A Deep Reinforcement Learning Method based on Deterministic Policy Gradient for Multi-Agent Cooperative Competition

Xuan Zuo*, Hui-Feng Xue, Xiao-Yin Wang**, Wan-Ru Du**, Tao Tian**, Shan Gao*, Pu Zhang***

** School of Automation, Northwestern Polytechnical University, Xi'an 710072 P.R.China (e-mail: 1069114233@qq.com).*
*** China Aerospace Academy of Systems Science and Engineering, Beijing 100048 P.R.China (e-mail: xhf0616@163.com)}*

Abstract: Deep reinforcement learning in multi-agent scenario is important for real-world applications but presents challenges beyond those seen in single agent settings. This paper proposes a method to train a team of multiple types of agents to cooperate against another team of agents. Furthermore, this paper studies how to train multiple types of agents to collaborate better on their team tasks, and analyses the influence of various factors on agents' policy. In the computer experiments, agents are divided into attacking agents and defending agents. The results show that attacking agents which play the roles of deceivers can attract most of defending agents and help the other attacking agents to reach their targets successfully. Choosing appropriate length of training could help agents learn better action policy. The experiments results reveal that the number of agents has an effect on the performance of our proposed method. Increasing the number of deceivers in attacking agents can significantly increase the mission success of attacking team, but the computational complexity will rise and more episodes are needed to train agents.

Keywords: Machine learning, reinforcement learning, multi-agent, cooperative competition, artificial intelligence.

1. INTRODUCTION

Much progress towards artificial intelligence has been made using supervised learning systems that are trained to replicate the decisions of human experts (Silver et al., 2017; Hastie et al., 2009; LeCun et al., 2015; Krizhevsky et al., 2012). Reinforcement Learning (RL) is one of the most general formulation of the learning problem. Unlike supervised learning, the feedback is partial and in many cases the rewards are delayed. It also differs from unsupervised learning because the aim is not to find hidden structure in unlabeled data but to solely maximize the reward signal. An RL researcher is conventionally expected to come up with a good reward function and subsequently provide a robust RL algorithm to generalize to unseen trajectories of the feedback loop. Deep reinforcement learning (DRL) represents a step towards building autonomous systems with a higher level understanding of the visual world. In DRL (Arulkumaran et al., 2017; Francois-Lavet et al., 2018), deep neural networks are trained to approximate the optimal policy or the value function. In this way the deep neural networks, serving as function approximator, enables powerful generalization. One of the key advantages of DRL is that it enables reinforcement learning to scale to problems with high-dimensional state and action spaces (Hernandez-Leal et al., 2019).

Recently, there has been rapid progress using deep neural networks trained by reinforcement learning. These systems have outperformed humans in games, such as AlphaGo (Silver et al., 2016), AlphaGo Zero (Silver et al., 2017) and Alpha Zero (Silver et al., 2017, 2018). Otherwise, in the study of antagonistic video games based on local visual information, Deepmind's AlphaStar (Vinyals et al., 2019) in the real-time strategy game Starcraft II, and OpenAI Five (OpenAI, 2018) in the multiplayer game DOTA 2 show outstanding performance beyond top human players. The outstanding ability of deep reinforcement learning in solving single individual policy optimization problem urges researchers to try to apply related methods to solve multi-agent cooperation or competition problems.

The nature of interaction between agents can either be co-operative, competitive, or both and many algorithms are designed only for a particular nature of interaction. These algorithms are generally not applicable in competitive or mixed settings (Lowe et al., 2017). As multi-agent learning needs to search for the optimal policy in the high-dimensional state space, it will bring about an unstable training environment, if each agent learns policy from its own perspective without cooperative communication or setting of sharing global utility. In recent years, the actor-critic algorithm framework which can be used to optimize agents' policy with centralized training and control agents' actions with decentralized execution, has been introduced into Multi-Agent Reinforcement Learning (MARL) to share agents' utility in various ways.

Previous works have shown how the MARL approach can be used to cooperation mission or competition mission between two types of agents. This paper presents a method to train multi types of agents to complete both competition and cooperation mission simultaneously, and demonstrates its performance by computer experiments. Centralized Critic is adopted to evaluate and optimize agents' policy, which make multi types of agents cooperate for team work.

In this paper, agents are divided into attacking agents and defending agents. Agents in attacking team which undertake the attacking mission can arrive their targets synchronously by minimizing the standard deviation of the distance between agents and targets. The other agents in attacking team which undertake deception mission can induce adversaries at close range by maximizing the value of a Gauss-Quadratic mixing function about the distance between them and defending agents. The defending agents intercept adversaries by minimizing the distance to the attacking agents and giving him a constant reward on contact. This paper introduces multi-agent deep deterministic strategy gradient algorithm which combines Actor-Critic framework and DDPG algorithm to train agents learning policy, so that the multi-agent multi-mission multi-target co-control could be realized. This paper also analyses the effects of the number of training episodes and the number of agents undertook deception mission on the training results.

2. RELATED WORKS

A number of complex problems in today's society can be modeled as multi-agent learning problems. A few examples include multi-robot control, analysis of social dilemmas, managing air traffic flow and energy distribution, etc. When one or more agents fail in a multi-agent system, the remaining agents can take over some of their tasks. This implies that multi-agent system is inherently robust. Furthermore, by design most multi-agent systems also allow the easy insertion of new agents into the system, leading to a high degree of scalability. While single-agent RL have a relatively strong theoretical foundation, a thorough understanding of the learning problem in multi-agent settings is still an open problem (Kapoor, 2018). Therefore, progress in Multi-Agent RL systems is due.

Many initial approaches have been focused on tabular methods to compute Q-values for general sum Markov games (Hu and Wellman, 2003). Another approach in the past has been to remove the non-stationarity in MARL by treating each episode as an iterative game, where the other agent is held constant during its turn. In such a game, the proposed algorithm searches for a Nash equilibrium (Conitzer and Sandholm, 2007). Naturally, for complex competitive or collaborative tasks with many agents, finding a Nash equilibrium is non-trivial. Building on the recent success of methods for deep RL, there has been a renewed interest in using high capacity models such as neural networks for solving MARL problems (Khan et al., 2018).

In most of the MARL algorithms, the training mechanism is assigned to each agent separately. For example, independent Q learning algorithm is adopted to train each agent. The distributed learning architecture above reduces the difficulty of implementing learning and the complexity of calculation. For the DRL problem of large-scale state space, a simple multi-agent DRL system can be constructed by using DQN algorithm instead of Q learning algorithm to train each agent individually (Liu et al., 2018). Tampuu et al. (Tampuu et al., 2017) used the above ideas to expand the framework of deep Q learning, dynamically adjusted the utility mode according to different goals, and proposed a DRL model in which multiple agents could cooperate and compete with each other. In the face of a class of reasoning missions that require multiple agents to communicate with each other, the DQN model cannot usually learn effective strategies. To solve this problem, Foerster et al. (Foerster et al., 2016) proposed a model called distributed deep loop Q network (DDRQN), which solved the problem of multi-agent communication and cooperation that can be observed in the state part.

