Eq. (24) with respect to θ of $\pi(\varepsilon|\theta^{\pi})$, we obtain:

$$\begin{aligned} \nabla_{\theta^{\pi}} J_{\beta}(\pi_{\theta}) &\approx \int_S \rho^{\beta}(\varepsilon) \nabla_{\theta^{\pi}} \pi(\varepsilon|\theta^{\pi}) Q(\varepsilon, \pi(\varepsilon)) d\varepsilon \\ &= \mathbb{E}_{\varepsilon \to \rho^{\beta}} \left[\nabla_{\pi} Q(\varepsilon, a)|_{a=\pi(\varepsilon)} \cdot \nabla_{\theta^{\pi}} \pi(\varepsilon|\theta^{\pi}) \right]. \end{aligned} \tag{25}$$

The gradient of the $\pi(\varepsilon|\theta^{\pi})$ can be unbiased estimated by averaging the data in mini-batch based on Monte-Carlo Method [39], and Eq. (25) can be written as:

$$\nabla_{\theta^{\pi}} J_{\beta}(\pi_{\theta}) = \frac{1}{N} \sum_{k=1}^{N} \left[\nabla_{\pi} Q(\varepsilon, a)|_{\varepsilon=\varepsilon_k, a=\pi(\varepsilon_k)} \cdot \nabla_{\theta^{\pi}} \pi(\varepsilon|\theta^{\pi})|_{\varepsilon=\varepsilon_k} \right], \tag{26}$$

where N is the number of the mini-batch.

Both the actor net and the critic net change at the same time since the actor–critic structure-based algorithms are iteratively updated. As a result, it is difficult to keep the nets stable in the face of changing targets. The Target Net technology [39] is used to solve it: the target nets $Q'(\varepsilon, a|\theta^{Q'})$ and $\pi'(\varepsilon|\theta^{\pi'})$ are constructed with the same structure and parameters as $Q(\varepsilon, a|\theta^Q)$ and $\pi(\varepsilon|\theta^{\pi})$, but they only take effect on target value generation. Thus the Eqs. (20) and (21) can be rewritten as:

$$y'_k = r_k + \gamma Q'(\varepsilon_{k+1}, a'_{k+1}|\theta^{Q'}). \tag{27}$$

$$L_c = \frac{1}{N}\sum_{k=1}^{N}\left(\left(r_k + \gamma Q'(\varepsilon_{k+1}, a'_{k+1}|\theta^{Q'})\right) - Q(\varepsilon_k, a_k|\theta^Q)\right)^2. \tag{28}$$

After the MAS runs for several steps, the target nets will be updated once using the Moving Average method, which can significantly improve the net stability. The update equations are as follows:

$$\begin{cases} \theta^{Q'} = \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\pi'} = \tau\theta^\pi + (1-\tau)\theta^{\pi'} \end{cases}, \tag{29}$$

where $\tau \in [0, 1]$ is the moving update factor.

While updating the nets, the following policies which can maximize the value of error-state will be used:

$$\theta_{k+1} = \theta_k + \alpha\left[r + \gamma \max Q'(\varepsilon, a'|\theta^{Q'}) - Q(\varepsilon, a|\theta^Q)\right]\nabla Q(\varepsilon, a|\theta^Q), \tag{30}$$

where α is the discount factor. Such a method leads to serious overestimation problem, resulting in relatively large consensus errors. The double Q Network technology [39] is proposed to solve it. Two critic nets with the same structure but different parameters can be constructed to evaluate the value, and the smaller one will be used to update the nets.

What is more, in order to increase the stability of the value estimation, the Target Policy Smoothing Regularization technology [34] is proposed: a gaussian noise can be added to the strategy when $\pi'(\varepsilon|\theta^{\pi'})$ is working:

$$\widetilde{a} = \pi'(\varepsilon|\theta^{\pi'}) + \xi, \quad \xi \sim N(0, c^2), \tag{31}$$

where c is the target-noise factor. This method will result in a more conservative controller, with the noise space included in the stability margin. Thus Eqs. (20) and (21) can be written as:

$$y'_k = r_k + \gamma \min_{i=1,2} {Q_i}'(\varepsilon_{k+1}, \widetilde{a}_{k+1}|{\theta^{Q_i}}'). \tag{32}$$

$$L_c = \frac{1}{N}\sum_{k=1}^{N}\left(y'_k - Q(\varepsilon_k, a_k|\theta^Q)\right)^2. \tag{33}$$

The delay Update technology [34] will be employed to improve training efficiency even more. When critic nets are changing all the time, it is inefficient and energy-consuming to find policies that maximize the value of error–action pairs. A good way is to update actor nets after several updates of critic nets, where *policy_freq* is the parameter used to record the number of delayed updates.

3.2. The value function and data processing

The proposed algorithm is a kind of centralized training and distributed control method. Given that each agent in the distributed topologies may only obtain information from its neighbors, the consensus error (3) or (12) will act as the input of the nets in order to generate the optimal policies. According to Lemma 1, when the MASs are synchronized, the consensus error tends to 0. Hence the opposite of the tracking error can be used as the current state value:

$$V_i = -\frac{\|\varepsilon_1^T, \varepsilon_2^T, \ldots, \varepsilon_i^T, \ldots, \varepsilon_{N_i}^T\|^2}{N_i}, \tag{34}$$

where N_i is the number of the neighbors for agent i.

Remark 7. The design of the value function is critical in RL tasks since it has a direct impact on the optimization results. In general, it is designed by final performance indicators rather than intermediate factors. Otherwise, some unexpected optimization results may be acquired. What is more, the consensus error is proportional to the control policies for linear time-invariant MASs (1) and (9). If the control policies are taken in to account, steady-state errors will occur with relatively small policies, which is not what we want.

For methods based on BP neural networks, the training data needs to be independent and identically distributed. However, the state data of MAS is time-dependent, leading to that the controllers are difficult to stabilize. The replay Buffer technology [36] is introduced to solve it. Before training the nets, a replay buffer will be built to store the agent's state, and the information of virtual neighbors is also randomly inserted into the data. When the training begins, a certain batch of data is detached. This method not only solves the problem of time-coupling of data but also enhances the robustness of policies by including both unstable and stable data.

If the critic nets are trained on error–action pairs with a wide range of values, it may lead to inaccurate value estimations. Furthermore, too big data has an impact on the neural network's stability. As a result, the tracking error and control policies will be normalized by dividing by *error_bound* and *action_bound*. After that, the training speed has also been improved to a certain extent. So far, all the structures of the controller have been introduced. The algorithm steps are shown in Algorithm 1 to show the running process more intuitively.

Remark 8. The TD3 based algorithm is suitable for both leaderless and leader-following MASs, and the former can be regarded as a special case of the latter. Furthermore, in this paper, the target nets' updating frequency and the delayed updating frequency of the actor net are the same. Of course, the above two update frequencies can be changed independently based on project requirements.

4. Proof of the convergence and stability

In this section, relevant proofs of convergence and stability of the proposed algorithm will be given respectively to ensure the robustness of the algorithm. The parameters of critic nets are updated by gradient derivation method [39] as follows:

$$\theta_k^Q = \theta_k^Q + \alpha_\theta L_{c(k)}\nabla Q(\varepsilon_k, a_k|\theta^Q), \tag{35}$$

where α_θ is the learning rate of the critic net, $L_{c(k)}$ is the MSE error at step k, $\nabla Q(\varepsilon_k, a_k|\theta^Q)$ is the gradient. And the above two parameters determine the step size and direction of update respectively. Note that the Target net technology and Double Net technology only affect the convergence speed, but they have nothing to do with the synchronization of the MAS, so we only need to analysis one critic net [34,39]. Since the critic nets try to minimize the MSE loss, which is proportional to the target value y_t, the following analysis will base on it.

