IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 16, 2020, accepted July 21, 2020, date of publication July 24, 2020, date of current version August 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011670

A Novel Multi-Agent Parallel-Critic Network Architecture for Cooperative-Competitive Reinforcement Learning

YU SUN[1,2], JUN LAI[1], LEI CAO[1], XILIANG CHEN[1], ZHIXIONG XU[1], AND YUE XU[2]
[1]Command and Control Engineering College, Army Engineering University of PLA, Nanjing 210000, China
[2]The PLA Unit 31102, Nanjing 210000, China

Corresponding authors: Lei Cao (caolei.nj@foxmail.com) and Xiliang Chen (383618393@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61806221, and in part by the Advanced research fund of equipment development department under Grant 61421010318.

ABSTRACT Multi-agent deep reinforcement learning (MDRL) is an emerging research hotspot and application direction in the field of machine learning and artificial intelligence. MDRL covers many algorithms, rules and frameworks, it is currently researched in swarm system, energy allocation optimization, stocking analysis, sequential social dilemma, and with extremely bright future. In this paper, a parallel-critic method based on classic MDRL algorithm MADDPG is proposed to alleviate the training instability problem in cooperative-competitive multi-agent environment. Furthermore, a policy smoothing technique is introduced to our proposed method to decrease the variance of learning policies. The suggested method is evaluated in three different scenarios of authoritative multi-agent particle environment (MPE). Multiple statistical data of experimental results show that our method significantly improves the training stability and performance compared to vanilla MADDPG.

INDEX TERMS Multi-agent system, deep reinforcement learning, parallel-critic architecture, training stability.

I. INTRODUCTION

Reinforcement learning (RL) [1] is an important branch of machine learning. The essence of RL is agents learning policies in the interaction process with the environment to maximize returns or achieve specific goals. Instead of guiding the agent on how to make actions correctly in supervised learning, RL usually evaluates and corrects the action selection based on the feedback signal from the environment. Therefore, RL is more suitable for solving complicated decision-making problems due to easier reward function design and lower information requirements. Recently, deep reinforcement learning (DRL) [2] that combines deep neural networks (DNN) with traditional RL methods has become a research hotspot and made tremendous breakthroughs in the computer vision system, robot control, large-scale real-time strategic games, etc.

The associate editor coordinating the review of this manuscript and approving it for publication was Inês Domingues.

Multi-agent system (MAS) [3] is a kind of dynamic system composed of multiple interactive agents. It is often used to solve coordination and cooperation problems. MAS has a wide range of applications in numerous fields due to its strong practicality and scalability. Multi-agent reinforcement learning (MARL) [4] can be seen as the application of RL methods in MAS. Generally speaking, traditional MARL can be divided into three categories according to the type of tasks: (1) fully cooperative. The agents have the same reward function: $R_1 = R_2 = \cdots = R_n$, which means that all agents are cooperating to achieve a common goal. Its representative algorithms include Distributed Q-learning [5], Team Q-learning [6], etc.; (2) fully competitive. The agents have the opposite reward function $R_1 = -R_2$, which means the goal of the agent is to maximize its private reward while minimizing the competitive agents' reward. Minimax-Q [7] is the typical algorithm for fully competitive settings; (3) Mixed. The reward functions of agents are different from each other, which means agents are self-interested and learning to achieve the equilibrium. Nash Q-learning [8],

Correlated Q-learning [9], Friend or Foe Q-learning [7] are designed for mixed tasks. However, traditional MARL algorithms are only suitable for small scale problems such as static games or grid worlds due to the limitation of computing power, while in real-world problems, the state and action space are usually large and continuous, few traditional algorithms can be effective in these kinds of environments. Moreover, Scalability and dealing with incomplete information environments are also unresolved problems in traditional MARL.

Recently, DNN has been applied to MARL to solve these problems, which leads to an emerging research field called multi-agent deep reinforcement learning (MDRL) [10]. Compared with single-agent DRL and MARL, learning agents in MDRL can communicate and collaborate to complete complicated tasks in broader scenarios. However, as an emerging research field, MDRL also faces many problems and challenges: (1) environmental non-stationarity. Each agent must consider the influence of dynamic environment and changing policies of other agents at the same time, that should seriously affect the learning stability and policy converging efficiency of each agent; (2) curse of dimensionality. MDRL environments usually have more agents and larger state-action space compared to single-agent DRL, which will increase the calculation and may cause the convergence failure; (3) partial observation problem. In most multi-agent systems, agents cannot perceive the complete information of the environment during the interaction, which makes it difficult to make optimal decisions thus affects the performance of learning algorithms.

Our work is focused on alleviating multi-agent environmental non-stationarity problem that affects learning stability of MDRL algorithm Multi-agent Deep Deterministic Policy Gradient (MADDPG) [11] in cooperative-competitive scenarios. Specifically, we modify the original actor-critic network architecture by increasing the number of centralized critics that operating parallelly to help consolidate the networks and training policies. The adopted parallel-critic architecture can jointly form more stable policies with relatively high efficiency. Besides, a policy smoothing technique is also designed for improving MADDPG to reduce variance during the training process.

In sum, the main contributions of this paper are outlined as follows:

- We propose a novel parallel-critic centralized network architecture based on MADDPG, which can not only stabilize the whole training process in non-stationary multi-agent environments but also improve the efficacy and effectiveness significantly.
- We design a target policy smoothing technique tailor-made for our proposed method by adding random noise as regularization to alleviate the high variance of policy training, thus further strengthen the policy robustness.
- We apply our improved algorithm to Multi-agent Particle Environment (MPE) [11]. The empirical experimental results show that the stability and effectiveness of our proposed method, which outperforms the original MADDPG algorithm on multiple cooperative and competitive tasks.

The organization of this paper is briefly shown below. Related work is presented in Section 2. Section 3 contains the preliminary knowledge, main structure and implementation details of our proposed method. Section 4 includes the experimental environment, implementation particulars and comparative analysis of results. Epilogue with the discussion of the future work of our proposed method is reported in the last section.

II. RELATED WORK

Many works have been done in MDRL field so far, the previous methods can be divided into the following 4 recommend categories:

(1) Extending single-agent DRL methods directly to multi-agent environments. Tampuu *et al.* [12] first extend the classic DQN algorithm to Atari Pong game and let the agents form cooperative and competitive behaviors. Gupta *et al.* [13] apply single-agent DDPG and TRPO to complex multi-agent environments with the combination of recurrent neural network (RNN), the improved algorithm has strong scalability; Bansal *et al.* [14] used another single-agent DRL algorithm PPO to train MuJoCo simulator. These methods are usually easy to implement but with the limitation of low training efficiency.

(2) Building explicit communication mechanisms among agents. Reinforced inter-agent learning (RIAL) [15] and differentiable inter-agent learning (DIAL) [15] use centralized Q network to build a communication channel for transferring information among agents, the communication model is primitive but capable of basic communication among agents; CommNet [16] builds a communication channel with the ability to transmit continuous information to form explicit team policies among cooperative agents, which ensures the real-time information transmission between all agents; bidirectional communication network (BiCNet) [17] combines actor-critic framework and bidirectional recurrent neural network to share and store information in each time step, BiCNet is also applied to real-time strategy game StarCraft 2 and successfully implement a variety of team tactics such as offense, retreat, attack and alternative coverage. Besides, attention communication MDRL [18], ACCNet [19], message pruning communication MDRL [20], double attentional actor-critic message processor (DAACMP) [21], attention-based message processing (AMP) [22], etc. are all the recent developments of MDRL methods with communication mechanisms. To sum up, the communication mechanism is always used in cooperative multi-agent environment to form joint policies but could be invalid in fully competitive scenarios. Moreover, these methods usually have high computational complexity.