The machine learning method based on Q learning is challenged by an inherent non-stationarity of the environment, while policy gradient suffers from a variance that increases as the number of agents grows. In recent years, some new approaches have been proposed, including deep reinforcement learning based on attention mechanism and deep reinforcement learning based on Actor-Critic framework. In 2017, Choi et al. (Choi et al., 2017) proposed a multi-focus attention network (MANet) method that can simulate human spatial extraction ability. This method first divides the low-level input into several segments representing local states. In 2017, Foerster et al. (Foerster et al., 2018) proposed a new multi-agent actor-critic method called counterfactual multi-agent (COMA) policy gradients based on the Actor-Critic. COMA uses a centralized critic to estimate the Q-function and decentralized actors to optimize the agents' policies. To address the challenges of multi-agent credit assignment, it uses a counterfactual baseline that marginalizes out a single agent's action, while keeping the other agents' actions fixed. Lowe R et al. (Lowe et al., 2017) present an adaptation of actor-critic methods named MADDPG (Multi-agent Deep Deterministic Policy Gradient) that considers action policies of other agents and is able to successfully learn policies that require complex multiagent coordination to maximizes the global utility. They also introduce a training regimen utilizing an ensemble of policies for each agent that weakened overfitting and leads to more robust multi-agent policies. They show the success of their approach compared to existing methods in cooperative as well as competitive scenario.

In the environment of large-scale multi-agent cooperation, it is difficult for agents to differentiate valuable information that helps cooperative decision making from globally shared information. The predefined communication architectures, on the other hand, restrict communication among agents and thus restrain potential cooperation. To tackle these difficulties, in 2018, Jiang et al. (Jiang and Lu, 2018) proposed an attentional communication model that learns when communication is needed and how to integrate shared information for cooperative decision making.

In recent years, MADDPG has been successfully used in the field of traffic control. Wu et al. (Wu, Jiang and Zhang, 2020) proposes a distributed conflict-free Cooperation MADDPG (CoMADDPG), for multiple connected vehicles at unsignalized intersection. CoMADDPG can reduce average travel time by 39.28% compared with the other optimization-based methods, which indicates that CoMADDPG has an excellent prospect in dealing with the scenario of unsignalized intersection control.

3. MULTI-AGENT COOPERATIVE COMPETITION AND DECEPTION TACTION

3.1 Multi-agent centralized training framework

In this paper, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is adopted to train all agents in the competitive scenario, so that both attacking and defending team can gradually learn cooperative tactics in the training process. Based on the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2015), MADDPG algorithm adds the policy information of other agents, including the state, action and reward value of each agent. Similar to DDPG, the MADDPG algorithm also uses the Actor-Critic algorithm framework (Sutton and Barto, 2018), as shown in Figure 1.

During training, the policy network in the Actor selects an action according to the observation information of the current agent, and the action is executed in the environment to get the new state and return value. After that, each agent's state, action, reward value, and new state form a sample to be saved to the experience replay buffer.

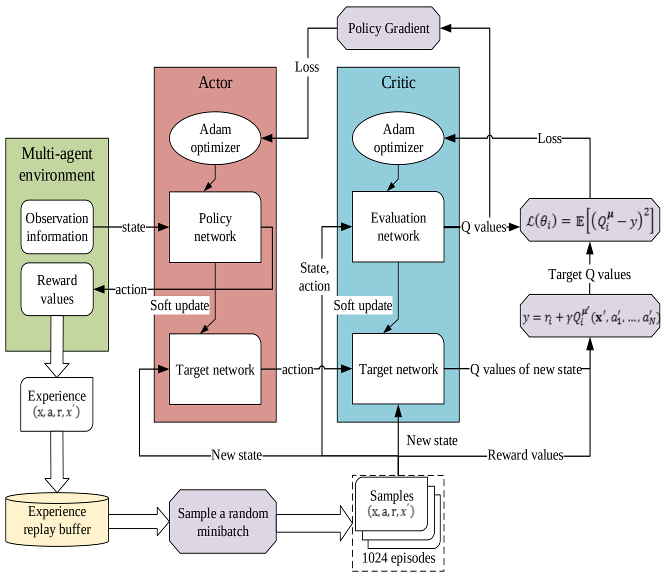

Fig. 1. Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm.

This method randomly collects 1024 samples from the experience replay buffer, and sends the new states to the target network in the Actor. Then the new actions output from the target network in the Actor are passed to the target network in the Critic. The target network in the Critic obtains the new states from the samples and the actions passed by the Actor, and outputs Q values of the new states. The target Q values can be calculated according to the Bellman equation using the reward values obtained from samples and the Q values of new states output from the target network in the Critic. The evaluation network in the Critic outputs the estimated centralized Q values after it gets the sampled states and actions.

In the Critic, the mean squares of the differences between the estimated centralized Q values and the target Q values are used as the loss value to train the evaluation network and update its parameters. Based on the estimated centralized Q values, the strategy gradient (Silver et al., 2014) can be calculated to train and update the policy network in the Actor.

The parameters of target networks are updated by superposing the parameters of policy network or evaluation network and the parameters of target network proportionally. In order to ensure the stability of the learning process, the MADDPG algorithm updates the parameters of target network using the soft update method similar to DDPG. It updates network parameters as follows:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \tag{1}$$

with $\tau \ll 1$, where θ_i' is the parameter of target network, and θ_i is the parameter of policy network or evaluation network.

This decision-making algorithm is a decentralized execution, centralized training approach which is not linked to the graphical interface until at the testing (decentralized executing) phase. At the training phase, the algorithm and data of dynamic multi-agent environment runs in the background. Then at the testing phase, every agent's action is given by policy network and the state of the multi-agent environment is updated per 0.1 second and mapped synchronously to the graphical interface.

3.2 Multi-Agent Deep Deterministic Policy Gradient

The MADDPG (Multi-Agent Deep Deterministic Policy Gradient) algorithm is a general-purpose multi-agent deep reinforcement learning algorithm based on policy gradient method (Lowe et al., 2017). MADDPG does not assume a differentiable model of the environment dynamics or any particular structure on the communication method between agents, and it is applicable not only to cooperative interaction but to competitive or mixed interaction. MADDPG adopts the framework of centralized training with decentralized execution to extend the Actor-Critic methods where the Critic is augmented with extra information about the policies of other agents, while the Actor only has access to local information. After training is completed, only the local actors are used at execution phase, acting in a decentralized manner.

