Applied Intelligence (2023) 53:7876–7891
https://doi.org/10.1007/s10489-022-03821-9

Switching-aware multi-agent deep reinforcement learning for target interception

Dongyu Fan[1] · **Haikuo Shen**[1,2] · **Lijing Dong**[1,2,3]

Accepted: 25 May 2022 / Published online: 29 July 2022

Abstract

This paper investigates the multi-agent interception problem under switching topology based on deep reinforcement learning. Due to communication restrictions or network attacks, the connectivity between every two intercepting agents may change during the entire tracking process before the successful interception. That is, the topology of the multi-agent system is switched, which leads to a partial missing or dynamic jump of each agent's observation. To solve this issue, a novel multi-agent level-fusion actor-critic (MALFAC) approach is proposed with a direction assisted (DA) actor and a dimensional pyramid fusion (DPF) critic. Besides, an experience adviser (EA) function is added to the learning process of the actor. Furthermore, a reward factor is proposed to balance the relationship between individual reward and shared reward. Experimental results show that the proposed method performs better than recent algorithms in the multi-agent interception scenarios with switching topologies, which achieves the highest successful interception with the least average steps. The ablation study also verifies the effectiveness of the innovative components in the proposed method. The extensive experimental results demonstrate the scalability of our method in different scenarios.

Keywords Multi-agent system · Reinforcement learning · Deep learning · Switching topology

1 Introduction

Multi-agent systems (MASs) are widely used to analyze and simulate the environment with multiple intelligent entities in many fields [1, 2]. As for multi-agent tracking, there are many research results on the application of robots systems [3–5], unmanned vehicles [6–8], spacecraft systems [9–11], etc. Multi-agent interception problem is one of the extensions of multi-agent tracking systems, which refers to when an external intruder breaks into the area guarded by MASs, multiple agents communicate and interact with each other to track and intercept the intruder before it reaches the defense zone.

A key problem of the multi-agent interception system is to design a decentralization strategy only by using the relative information between agents so that the state of tracking agents and the target agent tends to be consistent and finally realize interception. In practical applications, due to limiting factors such as communication ability and sensing range, agents cannot observe the information of other agents at every moment. Besides, agents often encounter many attacks in the process of information transmission. More specifically, the connection between any two agents may be broken due to communication restrictions and reconnected when communication is restored. Thus, in the whole tracking process, the connectivity between two agents is often time-varying, which results in the connections among multiple agents usually under switching topologies.

This work was supported by the National Natural Science Foundation of China under Grant 61903022 and funded by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2019IRS11.

✉ Haikuo Shen
shenhk@bjtu.edu.cn

[1] School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing 100044, People's Republic of China

[2] Key Laboratory of Vehicle Advanced Manufacturing, Measuring and Control Technology (Beijing Jiaotong University), Ministry of Education, Beijing 100044, People's Republic of China

[3] Beijing Advanced Innovation Center for Intelligent Robots and Systems, Beijing Institute of Technology, Beijing 100081, China

In the traditional control area, the switching problem of MASs has been intensively studied from various perspectives [12–14]. By constraints of certain linear matrix inequality and average dwell time, decentralized designs with Lyapunov stability theory are devoted to the linear MASs with extension from fixed topology to switching topology [15]. In [16], an adaptive protocol is proposed for second-order heterogeneous nonlinear MASs, where the switching signals of each subsystem are not the same. Jiang et al. [17] provide protocols for linear time-varying multi-agent systems, which solves the leader-following consensus problems on the premise of taking the leader as the root node of the spanning tree in the switching topologies. However, these multi-agent approaches without learning usually take a hypothetical known model as the premise or are based on several assumptions, which are generally difficult to implement in practical applications. Multi-agent deep reinforcement learning (MADRL) is based on reinforcement learning (RL) [18], which enables agents to learn to implement tasks even if the model is unknown. MADRL has been attracting considerable interest, the most recent achievements of which have given rise to innovative solutions to practical challenges [19–21]. Most existing studies in MADRL assume that each agent shares accurate information by an ideal fixed topology. However, in practical scenarios, the information of some agents cannot always be observed, that is, the system is in switching topologies. Under the strict assumption of fixed topology, these methods of training neural networks of agents are not sufficient for scenarios under switching topologies.

Motivated by the aforementioned discussion, in this paper, we aim to address the challenging issue of time-varying communication topology in multi-agent interception systems by deep reinforcement learning. The main contributions of this work are as follows.

- The multi-agent interception system under switching topology is modeled based on the multi-agent particle environment (MPE). We add obstacles that can interfere with the communication between agents to truly simulate the switching topology caused by connection and disconnection of communication links during the tracking of the external intruding agent by multiple agents.
- To intercept the intruder by the multi-agent system, each tracking agent should not only take its own tracking of the intruder as the premise but also aim to achieve the overall interception. We design a weighted sum of independent reward and shared reward as the total reward of the intercepting agent with a reward influence factor. The effectiveness of the proposed reward is verified by comparative experiments.
- A loss function as an experience adviser (EA) is added to ensure the stability of policy learning. The function performs loss operation between the action output by the policy and the experience replay buffer, and back propagates it to the policy for learning.
- We propose a novel multi-agent deep reinforcement learning method named multi-agent level-fusion actor-critic (MALFAC) with a direction assisted (DA) actor and a dimensional pyramid fusion (DPF) critic. The experiment results show that the proposed method outperforms the comparison methods, which obtains the highest reward and success rate at the least interception steps. In addition, the ablation study demonstrates the effectiveness of each innovative component.