(3) Modifying the network architectures. Value decomposition networks (VDN) [23] and QMIX [24]

use value function decomposition technique in training networks to decompose the global Q functions according to the contribution of collaborating agents; COMA [25] uses a global fully centralized critic network to improve the training efficiency in fully collaborative tasks, but its centralized architecture increases the input dimension and breadth of a single training network and thus may cause the curse of dimensionality, meanwhile, the single global network is not scalable; MADDPG first introduces the centralized training and decentralized execution (CTDE) [11], [15], [26] framework into MDRL field, in which each agent has a centralized critic to receive global information in the training phase, while performs the actions according to local observations (information) in the execution phase. With ideal performance in many multi-agent scenarios, the CTDE framework has been adopted in many MDRL methods since then with numerous advantages, MADDPG is applied in both cooperative and competitive scenarios such as MPE with satisfying performances.

(4) Other methods. These methods usually bring about some interdisciplinary tricks into MDRL field to help solve MAS problems. Q-value path decomposition (QDP) [27] integrates gradient attribution technique into MDRL to directly decompose global Q-values along trajectory paths to assign credits for agents, it has also been applied in StarCraft II micromanagement tasks with good performance; NCC-MARL [28] introduces neighborhood cognitive consistency (NCC) into MDRL to facilitate large-scale teamwork tasks like football player control. Mao *et al.* [29] propose a novel reward design method to accelerate the formation of better policies, the method is specially designed for packet routing application. Also, Yang *et al.* [30] borrow the mean field theory from stochastic process to deal with the large-scale multi-agent problems, its core idea is using mean values to simplify the multiple agents' interplay into double-agent's game, Mean field MDRL is extremely effective in scenarios with a massive amount of cooperative agents.

The above contents briefly summarize previous works in MDRL field. Among them, MADDPG has both advantages and disadvantages. First of all, its CTDE framework is perfectly aligned with multi-agent environment and can ease the training process; second, due to the independent training model and reward function of each agent, MADDPG can apply to diverse environments with fully cooperative, fully competitive or mixed learning agents. But in the meantime, MADDPG still has no good solution towards training instability problems caused by the multi-agent environment non-stationarity. Our proposed method not only inherits the advantages of MADDPG but also with better stability and training efficiency in learning optimal policies. Moreover, due to the simplicity of modifying architecture, our method is easier to implement and function. The specific architecture and implementation details of our method will be introduced in the following article.

III. METHODOLOGY

In this section, we first introduce the background knowledge in Section A, then the whole training architecture of our proposed method is presented in detail (Section B).

A. BACKGROUND

1) MULTI-AGENT MARKOV DECISION PROCESS (MAMDP)

In MDRL environments, agents follow multi-agent Markov decision process (MAMDPs) [4]. MAMDP is usually defined as a tuple: $\langle S, A_1, \cdots, A_n, R_1, \cdots, R_n, \rho, \gamma \rangle$, where n is the number of agents; S represents the state space of the environment; $A_i\ (i = 1, \cdots, n)$ is the action space of every single agent; $A = A_1 \times \cdots \times A_n$ means the joint action space of all agents; the joint state transition function $\rho : S \times A \times S \to [0, 1]$ determines the probability distribution of transition function from current state $s \in S$ to next state $s' \in S$ when the joint action $a \in A$ is executed. The learning goal of agent i is to maximize collective discounted reward $R = \sum_{1}^{n} \sum_{t'=t}^{T} \gamma^{t'-t} r_i^{t'}$ from time t to T ($\gamma \in [0, 1]$ is the discount coefficient to adjust the learning process). The model of standard MARL is shown in Figure 1.

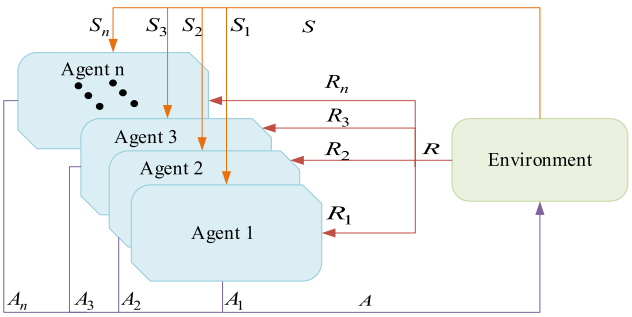

FIGURE 1. The model of standard MARL.

2) MULTI-AGENT DEEP DETERMINISTIC POLICY GRADIENT (MADDPG) AND EXTENSIONS

Multi-agent Deep Deterministic Policy Gradient (MADDPG) is an actor-critic algorithm designed for multi-agent environments. The joint policy space of N agents is $\pi = (\pi(\theta_1), \pi(\theta_2), \ldots, \pi(\theta_N))$ parameterized by $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$. The overall framework of MADDPG is derived from single-agent DRL algorithm DDPG [31], which consists of actor network, critic network, target actor network and target critic network in each agent's training structure. MADDPG adopts the CTDE framework (introduced in the above chapter) that each agent has a centralized critic to exchange information from other agents on the training phase while implementing policies separately based on its private observation. The agent i obtains current observation o_i^t at time t according to the policy $\pi(\theta_i)$, then interacts with environment to get experience $\left(o_i^t, a_i^t, o_i^{t+1}, r_i^t\right)$ and store in the replay buffer $\left(s_t, a_1^t, a_2^t, \ldots, a_N^t, r_1^t, r_2^t, \ldots, r_N^t, s_{t+1}\right)$,where $s_t = \left(o_1^t, o_2^t, \ldots, o_N^t\right)$ is the observation set of all agents

at time t. The input of each critic network includes the observations, actions and rewards of other agents: $Q(s_t, a_1, a_2, \ldots, a_N, \theta^Q)$, when the interaction process is finished, each agent from $i \sim N$ randomly extracts experiences from their own replay buffer for training and updates critic network parameter by minimizing the loss functions:

$$L_{\theta_i^Q} = \frac{1}{K}\sum_{t=1}^{K}(y_t - Q(s_t, a_1, a_2, \ldots, a_N, \theta_i^Q))^2 \quad (1)$$

$$y_t = r_t + Q_i'\left(s_{t+1}, a_1', a_2', \cdots a_N', \theta_i^{Q'}\right) \quad (2)$$

where $\theta_i^{Q'}$ is the parameter of agent i's target critic network. the parameter of each actor network from $i \sim N$ is updated by policy gradient descent method:

$$\nabla_{\theta_i^\pi} L = \frac{1}{K}\sum_{t=1}^{K}\nabla_{\theta_i^\pi}\pi(o, \theta_i^\pi)\nabla_a Q(s, a_1, a_2, \ldots, a_N, \theta_i^Q) \quad (3)$$

Same as DDPG, MADDPG soft updating the parameter of target critic and target actor networks in agent i to ease the convergence:

$$\theta_i^{Q'} = \tau\theta_i^Q + (1-\tau)\,\theta_i^{Q'}, \quad \theta_i^{\pi'} = \tau\theta_i^\pi + (1-\tau)\,\theta_i^{\pi'} \quad (4)$$

where τ is the updating coefficient to adjust the updating frequency.

3) IMPROVING EXTENSIONS BASED ON MADDPG

Many derivative works have been done focusing on inherent defects of MADDPG to improve the overall performance since then.

MADDPG-GCPN [32] proposes a decentralized generative policy network to guarantee the performance of cooperative agents in partially observatory environments, in which an extra actor network $\boldsymbol{\pi_i^{GC}}$ is set in each agent' network structure to imitate action samples of other agents, this avoids the concatenation of other agents' target policies in the training phase, so the training process can be conducted in an independent model, further, MADDPG-GCPN designs two sets of rewards: the global one with joint rewards of all the agents and the individual one with immediate private reward. Experiments show that MADDPG-GCPN has better performance than MADDPG in several scenarios where agents only have partial information from other agents and the environment.

