Neurocomputing 411 (2020) 206–215

Contents lists available at ScienceDirect

Neurocomputing

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

A TD3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment

Fengjiao Zhang [b], Jie Li [a], Zhi Li [b,*]

[a] The Center of Data Science and Service, Computer science Institute, Beijing university of Posts and Telecommunications, No. 10 Xitucheng Road, Haitian District, Beijing, China
[b] Sichuan University, No. 24 South Section 1, Yihuan Road, Chengdu, Sichuan, China

ARTICLE INFO

Article history:
Received 20 November 2019
Revised 25 April 2020
Accepted 31 May 2020
Available online 12 June 2020
Communicated by Peter. W Vamplew

Keywords:
Reinforcement learning
Overestimation error
Dual-critic
MADDPG
MATD3

ABSTRACT

We explored the problem about function approximation error and complex mission adaptability in multi-agent deep reinforcement learning. This paper proposes a new multi-agent deep reinforcement learning algorithm framework named multi-agent time delayed deep deterministic policy gradient. Our work reduces the overestimation error of neural network approximation and variance of estimation result using dual-centered critic, group target network smoothing and delayed policy updating. According to experiment results, it improves the ability to adapt complex missions eventually. Then, we discuss that there is an inevitable overestimation issue about existing multi-agent algorithms about approximating real action-value equations with neural network. We also explain the approximate error of equations in the multi-agent deep deterministic policy gradient algorithm mathematically and experimentally. Finally, the application of our algorithm in the mixed cooperative competition experimental environment further demonstrates the effectiveness and generalization of our algorithm, especially improving the group's ability of adapting complex missions and completing more difficult missions.

1. Introduction

Reinforcement learning (RL) is able to solve sequential decision problems in Markov Decision Processes (MDPs), and makes a tremendous progress recently such as AlphaGo [1], AlphaStar, OpenAI Five on Dota 2. With the development of deep reinforcement learning, single agent algorithm is gradually on the right track and has already addressed plenty of difficult problems giving researchers more powerful instruments to study in high dimension and more complex action spaces. As study more on single agent reinforcement learning, researchers realize that the introduction of multi-agent could achieve higher performance on complex mission environments. Jaderberg et al. [2] shows the advantages of multi-agent cooperation in the first-person game Quake III. The successful rate of agents after training surpasses human professional players as their teammates or opponents.

The development of multi-agent deep reinforcement learning is closely related with many basic theoretical problems. For example, there will be an overestimation bias while using a neural network to approximate the true state-action equation. Wang et al. [3] tried to make the value estimate more accurate by improving the neural network structure. However, some factors cannot be eliminated even if they are reduced due to the application of neural networks. The state-of-the-art model-free reinforcement learning algorithms, whether single-agent or multi-agent reinforcement learning algorithms, are based on Time Division (TD) or Monte Carlo (MC) [4]. These methods have an inevitable overestimation problem and perform badly in large state action space. The utility of policy gradient methods will bring in high variance of the estimation results, making policies more sensitive, even leading to a failed training. We will introduce the same overestimation problems faced by multi-agents in Section 4.1 and 4.2 below.

Inspired by the reduction of overestimation error in real state-action equations in the field of single-agent reinforcement learning [5], we present a new algorithm in multi-agents, which is used to reduce the bias of the function estimation by using dual-centered Q-network. The method using batch update and delayed policy update is adopted to reduce the high variance of estimation results and stabilize the training process. The convergence speed of results is faster and the effect is also improved.

Another important decisive factor in multi-agent reinforcement learning is the stability of state spaces. Q-learning [6] is a widely utilized method in discrete space using MDPs to analyse multi-agent reinforcement learning problems. Another popular framework applying well in the field of single-agent is actor-critic [4],

* Corresponding author.
E-mail address: lizhi@scu.edu.cn (Z. Li).

https://doi.org/10.1016/j.neucom.2020.05.097
0925-2312/

where the critic is also based on Q-learning and deep Q-network (DQN) [7]. So once other agents change their policy, that is

$$P(s'|s,a,\pi_1,\ldots,\pi_N) \neq P(s'|s,a,\pi'_1,\ldots,\pi'_N)$$

when any $\pi_i \neq \pi'_i$, where π_i is the policy of agent i, the environment does not meet the assumptions about environmental stability in Q-learning. Our team draws on the centralized critic idea in the multi-agent deep deterministic policy gradient (MADDPG) algorithm [8] allowing individuals in the group to learn the centralized critic architecture proposed separately without interfering with each other during training process. Due to different reward shapes of different individuals, it is possible to allow groups not only to execute monotonous tasks, but also execute cooperative and competitive missions at the same time without designing the reward architecture separately for opposite agents. In summary, we proposed our own multi-agent reinforcement learning algorithm multi-agent time delayed deep deterministic policy gradient (MATD3) for complex mixture environments.

This paper proves that the problem of overestimation and high variance also occurred in MADDPG at continuous motion control scenarios, and under the following assumptions and requirements:

(1) Individual agents only use local information observed during execution;
(2) No need to set differential dynamic environment modeling among intelligent community, especially the communication methods;
(3) Not only suitable for cooperation interactive environment, but also for competitive-cooperative hybrid environments.

We propose an approach solving the critic overestimation and reducing the variance of evaluation results in group reinforcement learning through double centralized critics and delayed policy update. Then, for the target policy smoothing, the correlative actions of individual agents in the group should have similar action values. Double centralized critics absorb some additional information during training in order to stabilize the state space, such as the policies of other agents. This condition can also be mitigated, as done in Lowe et al. [8]. Decentralized implementation of double centralized critics allows us to find a way to reduce the overestimation bias and variance, which can be applied to non-standard stable environments and used to stabilize the training process.

2. Related work

The common failure mode of deep deterministic policy gradient (DDPG) [9] is that the learning of Q-function starts to overestimate the Q-value and eventually breaks down the policy because it takes advantage of errors in the Q-function. TD3 [5] is an algorithm that solves this problem by introducing three key techniques that will be introduced in Section 3.

Estimation error in reinforcement learning algorithm and its effects have been studied in Mannor et al. [10]. We focus on the overestimation of function approximation and high variance of the results of policy network. In the process of double Q-learning to eliminate approximated errors, two separate estimators are used in the single agent domain [11] to make an unbiased estimation. Dueling DQN [3] improves the network structure based on double Q-learning. Another approach is to focus directly on how to reduce the estimated variance in the policy gradient method [12] and minimize overfitting for early high variance [13]. Recently, some researchers also pay attention to risk-aversion [10] and exploration [14].

In order to address the high variance caused by cumulative errors in the TD-like methods, a large part of solutions try to minimize the error range of each step or combine off-policy thoughts with MC methods. A new concept of action value is proposed by Nachum et al. [15], which is defined by a Gaussian smoothed version of the expected Q-value, and takes into account the ability to learn the covariance and mean during training. Zheng et al. [16] reduces the error by improving the network structure, but only applies to a single agent environment. The high variance caused by the cumulative error in the monomer is more serious in this situation, so this paper presents a new type of target network for multiple agents to reduce the pre-update error and provide a technique for reducing group variances by regularization and averaging value estimating.

Traditional reinforcement learning methods such as Q-learning and policy gradient are almost powerless for multi-agent environments. One of the reasons is that each agent's policy changes as the training progresses. Thus, the environment becomes very unstable for any independent agent, which poses a challenge for the stability of learning and results in difficulties in using experience replay buffer directly. But a stable past experience replaying is a very significant condition for stabilizing DQN training. There is also an option that a single agent can be optimized using a model-based policy gradient method, which have the ability to learn the optimization strategy through the backpropagation (BP) algorithm. But this requires each agent to have a discernible model that can be distinguished from the dynamic changes and assumptions of the world.