The remainder of this paper is summarized as follows. After giving a brief review of the related work to this study in Sections 2, and 3 provides the preliminary and problem statement. Section 4 presents the proposed method. Then, Section 5 shows the experiment results and the extensive study. At last, Section 6 concludes this work.

2 Related work

Reinforcement learning (RL) maximizes the cumulative reward by trial and error to solve sequential decision-making problems [18]. Different from supervised learning, RL does not tell the agent how to output the correct action but evaluates the action and modifies the policy according to the feedback signal. Deep reinforcement learning (DRL) has made amazing achievements by applying neural networks in traditional reinforcement learning [22–30]. As the cornerstone of deep reinforcement learning, the deep Q-learning network (DQN) [30] is a breakthrough in the overall training architecture by introducing a target network that can help the main network train stably, which is a value-based (VB) approach. Proximal policy optimization (PPO) [27] is one of the variants of the policy-based (PB) methods, which updates each policy by multiple epochs of stochastic gradient ascent. Deep deterministic policy gradient (DDPG) [28] adopts the actor-critic (AC) framework with four networks, which is a more mature training program for agents with continuous actions.

In practical applications, multiple agents need to make independent decisions based on their own observations to contribute to the overall goal most effectively [2, 31–33]. Since more than one agent interacts with the environment at the same time, it is usually unclear whether the action of a specific agent is generally positive or negative. In addition, the action of one agent is affected by the action of another agent, which makes policy learning

more challenging. Multi-agent deep reinforcement learning (MADRL) is an effective way to solve complex multi-agent applications [19–21]. Different from decentralized training decentralized execution (DTDE), centralized training decentralized execution (CTDE) is a successful paradigm adopted in MADRL methods [34–37]. This paradigm does not require large calculations resulting from centralized execution. In CTDE, each agent can take the observations and actions of other agents into account during the training process, and execute an action only by the trained policy after receiving its own observation during the test. CTDE has been used to solve a variety of challenging issues, such as non-stationarity, coordination, partial observability, credit assignment, scalability [21, 38–41].

Multi-agent deep deterministic policy gradient (MADDPG) [42] is a typical CTDE method. In MADDPG, each actor outputs its own deterministic policy, and each critic in training can receive the information of other agents that contains the observation and the action of each agent. In this way, during training, the information processed by each critic is transmitted back to each actor by gradient update, so that each agent can cooperate with other agents. As an extension of DDPG, MADDPG has been applied in many fields and achieved good performance [43–49], such as multi-agent defense and attack problem [43], trajectory planning for multi-UAV assisted mobile edge computing [46], and cooperative spectrum sensing in cognitive radio networks [48], etc.

In addition to the application of MADDPG, there are some studies that take MADDPG as the baseline of the proposed algorithm. Multi-agent time delayed deep deterministic policy gradient (MATD3) [50] improves the ability to address complex tasks by dual-centered critic, group target network smoothing and delayed policy updating. Multi actor attention critic (MAAC) [51] learns a centralized critic with an attention mechanism, which enables more flexible learning in complex multi-agent environments. The aforementioned related methods are summarized in Table 1.

The above studies are based on the assumption that the multi-agent systems are with fixed topology. However, in practical applications, the connection between two agents usually changes over time due to communication constraints, which triggers switches in the topology of the whole system. In this paper, we address multi-agent problems under switching topologies. The method presented in this work adopts novel actor-critic neural networks named multi-agent level-fusion actor-critic (MALFAC) with a direction assisted (DA) actor, a dimensional pyramid fusion (DPF) critic, and a new loss function as an experience adviser (EA). These proposed components are also listed in Table 1, which will be introduced in detail in the following sections.

3 Preliminary and problem statement

This section mainly introduces some preliminaries and problem statements in the multi-agent learning under switching topology. First, the multi-agent decision-making process and the role of the reward function are introduced. The basic knowledge of switching topology on observations in multi-agent learning are given. Moreover, the observation of the multi-agent interception system under switching topology is described. These are the foundation for the research in subsequent sections.