PS-MADDPG [33] combines the parameter sharing technique with MADDPG to improving the training efficiency. The author suggests a couple of implementation plans: independent critics training simultaneously without sharing private rewards; one critic training and sharing weight to others; one critic training with multiple heads. Experimental results show that parameter sharing has better compatibility with MADDPG than other methods such as PS-PPO, PS-TRPO.

MATD3 [34] extends the single-agent DRL algorithm TD3 to multi-agent scenarios just as DDPG to MADDPG, it aims to relieve the value function overestimation problem in MDRL. Same as TD3 [35], MATD3 sets double critic networks $\boldsymbol{Q_{\theta_1}^i}$ and $\boldsymbol{Q_{\theta_2}^i}$ in each agent $\boldsymbol{i}$ and choose the smaller $\boldsymbol{Q}$ value to update the network. Experimental results prove the feasibility of this method in multi-agent settings.

Moreover, ATT-MADDPG [36] adds the attention layer in each critic network to strengthen the policy learning ability among cooperating agents; R-MADDPG [37] builds communication channels amongst agents through recurrent neural networks for learning in partial observatory environments; MAAC [38] uses attention mechanism to select relevant information for each agent during the training thus reduce the computational pressure and improve the scalability. Different from the extension works summarized above, Our method concentrates on training instability problem in MADDPG, we set up a novel parallel-critic network architecture with multiple critic networks training simultaneously to stabilize the training process and also improves the effectiveness. As far as we know, we are the first one using parallel-critic architecture in MADDPG to help solve MAS problems.

4) EXISTING METHODS INVOLVE PARALLEL (ENSEMBLE) VALUE FUNCTIONS

Parallel (ensemble) value functions method is first introduced to single-agent RL by Marco *et al.* [39], they implement 4 traditional RL algorithms and combine the policies produced by all the value functions to improve the policy adaptability and performance. Faußer *et al.* [40] propose a function approximation method to learn the combined state-value function of multiple agents, the core of their idea is value functions voting and averaging. Many improving methods including [41]–[43] have been put forward since then, some of them even combine with actor-critic algorithms. But as far as we know, parallel (ensemble) value functions technique has never been used in MDRL field before.

B. MADDPG WITH PARALLEL-CRITIC NETWORK ARCHITECTURE

In this work, we propose a novel parallel-critic network architecture based on MADDPG, which can provide stable policies for each learning agent in cooperative-competitive environments. we name it MADDPG with parallel-critic networks (MADDPG-PC). Specifically, we modify the original actor-critic network of each agent i in MADDPG to multiple parallel critics which can be capable of synchronized training, we set the agent i's Q value function in time t as the mean value of parallel critics' counterparts:

$$Q_{\text{mean}}\left(s_t, a_1, a_2, \cdots a_N, \theta^Q\right) = \frac{1}{U}\sum_{j=1}^{U} Q_j\left(s_t, a_1, a_2, \cdots a_N, \theta_j^Q\right) \quad (5)$$

where U is the number of parallel critics, $\theta_j^Q \in \theta^Q$ is the parameter of agent i's j-th critic. the U networks are training parallelly and independently. Theoretically speaking, this parallel-critic architecture has the following advantages compared to traditional single actor-critic MADDPG: (1) multiple

parallel critics train simultaneously without affecting each other, even if one of them performs awfully in several time steps, the combination of all critics' networks can offset the oscillation of Q value thus the training process is stabilized; (2) multiple parallel critics could expand the exploration scope of and learn the knowledge in the environment more widely.

To be consistent with the target network structure in MADDPG, we also set up the corresponding U target-critic networks, so the expected reward of agent i in time t can be rewritten as:

$$y_t = r_t + Q'_{\text{mean}}\left(s_{t+1}, a'_1, a'_2, \cdots a'_N, \theta_j^{Q'}\right) \tag{6}$$

where

$$Q'_{\text{mean}}\left(s_{t+1}, a'_1, a'_2, \cdots a'_N, \theta_j^{Q'}\right) = \gamma \frac{1}{U} \sum_{j=1}^{U} Q'_j\left(s_{t+1}, a'_1, a'_2, \cdots a'_N, \theta_j^{Q'}\right) \tag{7}$$

And $\theta_j^{Q'} \in \theta^{Q'}$ is the parameter of agent i's j-th target-critic, parameters of critic networks is updated by minimizing the loss function:

$$L_{PC} = \frac{1}{N} \sum_{t=1}^{N} \left(y_t - Q_{\text{mean}}\left(s_t, a_1, a_2, \cdots a_N, \theta^Q\right)\right)^2 \tag{8}$$

But in actual practice, each critic j of agent i trained by the same TD-error function may cause homogenization during the training process, so that the difference between critics are cut back, obviously, it's not conducive to convergence, so we first calculate the private loss function of each critic and its corresponding target critic with using their own Q value and target Q value:

$$L_{\theta_j^Q} = \frac{1}{N} \sum_{t=1}^{N} \left(y_t - Q_j\left(s_t, a_1, a_2, \cdots a_N, \theta_j^Q\right)\right)^2 \tag{9}$$

$$y_t = r_t + Q'_j\left(s_{t+1}, a'_1, a'_2, \cdots a'_N, \theta_j^{Q'}\right) \tag{10}$$

Then the final loss function of agent i's j-th critic can be adapted to:

$$L_{\theta_j^Q} = \phi L_{PC} + (1 - \phi) L_{\theta_j^Q} \tag{11}$$

where ϕ is the coefficient to compromise private loss of j-th critic and general loss.

The parameter of agent i's actor network is updating in the same way as MADDPG:

$$\nabla_{\theta^\pi} J = \frac{1}{K} \sum_{t=1}^{K} \nabla_{\theta^\pi} \pi\left(o, \theta^\pi\right) \nabla_a Q_{\text{mean}}\left(s_t, a_1, a_2, \cdots a_N, \theta^Q\right) \tag{12}$$

Figure 2 shows our structure of parallel-critic network architecture.

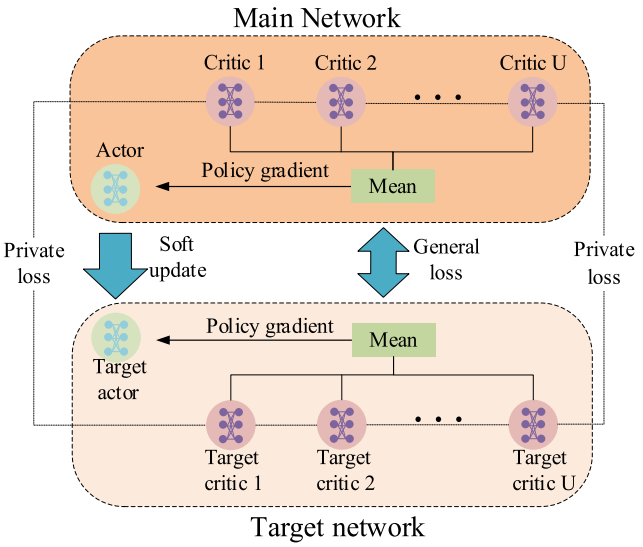

FIGURE 2. The structure of parallel-critic network architecture.

C. POLICY SMOOTHING TECHNIQUE

In some of the single-agent deterministic policy gradient methods such as DDPG, learning policy can be easily impacted by function approximation error [35], this causes the variance of target policies. Some improved algorithms attempt to alleviate this problem by applying disturbance factor (usually the clipped Gaussian noise) $\varphi = \text{clip}\left(\mathcal{N}\left(0, \zeta\right), -\lambda, \lambda\right)$ to actions of all the agents. This regularization method is also called policy smoothing technique.