Some recent multi-agent reinforcement learning algorithms such as MADDPG and QMIX [17] do not have made more considerations for joint overestimation in multi-agent environments. Our work builds directly on the idea of MADDPG and the relationship between them is discussed in Section 4. The MATD3 algorithm proposed in this paper can reduce the overestimation error of critic networks and also be extended to any other multi-agent reinforcement learning algorithm. Chen et al. [18] introduces an application of reinforcement learning in adapted optimal control. Its algorithm combines off-policy and experience replay to make the agent study in linear time system. However it does not refer how to deal with group experience replay and off-policy methods that are vital problems in multi-agent RL. Abed-alguni et al. [19] and He et al. [20] aim to reduce the errors of overestimate of Q-value by using more estimators. The latter chooses to use two famous estimators, maximum estimator (ME) and double estimator (DE) alternately. However, both of them do not consider about multi-agent overestimation problem and only useful for single agent in discrete training space.

Both Abed-alguni et al. [21] and Hengst et al. [22] explore the hierarchy structure that decomposes complex tasks in reinforcement learning, which improves the efficiency of agents study. The former considers about multi-agent situation but only suitable for cooperative environments and do not talk about the reward structure in competitive environments. Both of them do not deal with overestimation problems that definitely occurs during the estimation of Q-value as [5] referred. The latter paper and Kulkarni et al. [23] only consider about single-agent problem, but it performs well in environments with sparse rewards.

3. Background

Multi-agent reinforcement learning is to maximize the cumulative reward of community agents. Our environment can be modeled as a community Markov game that includes N agents. MDPs have a typical form of sequential decision-making. In MDPs, actions not only affect the immediate rewards obtained by the current agent, but also affect the subsequent environmental conditions. State $s \in S$defines the true state of the environment. The

observation set of individual agent $i = \{1, 2, \ldots, N\}$ is $O_i \in O = \{O_1, O_2, \ldots, O_N\}$, the action space of agent i is $A_i \in A = \{A_1, A_2, \ldots, A_N\}$. The initial state defined by distribution $\rho : S \to [0, 1]$. Given a state $s_t \in S$ at every discrete time step t, agent $i \in \{1, 2, \ldots, N\}$ receives a local observation $o_i : s_t \to O_i$, then agent i will interactive with environment following random policy $\pi_{\phi_i} : A_i \times O_i \mapsto [0, 1]$ abbreviated as $\pi_i : A_i \times O_i$, take an action $a_t \in A$ receive immediate reward R_{t+1}. Afterwards, the environment will transfer to next state $s_{t+1} \in S$ according to state transition function $\Gamma : S \times A_1 \times \ldots \times A_N \to S$, meanwhile original environment state will transfer from the old state $s_t \in S$to the new state $s_{t+1} \in S$. Individual rewards are related with the state and action of agent $i$as $r_i : S \times A_i \to \mathbf{R}$. The goal of every agent in the community is to maximize the expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where γ is a discount factor that limits the length of time and T is the end time.

Q-learning is a popular reinforcement learning method based on TD approach, and there are also lots of examples to apply it to the multi-agent domain [24]. Q-learning evaluates the value of the state-action pair called Q-value $Q^{\pi}(s, a) = E_{r,s'}[R|s_t = s, a_t = a]$, rewritten as the form of the Bellman equation $Q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma Q(s', a')]$. The formula for updating the Q-value is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

Q-value introduces an overestimation error while calculating the max operation. Hasselt [11] and Van Hasselt et al. [25] use two Q-value networks and reduce the estimation error with the smaller one. However, as the state space dimension becomes higher and higher, Q-learning's tabular method is too difficult to search for the calculation of action-value functions. Therefore, Mnih et al. [4] proposed the DQN method based on Q-learning theory, where one trick for stable training is to use a dual network structure that one is the Q-network and the other one is the target Q-network q_{tar} with delayed update outputting $\overline{q}$. The Q-function network learns an action-value function $Q*$ by minimizing the loss: $L(\theta) = E\left[\left(r + \gamma \max_{a'} \overline{q}^*(s', a'|\theta^-) - q^*(s, a|\theta)\right)^2\right]$, where $y = r + \gamma \max_{a'} \overline{q}^*(s', a'|\theta^-)$ is called TD target. θ, θ^- are the neural network parameter and target network parameter respectively. Another way of DQN stability training is to utilize the experience replay buffer in which experience tuple $\tau = (s, a, s', r)$ gained in the interaction with the environment that is stored breaking data association and improving its utilization. In a multi-agent environment, a simple idea is to let the agents in the community use their own Q-function network and training separately [26–28], but this makes the environment very unstable and affect the convergence conditions of Q-learning seriously. And experience replay method is no longer applicable, since $P(s'|s, a, \pi_1, \ldots, \pi_N) \neq P(s'|s, a, \pi'_1, \ldots, \pi'_N)$ when any $\pi_i \neq \pi'_i$.

In the continuous action space, there is another more common method better than Q-learning, called policy gradient (PG) [29]. The PG algorithm does not parameterize the Q-value like Q-learning, but directly parameterizes the policy, which is $\pi_{\phi}(s)$, and updates the parameters by the gradient descent method $\phi \leftarrow \phi + \alpha \nabla_{\phi} J(\phi)$, where α is the learning rate, $\nabla_{\phi} J(\pi_{\phi}) = E\left[\nabla_{\phi} \log \pi(a|s) q^{\pi}(s, a)\right]$ is the policy gradient of the stochastic policy. Finding the optimal parameters through the policy gradient maximizes the expectation of the cumulated return $E\left[\sum_{t=0}^{T} R_{t+1} | \pi_{\theta}\right]$. The random policy outputs a probability distribution $\pi_{\theta}(a|s) = P[a|s; \theta]$. When a random strategy is adopted, different actions may be taken even in the same state. The deterministic policy gradient (DPG) algorithm [30] extends on this basis. When the agent is in the same state, the action is uniquely determined that is the deterministic policy $\mu_{\theta}(s) = a$. DDPG [9] algorithm introduces neural networks based on DPG using neural networks to approximate policy μ_{ϕ} and critic $Q_{\theta}(s, a)$ through the gradient accent and off-policy which can extract experience trajectories from the replay buffer.

Although DDPG can reach good performance sometimes, it is very sensitive for hyper-parameters and other types of adjustments. The common failure mode of DDPG is the overestimation of Q-value when starting to learn the Q-function. Twin Delayed DDPG (TD3) [5] is an algorithm that solves this problem by introducing three key techniques. The first one is clipped dual Q-networks. TD3 learns two Q-functions and uses the smaller of the two Q-values to form the target of the Bellman error $y(r, s'd) = r + \gamma(1 - d) \min Q_{\theta_i, tar}\ (s', \mu'(s)), i = 1, 2$. The second is delayed policy update. The policy network parameters are updated after dual Q-function networks are updated. Thirdly, smoothed target policy utilization makes all action values within the specified range $\mu'(s') = clip\left(\mu_{\theta_{tar}}(s') + clip(\xi, -c, c), a_L, a_H\right),\ \xi \tilde{N}(0, 1)$.