4.1. Proof of convergence

Lemma 4. *For $\forall k \in \mathbb{N}$ and $\forall s \in \mathbb{N}$, the value of critic net $Q(\varepsilon, a|\theta^Q)$ is monotonous and non-decreasing under the driven of a set of policies $a(k)$ with a set of small initial values:*

$$Q_k^{s+1}(\varepsilon_k^{s+1}, a_k^{s+1}|\theta^Q) \geq Q_k^s(\varepsilon_k^s, a_k^s|\theta^Q) \tag{36}$$

where k is the running step of the MASs and s is the iteration step.

Proof. Considering that the reward r is related to the consensus error and policies at step k, the reward can be expressed as $r_k^s(\varepsilon_k^s, \pi^s(\varepsilon_k|\theta^\pi))$. Then the $TD-target$ can be written as:

$$y_k^{s+1} = r_k^{s+1}(\varepsilon_k^{s+1}, \pi^{s+1}(\varepsilon_k^{s+1}|\theta^\pi)) + \gamma Q_{k+1}^s(\varepsilon_{k+1}^s, \pi^s(\varepsilon_{k+1}^s|\theta^\pi)|\theta^Q). \tag{37}$$

Suppose that for $\forall k = 0, 1, 2, \ldots$, the inequality (36) holds:

$$Q_{t+1}^{s+1}(\varepsilon_{t+1}^{s+1}, a_{t+1}^{s+1}|\theta^Q) \geq Q_{t+1}^s(\varepsilon_{t+1}^s, a_{t+1}^s|\theta^Q). \tag{38}$$

Then let $k = t$, the following inequality holds:

$$\begin{aligned} y_t^{s+1} &= r_t^{s+1}(\varepsilon_t^{s+1}, \pi^{s+1}(\varepsilon_t^{s+1}|\theta^\pi)) + \gamma Q_{t+1}^s(\varepsilon_{t+1}^s, \pi^s(\varepsilon_{t+1}^s|\theta^\pi)|\theta^Q) \\ &= \max\left\{r_t^s(\varepsilon_t^s, \pi^s(\varepsilon_t^s|\theta^\pi))\right\} + \gamma Q_{t+1}^s(\varepsilon_{t+1}^s, \pi^s(\varepsilon_{t+1}^s|\theta^\pi)|\theta^Q) \\ &\geq r_t^s(\varepsilon_t^s, \pi^s(\varepsilon_t^s|\theta^\pi)) + \gamma Q_{t+1}^{s-1}(\varepsilon_{t+1}^{s-1}, \pi^{s-1}(\varepsilon_{t+1}^{s-1}|\theta^\pi)|\theta^Q) \end{aligned}$$

Algorithm 1: TD3 Based Synchronization Algorithm

Initialize:

1. Initialize two critic nets and one actor net $Q_{\theta^{c1}}$, $Q_{\theta^{c2}}$ and $\pi_{\theta^{\pi}}$ with parameters θ^{c1}, θ^{c2} and θ^{π}.
2. Initialize target nets $Q'_{\theta^{c1'}}$,$Q'_{\theta^{c2'}}$ and $\pi'_{\theta^{\pi'}}$ with the same structures and parameters $\theta^{c1'}$, $\theta^{c2'}$ and $\theta^{\pi'}$ as the critic net and the actor net.
3. Initialize the replay buffer R.

for $t = 1$ *to* $Episode$ **do**
  Initialize the MAS environment.(Leaderless MASs or Leader-following MASs environment)
  for $k = 1$ *to step* **do**
    (1) Generate policies with gaussian noise $a_k^i = \pi(\varepsilon_k) + \varsigma, \varsigma \sim N(0, \sigma^2)$ for each agent and work on the MAS.
    (2) Get the feedback from the environment and store the data (consensus error ε_k, policies a_k, reward r, consensus error for the next step ε_{k+1}) to R.
    (3) Extract a *mini_batch* data $(\varepsilon_k, a_k, r, \varepsilon_{k+1})$ from R, calculate the policies $\widetilde{a} = \pi'(\varepsilon|\theta^{\pi'}) + \xi, \xi \sim N(0, c^2)$ for target, and calculate the target value $y'_k = r_k + \gamma \min_{i=1,2} Q_i{}'(\varepsilon_{k+1}, \widetilde{a}_{k+1}|\theta^{Q'_i})$.
    (4) Update the critic nets $Q_{\theta^{c1}}$ and $Q_{\theta^{c2}}$ by gradient boosting algorithm.
    if *the system runs for* $policy\_freq$ *steps,* **then**
      ① Update the act net $\pi_{\theta^{\pi}}$ by deterministic policies gradient algorithm.
      ② Update the target nets $Q'_{\theta^{c1'}}$,$Q'_{\theta^{c2'}}$ and $\pi'_{\theta^{\pi'}}$ by Moving Average method:

$$\begin{cases} \theta^{Q'} = \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\pi'} = \tau\theta^{\pi} + (1-\tau)\theta^{\pi'} \end{cases}.$$

    end
    if *the state of an agent exceeds the set state boundary,* **then**
      Terminate this cycle and enter the next one.
    end
  end
end

$$= y_t^s. \tag{39}$$

According to the gradient derivation method, Eq. (36) holds at the finite number of iteration steps. When $k \to \infty$, the reward $r \approx 0$. Let $k = T$, suppose that when $T \to \infty$, then the following inequality holds:

$$Q_T^{s+1}(\varepsilon_T^{s+1}, a_T^{s+1}|\theta^Q) \geq Q_T^s(\varepsilon_T^s, a_T^s|\theta^Q). \tag{40}$$

Then let $k = T - 1$:

$$\begin{aligned} y_{T-1}^{s+1} &= r_{T-1}^{s+1}(\varepsilon_{T-1}^{s+1}, \pi^{s+1}(\varepsilon_{T-1}^{s+1}|\theta^{\pi}))) + \gamma Q_T^s(\varepsilon_T^s, \pi^s(\varepsilon_T^s|\theta^{\pi}))|\theta^Q) \\ &= \max\left\{r_{T-1}^s(\varepsilon_{T-1}^s, \pi^s(\varepsilon_{T-1}^s|\theta^{\pi}))\right\} + \gamma Q_T^s(\varepsilon_T^s, \pi^s(\varepsilon_T^s|\theta^{\pi})|\theta^Q) \\ &\geq r_{T-1}^s(\varepsilon_{T-1}^s, \pi^s(\varepsilon_{T-1}^s|\theta^{\pi})) + \gamma Q_T^{s-1}(\varepsilon_T^{s-1}, \pi^{s-1}(\varepsilon_T^{s-1}|\theta^{\pi})|\theta^Q) \\ &= 0 + \gamma Q_{t+1}^{s-1}(\varepsilon_T^{s-1}, \pi^{s-1}(\varepsilon_T^{s-1}|\theta^{\pi})|\theta^Q) \\ &= y_{T-1}^s. \end{aligned} \tag{41}$$

According to Eqs. (39) and (41), Eq. (36) holds in all the processes of MASs operation. □

Theorem 1. *Considering the leaderless MAS (1) and leader-following MAS (9), if the communication topologies of them satisfy the Assumptions 1∼3, and they are driven by the act net (22) and the critic net (19) of TD3 based algorithm, when $s \to \infty$, the actor nets $\pi^s(\varepsilon|\theta^{\pi})$ and critic nets $Q^s(\varepsilon, a|\theta^Q)$ will converge to the optimal act net $\pi^*(\varepsilon|\theta^{\pi})$ and critic net $Q^*(\varepsilon, a|\theta^Q)$.*