Consider a mission scenario with N agents with policies parameterized by $\boldsymbol{\theta} = \{\theta_1, \cdots, \theta_N\}$, and let $\boldsymbol{\mu} = \{\mu_1, \cdots, \mu_N\}$ be the set of all agent policies. Then the gradient of the expected return $J(\theta_i) = \mathbb{E}[R_i]$ for agent *i* can be written as:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}[\nabla_{\theta_i}\boldsymbol{\mu}_i(a_i|o_i)\nabla_{a_i}Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \dots, a_N)|_{a_i=\boldsymbol{\mu}_i(o_i)}] \tag{2}$$

where $Q_i^{\mu}(\mathbf{x}, a_1, \dots, a_N)$ is a centralized action-value function that takes as input actions of all agents, $a_1, \cdots, a_N$, in addition to some state information $\mathbf{x}$, and outputs the Q-value for agent i. In the simplest case, $\mathbf{x}$ could consist of the observations of all agents. The experience replay buffer $\mathcal{D}$ contains the tuples $(\mathbf{x}, \mathbf{x}', a_1, \cdots, a_N, r_1, \cdots, r_N)$, recording experiences of all agents. The parameters of policy network are updated using the policy gradient from formula (2) as the loss by Adam optimizer.

In the Actor-Critic method, the model of action-value function Q_i^{μ} is the evaluation network. The Adam optimizer is used to update the network parameters in the training process. The loss function is as follows:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}\left[\left(Q_i^{\mu}(\mathbf{x}, a_1, \dots, a_N) - y\right)^2\right] \tag{3}$$

where y represents the target Q value calculated by the Bellman equation, and y can be written as:

$$y = r_i + \gamma Q_i^{\mu'}(\mathbf{x}', a_1', \dots, a_N')|_{a_j' = \mu_j'(o_j)} \tag{4}$$

where $\boldsymbol{\mu}' = \{\boldsymbol{\mu}_{\theta_1'}, \cdots, \boldsymbol{\mu}_{\theta_N'}\}$ represents the policy of all agents in the new states, $Q^{\mu'}$ is the Q value output from the target network in the Critic, and γ is a discount factor.

3.3 Network model

This paper adopts a fully connected network to train multi-agent policy. The network model consists of 1 input layer, 2 hidden layers, and 1 output layer, each of which contains 64 units. The ReLU (Rectified Linear Unit) is used as activation function.

4. MULTI-AGENT COOPERATIVE COMPETITION AND REWARD FUNCTION

This paper focuses on the application of deep reinforcement learning algorithm and functionalized return value. Through centralized training, the agents that undertakes the deception mission and the agents that undertakes the attack mission realize the multi-agent, multi-mission, multi-target collaborative tactics. In the battlefield environment, there are often many mobile defence units near important targets. When a target encounters an attack or is about to encounter an attack, the defending units of the defender can move in time and intercept the incoming units. In order to safely break through the defence of the defending team and reach the targets, the attacking units of the attacking team need to have good tactical coordination. In this paper and experiments, we assume that the defending team has two types of agents, one is the commander with global insight, and the other is the ordinary guardian who can only observe the barrier-free area. The attacking team also has two types of agents, one is the attacker who undertakes the collaborative attack mission, and the other is the deceiver who actively approaches and attracts the defending units to cover attackers.

In this paper, the reward function of the defending agents consists of two parts: one is the punishment of the distance between the defending team and the nearest unit of the attacking team; the other part is the reward when the defending unit intercepts the attacking unit with a constant value. Thus, the reward function of the defending agents can be expressed as:

$$R_i^{defense} = \sum_{t=1}^{T}\left[r_{i1}^{defense}(t) + r_{i2}^{defense}(t)\right] \tag{5}$$

where $i = 1, 2, \cdots, m$ indicates the i^{th} defending agent, t indicates the t^{th} step in each episode. The first part of the above equation that represents punishment can be written as:

$$r_{i1}^{defense}(t) = a * \min_j\{d_{i,j}^{defense-attack}\}$$

where a is a negative constant, $j = 1, 2, \cdots, n$ indicates the j^{th} attacking agent, and $d^{defense-attack}$ is the distance between the attacking agent and the defending agent. The second part of the formula (5) that represents reward can be written as:

$$r_{i2}^{defense}(t) = \sum_i \sum_j g_{i,j}$$

With

$$g_{i,j} = \begin{cases} c, & if\ d_{i,j}^{defense-attack} < l + l' \\ 0, & if\ d_{i,j}^{defense-attack} \geq l + l' \end{cases}$$

where c is a positive constant, l and l' represent the radius of the attacking agents and defending agents respectively.

As the attackers and the deceivers in attacking side have different tactical missions, the settings of their reward functions are also different. The attackers' reward function consists of five parts: one is the distance from the attacker to the nearest target as punishment; the second is a constant value as the reward for the attacker to reach the target; the third is the standard deviation of the distance between each attacker and its nearest target that is used as punishment to train agents to attack synchronously; the fourth is to use a constant value as punishment when the attacker is intercepted by the defending agents; the fifth is the distance between the attacker and its nearest defending agent. Therefore, the attackers' reward function can be written as follows:

$$R_j^{attack} = \sum_{t=1}^{T}\left[r_{j1}^{attack}(t) + r_{j2}^{attack}(t) + r_{j3}^{attack}(t) + r_{j4}^{attack}(t) + r_{j5}^{attack}(t)\right] \tag{6}$$

The first part of the above equation can be written as:

$$r_{j1}^{attack}(t) = b * \min_k\{d_{k,j}^{target-attack}\}$$

where b is a negative constant, $k = 1, 2, \cdots, o$ indicates the k^{th} target, and $d^{target-attack}$ is the distance between the target and the attacker. The second part of formula (6) can be written as:

$$r_{j2}^{attack}(t) = \begin{cases} c', & if\ d_j^{attack-target} < l' + l'' \\ 0, & if\ d_j^{attack-target} \geq l' + l'' \end{cases}$$

where c' is a positive constant, l'' is the radius of the target. The third part of formula (6) can be written as:

$$r_{j3}^{attack}(t) = b' * std.\left\{\min_k\{d_{j,k}^{attack-target}\}\middle| j\right\}$$

where b' is a negative constant, $d^{attack-target}$ is the distance between the attacker and the target, and $std.$ represents the standard deviation of elements in the collection. The fourth part of formula (6) can be written as:

$$r_{j4}^{attack}(t) = \sum_{i=1}^{m} g'_{j,i}$$

with

$$g'_{j,i} = \begin{cases} c'', & if\ d_{i,j}^{defense-attack} < l + l' \\ 0\ , & if\ d_{i,j}^{defense-attack} \geq l + l' \end{cases}$$

where c'' is a negative constant. The fifth part of formula (6) can be written as:

$$r_{j5}^{attack}(t) = b'' * \min_i\{d_{i,j}^{defense-attack}\}$$

where b'' is a positive constant.