The MADDPG algorithm extends the DDPG algorithm to the multi-agent domain, and the agent i takes the centralized Q-value as $Q_i^{\boldsymbol{\mu}}(s, a)$, where $\boldsymbol{\mu} = \mu_1, \mu_2, \ldots, \mu_N$ is $\boldsymbol{\kappa} = \kappa_1, \kappa_2, \ldots, \kappa_N$ parameterized group deterministic policy set. s is the state of the current environment. $\boldsymbol{a} = a_1, a_2, \ldots a_N$ is the set of actions of N agents, and $\boldsymbol{x} = (o_1, o_2, \ldots, o_N)$ is the set of observations of all agents. Another reason for stable training in MADDPG is that since we know the actions taken by all agents, $P(s'|s, u, \mu_1, \ldots, \mu_N) = P(s'|s, u, \mu'_1, \ldots, \mu'_N)$ will be sufficient even$\mu_i \neq \mu'_i$. A stable training environment makes it possible to use the experience replay. Taking centralized critic discretized execution with an off-policy way, the policy gradient of agent is $\nabla_{\phi_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{s,a\tilde{B}}\left[\nabla_{\phi_i} \mu_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(s, a)|a_i = \mu_i(o_i)\right]$, where B is the replay buffer storing experience tuple $(\mathrm{x}, \mathrm{a}, r, \mathrm{x}')$.

4. Method

4.1. Double critic

We set up two independent centralized critic networks with corresponding target networks and one policy network with its target network for individual agents in the group. Its purpose is reducing overestimation by selecting a smaller Q-value when the neural network approximates the real state-action function.

Fig. 1 shows a double centralized critic network. Agent i obtains local observations and actions by interacting with the environment. For double critic networks of agent i, the input layer absorbs not only local observations o_i and action a_i, but also local observations and action information sets of other agents. The network out-

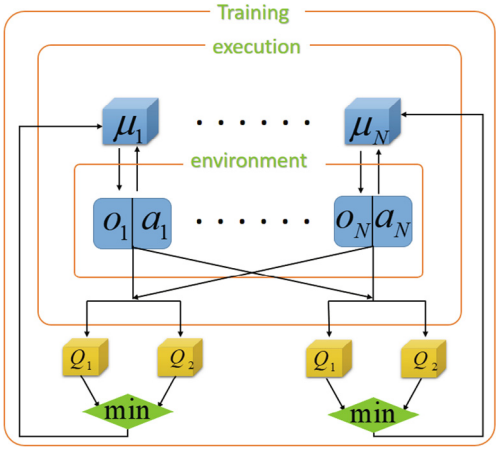

Fig. 1. Double centralized critic network.

puts an approximation of the true state-action equation, and selects a smaller Q-value through the minimization operator to the input layer of the policy network as a guide for its policy gradient.

4.2. Population overestimation

In Q-learning with discrete action space, the update formula for TD target is $y = r + \gamma \max_{a'} Q(s', a')$. Because each step involves a maximizing greedy operation, if the Q-value of a state action pair is overestimated, an error is introduced, which is accompanied by the Bellman update formula used by Q-learning which finally made

$$\mathbb{E}_{\omega}[\max_{a'}(Q(s', a') + \omega)] \geqslant \max_{a'} Q(s', a') \tag{2}$$

Thrun and Schwartz [31]. In some current reinforcement learning algorithms, like Fujimoto et al. [5] based on actor-critic, further proves that when the learning rate is satisfied, a single agent will have an overestimate error based on Thrun and Schwartz [31], as

$$\mathbb{E}\left[Q_{\theta}\left(s, \mu_{\phi_{true}}(o)\right)\right] \geqslant \mathbb{E}\left[Q^{\pi}\left(s, \mu_{\phi_{true}}(o)\right)\right], \quad \text{when } \alpha < \min(\omega_1, \omega_2) \tag{3}$$

where the real Q-function is $Q^{\pi}(s, a)$ and the approximate Q-function that actually estimates the real Q-function is $Q_{\theta}(s, a)$. ω_1 and ω_2 limit the learning rate to a very small value range. And it has proved that this deviation appeared in the DPG experiment. The most significant method of stabilizing training environment used in MADDPG is that the Q-network does not only absorb local observation and action of one agent when calculating Q-value but also absorbs observations and actions of other members if necessary. Thus, it doesnt change the intrinsic property of the Q-network. The overestimation in multi-agent scenarios still exist, as the input of formula (5) from local information changes to global information which means every individual's Q-network takes observations and actions from all members in the same 'cooperative' group. The experiments also proved that the overestimation error does exist in multi-agent environments as shown in Fig. 2. In the multiple agent reinforcement learning scenario, due to the environmental stability assumptions by Lowe et al. [8], the following errors exist:

$$\mathbb{E}_{\omega}\left[\max_{\mathbf{a}'} Q_{\theta,i}(\mathbf{x}', \mathbf{a}') + \omega\right] \geqslant \max_{\mathbf{a}'} Q^{\pi}(\mathbf{x}', \mathbf{a}'), \text{ where } \mathbf{x} = (o_1, o_2, \ldots, o_N) \tag{4}$$

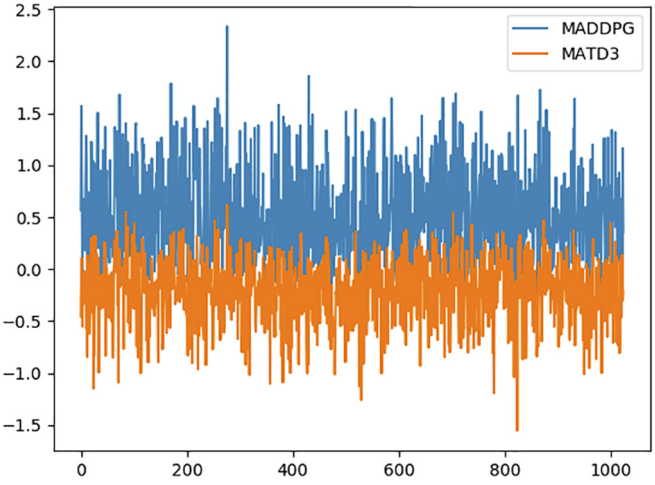

Fig. 2. The original estimated Q-value in MADDPG and the Q-value after taking reduction in MATD3.

This is an observation set for all agents. o_i is the local observation of agent i. $\mathbf{a} = \{a_1, a_2, .., a_N\}$ is the action set of all agents. a_i is the action of the agent i.

Our policy updating method in this paper is DDPG-like way. We let the real Q-function network of multiple agent reinforcement learning be annotated as $Q_{\theta,i}(x, a)$, and the actor guided by $Q_{\theta,i}(x, a)$, which $\mu_{\phi_{i,approx}}$ is abbreviated as $\mu_{i,approx}$. The approximate Q-function network is expressed as $Q_i^{\mu}(x, a)$ (unknown at training time), and the actor under its guidance work is $\mu_{\phi_{i,true}}$ abbreviated as $\mu_{i,t_{rue}}$. The group policy parameters are updated by:

$$\phi_{i,approx} = \phi + \frac{\alpha}{K_1} \mathbb{E}_{\mathbf{x}\mathbf{p}_{\mu}}\left[\nabla_{\phi} \mu_{\phi}(\mathbf{x}) \nabla_{a_i} Q_{\theta,i}(\mathbf{x}, \mathbf{a})|_{a_i = \mu_{\phi_i}(o_i)}\right] \tag{5}$$

$$\phi_{i,true} = \phi + \frac{\alpha}{K_1} \mathbb{E}_{\mathbf{x}\mathbf{p}_{\mu}}\left[\nabla_{\phi} \mu_{\phi}(\mathbf{x}) \nabla_{a_i} Q_i^{\pi}(\mathbf{x}, \mathbf{a})|_{a_i = \mu_{\phi}(o_i)}\right] \tag{6}$$