Proof. According to Lemma 4, the $TD-target$ (37) is monotonous and non-decreasing, so its limit exists:

$$y_k^{\infty} = \lim_{s\to\infty} y_k^s, \tag{42}$$

and the following inequality holds:

$$\begin{aligned} y_k^{\infty} &\geq y_k^s \\ &= r_k^{s+1}(\varepsilon_k^{s+1}, \pi^{s+1}(\varepsilon_k^{s+1}|\theta^{\pi})) + \gamma Q_{k+1}^s(\varepsilon_{k+1}^s, \pi^s(\varepsilon_{k+1}^s|\theta^{\pi})|\theta^Q) \\ &= \max\left\{r_k^s(\varepsilon_k^s, \pi^s(\varepsilon_k^s|\theta^{\pi}))\right\} + \gamma Q_{k+1}^s(\varepsilon_{k+1}^s, \pi^s(\varepsilon_{k+1}^s|\theta^{\pi})|\theta^Q), \end{aligned} \tag{43}$$

when $s \to \infty$:

$$y_k^{\infty} \geq \max\left\{r_k^{\infty}(\varepsilon_k^{\infty}, \pi^{\infty}(\varepsilon_k^{\infty}|\theta^{\pi}))\right\} + \gamma Q_{k+1}^{\infty}(\varepsilon_{k+1}^{\infty}, \pi^{\infty}(\varepsilon_{k+1}^{\infty}|\theta^{\pi})|\theta^Q). \tag{44}$$

Due to the features of $TD-target$ (37), an iteration step number P must exists that can make the following inequality holds:

$$\begin{aligned} y_k^{\infty} &\leq y_k^{P+1} - \rho \\ &= r_k^{P+1}(\varepsilon_k^{P+1}, \pi^{P+1}(\varepsilon_k^{P+1}|\theta^{\pi})) + \gamma Q_{k+1}^P(\varepsilon_{k+1}^P, \pi^P(\varepsilon_{k+1}^P|\theta^{\pi})|\theta^Q) - \rho \\ &= \max\left\{r_k^P(\varepsilon_k^P, \pi^P(\varepsilon_k^P|\theta^{\pi}))\right\} + \gamma Q_{k+1}^P(\varepsilon_{k+1}^P, \pi^P(\varepsilon_{k+1}^P|\theta^{\pi})|\theta^Q) - \rho, \end{aligned} \tag{45}$$

where ρ can be any constant, when $\rho = 0$:

$$y_k^{\infty} \leq \max\left\{r_k^P(\varepsilon_k^P, \pi^P(\varepsilon_k^P|\theta^{\pi}))\right\} + \gamma Q_{k+1}^P(\varepsilon_{k+1}^P, \pi^P(\varepsilon_{k+1}^P|\theta^{\pi})|\theta^Q). \tag{46}$$

Combining inequality (43) and inequality (46), we can get:

$$y_k^{\infty} = \max\left\{r_k^P(\varepsilon_k^P, \pi^P(\varepsilon_k^P|\theta^{\pi}))\right\} + \gamma Q_{k+1}^P(\varepsilon_{k+1}^P, \pi^P(\varepsilon_{k+1}^P|\theta^{\pi})|\theta^Q). \tag{47}$$

According to the gradient derivation method, the value of error–action will converge to a certain value. Thus the actor net will converge to a certain structure:

$$Q^{\infty}(\varepsilon, a|\theta^Q) = \lim_{s\to\infty} Q^s(\varepsilon, a|\theta^Q). \tag{48}$$

In order to ensure that the value of critic net $Q^{\infty}(\varepsilon, a|\theta^Q)$ is the optimal value, it is necessary to prove that:

$$Q^{\infty}(\varepsilon, a|\theta^Q) \geq Q^s(\varepsilon, a|\theta^Q), \forall s = 0, 1, 2, \ldots \tag{49}$$

Suppose that the inequality (49) holds when $s = m$:

$$\begin{aligned} y_k^m &= r_k^m(\varepsilon_k^m, \pi^m(\varepsilon_k^m|\theta^{\pi})) + \gamma Q_{k+1}^{m-1}(\varepsilon_{k+1}^{m-1}, \pi^{m-1}(\varepsilon_{k+1}^{m-1}|\theta^{\pi})|\theta^Q) \\ &\leq \max\left\{r_k^m(\varepsilon_k^m, \pi^m(\varepsilon_k^m|\theta^{\pi}))\right\} + \gamma Q_{k+1}^m(\varepsilon_{k+1}^m, \pi^m(\varepsilon_{k+1}^m|\theta^{\pi})|\theta^Q) \\ &\leq 0 + \gamma Q_{k+1}^{\infty}(\varepsilon_{k+1}^{\infty}, \pi^{\infty}(\varepsilon_{k+1}^{\infty}|\theta^{\pi})|\theta^Q) \\ &= y_k^{\infty}. \end{aligned} \tag{50}$$

When $s = n, n \to \infty$, we have

$$y_k^n = r_k^n(\varepsilon_k^n, \pi^n(\varepsilon_k^n|\theta^{\pi})) + \gamma Q_{k+1}^{n-1}(\varepsilon_{k+1}^{n-1}, \pi^{n-1}(\varepsilon_{k+1}^{n-1}|\theta^{\pi})|\theta^Q)$$

$$\begin{aligned}
&\leq \max\left\{r_k^n(\varepsilon_k^n, \pi^n(\varepsilon_k^n|\theta^\pi))\right\} + \gamma Q_{k+1}^n(\varepsilon_{k+1}^n, \pi^n(\varepsilon_{k+1}^n|\theta^\pi)|\theta^Q)\\
&= 0 + \gamma Q_{k+1}^n(\varepsilon_{k+1}^n, \pi^n(\varepsilon_{k+1}^n|\theta^\pi)|\theta^Q)\\
&\leq \gamma Q_{k+1}^\infty(\varepsilon_{k+1}^\infty, \pi^\infty(\varepsilon_{k+1}^\infty|\theta^\pi)|\theta^Q)\\
&= y_k^\infty. \qquad (51)
\end{aligned}$$

According to inequalities (50) and (51), the inequality (49) holds.

Let $a = a*$, then:

$$Q^s(\varepsilon, a|\theta^Q) \leq Q^\infty(\varepsilon, a|\theta^Q) \leq Q^\infty(\varepsilon, a*|\theta^Q). \qquad (52)$$

And according to inequality (48), the value of critic net is minimum when $s \to \infty$, then:

$$Q^\infty(\varepsilon, a|\theta^Q) = Q^*(\varepsilon, a|\theta^Q), \qquad (53)$$

combining inequalities (52) and (53), the following equation holds:

$$\begin{aligned}
Q^\infty(\varepsilon, a|\theta^Q) &= \lim_{s\to\infty} Q^s(\varepsilon, a|\theta^Q)\\
&= Q^*(\varepsilon, a^*|\theta^Q). \quad \square \qquad (54)
\end{aligned}$$

4.2. Proof of stability

Theorem 2. *Considering the Leaderless MAS (1) and Leader-following MAS (9), if the Assumptions 1∼3 holds, and the value of the error − action pair and the policies are given by critic net (19) and actor net (22), the consensus error (3) and (12) are asymptotic stable. Besides, when $k \to \infty$, $\varepsilon \to 0$.*

Proof. According to Eq. (20), the following equation holds:

$$-r_k = \gamma Q^*(\varepsilon_{k+1}, a_{k+1}^*|\theta^Q) - Q^*(\varepsilon_k, a_k^*|\theta^Q), \qquad (55)$$

then the Lyapunov Difference Function can be designed as:

$$\begin{aligned}
\Delta\left(\gamma^k Q^*(\varepsilon_k, a_k^*)\right) &= \gamma^{k+1} Q^*(\varepsilon_{k+1}, a_{k+1}^*|\theta^Q) - r^k Q^*(\varepsilon_k, a_k^*|\theta^Q)\\
&= -\gamma^k \cdot r_k\\
&\leq 0. \qquad (56)
\end{aligned}$$

The inequality (56) shows that the consensus error is asymptotic stable: when $s \to \infty, \varepsilon \to 0$, the states of all agents are finally synchronized. $\square$

5. Simulation results

5.1. Simulation for leaderless MASs

Considering a linear time-invariant leaderless MAS with 5 agents, the system matrices are designed as:

$$A = \begin{bmatrix} -2.5 \pm 0.3\Psi & 1.9 \\ -2.4 & 2 \pm 0.1\Psi \end{bmatrix}, \quad B = \begin{bmatrix} 1.5 \pm 0.02\Psi \\ 0 \end{bmatrix},$$

where Ψ is an environmental factor used to model the effects of environmental factors on the system matrices.