3.1 Preliminary

In the multi-agent decision-making process, multiple agents obtain observation information, perform actions, get the reward and the next observation when interacting with the environment. The decentralized partially observable Markov decision process (Dec-POMDP) and the partially observable stochastic game (POSG) are the general decision-making processes of the multi-agent system, where agents could only partially observe the environment. Different from Dec-POMDP where multiple agent share a same reward $\mathcal{R}$, in the POSG, each agent i owns an individual reward function $\mathcal{R}_i$. The goal of each agent i is to maximize its total expected return $\mathcal{R}_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where γ and T denote a discount factor and the time horizon, respectively. Considering the switching topologies in multi-agent decision-making process, this paper proposes

Table 1 Summary of the related methods and the proposal

Method	DTDE	CTDE	VB	PB	AC	EA	DA	DPF
DQN	✓		✓					
PPO	✓			✓				
DDPG	✓				✓			
MADDPG		✓			✓			
MATD3		✓			✓			
MAAC		✓			✓			
MALFAC		✓			✓	✓	✓	✓

an individual reward factor to balance the relationship between individual reward and shared reward in the design of reward function r_i.

In our study, the multi-agent intercepting system is abstracted as a graph $\mathcal{G}$. $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ stands for a weight graph, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, $\mathcal{E} \subseteq \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$, and $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N\times N}$ refer to a node set, an edge set, and an adjacency matrix, respectively. Each agent can be considered as a node and the connection between every two agents is called an edge. $\mathcal{N}^i = \{v_i \in \mathcal{V} : (v_i, v_j) \in \mathcal{E}\}$ denotes a neighbor set of the node v_i. The index of each node belongs to a finite set $\mathcal{I} = \{1, 2, ..., N\}$. If $(v_i, v_j) \in \mathcal{E}$ and $i \neq j$, $a_{ij} > 0$, otherwise $a_{ii} = 0$. The topology that switches between a finite number of possible connected or non-connected graphs over time is called a switching topology, which is expressed as $\mathcal{G}_{\sigma(t)} = (\mathcal{V}, \mathcal{E}_{\sigma(t)})$, where $\sigma : t \to \mathbb{V}$ is a piecewise constant function and $\mathbb{V}$ is a finite set of all possible graphs with node set $\mathcal{V}$. In this work, the switching topology of the system caused by the change of connectivity between agents affects the absence or recurrence of the observation objects at the input of the neural networks.

3.2 Switching observation problem description

Consider a simulated multi-agent intercepting scenario, where N agents are supposed to track the intruding target agent and intercept it to prevent the defense zone from being invaded. Any agent in the scenario is capable of communicating, computing, and moving. Besides, there are also some obstacles in this environment, which hinder the communication among agents. When an agent is in the blind area of another agent, the information of the former can not be observed by the latter. Thus, the interconnection topology of the multi-agent system is dynamically time-varying in the process of tracking.

As depicted in Fig. 1, in order to be concise, the i-th, j-th, and k-th ($i, j, k \in \mathcal{I}$) agents and an obstacle are taken as examples to illustrate the switching problem, which is described from the perspective of the i-th agent as follows. At instant t, the k-th agent in the blind area of the i-th agent cannot be observed, while the j-th agent in the visible area of the i-th agent can be observed. At this instant, the j-th agent is the neighbor of the i-th agent ($j \in \mathcal{N}_t^i$), while k-th agent is not ($k \notin \mathcal{N}_t^i$). Then, at instant $t + \Delta t$, the k-th agent that has not been observed previously can be observed, while the j-th agent is not able to be observed. At this instant, the k-th agent is the neighbor of the i-th agent ($k \in \mathcal{N}_{t+\Delta t}^i$), while j-th agent is not ($j \notin \mathcal{N}_{t+\Delta t}^i$). The neighbor agent of the i-th agent has changed and the topology of the system switches. In the practical tracking scene, there is more than one obstacle, so that the topology among agents may switch many times until the intruding agent is intercepted.

In this paper, the switching observation of the i-th agent at instant t is expressed by:

$$o_t^i = \left[p_t^i, v_t^i, \Delta\hat{p}_t^{ij}, \Delta\hat{v}_t^{ij} (j \in \mathcal{N}_t^i)\right] \tag{1}$$

where p_t^i and v_t^i refer to the position and velocity of the i-th agent at instant t respectively. $\Delta\hat{p}_t^{ij}$ and $\Delta\hat{v}_t^{ij}$ refer to the relative positions and relative velocities between the i-th agent and the j-th agent respectively, if the j-th agent is the i-th agent's neighbor according to the topology at instant t.