K_1 and K_2 are used to normalize the gradient. The overestimation error will still appear under some more stringent conditions without normalized gradient [5], because gradient direction is a local optimum rate of increase of the equation. Thus existing ω_1, when ω_1 enough small, if $\alpha < \omega_1$, the true Q-function value under the guidance of $\mu_{i,approx}(x_i)$ will limit the true value of the Q-function under the policy $\mu_{i,true}(o_i)$ guidance:

$$\begin{aligned} &\mathbb{E}\left[Q_{\theta,i}(o_1, \ldots, o_N, a_1, \ldots, a_N)|_{a_i = \mu_{i,approx}(o_i)}\right] \\ &\geqslant \mathbb{E}\left[Q_{\theta,i}(o_1, \ldots, o_N, a_1, \ldots, a_N)|_{a_i = \mu_{i,true}(o_i)}\right] \end{aligned} \tag{7}$$

On the contrast, there exist ω_2 while it is enough small, if $\alpha < \omega_2$, the true value of Q-function under the policy $\mu_{i,approx}(x)$ guidance will limit that under the policy $\mu_{i,true}(x)$ guidance:

$$\begin{aligned} &\mathbb{E}\left[Q_i^{\pi}\left(o_1, \ldots, o_N, a_1, \ldots, a_N|_{a_i = \mu_{i,true}(o_i)}\right)\right] \\ &\geqslant \mathbb{E}\left[Q_i^{\pi}\left(o_1, \ldots, o_N, a_1, \ldots, a_N|_{a_i = \mu_{i,approx}(o_i)}\right)\right] \end{aligned} \tag{8}$$

When both of their policies are assumed to be $\mu_{\phi_{i,true}}$ because of the formula (1) and $\alpha < \min(\omega_1, \omega_2)$, the value of the group Q-function value will be overestimated:

$$\begin{aligned} &\mathbb{E}\left[Q_{\theta,i}\left(o_1, \ldots, o_N, a_1, \ldots, a_N|_{a_i = \mu_{i,approx}(o_i)}\right)\right] \\ &\geqslant \mathbb{E}\left[Q_i^{\pi}\left(o_1, \ldots, o_N, a_1, \ldots, a_N|_{a_i = \mu_{i,approx}(o_i)}\right)\right] \end{aligned} \tag{9}$$

Although this overestimated error is relatively small at each update, the presence of this error can cause two concerns. First, if not confirmed, this overestimation bias may accumulate as a large deviation. Second, an unhealthy inaccurate value estimation may also cause the strategy to be updated in a bad direction. In the current multi-agent algorithm MADDPG, we show the difference between the estimate of the true value and the true value of the estimated value after the elimination. As shown in Fig. 2, it can be clearly seen that the deviation does exist.

4.3. Reduce the population overestimate

There have been some ways to reduce the overestimation bias that occurs when using Q-learning. In the field of multi-agents, Zheng et al. [16] explores the extension of double Q-learning to the multi-agent domain, but only suitable for cooperative situation. We improve the critic under the actor-critic setting, and make the algorithm suitable for the mixed cooperative competition environment. Besides, we also reduce the overestimation bias of the real state-action equation.

In double Q-learning, the update to the TD target is through two independent Q-networks, each of which is used to update the esti-

mate of another Q network. If the estimates of values are independent of each other, they can select actions for the policy, making an unbiased estimate of actions chose by the other 's policy.

In WMA double weighted multi-agent [16], the update of two Q-networks Q_v and Q_u randomly select a Q-network for updating, and one network is used as the target network of the other network. Both Q_u and Q_v are Q-networks in Fujimoto et al. [5] which balance two Q-networks with β as Eq. (9). If one of them overestimates issues during evaluating the Q-value of same pair of observation and action, it is able to reduce overestimation partly using this method. Because the Q-value is balanced with β, Q-target calculating will be more precise as shown in Eq. (10). The TD target update function is:

$$Q_U^w(s', a*) \leftarrow \beta Q^U(s', a*) + (1-\beta)Q^V(s', a*), a* \\ \leftarrow \arg\max_a Q^U(s', a) \tag{10}$$

$$Q_x^{target}(s', a*) \leftarrow R(s, a) + Q_U^w(s', a*) \tag{11}$$

Where U is the Q-function network 1, V is the Q-function network 2, or turn them upside down.

Due to MADDPG is under actor-critic setting, its TD target update function is:

$$y = r_i + \gamma Q_i^{\mu'}(\times', a'_1, \ldots, a'_N)|_{a'_j=\mu'_{\phi_j}(o_i)} \tag{12}$$

In the experiment, we found that if the policy is slowly changed in the actor-critic structure, the difference between the current and target value network will be very small as well as the update, and the improvement effect will not be obvious. Let critics of the agent i be $(Q_{i,\theta_1}, Q_{i,\theta_2})$, and policies under their guidance are $(\mu_{i,\phi_1}, \mu_{i,\phi_2})$ respectively. We draw on the TD update method in WMA Double DQN and MADDPG, and propose our own TD target update mode of MATD3 agent i:

$$y_i = r_i + \gamma \min_{n=1,2} Q_{i,\theta'_n}(\times', a'_1, \ldots, a'_N)|_{a'_j=\mu'_{\phi_j}(o_j)} \tag{13}$$

Another benefit of setting the independent variable of the estimated error to be variable is that we can see that the minimized operator should provide higher values for states with low variance estimation errors, because the expected smallest value of random variables set will decrease as the variance of the random variable increases. This effect means that the minimum of Eq. (9) will have some preference when estimating state value with low variance. This preference is beneficial and can lead to a more secure policy update that stabilizes learning. This method reduces the drawbacks that the algorithms only apply cooperative environment caused by using DQN and MADDPG cannot alleviate the overestimate error.

4.4. Population policies variance

If the group overestimate error does not be suppressed or reduced in time, it will cause the accumulation of the overestimate error of the group, and finally seriously affect the policy update. This means that when a certain state-value pair is evaluated, there will be a large divergence, causing a large variance in value evaluation so that seriously affects the policy to update in a beneficial direction. Sutton [29] mentions that the high variance of the estimate results will produce a noise gradient when the policy is updated, which can be reduced by decreasing the learning rate, but it is not easy to perform well in actual training. In this section, we emphasize:

(1) The description of relationship between target network and estimate error.
(2) The importance of minimizing population error in each time step of updating.
(3) Proposing how to reduce variance of estimate results based on actor-critic.