The communication topology shown in Fig. 5 includes a directed spanning tree, and the Laplace Matrix is as follows:

$$L = \begin{bmatrix} 3 & 0 & 0 & -3 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 2 & -2 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}.$$

The *max_state* of the simulation environment is 1000, the *max_action* is 1500, and the *max_step* is 500. All of them are set to ensure sufficient space margin and time margin. A large enough operating space not only provides sufficient data needed for net training, but also allows the controller to gain experience through trial and error. Besides, the initial states are randomly generated within the range [−20, 20].

The controller contains 2 actor nets and 4 critic nets. In addition to the input layer and the output layer, each net contains 3 hidden layers, and each layer has 256 neurons. All the parameters required for training are shown in Table 1 for convenience.

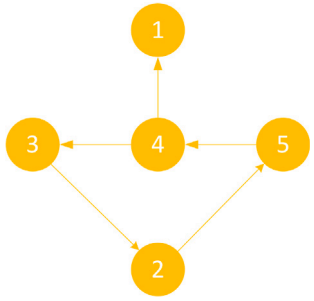


Fig. 5. The communication topology diagram of the leaderless MAS (In order to show the topology more intuitively, the virtual neighbors are not shown).

Table 1
The hyper parameters of TD3 controller for leaderless MAS.

Variable	Value	Variable	Value
max_s	1000	max_step	500
max_action	1500	max_initial_s	20
Buffer size	1 000 000	a_lr	0.0002
c_lr	0.0002	γ	0.99
τ	0.005	policy_noise	0.02
g_initial_noise	0.5	noise_clip	0.1
batch_size	128	Episode	50
policy_freq	3		

Table 2
The relationship between the parameters and the controller's performance.

Performance	Controller variables
Explore Space	↑:max_state, max_action, Buffer size, max_step, max_initial_state, Episode
Learn speed	↑:a_learning_rate, c_learning_rate, batch_size
Robustness	↑: Explore Space(All), policy_noise, g_initial_noise, noise_clip
Control accuracy	↓: Robustness(All)

The reward curve is shown in Fig. 8. The initial high rewards in the first 9 episodes are due to the small initial values. At this stage, the controller is still collecting data, and has not begun learning. As the episodes rises, the reward lowers sharply and fluctuates rapidly, indicating that the controller is beginning to learn. The return then gradually increases while the volatility decreases. At about the 32th episode, the reward grows sharply and the leaderless MAS is synchronized, meaning that the controller has finished optimizing.

Remark 9. By initializing the neural network parameters with suitable small values, the MASs can remain stable, collect more information for training, and converge quickly. This kind of initialization method is widely used in various neural network-related applications and does not require any system information, yet can have similar effects to admission policies [40,41].

Remark 10. The rich set of parameters makes the controller highly robust and flexible, allowing for adjustment as needed. Table 2 presents the main parameters and their impact on controller performance. ↑ represents positive correlation, ↓ represents negative correlation, Explore Space (All) represents all elements of Explore Space, and Robustness (All) represents all elements of Robustness.

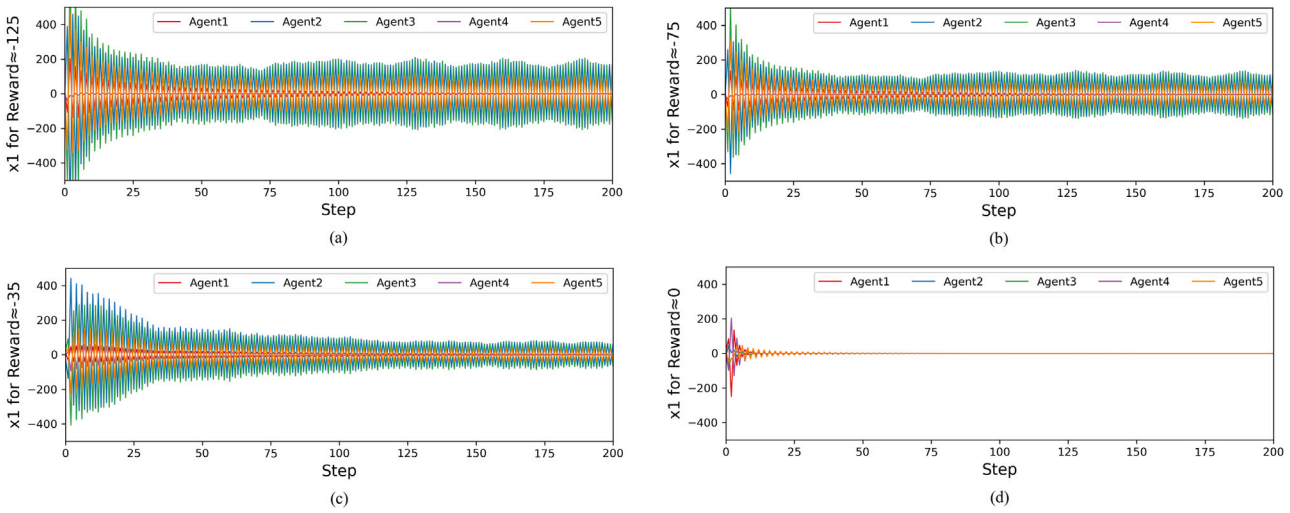


Fig. 6. The graph of agents states under 4 different returns on the leaderless MAS (Subgraphs (a), (b), (c), (d) are the states of agents when the reward r is approximately −125, −75, −35, 0, respectively.).

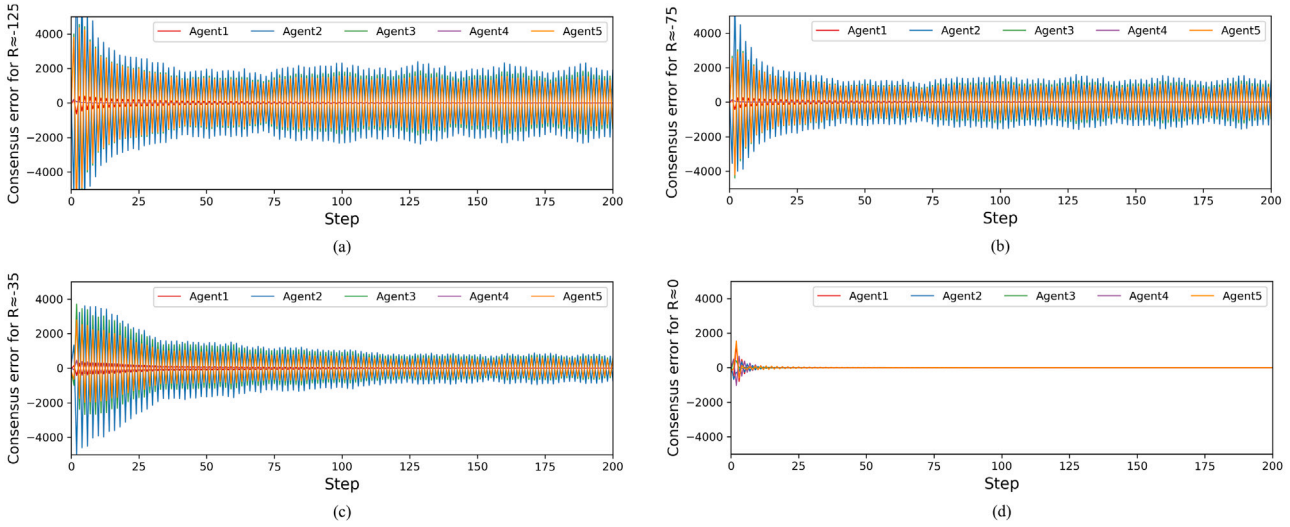


Fig. 7. The graph of consensus errors under 4 different returns on the leaderless MAS (Subgraphs (a), (b), (c), (d) are the consensus errors of agents when the reward r is approximately −125, −75, −35, 0, respectively.).