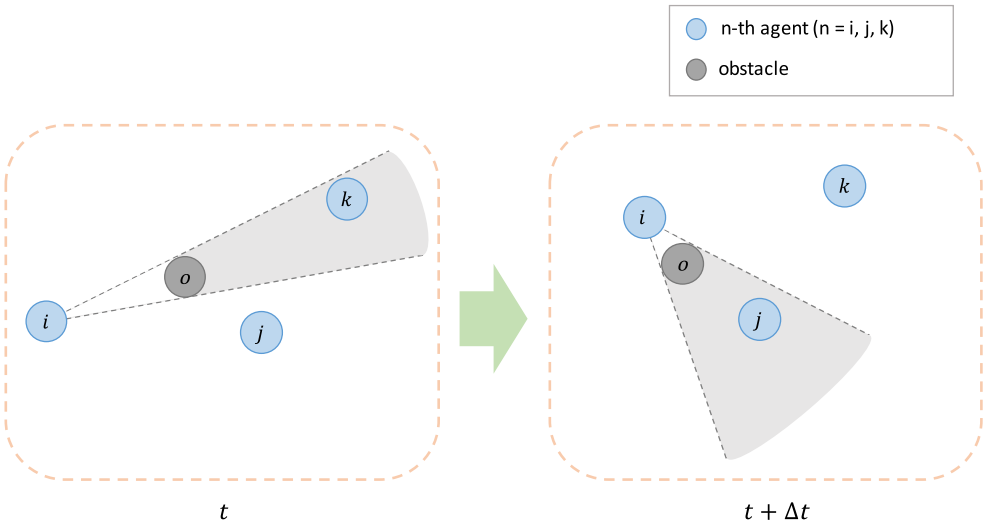


Fig. 1 Schematic diagram of switching in the multi-agent tracking process

4 Method

In this section, we propose a multi-agent level fusion actor-critic (MALFAC) method. First, the framework of the proposed method is presented. Then, the proposed direction assisted actor and dimensional pyramid fusion critic are provided in detail. In addition, we summarize the learning process of our method in Algorithm 1.

4.1 Framework

We propose a multi-agent level fusion actor-critic (MALFAC) method for multiple agents to learn cooperative interception, the pipeline of which is illustrated in Fig. 2. The proposed method adopts the framework of centralized training with decentralized execution. Each agent has four neural networks (μ_i, Q_i, μ_i', and Q_i'). μ_i and Q_i denote the neural networks of the current direction assisted (DA) actor and the current dimensional pyramid fusion (DPF) critic of the i-th agent respectively. μ_i' and Q_i' stand for the neural networks of the target direction assisted (DA) actor and the target dimensional pyramid fusion (DPF) critic of the i-th agent respectively. To simplify the description, the explanation that a symbol with a superscript or subscript i means that it belongs to the i-th agent will be omitted in the following. From the perspective of the i-th agent, the current DA actor μ_i interacts with the environment, when the switching observation o_i is obtained by the i-th agent, the actor outputs an action a_i to be executed, and the agent gains a reward r_i and the next switching observation o_i'.

There is an experience replay buffer $\mathbf{D}$ to store the data (o_i, a_i, r_i, o_i'). When the number of stored data is equal to the threshold of $\mathbf{D}$, the training of each critic starts. The loss function $\mathcal{L}_c^i$ is used to train the neural networks of the DPF critic. As loss functions of actor, $\mathcal{L}_{a_1}^i$, $\mathcal{L}_{a_2}^i$, and $\mathcal{L}_{a_3}^i$ are jointly involved in the updating of the actor's network. Each actor and each critic update the parameters of their neural networks separately. Besides, each critic can obtain the data collected by all actors. To maximize the contribution of each critic to the overall goal, the data generated by all actors is considered when updating parameters. The target neural networks are updated by the corresponding current neural networks at a certain updated interval with a soft updated rate τ. The parameters of the neural network model can be optimized by repeating this training process.

In the following subsections, the learning process of the proposed direction assisted actor and dimensional pyramid fusion critic is detailed, including the network architectures and loss functions.

4.2 Direction assisted actor

The direction assisted (DA) actor takes the observation as input, and outputs the action. The basic neural networks of the DA actor follow [50], which consist of 3 fully connected layers and each layer has 64 neurons. With the forward propagation of information by the networks, some key information may be forgotten. Inspired by the shortcut connection in ResNet [52], an extraction layer is added between the input layer and the last hidden layer, as shown