4.4.1. Accumulated population error

The policy update of our algorithm uses a time difference method similar to DDPG update. Each evaluation of state action pairs is based on the evaluation of next state, so agent i will introduce an estimation error at each time step t:

$$\delta_{i,t}(s_1, \ldots, s_N, a_1, \ldots, a_N) = \delta_{i,t}(\mathrm{s}, \mathrm{a}) \tag{14}$$

$$Q_{\theta_i}(s_1, \ldots, s_N, a_1, \ldots, a_N) = r + \gamma \mathbb{E}\left[Q_{\theta_i}(s'_1, \ldots, s'_1, a'_1, \ldots, a'_N)\right] \\ - \delta_i(s1, \ldots, sN, a1, \ldots, aN) \tag{15}$$

$$Q_{\theta_i}(\mathrm{s}, \mathrm{a}) = r + \gamma \mathbb{E}\left[Q_{\theta_i}(\mathrm{s}', \mathrm{a}')\right] - \delta_i(\mathrm{s}, \mathrm{a}) \tag{16}$$

Although we hope that this estimation error is as small as possible, it may be accumulated and finally makes the policy update to the local optimal direction. The above formula can be rewritten as:

$$\begin{aligned} Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t) = \; & r_t + \gamma \mathbb{E}\left[Q_{\theta_i}(\mathbf{s}'_{t+1}, \mathbf{a}'_{t+1})\right] - \delta_{i,t}(s, a) \\ & = r_t + \gamma \mathbb{E}\left[r_{t+1} + \gamma \mathbb{E}[Q_\theta(\mathbf{s}_{t+2}, \mathbf{a}_{t+2})] - \delta_{i,t+1}\right] - \delta_{i,t} \\ & = \mathbb{E}_{s_i \xi_\mu, a_i \mu}\left[\sum_{j=t}^{T} \gamma^{j-t}(r_j - \delta_{i,j})\right], \text{ while } \mathrm{i} = 1, \ldots, \mathrm{N} \end{aligned} \tag{17}$$

As can be seen from the function 11, when we explicitly write the estimated error into the equation for the Q-value, the variance of the Q-value is not only related to the expected variance of the future reward, but also of the estimation error. If the discount factor γ is large at each step of updating, and does not pay attention to the reduction of the estimation error, the accumulation of errors will be accumulated, so that the variance of the Q-value will increase rapidly every time an update is made. Gradient updates only reduce errors in this mini-batch, but do not help reduce the remaining errors in the entire set of value estimates.

4.4.2. Individual target network and delayed population policy update

The target network is an important tool for stable deep learning. Because the approximator of function needs multi-step gradient update to converge, the target network provides a stable target and makes possible that the training data more effectively convergent during the individual learning process in the group. Without a fixed personal target, each update may have residual errors and begin to accumulate. The accumulation of errors can be harmful to itself. when the residual error appear with policy maximization along the estimation of values, it will result in a uncontrollable divergence.

Fig. 3 shows that the failure of actor critic may be due to the interaction between critic and actor. If our individual policy is updated with the Q-value estimated by its dual centralized critic after each individual critic update, it is easy to update in a terrible direction due to the high variance and inaccurate estimation of the Q-value distribution. And because we adopt a centralized critic, bad policies will spread throughout the group. So it is better to consider updating when critic is stable.

If the policy is updated on a high error state, it will lead to decentralized behavior. And since the target network can reduce the cumulative error caused by multi-step updates, our individual target network should be updated with low frequency. Our algorithm allows the individual's target network to be updated after the critic updating. In order to wait for the critic to be slightly stable, the value error is small enough before updating the policy

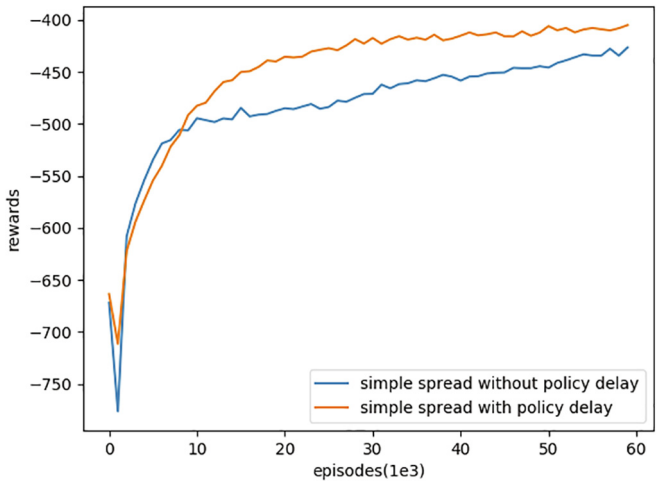

Fig. 3. Comparison of policy delay update.

and target network. We limit the possibility of repeated updates for updating the policy with appropriate delays when the critic does not change. The lower the frequency of policy updates, the higher probability of using low variance for value estimation in principle making the policy update in a better direction. The effectiveness of this method has been demonstrated in the experiment of Section 5, and the performance has been effectively improved.

4.5. Target policy smoothing

The shortcoming of MADDPG is that it may overfit to local peaks. The TD target of individual agent i is:

$$y_i = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(x', a'_1, \ldots, a'_N)|_{a'_j=\mu'_k(o_j)} \tag{18}$$

It can be seen that when calculating the TD target of the agent i, two approximate networks are adopted. One is the approximate network of the value network Q_i, and the other is the approximate network of the deterministic policy set μ', so the error of the TD target due to the approximation of the equation is very sensitive. Therefore, the variance of the TD target is also increased that can be mitigated by regularization. By referring to TD3s thoughts that a single agent should have similar Q-values under similar state-actions, our algorithm mitigates the bias and variance caused by the approximation of the equation with smoothing the target policy μ'_i of each agent i:

$$y_i = r_i + \mathbb{E}_{\omega_i}\left[Q_{i,\theta'}(x', a'_1, \ldots, a'_N) + \omega_i\right]|_{a_j=\boldsymbol{\mu}_{j,\phi'}\left(o'_j\right)} \tag{19}$$

In the actual experiment, the processed random noise ω_i is added to the target policy $\mu_{i,\phi'}$ of each agent i:

$$y_i = r_i + \gamma Q_{i,\theta'}(x', a'_1, \ldots, a'_N) + \xi_i|_{a_j=\boldsymbol{\mu}_{j,\phi'}\left(s'_j\right)}, \xi_i$$
$$\sim clip(N(0,\delta), action\_low, action\_high) \tag{20}$$

where action low and action high are the minimum and maximum of the total action space respectively. $v(0,\delta)$ is a Gaussian distribution with an expectation of 0 and a variance of δ.

5. Experiments

In this section, we show that the MATD3 algorithm has a better performance than the multi-agent environment used by the MADDPG algorithm. With the critic of truncated duel centralized Q-network to reduce the overestimation bias of a single agents critic networks in the population, the stability of the training environment is increased and the overestimation error is reduced. Besides, the performance of the algorithm is improved. In MATD3, there are two Q-networks for critic, which are used to compare the estimate values, and take a smaller estimate value as the reference Q-value used in actor network update. Critics are updated by minimizing the difference between the target network and the Q-value estimation network. The target network and actor are updated after critics updating certain times.

5.1. Comparison

We performed our algorithm testing in MADDPG's self-built multi-agent particle environment [32]. This environment includes nine scenarios, and the detailed description of the scene is in the MADDPG article, we will not repeat them. The nine scenarios are: simple, physical deception, covert communication, keep-away, simple reference, cooperative communication, cooperative navigation, predator–prey and simple world comm. Our algorithm is compared to DQN, DDPG and MADDPG in nine scenarios. Our results include tables and images shows as following. In our algorithm, each agent i has two centralized critic networks and one actor network, and those networks have their own corresponding target network respectively. The neural network structure is used to set up adversaries and good agents are the same as what uses three full-connected layers as Fig. 4. There is one active unit Relu between each layer, and every layer has 64 units. The parameters of network are updated by Adam. The learning rate is $10e-2$ and set 1024 transitions for mini-batch training which are uniformly sampled from the experience replay buffer. The two independent centralized critic target networks of each agent receive the state set $x\prime = o1, o2, \ldots, oN$ and action set $a_1, a_2, \ldots, a_N$ in the tuple uniformly is sampled from the replay buffer as the input of the first layer. We compare two outputs and take a smaller Q-value to update the target value of the two critics.

From the Fig. 4, double centralized critic networks have their own streams to estimate the Q-value of current population state-action set and output a smaller Q-value to the policy network by the minimize operator.