The deceivers' reward function consists of three parts: one is a Gauss-Quadratic mixing function about the distance between the deceiver and defending agent as a reward to train the deceiver to learn to approach defending agent actively and keep the distance from defending agent; the second is a constant as punishment when the deceiver is contacted by defending agent; the third is the distance between the deceiver and attacker as a reward to train the deceiver to learn to evade attackers in the mission. Therefore, deceivers' reward function can be written as follows:

$$R_{j'}^{deception} = \sum_{t=1}^{T}[r_{j'1}^{deception}(t) + r_{j'2}^{deception}(t) + r_{j'3}^{deception}(t)]$$

(7)

the first part of above equation can be written as:

$$r_{j'1}^{deception}(t) = \sum_{i=1}^{m}\left\{h_1 * e^{-\alpha_1 * \left(d_{i,j'}^{defense-deception} - f_1\right)^2} + \alpha_2 * \left(h_2 - \left(d_{i,j'}^{defense-deception} - f_2\right)^2\right)\right\}$$

where $d^{defense-deception}$ is the distance between the defending agent and the deceiver, $f_1, f_2, h_1, h_2, \alpha_1, \alpha_2$ are all positive constant parameters. The second part of formula (7) can be written as:

$$r_{j'2}^{deception}(t) = \sum_{i=1}^{m} g''_{j',i}$$

with

$$g''_{j',i} = \begin{cases} c''', & if\ d_{i,j'}^{defense-deception} < l + l' \\ 0\ , & if\ d_{i,j'}^{defense-deception} \geq l + l' \end{cases}$$

where c''' is a negative constant. The third part of formula (7) can be written as:

$$r_{j'3}^{deception}(t) = b''' * \min_j\{d_{j',j}^{deception-attack}\}$$

where b''' is a positive constant, $d^{deception-attack}$ is the distance between the deceiver and the attacker.

In addition to the targets and agents in both attacking and defending team, obstacles and forests are also set up in the mission scenario in this paper. The obstacles are impassable, agents can only detour after touching. The forest can cover the observation ability of the agent. When agents are outside the forest, they cannot observe the agents in the forest. Similarly, when agents are in the forest, they cannot observe other agents outside the forest. The commander in defending agents is the only agent with global observation capability, and all the agents in the scenario can be observed regardless of whether the commander is in forests or outside forests.

The setting of obstacles and forests can be used to simulate some dangerous obstacles and low-detection areas in complex battlefield environment, respectively. Agents are trained to avoid obstacles and move purposefully into and out of forests. The attacking agents can learn to use the forests intelligently to avoid being intercepted, and the defending agents can learn to cooperate with each other to deal with the attacking agents in forests.

5. EXPERIMENTS

5.1 The experiment of cooperative competition and deception tactics

In the computer experiments, the two types of attacking agents can be trained to learn policy to maximize the global utility without particular communication method. Thus, the agents can cooperate on the attacking missions.

The multi-agent environment in this paper is a 600*600 RGB image modelled using OpenAI Gym (a toolkit for developing and comparing reinforcement learning algorithms). At the beginning of each episode, locations of all agents, targets, obstacles and forests are set randomly in a two-dimensional plane. Agent i ($i=1, 2, \ldots, N$) moves in the environment with speed $\boldsymbol{v}_i(v_i^x, v_i^y)$. The value of agent's speed is determined by the action outputted from its policy network. Agents' position at the current step is the sum of the previous position and the current speed multiplied by $dt=0.1$s. In this mission, any attackers in the attacking team covering the targets will score. On the other hand, the defending team tries to intercept the attacking team to prevent them from reaching the targets. Each episode ends after 25th step no matter which team wins.

In order to measure the performance of the multi-agent cooperative tactics quantitatively, three indicators are used to describe the process of attackers and deceivers learning cooperative tactics, including the total number of times attacking agents were intercepted by defending agents, the number of times the attackers were intercepted, and the number of times the attackers reached the targets for each 1000 episodes during the training process. The experimental environments will be reset at the beginning of the new episode, randomly generating new locations for targets, agents, obstacles and forests. There are three defending agents, one of which is the commander and others are ordinary guardians, and there are five attacking agents, two of which are attackers and three are deceivers. In addition, there are 2 targets, 1 obstacle, and 2 forests in the experimental environments, as shown in Figure 2.

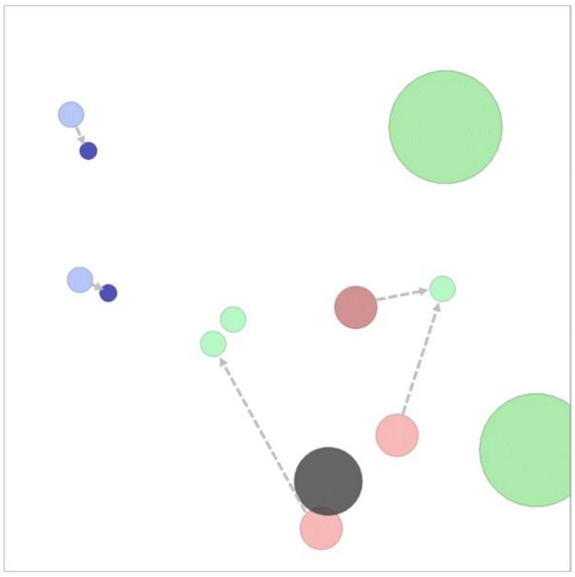

Fig. 2. Multi-agent cooperative attack and deception mission scenario. The larger red circles indicate the defending agents, among which the dark red one is the commander; the smaller blue and green circles are the attackers and deceivers in attacking agents respectively. The dark blue dots indicate the targets, black circles indicate obstacle, and green large circles indicate forests.

The experiment results show that multi-agent cooperative attack and deception mission in the above experimental environments can achieve satisfied results after about 80,000 times of training. The experiment results after 81,000 episodes of training are given below. It can be observed that the three defending agents are completely confused by the deceivers in the attacking team, neither defensing the attackers nor defending and staying near the targets. The attackers successfully avoid the defending agents and approach the targets in the mission of this episode.

Figure 3 shows the reward value curves of all agents in the training process. In the first 10,000 episodes, with the various agents moving from blind action to gradually learning their own tasks, the reward values rise rapidly. Then, as the opponents' tactical level improved, their own reward values decrease significantly. In the subsequent training process, the cooperative tactics of both the attack and defence sides have gradually matured, and the reward values of all types of agents have increased. After training up to 70,000 episodes, the attacking agents gain a comparative advantage and their reward values continue to rise, while the return values of the defending agent decrease significantly.