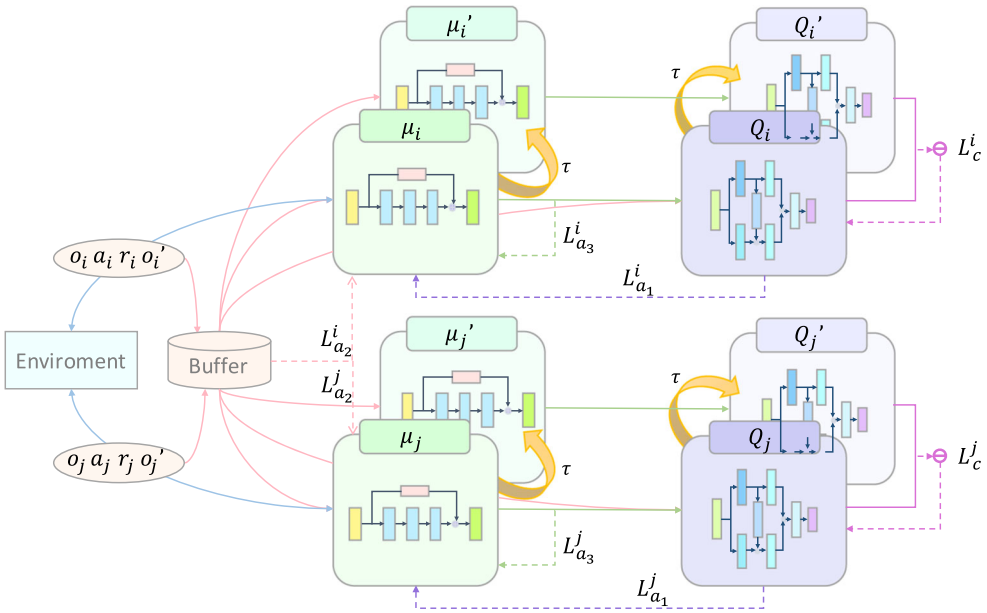


Fig. 2 Pipeline of the proposed method

in Fig. 3. In the extraction layer, only the position and velocity information of the intruding agent is extracted, combined with the weight calculation and the addition of the last hidden layer, the action to be executed is output after the operation of the *tanh* activation function.

Firstly, the DA actor aims to make the state-action value which is output by the critic as large as possible. The first loss function of the DA actor is expressed by

$$\mathcal{L}_{a_1}^i = -\mathbb{E}[Q_i(o_1, o_2, ..., o_n, \mu_1(o_1), \mu_2(o_2), ..., \mu_n(o_n))] \tag{2}$$

Secondly, to make the policy learning more stable, the policy loss is not only related to the output of state-action value function, but also related to the data from experience replay buffer. We design a loss function of experience adviser (EA) as (3), which can consolidate the knowledge learned before and learn on this basis to maintain the stability of training.

$$\mathcal{L}_{a_2}^i = \mathbb{E}[\mu_i(o_i) - a_i] \tag{3}$$

Thirdly, there is a constraint on the output of the actor to make the action value as small as possible, which is expressed as

$$\mathcal{L}_{a_3}^i = \mathbb{E}(\|\mu_i(o_i)\|) \tag{4}$$

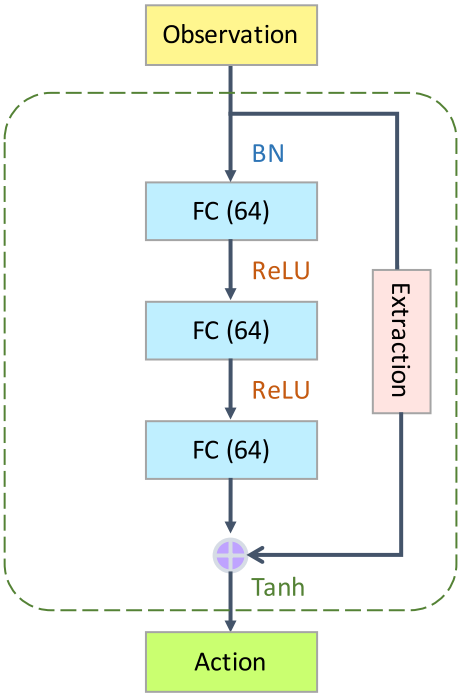


Fig. 3 The neural networks of the DA actor

Thus, the total loss of DA actor is summarized as

$$\mathcal{L}_a^i = \mathcal{L}_{a_1}^i + \alpha\mathcal{L}_{a_2}^i + \beta\mathcal{L}_{a_3}^i \tag{5}$$

where α and β are weight hyper-parameters of the EA loss and constraint loss, respectively, to adjust the proportions of the three loss functions.

The parameters θ_i' of target DA actor networks for each agent i are updated by the parameters θ_i of corresponding current DA actor, which can be described as

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \tag{6}$$

4.3 Dimensional pyramid fusion critic

The dimensional pyramid fusion (DPF) critic takes the concatenation of observation and action as input, and outputs the state-action value. The neural networks of the DPF critic adopt several convolutional layers, as presented in Fig. 4.

There are two branches, which process the input information and the information segmented by the direction dimension respectively. Then the branches are fused with each other to output the state-action value function. The loss function of the DPF critic is defined as

$$\mathcal{L}_c^i = \mathbb{E}_{(\mathbf{o},\mathbf{a},r_i,\mathbf{o}')\in\mathbf{D}}\left[\left(Q_i^{\mu}(\mathbf{o},\mathbf{a}) - y_i\right)^2\right] \tag{7}$$

where

$$y_i = r_i + \gamma Q_i^{\mu'}(\mathbf{o}', \mathbf{a}')|_{a_j'=\mu_j'(o_j)} \tag{8}$$

is taken as the label of the critic networks.

In the multi-agent interception task, N agents should protect a field from the intrusion of other agents. In addition, due to the existence of obstacles, the observed objects of each agent may disappear occasionally. The agents must first learn to track the mobile intruder under switching topologies $\mathcal{G}_{\sigma(t)}$. The setting of rewards can guide agents to carry out tasks. In this task, each agent needs not only an individual reward to ensure its own tracking, but also a collective reward to cooperate with other agents under switching observations to finally realize the interception. Thus, the reward of each agent is designed as

$$r_i = \frac{1-\lambda}{N}\sum_{j=1}^{N} d_j + \lambda d_i \tag{9}$$

where $\lambda \in [0, 1]$ denotes an individual reward factor and $d_i = \|p_i - p_t\|_2$ refers to the distance between the i-th agent and the intruding agent.