To achieve target policy smoothing, the action is eventually limited to the action space of corresponding environment by adding noise $\xi \in N(0, noise)$ to the action. After critic updating certain times, all target networks and actor networks are updated. We use the Q-estimated-value of critic network 1 to update the agent's unique actor network parameters. Update the actor by policy gradient: $\nabla_{\phi_i} J(\phi_i) = \mathbb{E}_{x,a\tilde{A}}\left[\nabla_{a_i} Q_i^{\mu}(x, a_1, \ldots, a_N) \nabla_{\phi_i} \boldsymbol{\mu}_{\phi_i}(o_i)|_{a_i=\boldsymbol{\mu}_{\theta_i}(o_i)}\right]$. Each task of MATD3 ran between 60,000 and 160,000 times as we marked in tables and figures, and the policy evaluation was conducted every 1000 times.

We don't make any special changes to other three algorithms' original setting when comparing with our work. It should be

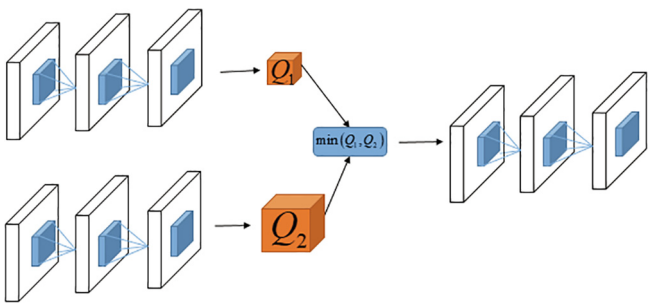

Fig. 4. The double centralized critic networks.

emphasized that the network structure of these algorithms adopts the same network structure and parameter setting. We did not make changes to achieve our goals deliberately. The detail of our algorithms is shown in the appendix.

Our learning curves are shown in Fig. 5. From this figure, we can see that the curves of DQN are strongly undulant and do not reach at a stable state in most of secarios within limited number of iterations. What's more, MATD3 has smaller oscillation range. For the purpose of more detailed evaluation, the size and figure are adjusted appropriately as Fig. 6 in order to compare MATD3 and other algorithms.

Benifited from reducing overestimation errors, smoothing the target policy and delayed policy update, our algorithm finally performs well in most of scenarios and accelerates the convergence speed of previous period.

In the cooperation scenario, the reward obtained by MATD3 in cooperative navigation is higher than the reward obtained by the original algorithm in the environment. In the convergence of the previous period of MATD3 in cooperative communication, the convergence speed is accelerated due to the reduction of the overestimation error. In the competition scenario, we uniformly set good agents to the agent using MADDPG algorithm, and set adversaries to one of DQN, DDPG, MADDPG and MATD3, then compare good ones and adversaries. In the most complex scene simple world comm, our algorithm performs much better, especially it has improved performance in the confrontation with MADDPG.

5.2. Competition experiments

In the scenario physical deception, we let adversaries' algorithm be MATD3, and good agents be MADDPG. Compare the confrontation between two sides as we showed in Table 1.

The definition of adv success once is that the center distance between the agent and the target landmark less than a certain threshold which is set to the sum of the radius of the agent and the target landmark. Take the 20,000 times after the reward is stabilized to calculate the number of successes in 60,000 times. Take the 50,000 times after the reward is stabilized to calculate the number of successes in 100,000 times. successful times is the number of MATD3 (adversaries successful times) minus MADDPG (good agents' successful times).

5.3. Complexity experiment

The number of agents and targets involved in the nine experiments are shown in Table 2. The most complicated scenario is simple world comm which involves the largest agents number and includes communication and competition at the same time. In

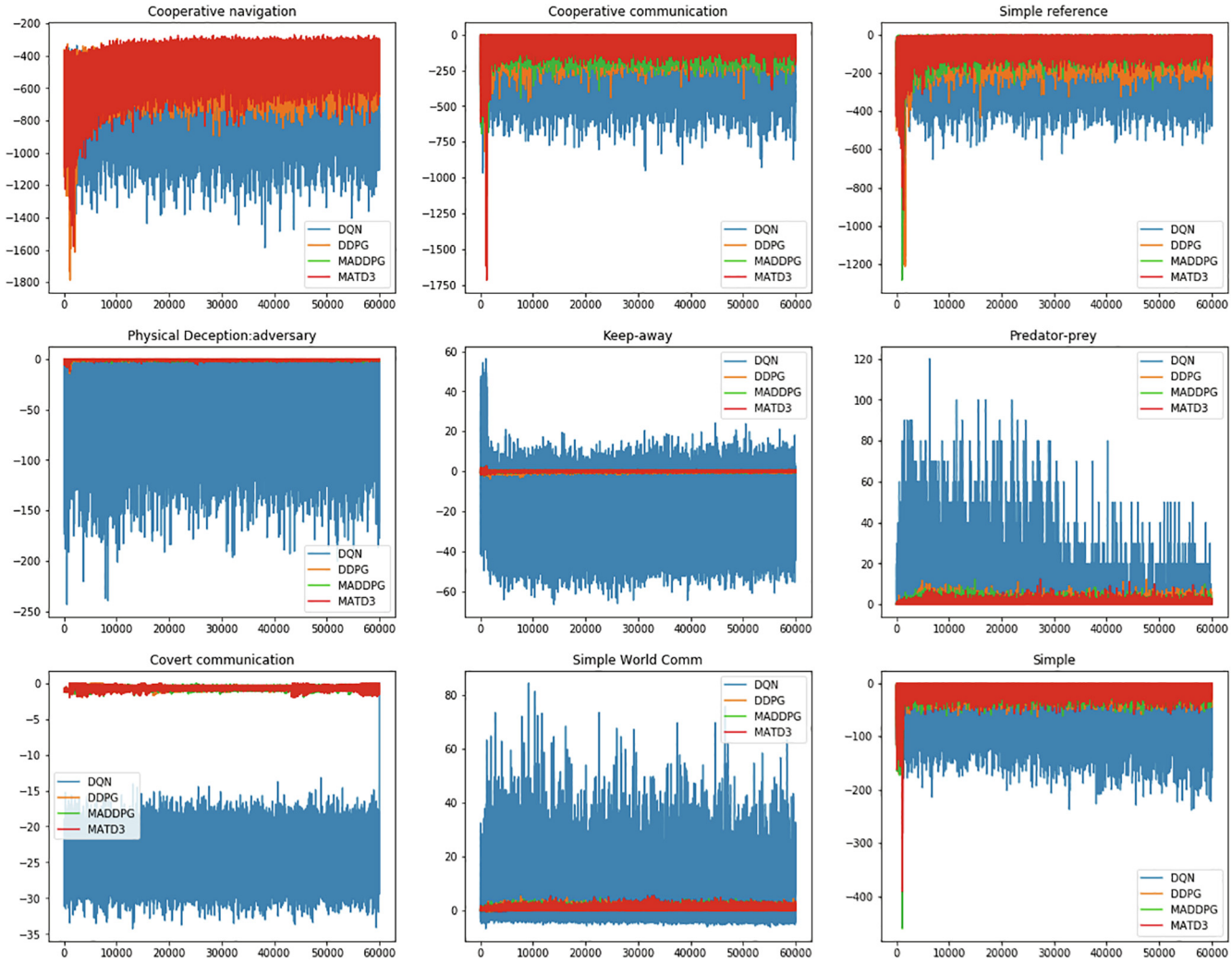

Fig. 5. MATD3 compared with DQN, DDPG and MADDPG in nine scenarios.