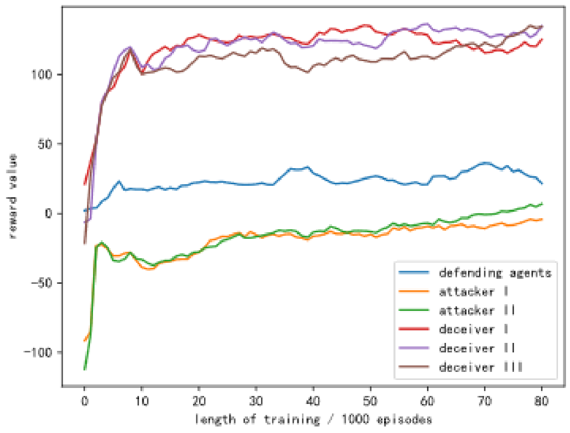

Fig. 3. The reward values of agents for the cooperative attack and deception mission during 81,000 training episodes. The ordinate indicates the sum of the reward values per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

Figure 4 and Figure 5 show the number of times all the attacking agents are intercepted by the defending agents and the number of times attackers are intercepted per 1,000 episodes during the training process respectively. It can be observed that as the tactical ability of the defending agents continue to enhance after the start of training, the number of times attacking agents are intercepted increases rapidly, reaching a maximum of 14,724 times per thousand episodes.

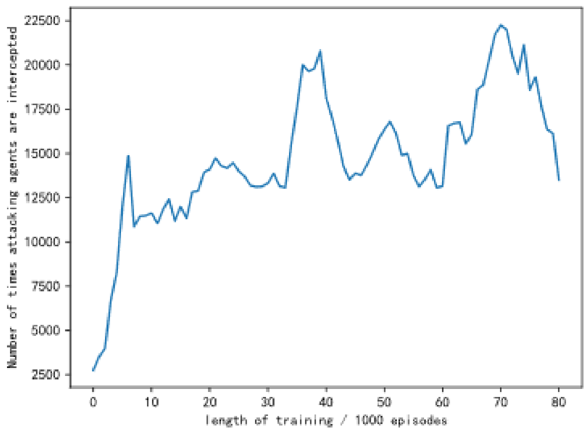

Fig. 4. The number of times attacking agents are intercepted by defending agents during the 81,000 training episodes. The ordinate indicates the sum of the number of interceptions per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

However, after only a few thousand episodes of training, the attacking agents soon learn to avoid being intercepted by the defending agents, and the number of interceptions decreases significantly. After about 30,000 episodes of training, the number of interceptions fluctuates between 12,000 and 22,500 per thousand episodes as the competition between the defending and attacking team becomes more intense. The number of times attackers are intercepted increases rapidly during the initial stage of training, and then decreases significantly. Between 29,000 and 35,000 episodes, the number of times attackers are intercepted is only about one-tenth of the number of times all attacking agents are intercepted, indicating that most of the defending agents are induced by the deceivers. However, as can be seen from Figure 6, the attacker can only reach the target about once per episode on average, which means that the attackers have not learned to attack the targets cooperatively.

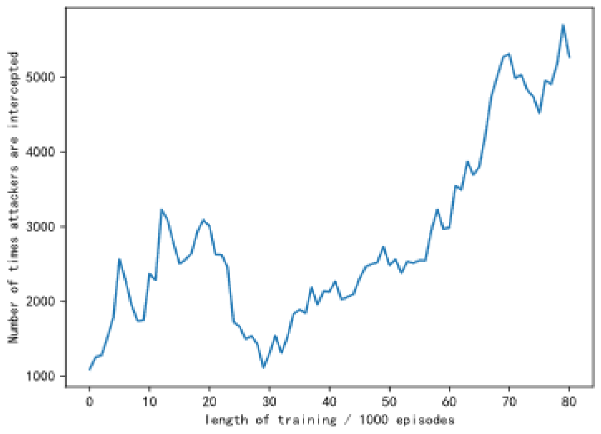

Fig. 5. The number of times attackers are intercepted by defending agents during the 81,000 training episodes. The ordinate indicates the sum of the number of interceptions per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

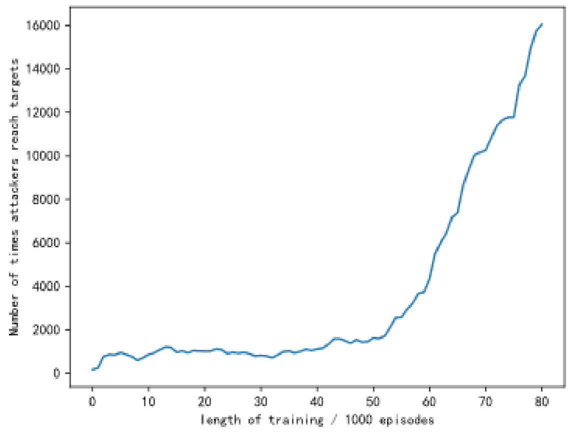

Fig. 6. The number of times the attackers reach the targets during 81,000 episodes of training. The ordinate indicates the sum of the number of times attackers reach targets per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

As shown in Figure 5 and Figure 6, the number of times the attackers reach the targets is significantly more than the number of interceptions after about 70,000 episodes of training, indicating that the cooperative attack tactics of the attacking agents are effective at this time. When training to 80,001 to 81,000 episodes, the number of times attackers reach targets peaks at 16,040 per 1,000 episodes. As the tactical ability of the defending agents continue to enhance at this time, the number of times attackers are intercepted also rises rapidly. If training continues, the attackers' tactics will tend to avoid the risk of being intercepted and even give up many opportunities to get close to targets. Therefore, the training process is terminated at 81,000th episodes. Fig. 1 meant 16000 A/m or 0.016 A/m. Figure labels should be legible, approximately 8 to 12 point type.

5.2 *The controlled experiment*

In order to reveal the role of deceivers more directly and verify the effect of deception tactics, the controlled experiment has been carried out. In the controlled experiment, the mission and the experimental environment are exactly the same as the previous experiment, the only difference is the setting of the agents' reward function. The deceivers' action policy is simply to avoid being intercepted by the defending agents, instead of actively approaching defending agents at appropriate distance. In the following controlled experiment, agents are also trained 81,000 episodes. The results are shown below.

In Figure 7, it can be observed that the attackers are surrounded by defending agents, while the attacking agents that originally played the roles of deceivers can only elude the defending agents and can no longer effectively deceive defending agents. Figure 8 shows the reward value curves of agents in the training process. It can be observed that after training to 65,000 episodes, the reward values of defending agents are rapidly increased, while the reward value of an attacker in attacking agents decreases significantly. This shows that the tactical mature defending agents can distinguish different types of attacking agents. After the attackers lose the cover of the deceivers, they are more likely to be intercepted by defending agents.