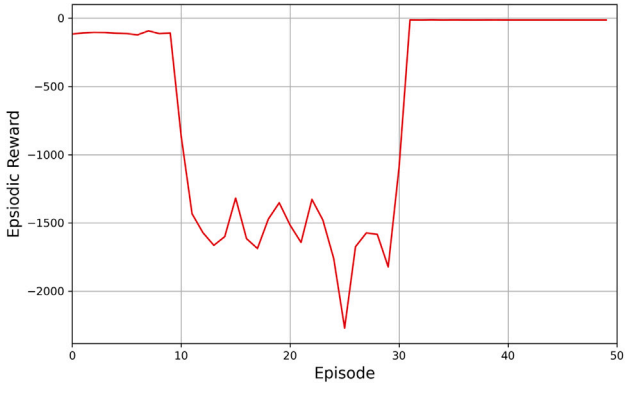


Fig. 8. The graph of the return changes with the episode steps of the leaderless MAS.

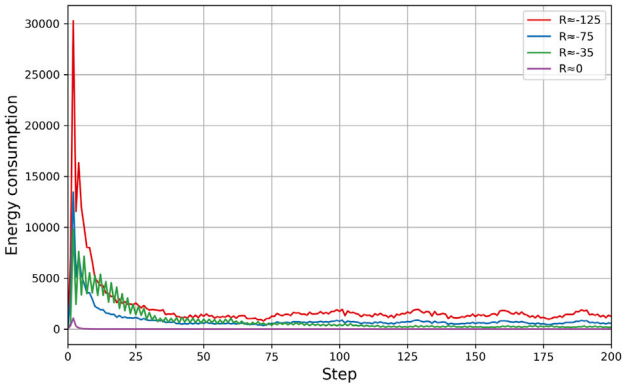


Fig. 9. The graph of energy consumption under 4 different returns on the leaderless MAS.

4 performance curves under different return conditions are chosen for analysis in order to depict the controller's learning process more intuitively. The state curve is shown in Fig. 6. On the one hand, the agents' states can be maintained within a certain maximum value under a certain return, indicating that the controller has different levels of

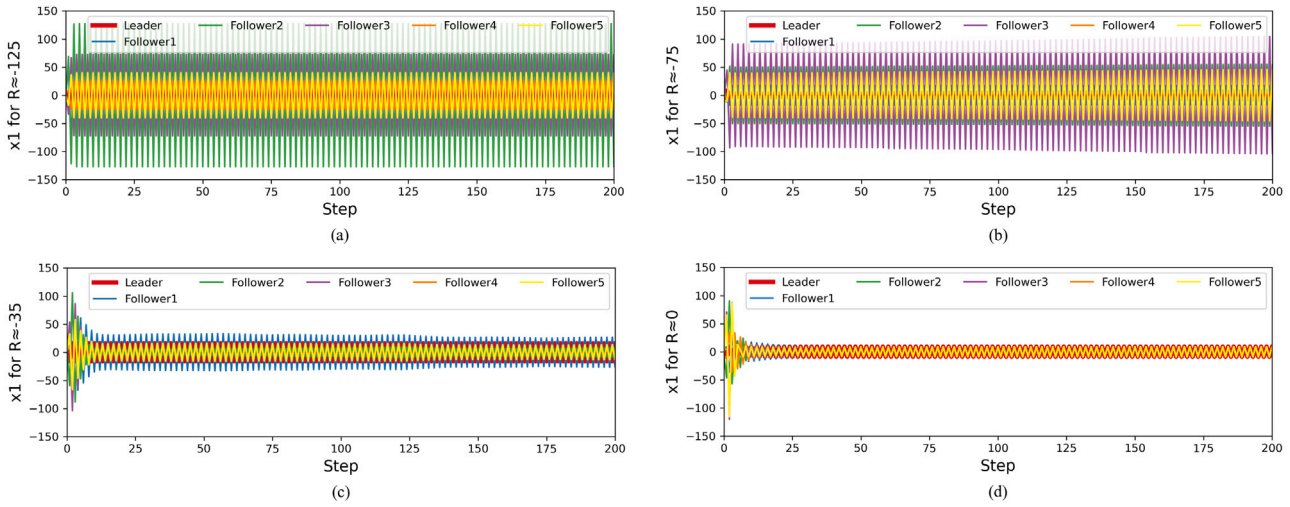


Fig. 10. The graph of states under 4 different returns on the leader-following MAS (Subgraphs (a), (b), (c), (d) are the states of agents when the reward r is approximately −125, −75, −35, 0, respectively.).

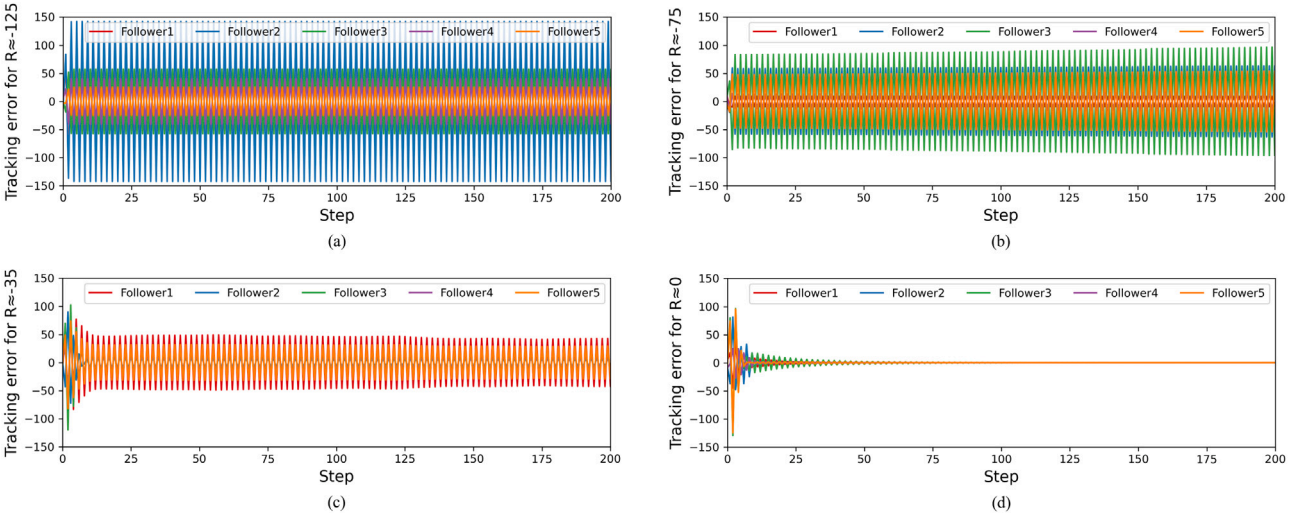


Fig. 11. The graph of tracking errors under 4 different returns on the leader-following MAS (Subgraphs (a), (b), (c), (d) are the tracking errors of agents when the reward r is approximately −125, −75, −35, 0, respectively.).

control abilities. On the other hand, as the step increases, the states of the agents gradually decrease at the 200th step and eventually reach consensus, which shows that the proposed controller can achieve synchronization control after learning.

As shown in Fig. 7, the trend of the consensus error is the same as that of the state curve. As the reward grows, the consensus error at step 200 become smaller and smaller, meaning that the learning process is to reduce the consensus error as much as possible, and finally realizing synchronization.

As shown in Fig. 9, the energy consumption decreases as the returns increase and ultimately reaches 0, indicating that optimal control has been achieved. Besides, the oscillations gradually decrease, which shows that MAS is becoming more and more stable. Furthermore, the initial energy consumption is occasionally high, which is related to the initial state of the intelligent agents. Energy consumption is higher when the gap between their states is larger. Otherwise, energy consumption will be lower.