In our method, we also introduce policy smoothing technique, so the expected reward of agent i with policy smoothing regularization is defined as:

$$y_t = r_t + Q'_{\text{mean}}\left(s_{t+1}, a'_1 + \varphi, a'_2 + \varphi, \cdots a'_N + \varphi, \theta_j^{Q'}\right) \tag{13}$$

where $\varphi = \text{clip}(\mathcal{N}(0, \varsigma), -\lambda, \lambda)$.

Table 1 describes the MADDPG-PC method Briefly, while the mainframe of the MADDPG algorithm remains unchanged. Figure 3 shows the framework of MADDPG-PC.

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

Our experimental hardware are Intel SSD PEKNW 512G + Nvidia GeForce RTX 2080TI + 64G memory, software environment are ubantu18.04 + TensorFlow = 1.7.0 + gym = 0.10.5. the hyper-parameters setup is show in table 2.

B. MPE ENVIRONMENT SETUP

Our test environment consists of three scenarios in OpenAI MPE [11] environment: cooperative navigation, predator-and-prey and physical deception.

1) COOPERATIVE NAVIGATION

Cooperative navigation is a fully cooperative scenario. It is set in a two-dimensional coordinate plane with N agents and

TABLE 1. MADDPG with parallel-critic architecture.

Algorithm 1 MADDPG-PC

Initialize the actor network $\pi_i(o, \theta^{\pi_i}) + \varphi$, U critic networks $Q_j(s_t, a_1 + \varphi, a_2 + \varphi, \cdots a_N + \varphi, \theta_j^Q)$ and $\theta_j^Q, \theta^{\pi_i}$
Initialize the target-actor network $\pi_i'(o, \theta^{\pi_i}) + \varphi$, U target-critic networks $Q_j'(s_{t+1}, a_1' + \varphi, a_2' + \varphi, \cdots a_N' + \varphi, \theta_j^{Q'})$ and θ^{π_i}, $\theta_j^{Q'}$
Initialize experience replay buffer R_i and random noise $\mathcal{N}$ for action exploration
For $episode = 1, \cdots, M$ **do**
  Initialize the environment and state set of all agents s_1
  For $t = 1$ to max episode length and each agent i in the environment **do**
    Select action: $a_i^t = \pi_i(o_i^t, \theta^{\pi_i}) + \mathcal{N}_t$
    Executes the action a_i^t, get the next state o_i^{t+1} and instant reward r_i^t
    Store the experience $(o_i^t, a_i^t, o_i^{t+1}, r_i^t)$ in replay buffer R_i
    For agent $i = 1, \cdots, N$ **do**
      Sample a random mini-batch from D
      Calculate the expected reward:
      $$y_t = r_t + Q'_{\text{mean}}\left(s_{t+1}, a_1', a_2', \cdots a_N', \theta_j^{Q'}\right)$$
      Update the parameters of U critic networks:
      $$L_{MC}\left(\theta_j\right) = \phi L_{MC} + (1-\phi) L_{\theta_j}$$
      Update the parameters of actor network
      $$\nabla_{\theta^\pi} J = \frac{1}{K}\sum_{t=1}^{K} \nabla_{\theta^\pi} \pi\left(o, \theta^\pi\right) \nabla_a Q_{\text{mean}}\left(s_t, a_1, a_2, \cdots a_N, \theta^Q\right)$$
      Update target network parameters for each agent i:
      $$\theta_j^{Q'} = \tau\theta_j^Q + (1-\tau)\theta_j^{Q'}$$
      $$\theta^{\pi'} = \tau\theta^\pi + (1-\tau)\theta^{\pi'}$$
    End for
  End for
End for

TABLE 2. Hyper-parameters.

Hyper-parameters	MADDPG-PC	MADDPG
Learning rate l	0.01	0.01
discount factor γ	0.95	0.95
batch size	1024	1014
MLP units	64	64
Updating rate/steps	100	100
Update factor τ	0.01	0.01
Parallel critics	2,3,4	1

the $\boldsymbol{K}$ target points, the learning goal of agents is reaching the target points and avoid colliding with each other. As shown in Figure 4 (left), we set $\boldsymbol{N} = \boldsymbol{K} = \mathbf{3}$. The reward of each agent depends on the distance to the nearest target and whether it collides with other agents, the reward function of each agent can be written as:

$$r_i = -\min_{i \le N, tr \le K}(D(i, tr)) + C \tag{14}$$

where $D(i, tr) = \sqrt{(x_i - x_{tr})^2 + (y_i - y_{tr})^2}$ is the distance between agent $\boldsymbol{i}$ and target point $\boldsymbol{tr}$, which means the closer to the nearest target point, the greater the reward will be. furthermore, once a collision occurs, the colliding agents will be punished, we define the collision reward as:

$$C = \begin{cases} -1, & \text{if the collision happens} \\ 0, & \text{if there is no collision} \end{cases} \tag{15}$$

that means for achieving the overall optimal goal, each agent needs to simultaneously consider the distance to the nearest target and the relative positions to other agents.

2) PREDATOR-AND-PREY

In cooperative learning environments, the policies and rewards of learning agents improving continuously during the training time. But in competitive environments, the rewards of agents depend not only on their own policies but also on the adversarial policies learned by their opponents. predator-and-prey is a typical competitive scenario. There are N relatively slow predators cooperating to hunt down a faster prey and $\boldsymbol{L}$ large obstacles that can block the movement and observation of agents in a two-dimensional coordinate plane. As shown in Figure 4 (middle), we set $\boldsymbol{N} = \mathbf{3}$ and $\boldsymbol{L} = \mathbf{1}$. The reward of each predator depends on the distance to the prey and whether the collision happens. The smaller the distance, the greater the reward. We set $\boldsymbol{D}(\boldsymbol{i}, \boldsymbol{j})$ to be the distance between predator $\boldsymbol{i}$ and predator $\boldsymbol{j}$:

$$D(i, j) = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{16}$$

When the predators collide with the prey (capture the prey successfully), the predators receive rewards whereas the prey receives a punishment (negative reward):

$$C_1 = \begin{cases} 10, & \text{if the collision happens} \\ 0, & \text{if there is no collision} \end{cases} \tag{17}$$

To maintain the normal functioning of the environment, we also exert punishments to prevent agents from escaping outside the border, the punishments depend on how far away the agents from the border:

$$C_2 = \begin{cases} 0 & \max(x_i, y_i) < 0.9 \\ (\max(x_i, y_i) - 0.9) \times 200 & \max(x_i, y_i) \ge 0.9 \end{cases} \tag{18}$$