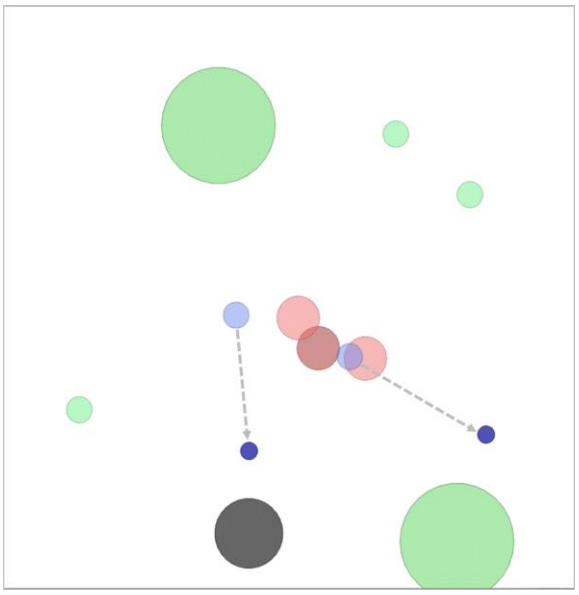

Fig. 7. The controlled experimental scenario without deception tactics. The larger red circles indicate the defending agents, among which the dark red one is the commander; the smaller blue circles are the attackers, and the small green circles are deceivers without deception tactics. The dark blue dots indicate the targets, black circles indicate obstacle, and green large circles indicate forests.

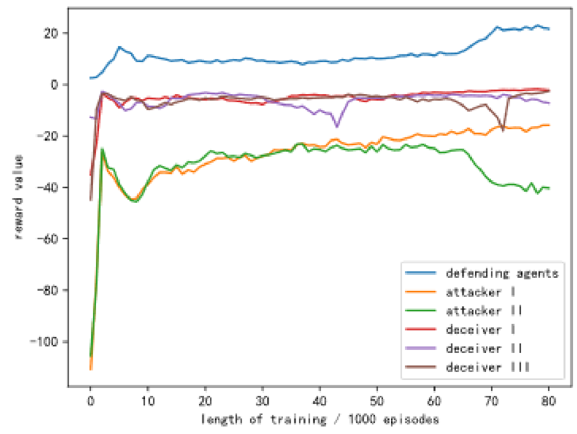

Fig. 8. The reward values of agents in the controlled experiment. The ordinate indicates the sum of the reward values per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

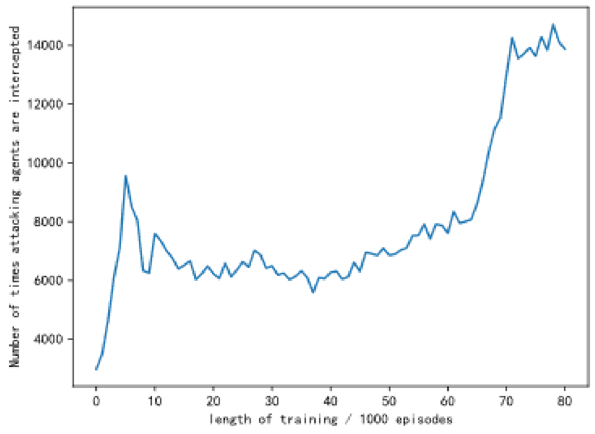

Fig. 9. The number of times attacking agents are intercepted by defending agents during the 81,000 training episodes. The ordinate indicates the sum of the number of interceptions per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

Figure 9 and Figure 10 respectively show the number of times all attacking agents are intercepted and the number of times attackers are intercepted during the training of the controlled experiment. It is easy to see that the proportion of the number of times attackers are intercepted accounts for the number of times all attacking agents are intercepted is significantly higher than that of the above situation with deception tactics. When training to around 70,000 episodes, the number of times attackers are intercepted quickly rises to more than 10,000 times. The attacking agents intercepted by defending agents at this time are almost all attackers. In other words, the three attacking agents without deception policy in the controlled experiment can hardly cover the attackers.

Figure 11 shows the number of times the attackers reach targets during the training process in the controlled experiment, which shows a nearly linear increase with the number of training episodes. When train to 79,001 to 80,000 episodes, the attackers reach the targets 5,946 times per thousand episodes. Compared with above scenarios with deception tactics, the number of times of reaching targets in the controlled experiment decreased by 62.5%. Since the number of times attacker are intercepted has far exceeded the number of times of reaching targets, most attacks are actually failed.

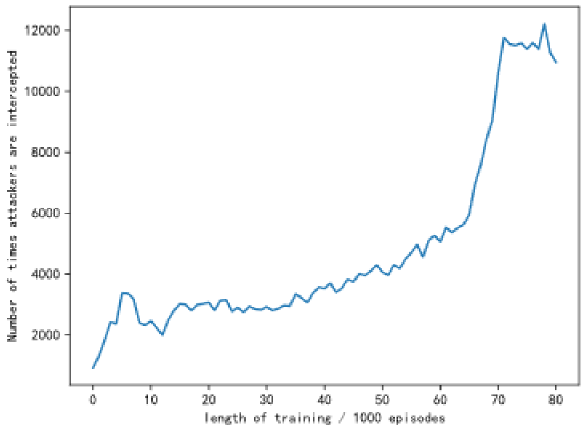

Fig. 10. The number of times attackers are intercepted by defending agents during the 81,000 training episodes. The ordinate indicates the sum of the number of interceptions per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

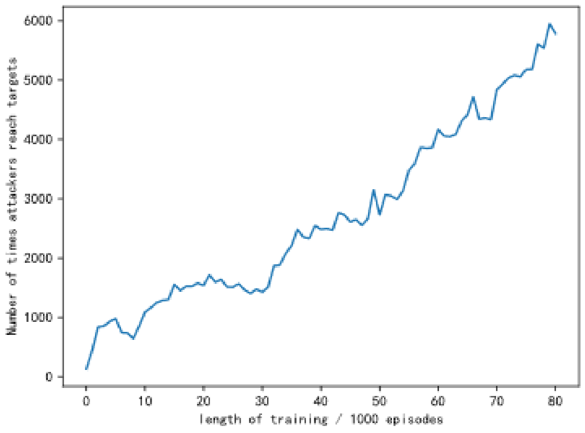

Fig. 11. The number of times the attackers reach the targets during 81,000 episodes of training. The ordinate indicates the sum of the number of times attackers reach targets per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

The results of the above controlled experiments reflect that the deception tactics play an important role in multi-agent cooperative competition. By actively approaching the defending agents and keeping an appropriate distance, the deceivers can confuse and induce opponents, thus covering and helping the attackers to safely complete their attacking mission.

5.3 The effect of training episodes on multi-agent policy

In the training process, the competition situation and agents' tactical level will directly affect the opponents' policy optimization process. To achieve the goal of the cooperative mission in this paper, the two types of attacking agents should learn to cooperate with each other to perform their tasks. On the other hand, the two types of defending agents should learn to work together to besiege and intercept attacking agents, so as to maximize the reward value of all of them.