5.2. Simulation for leader-following MASs

Considering a linear time-invariant leader-following MAS with a leader and 5 following agents, the system matrix are designed as follows:

$$A = \begin{bmatrix} -2 \pm 0.4\Psi & -1 \\ 2 \pm 0.1\Psi & 1 \pm 0.02\Psi \end{bmatrix}, \quad B = \begin{bmatrix} 1.5 \\ \Psi \end{bmatrix}. \tag{57}$$

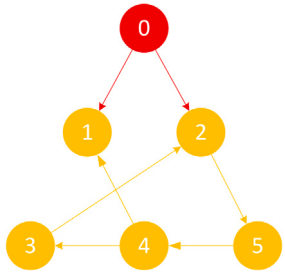


Fig. 12. The communication topology diagram of leader-following MAS (The virtual neighbors are not shown).

The communication topology is shown in Fig. 12. The Laplace Matrix is the same as that of the leaderless MAS, and the communication matrix between the leader and following agents is $\Gamma = diag\{1, 1, 0, 0, 0\}$.

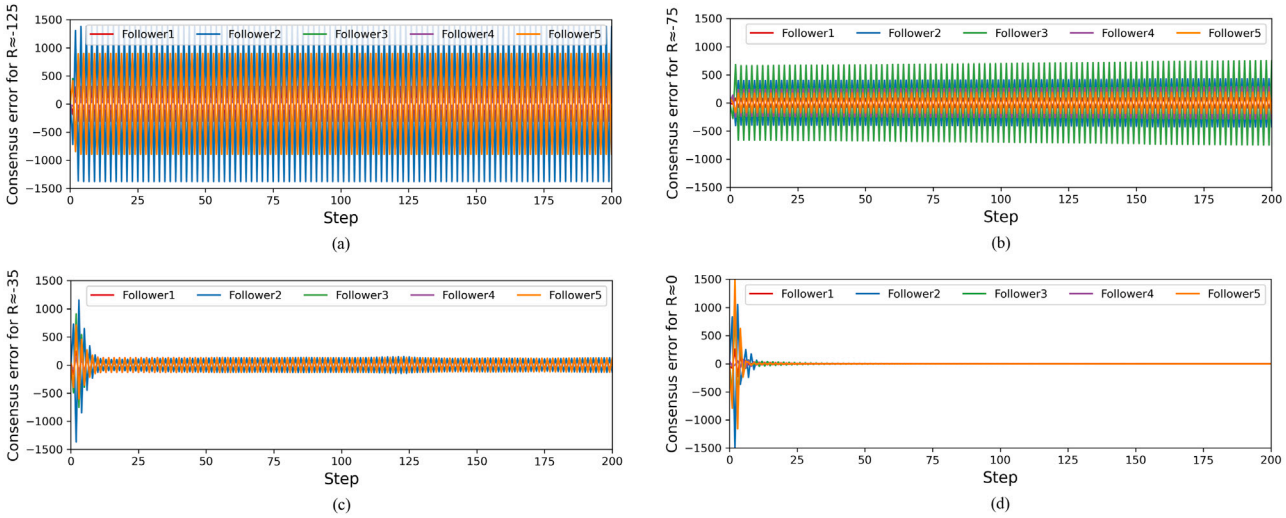


Fig. 13. The graph of consensus errors under 4 different returns on the leader-following MAS (Subgraphs (a), (b), (c), (d) are the consensus errors of agents when the reward r is approximately −125, −75, −35, 0, respectively.).

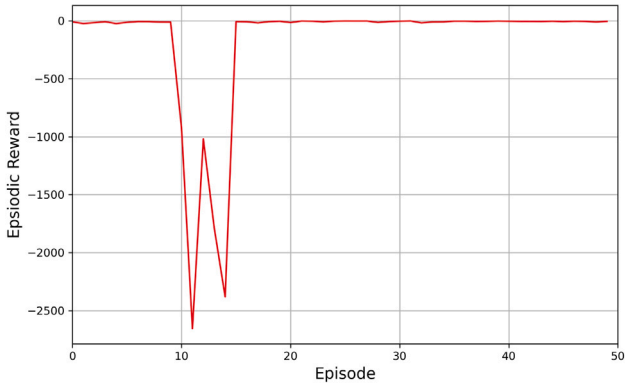


Fig. 14. The graph of the return changes with the episode steps of the leader-following MAS.

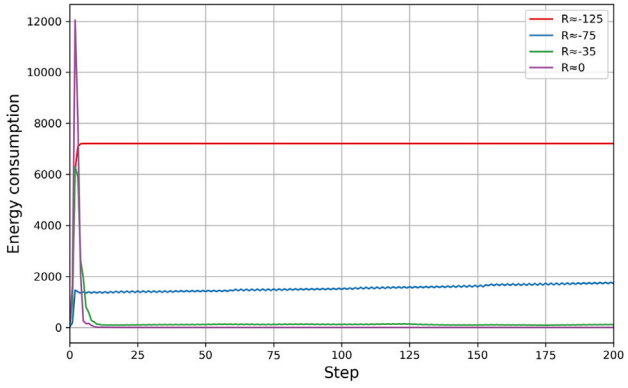


Fig. 15. The graph of energy consumption under 4 different returns on the leader-following MAS.

The simulation environment is designed in the same way as the leaderless MAS, with the same max state, max action and so on. In this paper, we only discuss the homogeneous MASs, so the inherent matrix of the leader is the same as that of the following agents. In addition, the structure of the controller for the leader-following MAS is the same as that of leaderless MAS, with most of the parameters being the same. The parameters that need to be set separately are: *a_learning_rate*: 0.0003, *c_learning_rate*: 0.0003, *policy_freq*: 2.

The reward curve is shown in Fig. 14. The leader-following MAS has gone through the stage of data reserve, learning, and stability stages, just like the leaderless MAS. The first stage has small losses due to the small consensus errors, the second stage has severe losses and significant shocks, and the third stage achieves optimal control.

Fig. 10. shows the state change of the leader and following agents. Different from the leaderless MAS, the state trajectories of the leader-following MAS are volatile, which means it is harder to synchronize. On the one hand, the volatility of the states of the following agents decreases with the increasing of the return and finally synchronized. On the other hand, the states of following agents gradually approaches that of the leader as the step increases, indicating that the controller gradually learns to achieve synchronization. When $r \approx 0$, the system will be synchronized under the policies generated by the controller.

The tracking error and consensus error are shown in Figs. 11 and 13 respectively. The volatility of above two errors are both decreased as the reward increases. The tracking error is consistently smaller than the consensus error, which can be attributed to the network topology and communication weights. What is more, the errors is always kept within a range under the same reward, and the range will decreases to 0 when $r \approx 0$, which shows that the controller is learning based on the consensus error.

The energy consumption is shown in Fig. 15. As we can see, the energy consumption gradually decreases as the returns increase, demonstrating that the controller is gradually learning to synchronize the system with the lowest energy consumption. Besides, the experiment shows that when $r \approx -125$, the energy consumption associated with the leader-following synchronization task is approximately −7000, which is significantly higher than that observed for the leaderless MAS. It indicates that the leader-following synchronization task is more challenging. In addition, the initial energy consumption of the MASs may be high due to their different initial states, which presents another challenge that needs to be addressed.

Furthermore, an additional experiment has been added in the Appendix to ensure the applicability of the proposed algorithm on practical platforms.

6. Conclusion

In this paper, the synchronization problem of two types of discrete-time linear and time-invariant MASs are considered, and a TD3 based

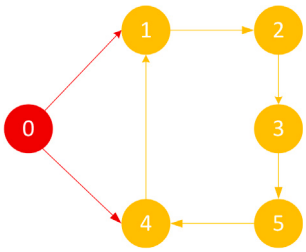


Fig. 16. The communication topology diagram of the leader-following MAS for the practical platform (The virtual neighbors are not shown).