Since the predators are collaborating to complete the hunting task, a good joint hunting policy is not the overlay of greedy policy for each agent. Some agents may sacrifice individual current rewards for completing the joint task, so we set the reward of predator $\boldsymbol{i}$ with minimal:

$$r_i = -0.1 \times \min_{i,j \le N}(D(i, j)) + C_1 - C_2 \tag{19}$$

As the prey has relatively faster moving speed and needs to face many predators s1multanously, we set the distance to be the sum of all agents in the environment. the reward of predator $\boldsymbol{j}$ is:

$$r_j = 0.1 \times \sum_{i=1}^{N}(D(i, j)) - C_1 - C_2 \tag{20}$$

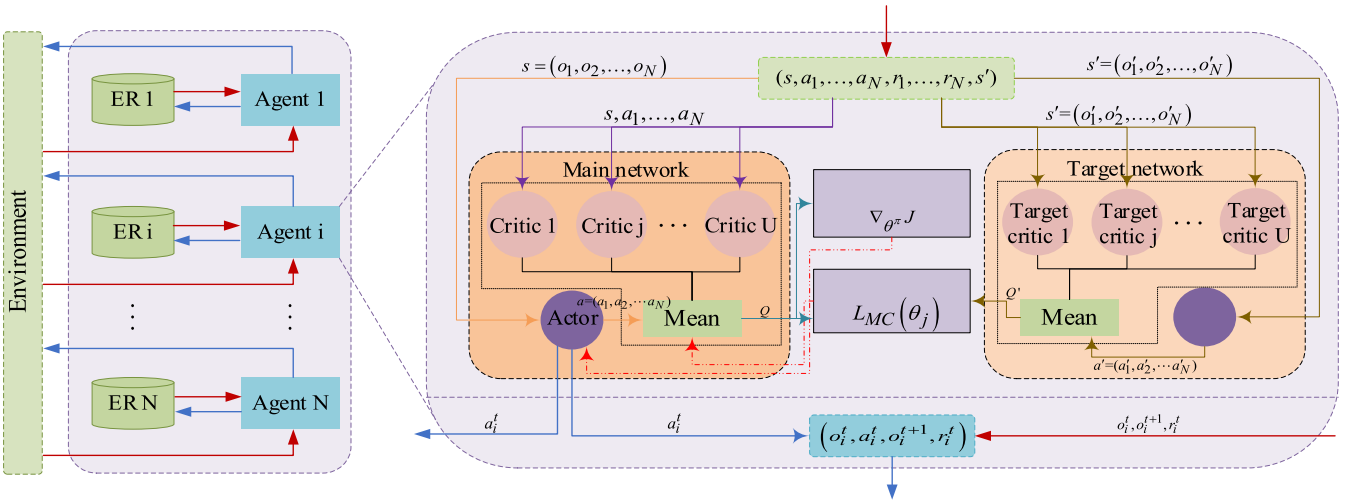

FIGURE 3. The framework of MADDPG-PC.

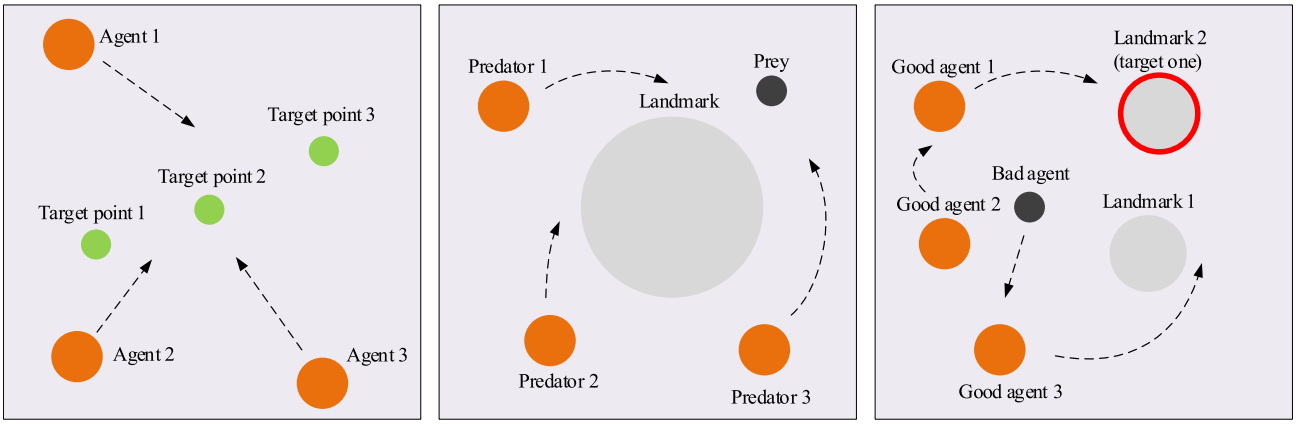

FIGURE 4. Diagram of three testing scenarios: cooperative navigation (left), predator-and-prey (middle), physical deception (right).

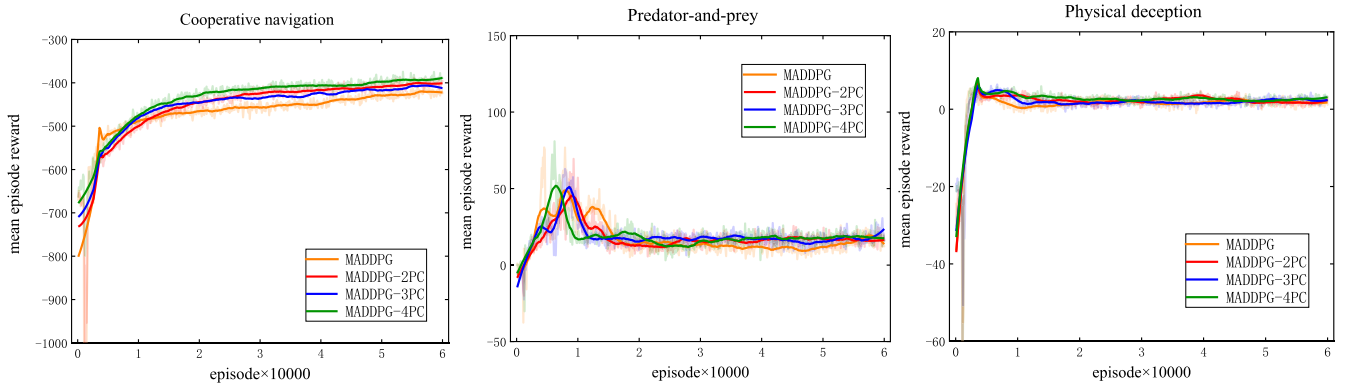

FIGURE 5. Mean episode reward of MADDPG-PC(U = 2,3,4) and original MADDPG in cooperative navigation (left), predator-and-prey (middle), physical deception (right).

3) PHYSICAL DECEPTION

Similar to predator-and-prey, physical deception is also a competitive scenario with N good agents, a bad agent and L landmarks, where one of the landmarks is the target landmark. As shown in Figure 4 (right), we set $N = 2$ and $L = 2$. The reward of each good agent depends on the distance to the target landmark and whether the collision happens, the smaller the distance, the greater the reward, if one of the good agents collides with the target landmark, all of them will get greater positive rewards. In the training process, good agents have to learn to scatter and cover all landmarks to deceive the bad agent.

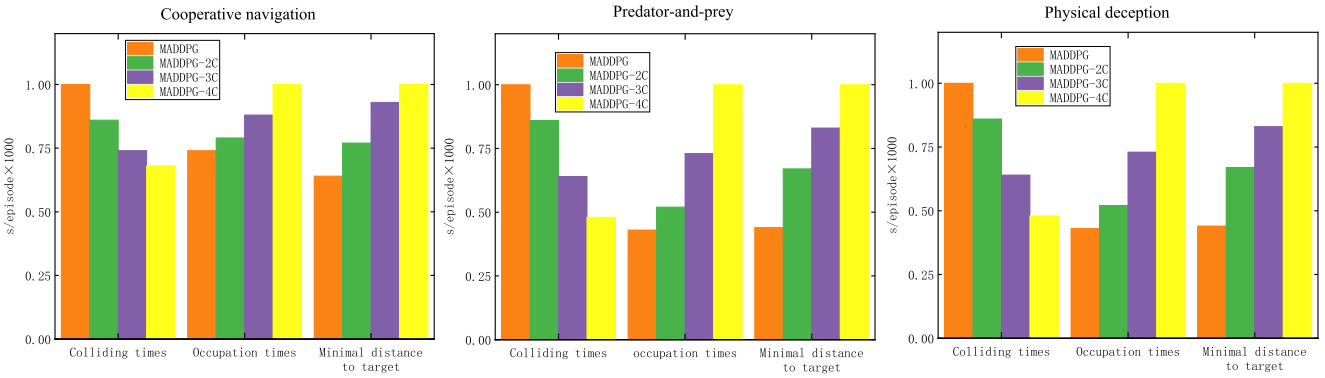

FIGURE 6. Important indicators with 0-1 regularization reflecting agents' performance of MADDPG-PC(U = 2,3,4) and MADDPG in cooperative navigation (left), predator-and-prey (middle), physical deception (right).