The parameters ϕ_i' of target DPF critic networks for each agent i are updated by the parameters ϕ_i of corresponding current DPF critic, which can be expressed as

$$\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i' \tag{10}$$

In summary, the learning process of the proposed method is summarized as Algorithm 1.

Algorithm 1 Learning process of MALFAC.

Initialize the environment with N intercepting agents and an intruding agent
Initialize an experience replay buffer $\mathbf{D}$
for episode=1 to E, where E refers to number of training episodes **do**
  Reset the environment, get a new state $\mathbf{x}$, and give each agent i a new observation o_i
  for $t = 1$ to l, where l denotes the maximum number of steps in an episode **do**
    for agent $i = 1$ to N **do**
      Select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ based on the actor with exploration
      Execute action a_i, then obtain reward r_i and new observation o_i'
      Store (o_i, a_i, r_i, o_i') in $\mathbf{D}$
    end for
    State $\mathbf{x}$ is replaced by next state $\mathbf{x}'$
    for agent $i = 1$ to N **do**
      Randomly sample a mini-batch B of (o, a, r, o') from $\mathbf{D}$
      Set the label of the DPF critic
      $y_i = r_i + \gamma\, Q_i^{\mu'}(\mathbf{o}', \mathbf{a}')|_{a_j' = \mu_j'(o_j)}$
      Update the DPF critic by
      $\mathcal{L}_c^i = \mathbb{E}_{(\mathbf{o},\mathbf{a},r_i,\mathbf{o}')\in\mathbf{D}}\left[\left(Q_i^{\mu}(\mathbf{o},\mathbf{a}) - y_i\right)^2\right]$
      Update the DA actor by
      $\mathcal{L}_a^i = -\mathbb{E}[Q_i(o_1, o_2, ..., o_n, \mu_1(o_1), \mu_2(o_2), ..., \mu_n(o_n))] + \alpha\mathbb{E}[\mu_i(o_i) - a_i] + \beta\mathbb{E}(\|\mu_i(o_i)\|)$
    end for
    if t mod ρ, where ρ represents the target update interval **then**
      Update the parameters of target DA actor and target DPF critic for each agent i:
      $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
      $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$
    end if
  end for
end for

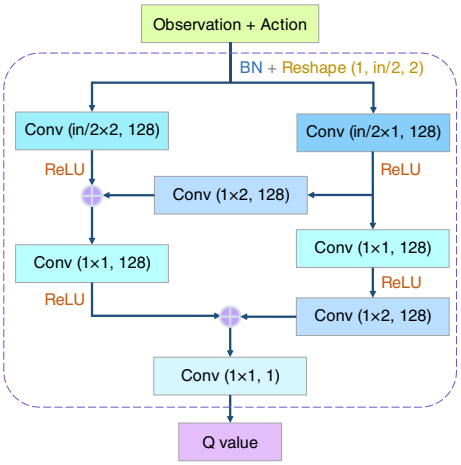


Fig. 4 The neural networks of the DPF critic

5 Experiments

This section first presents the implementation details of the experiments. Then the experimental results of the proposed method and related methods are reported. In addition, we extensively study the effectiveness and scalability of our method. At last, the discussion and limits of the proposed method are provided.

5.1 Implementation details

Different from supervised learning in general deep learning, the data set of deep reinforcement learning is obtained by the interaction between the agent and the environment. To verify the performance of the proposed method, experiments are conducted based on the multi-agent particle environment (MPE) [42], where agents can interact in a two-dimensional platform. The action space dimension of each agent is two. In this work, there are $N(N \in \{3, 4, 5\})$ agents guarding an area with a defense zone and $M(M \in \{15, 20, 25\})$ obstacles. Once an external agent invades, these agents will track it and intercept it to prevent it from reaching the defense zone. At the beginning of each training or test episode, tracking agents, an intruding target agent, a defense zone, and obstacles are initialized randomly in a certain area. Specifically, taking the area center as the coordinate origin, the defense zone, the intruding target agent, tracking agents and obstacles are randomly initialized at (x_d, y_d), (x_t, y_t), (x_i, y_i) and (x_o, y_o), where $x_d \in [0.7, 0.9]$, $y_d \in [0.7, 0.9]$, $x_t \in [-0.95, -0.75]$, $y_t \in [-0.95, -0.75]$, $x_i \in [-0.5, 0.5]$, $y_i \in [-0.5, 0.5]$, $x_o \in [-0.9, 0.9]$ and $y_o \in [-0.9, 0.9]$. The initial velocity of all agents is $(0, 0)$. The simulation time step is set as 0.1. The maximum velocity of the intruding target agent is 0.25. The maximum velocity of tracking agents is set to be $\kappa(\kappa \in \{1, 1.3, 1.6\})$ times that of the intruding target agent. We

take a scenario with 3 tracking agents and 20 obstacles as an example, as shown in Fig. 5.