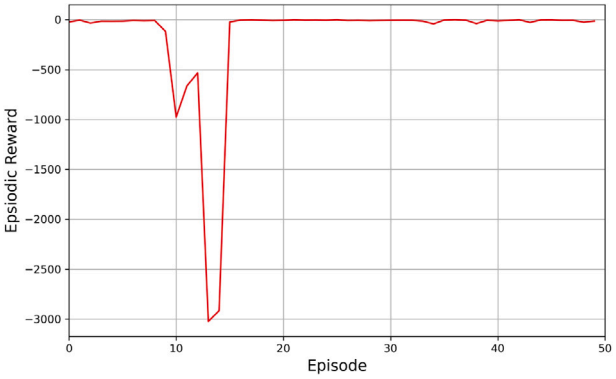


Fig. 17. The graph of the return changes with the episode steps of the leader-following MAS for the practical platform.

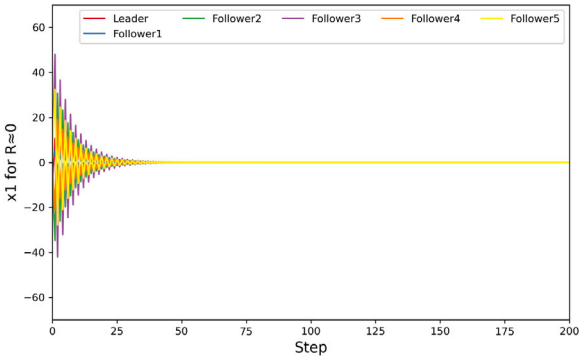


Fig. 18. The graph of state change on the leader-following MAS for the practical platform when $r \approx 0$.

method is proposed. 4 critic nets and 2 actor nets are designed to evaluate the value of the error–action pair and generate the optimal policies, respectively. After a period of training, the MASs can achieve synchronization with the lowest energy consumption. The proposed method not only avoids the time-consuming system modeling, but also overcomes the problem of dimensional explosion and removes the limitations of the same number of neighbors, giving it a wide range of applications. Besides, the real-time learning ability increases its robustness. The proofs of convergence and stability are also given to provide theoretical guarantees. Finally, some simulation cases are shown to verify the effectiveness of the algorithm. In the future, we will consider using deep reinforcement learning methods to solve problems related to multi-agent formation and investigating fault tolerance control strategies.

Table 3
The table of basic information for the experimental platform.

Platform	Configuration
Jetson Xavier NX	Computing Power: 21 TOPS CPU: Six-core NVIDIA Carmel ARM v8.2 64-bit GPU: NVIDIA Volta architecture, 384 NVIDIA CUDA cores, 48 Tensor cores Storage: 64 GB Memory: 8 GB DDR4 Usage: Drones, Unmanned Vehicles
YanHua-IPC-610L	Motherboard: SIMB-A683 CPU: 4th Generation Intel Core i5 Chipsets: Intel H81 Memory: 16 GB DDR3*2 Storage: 512 GB Usage: Robot arms, assembly lines, robots

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the Tianjin Natural Science Foundation of China (Grant No. 20JCYBJC01060), the National Natural Science Foundation of China (Grant No. 62103203) and the General Terminal IC Interdisciplinary Science Center of Nankai University, China.

Appendix

In order to fully demonstrate the practical applicability of the proposed algorithm, some experiments are conducted on various physical platforms. The platform information is shown in Table 3. Experiments have shown that the proposed algorithm can perfectly run on both of them, and the experimental results on Platform 1 will be shown below:

Considering a linear time-invariant leader-following MAS with a leader and 5 following agents, the system matrices are designed as follows:

$$A = \begin{bmatrix} 0.985 \pm 0.01\Psi & -0.09887 \pm 0.002\Psi \\ 0.9887 \pm 0.001\Psi & 0.985 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.8 \pm 0.15\Psi \\ \Psi \end{bmatrix},$$

where Ψ is an environmental factor. The communication matrix between the leader and following agents is $\Gamma = diag\{1, 0, 0, 1, 0\}$, and the Laplace matrix is:

$$L = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

The system topology is shown in Fig. 16. The controller parameters are set to be the same as those in Section 5.2 for the leader-following controller, except that the buffer size is set to 10 000.

As shown in Fig. 17, through going through 3 stages: data collection, fast learning, and stable control on the practical platform, the system reward reaches its maximum value. The state changes are shown in Fig. 18. When $r \approx 0$, the agents achieve synchronization. In addition, the consensus error and tracking error also reach their minimum values,

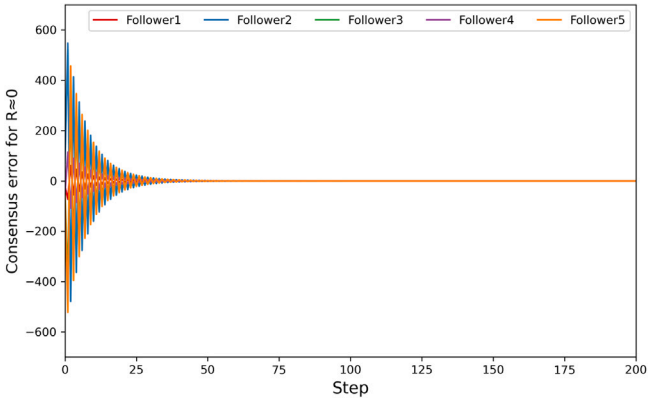


Fig. 19. The graph of consensus errors on the leader-following MAS for the practical platform when $r \approx 0$.

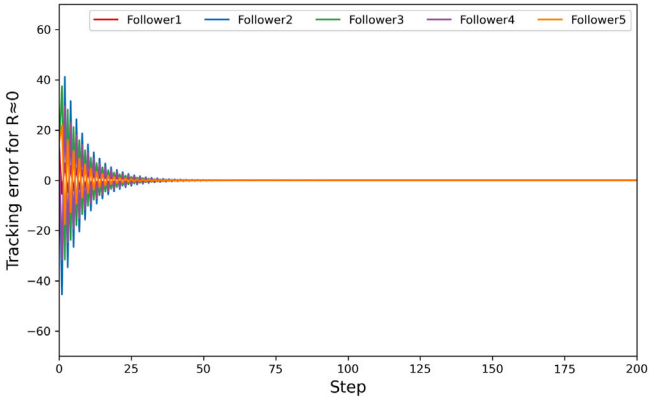


Fig. 20. The graph of tracking errors on the leader-following MAS for the practical platform when $r \approx 0$.

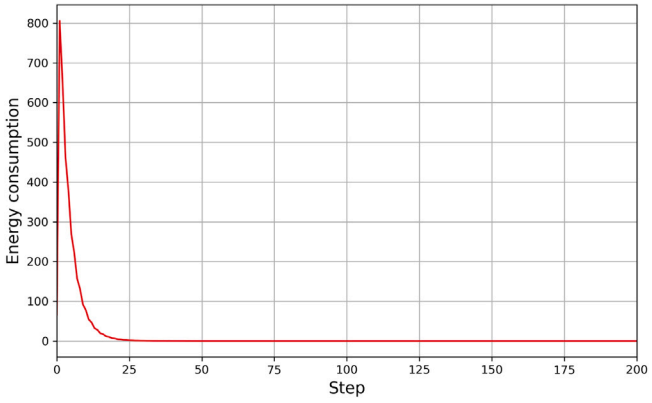


Fig. 21. The graph of energy consumption on the leader-following MAS for the practical platform when $r \approx 0$.

which are shown in Figs. 19 and 20 respectively. The energy consumption shown in Fig. 21 also reaches its minimum value at the same time. Therefore, the proposed algorithm can achieve optimal synchronization control on different practical platforms.

References

[1] J. Cui, Y. Liu, A. Nallanathan, Multi-agent reinforcement learning-based resource allocation for UAV networks, IEEE Trans. Wireless Commun. 19 (2) (2019) 729–743.

[2] Y. Gao, D. Li, Unmanned aerial vehicle swarm distributed cooperation method based on situation awareness consensus and its information processing mechanism, Knowl.-Based Syst. 188 (2020) 105034.