TABLE 3. Experimental data between MADDPG-PC(U = 2,3,4) and MADDPG in testing scenarios after 40000 episodes.

Scenario	Algorithm	Max reward	Mean reward (increasing rate)		Reward STD (reduction rate)	
			After 40000 episodes/100 episodes			
Cooperative navigation	MADDPG	-402.86	-431.13		10.61	
	MADDPG-2PC	-388.51	-408.43	5.27%	8.73	17.71%
	MADDPG-3PC	-391.04	-415.69	3.58%	9.52	10.29%
	MADDPG-4PC	-374.38	-398.80	7.50%	8.99	15.25%
Predator-and-prey	MADDPG	25.93	13.06		4.42	
	MADDPG-2PC	25.54	16.85	29.09%	3.10	29.95%
	MADDPG-3PC	30.66	16.80	28.68%	3.95	10.68%
	MADDPG-4PC	29.55	18.06	38.31%	3.33	24.60%
Physical deception	MADDPG	3.92	1.95		0.79	
	MADDPG-2PC	4.49	2.09	6.83%	0.73	8.35%
	MADDPG-3PC	4.28	2.10	7.73%	0.76	4.47%
	MADDPG-4PC	3.92	2.43	24.37%	0.71	10.25%

C. RESULTS AND ANALYSIS

We set original MADDPG as the comparison algorithm to verify the efficiency of our improved MADDPG-PC. The prey and bad agent in three scenarios all use DDPG algorithm. We set U = 2,3,4 in MADDPG-PC separately as the antithesis to juxtapose the effect of critics with different numbers on performance, so each scenario has four comparison tests.

Unlike other ML methods such as deep learning, MDRL does not have data sets, so we mainly use the following indicators to assess the training stability, performance, effectiveness and adaptability of our method: (1) we use standard deviation (STD) of mean episode reward to evaluate the training stability, the smaller the deviation, the more stable the algorithms; (2) we use the mean episode reward to evaluate the efficacy of algorithms, the greater the reward, the better the performance; (3) we use the performance of agents to assess the effectiveness of algorithms, the better the performance of agents, the higher the effectiveness; (4) we change the number of agents to evaluate the adaptability of our algorithms. The comparison of experimental results with the above indicators is elaborated below.

1) EVALUATION OF TRAINING STABILITY OF ALGORITHM

Figure 5 shows the mean episode reward curves of the original MADDPG and MADDPG-PC (U = 2,3,4) in three scenarios after 60000 episodes of training, we can infer that after 30000 episodes the curves tend to be stable in each scenario, so our calculation is based on testing data after 30000 episodes. As shown in TABLE 3, MADDPG-PC (U = 2,3,4) reduce by 17.71%, 10.29%, 15.25% on reward STD in cooperative navigation compared with MADDPG; reduce by 29.95%, 10.68%, 24.60% in predator-an-prey; reduce by 8.35%, 4.47%, 10.25% in physical deception. We can thus conclude that MADDPG-PC can stabilize the training process to a significant degree by parallel critics training simultaneously, we can also speculate that most of the time the stability improves with the increasing number of critics because more critics can supply more stable policies. On the other hand, according to experimental comparison data we can infer that the increase in the number of critics doesn't have a linear relationship with the stability improvement, this may depend on the complexity of the certain scenario or the complexity of the algorithm.

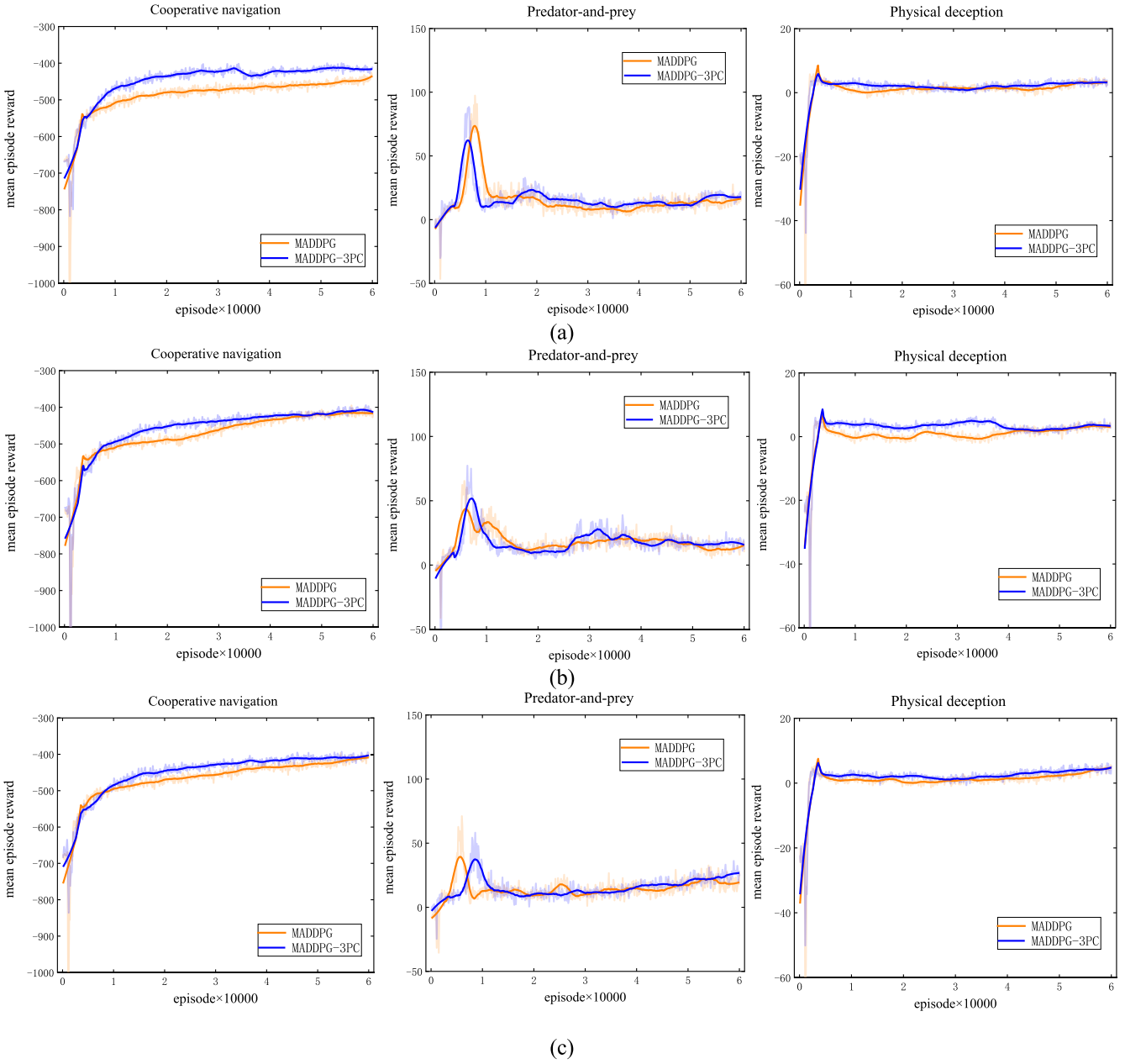

FIGURE 7. Mean episode reward of MADDPG-3PC and original MADDPG with 2 agents (a), 4 agents (b) and 5 agents (c) in 3 scenarios.