In the setting of agents, the viewpoint of each agent is set at its geometric center with a viewpoint of 360 degrees. When an agent enters the blind area of another agent's field of view, the latter cannot observe the former's information. Correspondingly, when an agent enters the field of view of another agent, the latter can obtain the observation of the former. Thus, the existence of obstacles affects the connections among agents, leading to switching topologies of the whole system. Before tracking agents realize interception, the possible switching topologies of the multi-agent system are displayed in Fig. 6. There are three tracking agents (blue) and an intruding agent (red). The green line connection between the agents depicts that they can observe each other.

In the multi-agent training process, following [42], the capacity of the experience replay buffer is set to 10^6. Each batch size in training is set to 1024. Adam [53] is adopted as an optimizer with a learning rate of 10^{-3}. The weight hyper-parameters α and β in (3) are set to 10^{-3}. For both training and testing, there are 100 steps in each episode. We implement the experiments using Python as the programming language on the Ubuntu 18.04 operating system. We also adopt the PyTorch machine learning library, which can use GPU to accelerate tensor computation and automatically derive neural networks.

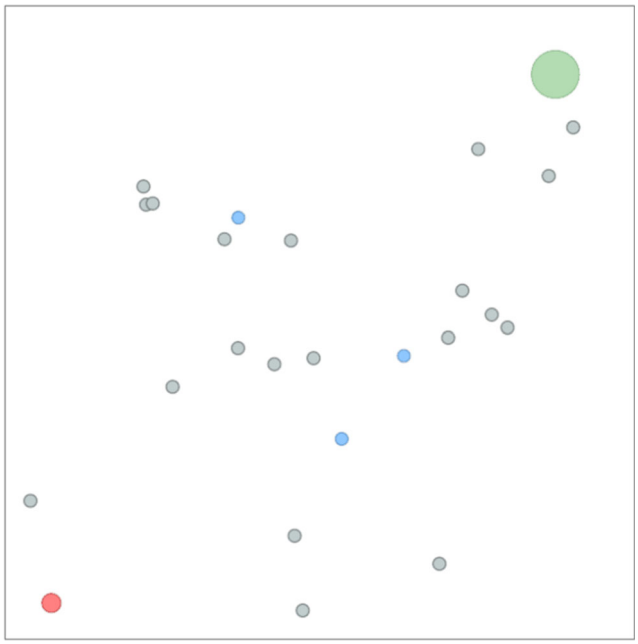


Fig. 5 The initial scenario with three tracking agents (blue), an intruding agent (red), a defense zone (green), and twenty obstacles (grey)

5.2 Comparison with related methods

First, we conduct experiments on the scenario with 3 tracking agents and 20 obstacles. The training curves of mean episode reward are plotted for the proposed method and compared methods (DQN [30], PPO [27], DDPG [28], MADDPG [42], MATD3 [50], and MAAC [51]) in Fig. 7. The reward is positively correlated with the learning ability of the multi-agent system. Without centralized training, DDPG, PPO, and DQN do not perform well. MADDPG and MATD3 performs similarly and outperforms DDPG, PPO, and DQN, but cannot adapt to switching topology scenarios. Since MAAC adopts an attentive critic to select relevant information for each agent, the upward trend of the reward obtained by MAAC is much more obvious than those of the other compared methods. However, as displayed in Fig. 8, MAAC consumes the longest calculation time of the learning period among all comparison methods. Combined with the experimental results in Figs. 7 and 8, our method obtains a higher mean episode reward than the second-best method (MAAC) with less training time for the scenario under switching topology.

Table 2 illustrates the testing results of our method and compared methods with the models saved at 6,000, 12,000, and 18,000 episodes. $succ\ \%$ stands for success rate. s and s' refer to the number of average steps for successful interception when the failed interception episode does not participate and participates in the calculation respectively. When calculating s', the interception steps of the failed round are recorded as the number of maximum steps (100) in each episode. $E(r)$, $D(r)$ represent the mean and variance of reward respectively.

As listed in Table 2, during the training, the DQN, PPO, DDPG, MADDPG, and MATD3 are not competent for the interception task, while MAAC and our method can achieve 36%, 36% and 72%, 96% success rates after 6,000 episodes and 12,000 episodes of training respectively. At the end of the training episode (18,000), the success rate of our method (100%) is higher than that of MAAC (89%), MATD3 (1%), and MADDPG (1%). As for average interception steps, our method uses the least number of steps (32.17) to realize successful interception, whether or not it includes failed interception episodes. Moreover, our method obtains the highest mean episode reward (-0.1184) with the smallest (0.0014) variance. Figure 9 compares the average rewards obtained by testing the model after training 18,000 episodes with a box plot. Compared with other methods, our method achieves the best reward performance, which has the largest median, smallest interquartile range, and the least outliers in the test results.