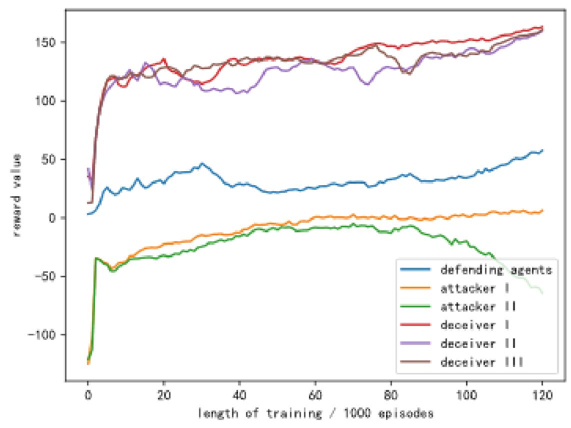

Fig. 12. The reward values of agents for the cooperative attack and deception mission during 121,000 training episodes. The ordinate indicates the sum of the reward values per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

Figure 12 to Figure 15 show the results of three defending agents against two attackers and three deceivers during 121,000 episodes of training. As the number of training episodes increases, it can be observed that the tactics of the defending agents are more and more tend to work together to besiege a single attacking agent, and the total number of interceptions and the number of times the attackers are intercepted are rapidly increased, as shown in Figures 12, 13 and 14. After training to about 90,000 episodes, this trend will force the attackers to adopt a more conservative policy of avoiding being intercepted rather than getting close to targets, as shown in figure 15.

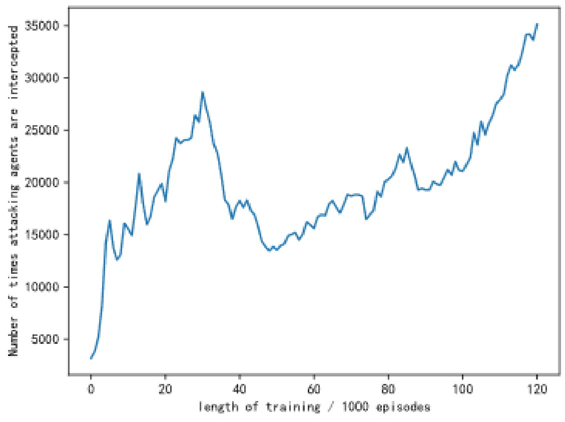

Fig. 13. The number of times attacking agents are intercepted by defending agents during the 121,000 training episodes. The ordinate indicates the sum of the number of interceptions per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

The above results show that if the agents encounter powerful opponents and accumulates too much failure experience in the training process, the agents' policy will be more and more conservative. As the training episodes increases, the attacking agents will gradually give up the opportunity to take the risk to complete their mission, which will eventually lead to the agents learning passive escape policy. This is obviously inconsistent with the expected goal of the cooperative competitive mission. Therefore, the number of training episodes should be reasonably set according to the requirements of the mission and the data of experiments.

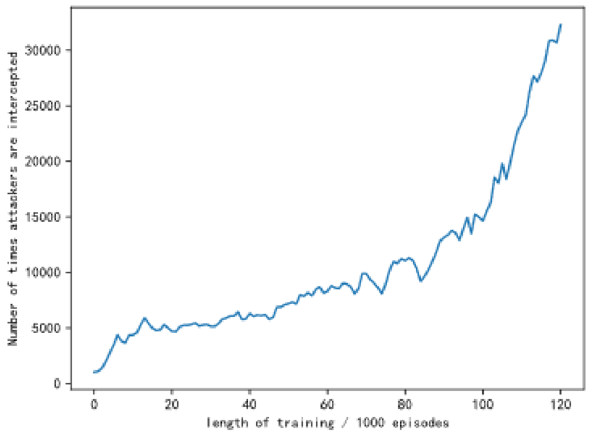

Fig. 14. The number of times attackers are intercepted by defending agents during the 121,000 training episodes. The ordinate indicates the sum of the number of interceptions per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

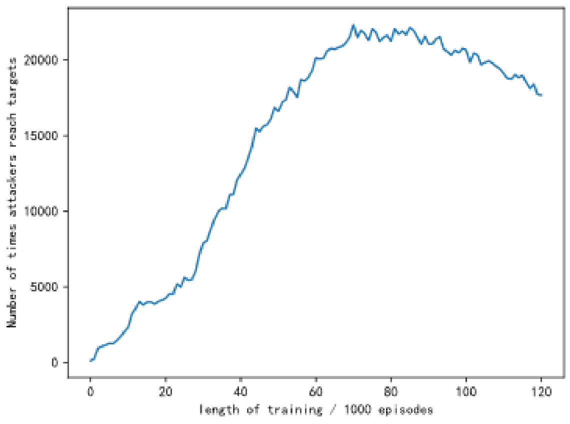

Fig. 15. The number of times the attackers reach the targets during 121,000 episodes of training. The ordinate indicates the sum of the number of times attackers reach targets per 1,000 training episodes; the abscissa indicates the number of training episodes in thousands of episodes.

By comparing the data of above experiments where three defending agents (including one commander and two guardians) and five attacking agents (including two attackers and three deceivers) are set up, it can be observed that the attacking agents can learn relatively satisfied tactical policy after about 80,000 episodes of training.

5.4 The effect of the number of deceivers on cooperative attacking

In the cooperative competition mission in this paper, the deceivers play important roles in inducing defending agents and covering attackers to reach targets safely. This paper researches the computer experiment results of cooperative attacking and deception tactics with variable number of deceivers by tuning the number of deceivers. It is found that the number of deceivers has a significant effect on the performance of deception tactics. The length of the training is set to 81,000 episodes in the experiments, the number of deceivers ranges from 0 to 4, and other settings in the experimental environment are consistent with the previous experiments. Table 1 statistics the experiments data of the last 1,000 episodes of the training process, i.e., the total number of interceptions, the number of times attackers are intercepted and the number of times attackers reach targets in the 80,001-81,000 episodes. The proportion of the number of times attackers are intercepted accounts for the total number of interceptions is also listed in Table 1.

Table 1. Statistical results of training data for the 80001-81000 episodes. N_d is the number of deceivers, T is the total number of interceptions, T_a is the number of times attackers are intercepted, T_r is the number of times attackers reach targets.

N_d	T	T_a	T_a /T	T_r
0	9666	9666	100%	1446
1	10938	7818	71.48%	3950
2	17679	8970	50.74%	10065
3	13512	5271	39.00%	16040
4	17793	4929	27.70%	5717

From the data in Table 1, it can be observed that the number of times attackers are intercepted shows a roughly downward trend with the increase of the number of deceivers. But when the number of deceivers increases to two, the number of times attackers are intercepted exceeded that of only one deceiver, which is caused by the inherent randomness of reinforcement learning. This random disturbance is reflected in the statistical data, which inevitably caused the training results to deviate from the overall trend to some extent. However, the proportion of the number of times attackers are intercepted accounts for the total number of interceptions decreases monotonously as the number of deceivers increases. This demonstrates that the more deceivers there are, the more defending agents they can induce, and the better the attackers' chances of surviving.

This paper tests the policy network model obtained after 81,000 episodes of training. The testing environment is set as the same as the training environment. The length of test is set to 1,000 episodes, and the test data are listed in Table 2.