[3] M. Pipattanasomporn, H. Feroze, S. Rahman, Multi-agent systems in a distributed smart grid: Design and implementation, in: 2009 IEEE/PES Power Systems Conference and Exposition, IEEE, 2009, pp. 1–8.

[4] J. Sun, H. Zhang, Y. Yan, S. Xu, X. Fan, Optimal regulation strategy for nonzero-sum games of the immune system using adaptive dynamic programming, IEEE Trans. Cybern. (2021).

[5] S. Satunin, E. Babkin, A multi-agent approach to intelligent transportation systems modeling with combinatorial auctions, Expert Syst. Appl. 41 (15) (2014) 6622–6633.

[6] J. Ota, Multi-agent robot systems as distributed autonomous systems, Adv. Eng. Inform. 20 (1) (2006) 59–70.

[7] J. Hu, H. Zhang, L. Liu, X. Zhu, C. Zhao, Q. Pan, Convergent multiagent formation control with collision avoidance, IEEE Trans. Robot. 36 (6) (2020) 1805–1818.

[8] V.P. Tran, M. Garratt, I.R. Petersen, Switching time-invariant formation control of a collaborative multi-agent system using negative imaginary systems theory, Control Eng. Pract. 95 (2020) 104245.

[9] R. Sakthivel, R. Sakthivel, B. Kaviarasan, H. Lee, Y. Lim, Finite-time leaderless consensus of uncertain multi-agent systems against time-varying actuator faults, Neurocomputing 325 (2019) 159–171.

[10] J. Bai, G. Wen, A. Rahmani, Leaderless consensus for the fractional-order nonlinear multi-agent systems under directed interaction topology, Internat. J. Systems Sci. 49 (5) (2018) 954–963.

[11] C. Ding, X. Dong, C. Shi, Y. Chen, Z. Liu, Leaderless output consensus of multi-agent systems with distinct relative degrees under switching directed topologies, IET Control Theory Appl. 13 (3) (2019) 313–320.

[12] L. Mo, X. Yuan, Y. Yu, Neuro-adaptive leaderless consensus of fractional-order multi-agent systems, Neurocomputing 339 (2019) 17–25.

[13] T. Han, Z.-H. Guan, M. Chi, B. Hu, T. Li, X.-H. Zhang, Multi-formation control of nonlinear leader-following multi-agent systems, ISA Trans. 69 (2017) 140–147.

[14] F.-Y. Wang, Y.-H. Ni, Z.-X. Liu, Z.-Q. Chen, Containment control for general second-order multiagent systems with switched dynamics, IEEE Trans. Cybern. 50 (2) (2018) 550–560.

[15] Q. Wei, X. Wang, X. Zhong, N. Wu, Consensus control of leader-following multi-agent systems in directed topology with heterogeneous disturbances, IEEE/CAA J. Autom. Sin. 8 (2) (2021) 423–431.

[16] F. Wang, H. Yang, Z. Liu, Z. Chen, Containment control of leader-following multi-agent systems with jointly-connected topologies and time-varying delays, Neurocomputing 260 (2017) 341–348.

[17] Q. Wang, Z. Duan, J. Wang, Distributed optimal consensus control algorithm for continuous-time multi-agent systems, IEEE Trans. Circuits Syst. II 67 (1) (2019) 102–106.

[18] Y. Jiang, J. Fan, W. Gao, T. Chai, F.L. Lewis, Cooperative adaptive optimal output regulation of nonlinear discrete-time multi-agent systems, Automatica 121 (2020) 109149.

[19] D.R. Robinson, R.T. Mar, K. Estabridis, G. Hewer, An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments, IEEE Robot. Autom. Lett. 3 (2) (2018) 1215–1222.

[20] T. Doan, S. Maguluri, J. Romberg, Finite-time analysis of distributed TD (0) with linear function approximation on multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 1626–1635.

[21] G. Chen, Z. Li, A fixed-time convergent algorithm for distributed convex optimization in multi-agent systems, Automatica 95 (2018) 539–543.

[22] D. Wang, Z. Wang, M. Chen, W. Wang, Distributed optimization for multi-agent systems with constraints set and communication time-delay over a directed graph, Inform. Sci. 438 (2018) 1–14.

[23] S. Chen, D.W. Ho, L. Li, M. Liu, Fault-tolerant consensus of multi-agent system with distributed adaptive protocol, IEEE Trans. Cybern. 45 (10) (2014) 2142–2155.

[24] S.K. Rakesh, M. Shrivastava, Performance analysis of fault tolerance algorithm for pattern formation of swarm agents, Knowl.-Based Syst. (2021) 108020.

[25] J. Sun, H. Zhang, S. Xu, Y. Liu, Full information control for switched neural networks subject to fault and disturbance, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[26] H. Wang, X. Wang, X. Zhang, Q. Yu, X. Hu, Effective service composition using multi-agent reinforcement learning, Knowl.-Based Syst. 92 (2016) 151–168.

[27] P.J. Werbos, W. Miller, R. Sutton, A menu of designs for reinforcement learning over time, in: Neural Networks for Control, volume 3, MIT press Cambridge, MA, 1990, pp. 67–95.

[28] H. Modares, F.L. Lewis, Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning, IEEE Trans. Autom. Control 59 (11) (2014) 3051–3056.

[29] H. Modares, F.L. Lewis, Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning, Automatica 50 (7) (2014) 1780–1792.

[30] J. Gadewadikar, F.L. Lewis, M. Abu-Khalaf, Necessary and sufficient conditions for H-infinity static output-feedback control, J. Guid. Control Dyn. 29 (4) (2006) 915–920.

[31] F.L. Lewis, K.G. Vamvoudakis, Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data, IEEE Trans. Syst. Man Cybern. B 41 (1) (2010) 14–25.

[32] M.I. Abouheaf, F.L. Lewis, K.G. Vamvoudakis, S. Haesaert, R. Babuska, Multi-agent discrete-time graphical games and reinforcement learning solutions, Automatica 50 (12) (2014) 3038–3053.
[33] C. Mu, Q. Zhao, Z. Gao, C. Sun, Q-learning solution for optimal consensus control of discrete-time multiagent systems using reinforcement learning, J. Franklin Inst. B 356 (13) (2019) 6946–6967.
[34] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.
[35] K. Hengster-Movric, K. You, F.L. Lewis, L. Xie, Synchronization of discrete-time multi-agent systems on graphs using riccati design, Automatica 49 (2) (2013) 414–423.
[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.
[37] H. Zhang, H. Jiang, Y. Luo, G. Xiao, Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method, IEEE Trans. Ind. Electron. 64 (5) (2016) 4091–4100.
[38] M.I. Abouheaf, F.L. Lewis, M.S. Mahmoud, D.G. Mikulski, Discrete-time dynamic graphical games: model-free reinforcement learning solution, Control Theory Technol. 13 (1) (2015) 55–69.
[39] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[40] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[41] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

Zhongxin Liu received the B.S. degree in automation and Ph.D. degree in control theory and control engineering from the Nankai University, Tianjin, China, in 1997 and 2002, respectively. He has been at Nankai University, where he is currently a Professor at the College of Artificial Intelligence. His current research interests include predictive control, complex networks and multi-agent systems.

Ye Li received the B.S. degree in the College of Artificial Intelligence from Nanjing Agricultural University in 2016. He is currently pursuing the M.S. degree with the College of Artificial Intelligence, Nankai University. His current research interests include multi-agent system, deep reinforcement learning.

Ge Lan received her B.S. and M.S. degrees in the College of Software from Nankai University, in 2017 and 2019, respectively. She is currently pursuing her Ph.D. degree with the College of Software, Nankai University, Tianjin, China. Her current research interests include graph neural networks, text classification and knowledge graph.

Zengqiang Chen received his B.S. degree in mathematics, M. S. degree and Ph.D. degree in control theory from Nankai University, China, in 1987, 1990 and 1997, respectively. He is now a professor at the College of Artificial Intelligence, Nankai University. His current research interests include complex networks, adaptive control, intelligent predictive control and chaos system.