2) EVALUATION OF TRAINING EFFICACY OF ALGORITHM

Combining Figure 5 and TABLE 3, the mean rewards/ 100 episodes of MADDPG-PC (U = 2,3,4) 5.27%, 3.58%, 7.50% compared with MADDPG in cooperative navigation; increase by 29.09%, 28.68%, 38.31% in predator-and-prey; increase by 6.83%, 7.73%, 24.37% in physical deception. We thus conclude that parallel-critic architecture can also improve the training efficacy by expanding the exploring range of agents, moreover, the increasing number of parallel learning critics indeed have beneficial effects on training (MADDPG-4PC performs the best).

3) EVALUATION OF EFFECTIVENESS OF ALGORITHM

According to the introduction of scenarios in the above section, we mainly adopt the following three indicators to evaluate the performance of agents: mean colliding times, mean occupation (capture) times and mean minimal distance to target point (prey). Figure 6 shows the above three indicators with 0-1 regularization of MADDPG-PC (U = 2,3,4) and MADDPG in three scenarios. We can see that in all three scenarios, the MADDPG-PC (U = 2,3,4) reduces in mean colliding times, improves in the mean minimal distance to target point (prey), and mean occupation (capture) times distinctively. So we can conclude that with the

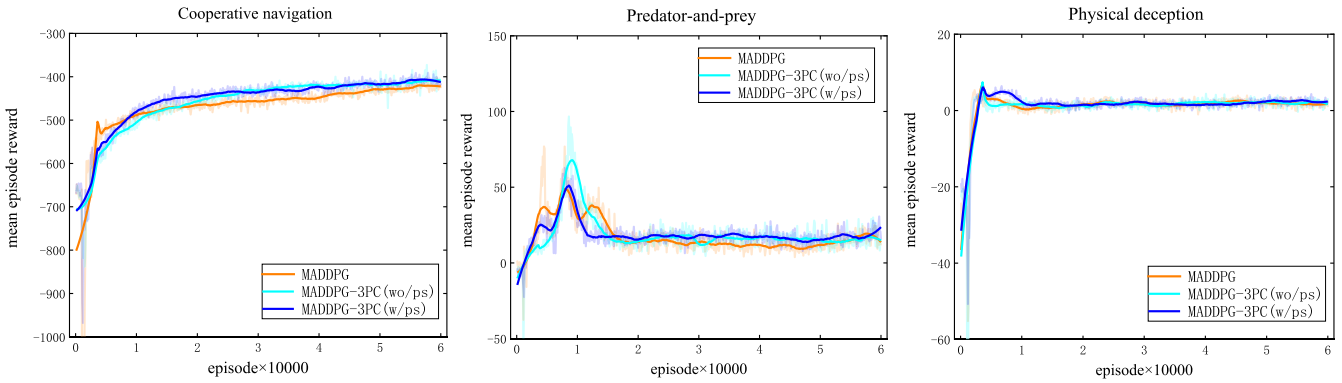

FIGURE 8. Mean episode reward of MADDPG-3PC (with policy smoothing), MADDPG-3PC (without policy smoothing) and original MADDPG in cooperative navigation (left), predator-and-prey (middle), physical deception (right).

TABLE 4. Experimental data between MADDPG-3PC and MADDPG with different number of agents.

Number of agents	scenario	Algorithm	Max reward	Mean reward (increasing rate)		Reward STD (reduction rate)	
				After 40000 episodes/100 episodes			
2	Cooperative navigation	MADDPG	-426.04	-455.34		9.62	
		MADDPG-3PC	-400.32	-417.80	8.24%	6.62	31.17%
	Predator-and-prey	MADDPG	27.56	13.13		4.03	
		MADDPG-3PC	27.79	14.71	12.00%	3.75	6.95%
	Physical deception	MADDPG	4.52	1.99		1.19	
		MADDPG-3PC	4.91	2.69	35.50%	0.82	30.62%
4	Cooperative navigation	MADDPG	-395.17	-421.75		8.71	
		MADDPG-3PC	-394.50	-416.55	1.23%	7.61	12.59%
	Predator-and-prey	MADDPG	30.42	15.71		4.49	
		MADDPG-3PC	30.63	16.91	7.66%	3.07	31.72%
	Physical deception	MADDPG	4.72	2.34		1.01	
		MADDPG-3PC	5.53	2.69	15.16%	0.87	13.63%
5	Cooperative navigation	MADDPG	-396.58	-425.25		11.05	
		MADDPG-3PC	-393.20	-411.09	3.33%	7.03	36.38%
	Predator-and-prey	MADDPG	32.55	17.19		5.08	
		MADDPG-3PC	36.15	20.42	18.78%	4.48	11.75%
	Physical deception	MADDPG	5.58	2.58		1.28	
		MADDPG-3PC	6.11	3.60	39.42%	0.98	23.49%

increasing number of our parallel-critic architecture, the performance of agents is getting better, so does the training effectiveness.

4) EVALUATION OF ADAPTABILITY OF ALGORITHM WHILE CHANGING THE NUMBER OF AGENTS

We also evaluate the performance of our algorithm while changing the number of agents in each scenario. We choose MADDPG-3PC and original MADDPG as comparison for simplifying the experimental procedure.

As shown in Figure 7 and TABLE 4, indicators of our algorithm do not degenerate when the number of agents changes, this confirms that MADDPG-PC has strong adaptability when training agents are changing.

5) ABLATION STUDIES ON POLICY SMOOTHING TECHNIQUE

In order to identify the effect of our proposed policy smoothing technique, we choose the following 3 algorithms: original MADDPG, MADDPG-3PC (without policy smoothing) and MADDPG-3PC (with policy smoothing) for comparative analysis. The testing scenarios are the same as above. Figure 8 shows the mean episode reward curves of 3 algorithms, we can easily conclude that policy smoothing technique can not improve the training efficacy of MADDPG agents. TABLE 5 compares the reward STD in the whole 60000 episodes and after 40000 episodes respectively of the 3 algorithms, we can conclude that MADDPG-3PC (with policy smoothing) outperform the rest in both indexes, and MADDPG-3PC (without policy smoothing) is even less

TABLE 5. Reward STD in whole 60000 episodes and after 40000 episodes of 3 algorithms in 3 scenarios.

scenario	Algorithm	Reward STD (after 40000 episodes)	Reward STD (60000 episodes)
Cooperative navigation	MADDPG	10.61	81.75
	MADDPG-3PC (wo/ps)	10.76	69.95
	MADDPG-3PC (w/ps)	9.52	66.29
Predator-and-prey	MADDPG	4.42	12.22
	MADDPG-3PC (wo/ps)	4.61	13.20
	MADDPG-3PC (w/ps)	3.95	9.05
Physical deception	MADDPG	0.79	5.21
	MADDPG-3PC (wo/ps)	0.85	6.33
	MADDPG-3PC (w/ps)	0.76	4.70

stable than original MADDPG in some scenarios. That means policy smoothing can help stabilize the training process significantly by adding and clipping noise in action inputs.

V. CONCLUSIONS AND FUTURE WORK

In this article, we present a novel parallel-critic architecture based on MDRL algorithm MADDPG called MADDPG-PC, which can not only stabilize the whole training process in non-stationary multi-agent environments but also improve the efficacy and effectiveness significantly. We also design a target policy smoothing technique tailor-made for our proposed method by adding random noise as regularization to alleviate the high variance of policy training, thus further strengthen the policy robustness. Empirical evaluation results in 4 aspects show that our method outperforms the original MADDPG algorithm on multiple cooperative-competitive tasks.