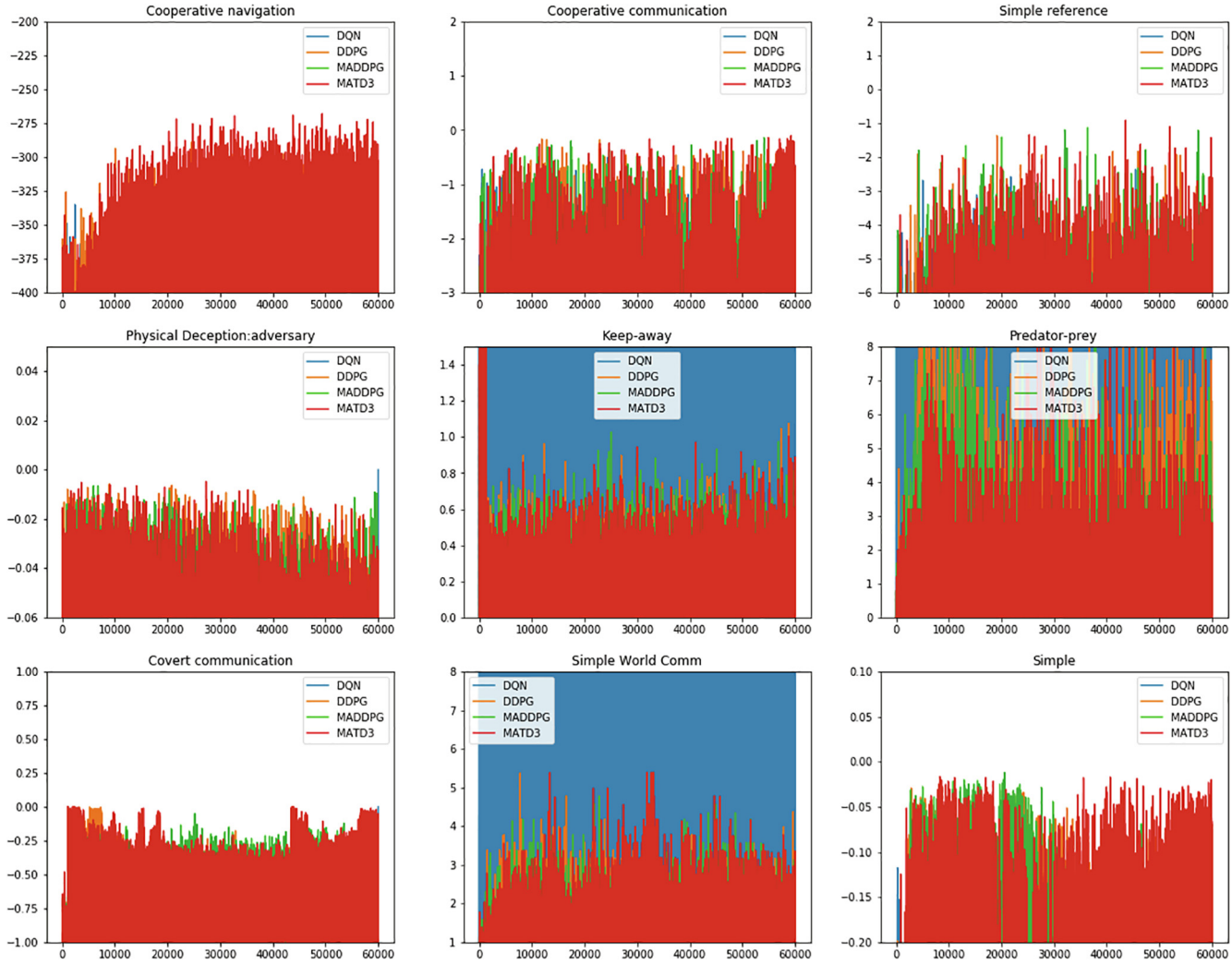

Fig. 6. Adjusted MATD3 compared with DQN, DDPG and MADDPG in nine scenarios.

Table 1
Results in competitive scenario physical deception.

Good agent 1, 2	Adversary agent	Good successful times	Adv successful times	δ successful times
MADDPG	MATD3	49,137	10,863	1369 (60 k)
MATD3	MADDPG	50,405	9494	1369 (60 k)
MATD3	MATD3	50,485	9515	1695 (60 k)
MADDPG	MADDPG	48,790	11,210	1695 (60 k)
MADDPG	MATD3	75,672	24,328	3538 (100 k)
MATD3	MADDPG	79,210	20,790	3538 (100 k)

Table 2
Complexity and improvement of space and tasks in nine scenarios.

Number	Experiment name	Competitive?	Comm?	Number of Agents	Number of	Raise by (%)
1	Simple	N	N	1	1	9.55%
2	Keep-away	Y	N	2	2	12.05%
3	Simple reference	N	Y	2	3	3.48%
4	Cooperative communication	N	Y	2	3	−3.64%
5	Physical deception	Y	N	3	2	−1.04%
6	Cooperative navigation	N	N	3	3	6.93%
7	Predator–prey	Y	N	4	2	−7.75%
8	Covert communication	Y	Y	3	2	12.08%
9	Simple world comm	Y	Y	6	5	25.04%

Fig. 5, it clearly shows that we get the highest score increase in this scenario, which confirms that the more complex the scenario of our MATD3 algorithm, the higher the execution of the group task.

Adv vs good means the competition between adversaries and good agents. Simple world comm involves six agents. Correspondingly, the adversaries group has four agents, the good group has two, and the landmark has five. The landmarks consist of two green areas representing the forest, two blue areas representing the water source, and a black obstruction area. Good agents try to get close to the water source to get rewards, and the adversaries prevents good agents from getting close to the water source. Agents hiding in the forest cannot be discovered by agents other than leaders. Agents in the same group learn to cooperate and can communicate to learn tasks such as chasing and pushing away.

6. Conclusion

This paper explores the overestimation errors and high variance problems in multiple agent deep reinforcement learning. We alleviate the overestimation problem by reducing the population centralization Q-value, and constrain the individuals action value additionally. The delayed policy update further stabilizes the multi-agent training environment, finally achieves good results in the experimental scenarios.

One of the advantage of delayed update policy is that the policy will improve itself towards a better direction when the critic is stable. However, we found that each agent's individual delayed policy update will cause a large group delay when extending the single agent TD3 to the multi-agent field, which limits the convergence speed and may even lead to lower exploration rate. Therefore, future research can consider increasing the exploration rate and increasing the utilization of high-value trajectories in the experience replay buffer.

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.

CRediT authorship contribution statement

Fengjiao Zhang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Jie Li:** Supervision. **Zhi Li:** Resources, Data curation, Project administration, Funding acquisition.

Acknowledgments

This work was supported by the Key Research and Development Project of Science & Technology Department of Sichuan Province under Grant 2019YFG0192.

References

[1] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484.
[2] M. Jaderberg, W.M. Czarnecki, I. Dunning, L. Marris, G. Lever, A.G. Castaneda, C. Beattie, N.C. Rabinowitz, A.S. Morcos, A. Ruderman, Human-level performance in first-person multiplayer games with population-based deep reinforcement learning, arXiv preprint arXiv:1807.01281.
[3] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, N. De Freitas, Dueling network architectures for deep reinforcement learning, arXiv preprint arXiv:1511.06581.
[4] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, pp. 1928–1937.
[5] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, arXiv preprint arXiv:1802.09477.
[6] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.