In Table 2, as the number of deceivers increases, the number of times attackers are intercepted monotonically decreases, and the proportion of the number of times attackers are intercepted accounts for the total number of interceptions also monotonically decreases. As shown in Table 1 and Table 2, when the number of deceivers increase from 0 to 4 in training and testing, the number of times attackers reach targets increases as the number of deceivers increases. This shows that, as more deceivers join, the attacking agents get a higher winning percentage.

Table 2. Statistical results of test data of 1000 episodes. N_d is the number of deceivers, T is the total number of interceptions, T_a is the number of times attackers are intercepted, T_r is the number of times attackers reach targets.

N_d	T	T_a	T_a /T	T_r
0	6030	6030	100%	1287
1	7515	5415	72.06%	3067
2	11436	5259	45.99%	5856
3	10365	4254	41.04%	10810
4	14865	3999	26.90%	3744

However, when the number of deceivers increases to more than 4, the previous training times are not enough for attackers to learn better policy in the face of more complex situations, so that the number of times attackers reach targets decreases compare to previous situation with less agents.

6. CONCLUSIONS

This paper proposes a method to realize multi-agent cooperative competition and deception tactics by designing functional reward values for multi-agent based on MADDPG algorithm, and verifies the role of deception tactics through experiments. We set up multiple types of agents to share various missions and obtain the cooperative attacking and deception policy modes through computer experiments to realize the cooperation of multi-mission among multiple agents. This paper also researches the effect of training episodes on multi-agent learning policy in the experiments. The experiments data shows that too little training is not enough for agents to learn satisfied policy, while too much training may make agents learn overly aggressive or conservative policy, so the number of training episodes should be reasonably set according to the mission requirements and test results. Otherwise, this paper analyses the effect of the number of deceivers on the performance of deception tactics by tuning the number of deceivers in the experiments. The training and testing results show that increasing the number of deceivers in the mission can significantly improve the performance of deception tactics by covering attackers to reach targets safely from the interception of defending agents. But the computational complexity of the multi-agent environments will rise with the increase of the number of agents, and more training episodes are needed to ensure a satisfied learning effect.

MARL has been applied to a variety of problem domains, mostly in simulation but also in some real-life tasks. Simulated domains dominate for two reasons. The first reason it is easier to understand and to derive insight from results in simpler domains. The second reason is that scalability and robustness to imperfect observations are necessary in real-life tasks, and few MARL algorithms exhibit these properties. In real-life applications, more direct derivations of single-agent RL are preferred (Buşoniu et al., 2010). Although deep reinforcement learning method has made a significant breakthrough in solving cooperative mission of multi-agent, there are still some problems that have not been solved well. First, the number of training episodes relies on human experience to set in the existing multi-agent algorithms based on deep reinforcement learning. It is difficult to define how many episodes of training can obtain a model that best conforms to the expected result before a number of training and testing. Second, the existing deep reinforcement learning algorithms continuously optimize the agents' policy in the direction of maximizing the Q values in training process by updating network parameters. Training a type of agents in this way can only perform one type of mission, unable to balance multi-mission goals and flexibly adjust policy according to the dynamic environment, and the problem of multi-agent multi-mission cooperation can only be solved by assigning different missions to different types of agents. At last, as the deep reinforcement learning method has a strong inherent randomness, the experiments results are difficult to be reproduced. The random multi-agent dynamic environment and experience sampling process will cause the policy learning process to be unstable and make the experimental data fluctuate in a large range. It is a practical problem in the cluster control of unmanned intelligent system, which is worth further exploring.

REFERENCES

Arulkumaran K., Deisenroth M., Brundage M. and Others (2017). A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine*, 34, 26–38.

Buşoniu, L., Babuška, R., & Schutter, B. D. (2010). Multi-agent Reinforcement Learning: *An Overview*, 183–221.

Choi J., Lee B., and Zhang B. (2017). Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

Conitzer, V., & Sandholm, T. (2007). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1), 23–43.

Foerster J., Assael I., de Freitas N. and Others (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145.

Foerster J., Farquhar G., Afouras T. and Others (2018). Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

François-Lavet V., Henderson P., Islam R. and Others (2018). An introduction to deep reinforcement learning. Foundations and Trends ® in *Machine Learning*, 11 (3-4), 219–354.

Hastie T., Tibshirani R., and Friedman J. (2009). The elements of statistical learning: data mining, inference, and prediction. *Springer*.

Hernandez-Leal P., Kartal B., and Taylor M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33 (6), 750–797.

Hu, J., & Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4, 1039–1069.

Jiang J., and Lu Z. (2018). Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pp. 7254–7264.

Kapoor, S. (2018). Multi-Agent Reinforcement Learning: A Report on Challenges and Approaches. *arXiv Preprint ArXiv*:1807.09427.

Khan, A., Zhang, C., Lee, D., Kumar, V., & Ribeiro, A. (2018). Scalable Centralized Deep Multi-Agent Reinforcement Learning via Policy Gradients. *ArXiv, abs/1805.08776.*

Krizhevsky A., Sutskever I., and Hinton G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105.

LeCun Y., Bengio Y., and Hinton G. (2015). *Deep learning. nature*, 521 (7553), 436–444.

Lillicrap T., Hunt J., Pritzel A. and Others (2015). Continuous control with deep reinforcement learning. In *arXiv preprint*, Vol. 1509.02971.

Liu Q., Zhai J., Zhang Z. and Others (2018). A survey on deep reinforcement learning. *Chinese Journal of Computers*, 40, 1–27.

Lowe R., Wu Y., Tamar A. and Others (2017). Multi-agent actor-critic for mixed cooperative competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390.

OpenAI (2018). Openai five. In *https: //blog.openai.com/ openai-five/.*

Silver D., Huang A., Maddison C. and Others (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529, 484–489.

Silver D., Hubert T., Schrittwieser J. and Others (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. In *arXiv preprint*, Vol. 1712.01815.

Silver D., Hubert T., Schrittwieser J. and Others (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362, 1140–1144.

Silver D., Lever G., Heess N. and Others (2014). Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395.

Silver D., Schrittwieser J., Simonyan K. and Others (2017). Mastering the game of go without human knowledge. *Nature*, 550, 354–359.

Sutton R., and Barto A. (2018). Reinforcement learning: An introduction. *MIT press*.

Tampuu A., Matiisen T., Kodelja D. and Others (2017). Multiagent cooperation and competition with deep reinforcement learning. *Plos One*, 12(4), e0172395.

Vinyals O., Babuschkin I., Czarnecki W. and Others (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 350–354.

Wu, T., Jiang, M., & Zhang, L. (2020). Cooperative Multiagent Deep Deterministic Policy Gradient (CoMADDPG) for Intelligent Connected Transportation with Unsignalized Intersection. *Mathematical Problems in Engineering*, 2020, 1–12.