However, our research has the following shortcomings: (1) the structure and training pattern of our method may increase the general training time, especially in some complex scenarios. (2) Our method is not applicable to multiple environments apart from MPE. (3) Our method lacks an explicit communication mechanism, cooperative agents form collective policies in a spontaneous way, which is very inefficient compared to novel algorithms like COMA, CommNet, etc.

Thus, we will focus on the following directions in the next work: (1) reducing the training time by incorporating novel tricks such as parameter sharing, removing unnecessary information and compressing state or action space in our current structure. (2) Developing the adaptability of our method to different environments by incorporating transfer learning [44] or curriculum learning [45] techniques. (3) Improving the algorithm's ability to generate better cooperative policies in mixed environments by combing game theory, communication rules, and other traditional RL methods.

ACKNOWLEDGMENT

(Yu Sun and Jun Lai contribute equally to this work.)

REFERENCES

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[3] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," in *Proc. IEEE/PES Power Syst. Conf. Expo.*, Mar. 2009, pp. 1–8.

[4] L. Buşoniu, R. Babuška, and S. B. De, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications*. Berlin, Germany: Springer, 2010, pp. 183–221.

[5] A. Galindo-Serrano and L. Giupponi, "Distributed Q-learning for aggregated interference control in cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1823–1834, May 2010.

[6] Y. Wang and C. De Silva, "Multi-robot box-pushing: Single-agent Q-learning vs. Team Q-learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 3694–3699.

[7] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. ICML*, 2001, pp. 322–328.

[8] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[9] A. Greenwald, K. Hall, and R. Serrano, "Correlated Q-learning," in *Proc. 20th Int. Conf. Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 242–249.

[10] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, Nov. 2019.

[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[12] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS ONE*, vol. 12, no. 4, pp. 1–15, 2017.

[13] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.* Cham, Switzerland: Springer, 2017, pp. 66–83.

[14] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," 2017, *arXiv:1710.03748*. [Online]. Available: http://arxiv.org/abs/1710.03748

[15] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.

[16] S. Sukhbaatar and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.

[17] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play StarCraft combat games," 2017, *arXiv:1703.10069*. [Online]. Available: http://arxiv.org/abs/1703.10069

[18] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7254–7264.

[19] H. Mao, Z. Gong, Y. Ni, and Z. Xiao, "ACCNet: Actor-coordinator-critic net for 'learning-to-communicate' with deep multi-agent reinforcement learning," 2017, *arXiv:1706.03235*. [Online]. Available: http://arxiv.org/abs/1706.03235

[20] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, "Learning agent communication under limited bandwidth by message pruning," 2019, *arXiv:1912.05304*. [Online]. Available: http://arxiv.org/abs/1912.05304
[21] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, "Learning multi-agent communication with double attentional deep reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 34, pp. 1–34, Mar. 2020.
[22] Z. Peng, L. Zhang, and T. Luo, "Learning to communicate via supervised attentional message processing," in *Proc. 31st Int. Conf. Comput. Animation Social Agents (CASA)*, 2018, pp. 11–16.
[23] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2018, pp. 2085–2087.
[24] T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," 2018, *arXiv:1803.11485*. [Online]. Available: http://arxiv.org/abs/1803.11485
[25] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," 2017, *arXiv:1705.08926*. [Online]. Available: http://arxiv.org/abs/1705.08926
[26] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *J. Artif. Intell. Res.*, vol. 32, pp. 289–353, May 2008.
[27] Y. Yang, J. Hao, G. Chen, H. Tang, Y. Chen, Y. Hu, C. Fan, and Z. Wei, "Q-value path decomposition for deep multiagent reinforcement learning," 2020, *arXiv:2002.03950*. [Online]. Available: http://arxiv.org/abs/2002.03950
[28] H. Mao, W. Liu, J. Hao, J. Luo, D. Li, Z. Zhang, J. Wang, and Z. Xiao, "Neighborhood cognition consistent multi-agent reinforcement learning," 2019, *arXiv:1912.01160*. [Online]. Available: http://arxiv.org/abs/1912.01160
[29] H. Mao, Z. Gong, and Z. Xiao, "Reward design in cooperative multi-agent reinforcement learning for packet routing," 2020, *arXiv:2003.03433*. [Online]. Available: http://arxiv.org/abs/2003.03433
[30] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," 2018, *arXiv:1802.05438*. [Online]. Available: http://arxiv.org/abs/1802.05438
[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971
[32] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with generative cooperative policy network," 2018, *arXiv:1810.09206*. [Online]. Available: http://arxiv.org/abs/1810.09206
[33] X. Chu and H. Ye, "Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning," 2017, *arXiv:1710.00336*. [Online]. Available: http://arxiv.org/abs/1710.00336
[34] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," 2019, *arXiv:1910.01465*. [Online]. Available: http://arxiv.org/abs/1910.01465
[35] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*. [Online]. Available: http://arxiv.org/abs/1802.09477
[36] H. Mao, Z. Zhang, Z. Xiao, and Z. Gong, "Modelling the dynamic joint policy of teammates with attention multi-agent DDPG," 2018, *arXiv:1811.07029*. [Online]. Available: http://arxiv.org/abs/1811.07029
[37] R. E. Wang, M. Everett, and J. P. How, "R-MADDPG for partially observable environments and limited communication," 2020, *arXiv:2002.06684*. [Online]. Available: http://arxiv.org/abs/2002.06684
[38] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.
[39] M. A. Wiering and H. van Hasselt, "Ensemble algorithms in reinforcement learning," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 38, no. 4, pp. 930–936, Aug. 2008.
[40] S. Faußer and S. Friedhelm, "Ensemble methods for reinforcement learning with function approximation," in *Proc. Int. Workshop Multiple Classifier Syst.*, 2011, pp. 56–65.
[41] S. Faußer and F. Schwenker, "Neural network ensembles in reinforcement learning," *Neural Process. Lett.*, vol. 41, no. 1, pp. 55–69, Feb. 2015.
[42] A. Hans and S. Udluft, "Ensembles of neural networks for robust reinforcement learning," in *Proc. 9th Int. Conf. Mach. Learn. Appl.*, Dec. 2010, pp. 401–406.
[43] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "EPOpt: Learning robust neural network policies using model ensembles," 2016, *arXiv:1610.01283*. [Online]. Available: http://arxiv.org/abs/1610.01283
[44] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[45] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

YU SUN received the B.E. degree from the Nanjing University of Posts and Telecommunications, in 2015. He is currently pursuing the master's degree with Army Engineering University. His research interests include intelligent decision making and mission planning.

JUN LAI received the M.S. degree in military communication from PLAUST, in 2005. He is currently an Associate Professor with the Army Engineering University of PLA. His current research interests include deep reinforcement learning and computer simulation.

LEI CAO received the B.S. and M.S. degrees from the China University of Science and Technology and PLA University of Science and Technology, in 1987 and 1990, respectively. He is currently a Professor with the PLA University of Science and Technology. His research interests include machine learning, command information system, and intelligent decision making.

XILIANG CHEN received the B.S. and M.S. degrees from the PLA University of Science and Technology, in 2007 and 2009, respectively, where he is currently pursuing the Ph.D. degree. He is also Teaching at the PLA University of Science and Technology. His research interests include reinforcement learning and intelligent decision making.

ZHIXIONG XU received the B.E. and M.S. degrees from the PLA University of Science and Technology, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree. His research interests include machine learning and intelligent decision making.

YUE XU received the B.E. and M.S. degrees from Southeast University, in 1988 and 1991, respectively. His current research field involves simulation and intelligent decision-making.