Appendix A

Algorithm 1: Delayed Double Critics DDPG for N-Agents Population

Initial two critic networks for each agent $i, Q^{\boldsymbol{\mu}}_{\theta_{i,1}}, Q^{\boldsymbol{\mu}}_{\theta_{i,2}}$ and actor network μ_{ϕ_i} with random parameters $\theta_{i,1}, \theta_{i,2}$, ϕ_i. Initial target networks for each agent $i, \theta'_{i,1} \leftarrow \theta_{i,1}$, $\theta'_{i,2} \leftarrow \theta_{i,2}$, $\phi'_i \leftarrow \phi_i$,Initial replay buffer $\mathscr{B}$
bf forepisode = 1 to Num-episodes **do**
  Initial a random noise $\xi$for exploring actions, and receive initial state $\mathbf{x} = (o_1, o_2, \ldots, o_N)$
  for t = 1 to Max-length-episodes **do**
    for each agent i, select random action $a_i \sim \boldsymbol{\mu}_i(o_i) + \xi_i$, w.r.t exploration and the current deterministic policy, $\xi_i \sim clip(N(0, \delta), a\_low, a\_high)$
    Execute actions $\mathrm{a} = (a_1, \ldots, a_N)$ and observe reward r and new state x'
    Store experience tuple $(\mathrm{x}', \mathrm{a}, r, \mathrm{x}')$ in replay buffer $\mathscr{B}$, and update state $\mathrm{x} \leftarrow \mathrm{x}'$
    foreach agent = 1 to N **do**
      Randomly sample a mini-batch including N samples $(\mathrm{x}^j, \mathrm{a}^j, r^j, \mathrm{x}'^j)$ from $\mathscr{B}$
      Set target $y^j = r_i^j + \gamma \min_{n=1,2} Q^{\mu'}_{\theta'_{i,n}}(\mathrm{x}'^j, a'_1, \ldots, a'_k, \ldots, a'_N)|_{a_k = \mu'_k(o_i^j)}$
      Update critic $\theta_{i,n} \leftarrow \mathrm{argmin}_{\theta_{i,n}} N^{-1} \sum \left(y^j - Q^{\boldsymbol{\mu}}_{\theta_{i,n}}\left(\mathbf{x}'^j, a_1^j, \ldots, a_N^j\right)\right)^2$
      if t mod d **then**
        Update actor parameter $\phi$by policy gradient: $\nabla_{\phi_i} J(\phi_i) = N^{-1}\left(\sum_j \nabla_{\phi_i} \boldsymbol{\mu}_i\left(o_i^j\right) \nabla_{a_i} Q^{\boldsymbol{\mu}}_{\theta_{i,1}}\left(\mathbf{x}^j, a_1^j, \ldots, a_i^j, \ldots, a_N^j\right)|_{a_i = \mu_i(o_i^j)}\right)$
        Update target networks:
        $\theta'_{i,n} \leftarrow \tau\theta_{i,n} + (1-\tau)\theta'_{i,n}$, $n = 1, 2$
        $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
      end if
    end for
  end for
end for

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[8] R. Lowe, Y. Wu, A. Tamar, J. Harb, O.P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems, pp. 6379–6390.
[9] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971.
[10] S. Mannor, D. Simester, P. Sun, J.N. Tsitsiklis, Bias and variance approximation in value function estimates, Manage. Sci. 53 (2) (2007) 308–322.
[11] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Thirtieth AAAI Conference on Artificial Intelligence.
[12] E. Greensmith, P.L. Bartlett, J. Baxter, Variance reduction techniques for gradient estimates in reinforcement learning, J. Mach. Learn. Res. 5 (2004) 1471–1530.
[13] R. Fox, A. Pakman, N. Tishby, Taming the noise in reinforcement learning via soft updates, arXiv preprint arXiv:1512.08562.
[14] B. O'Donoghue, I. Osband, R. Munos, V. Mnih, The uncertainty bellman equation and exploration, arXiv preprint arXiv:1709.05380.
[15] O. Nachum, M. Norouzi, G. Tucker, D. Schuurmans, Smoothed action value functions for learning gaussian policies, arXiv preprint arXiv:1803.02348.
[16] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, Weighted double deep multiagent reinforcement learning in stochastic cooperative environments, in: Pacific Rim International Conference on Artificial Intelligence, Springer, pp. 421–429.
[17] T. Rashid, M. Samvelyan, C.S. de Witt, G. Farquhar, J. Foerster, S. Whiteson, Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning, arXiv preprint arXiv:1803.11485.
[18] C. Chen, H. Modares, K. Xie, F.L. Lewis, Y. Wan, S. Xie, Reinforcement learning-based adaptive optimal exponential tracking control of linear systems with unknown dynamics, IEEE Trans. Autom. Control 64 (11) (2019) 4423–4438.
[19] B.H. Abed-alguni, M.A. Ottom, Double delayed q-learning, Int. J. Artif. Intell. 16 (2) (2018) 41–59.
[20] M. He, H. Guo, Interleaved q-learning with partially coupled training process, in: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 449–457.
[21] B.H. Abed-Alguni, S.K. Chalup, F.A. Henskens, D.J. Paul, A multi-agent cooperative reinforcement learning model using a hierarchy of consultants, tutors and workers, Vietnam J. Comput. Sci. 2 (4) (2015) 213–226.
[22] B. Hengst, Discovering hierarchy in reinforcement learning with hexq, in: ICML, vol. 19, pp. 243–250.
[23] T.D. Kulkarni, K. Narasimhan, A. Saeedi, J. Tenenbaum, Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation, in: Advances in Neural Information Processing Systems, pp. 3675–3683.
[24] G. Tesauro, Extending q-learning to general adaptive multi-agent systems, in: Advances in Neural Information Processing Systems, pp. 871–878.
[25] H.V. Hasselt, Double q-learning, in: Advances in Neural Information Processing Systems, pp. 2613–2621.
[26] J. Foerster, I.A. Assael, N. de Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems, pp. 2137–2145.
[27] T. Kasai, H. Tenmoto, A. Kamiya, Learning of communication codes in multi-agent reinforcement learning problem, in: 2008 IEEE Conference on Soft Computing in Industrial Applications, IEEE, pp. 1–6.
[28] M. Lauer, M. Riedmiller, An algorithm for distributed reinforcement learning in cooperative multi-agent systems, in. In Proceedings of the Seventeenth International Conference on Machine Learning, Citeseer.
[29] R.S. Sutton, Reinforcement learning, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Boston, 1992. URL Publisher description http://www.loc.gov/catdir/enhancements/fy0820/92007567-d.html Table of contents only http://www.loc.gov/catdir/enhancements/fy0820/92007567-t.html.
[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, 2014.
[31] S. Thrun, A. Schwartz, Issues in using function approximation for reinforcement learning, in: Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum.
[32] A.T. Ryan Lowe, Yi Wu, Multi-agent particle environment (11 2018). https://github.com/openai/multiagent-particle-envs.

Fengjiao Zhang, master of Sichuan University.
She mainly focuses on the research of multi-agent intelligence field and multi-agent reinforcement learning algorithms.

Jie Li, Deputy Director of Data Science and Service Center, Computer Science School, Beijing University of Posts and Telecommunications.
His research interests are in the areas of analysis of medical big data base on cloud computing, cloud-based operator traffic solution, and medical Internet of Things.

Zhi Li, Professor of Sichuan University.
He mainly focuses on the research of cognitive computing field, multi-agent reinforcement learning and signal intelligent sensing. His research interests include numerical simulation in electronic signals, analyzation of sensing data, design of sensors networks.