It is displayed in Fig. 10 that the testing results of models in different episodes by the proposed method. There is a

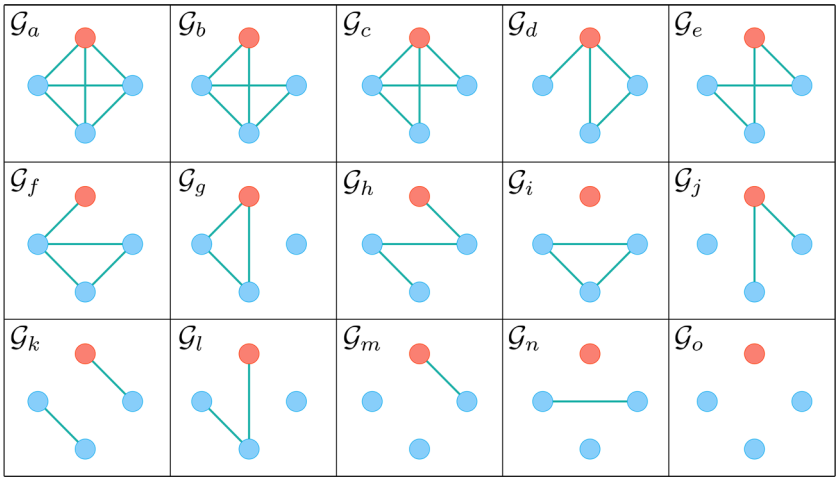


Fig. 6 Switching topologies of the multi-agent system

gradual improvement in the success rate of interception with the increase of training episodes corresponding to the saved models, as shown in Fig. 10a. Besides, the number of steps of successful interception reduces gradually with the increase of training episodes, as shown in Fig. 10b. From 9,000 to 12,000 episodes, the success rate improves significantly, and the number of interception steps decreases sharply. From 15,000 to 18,000 episodes, the success rate remains static at 100%, and the number of interception steps decreases steadily.

Figure 11 depicts one of the successful interceptions by our method in the designed multi-agent switching scenario. The intruding target agent starts from the initial position and moves towards the defense zone. The three guarding agents start from their respective initial positions to intercept the intruding target agent. The red curve plots the trajectory of the intruding target agent. The blue, orange, and green curves represent the trajectories of three tracking agents respectively. Although the observations of the tracking agent with a blue trajectory to the target agent and other tracking agents are sometimes obscured by obstacles in the tracking process, the multi-agent system still achieves the interception of the intruding target agent under the switching topology.

5.3 Extensive study

The proposed components (DA, DPF, and EA) are adopted in our method, which results in a good performance. To verify the effectiveness of each component more rigorously, a series of ablation experiments are conducted for each component. Table 3 demonstrates the results of the ablation experiments. With DA, DPF, and EA, the proposed method achieves the highest success rate and shortest interception steps. Without EA, the model takes 1.52 more steps before achieving an interception at a 100% success rate. Without DPF, the success rate of the model dropped to 75%. Without DA, the success rate of the model is 1%. The model with DA achieves a success rate of 66%. DA contributes more to

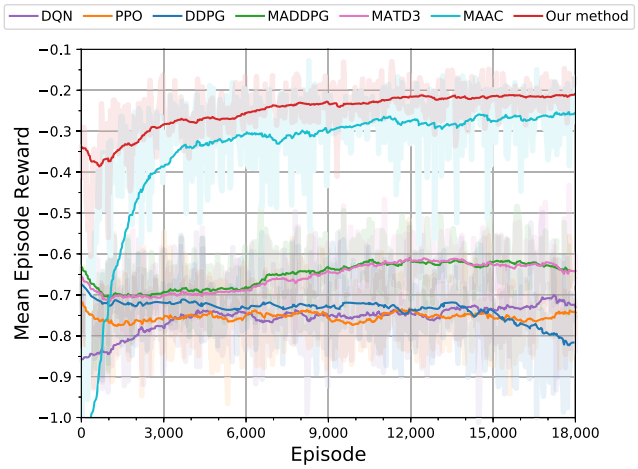


Fig. 7 Comparison of mean episode rewards in training

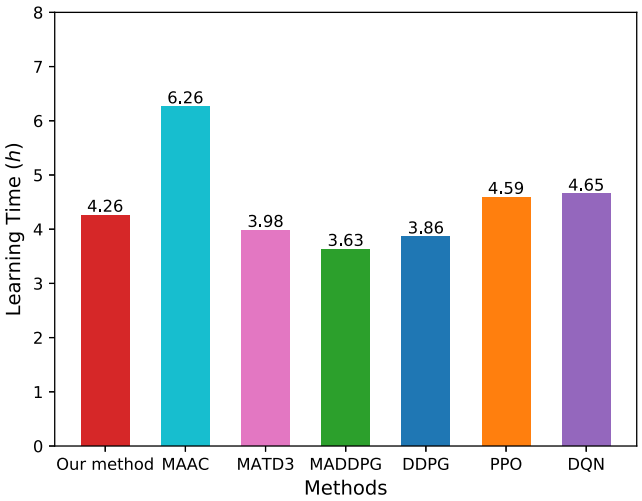


Fig. 8 Comparison of calculation time of the learning period

Table 2 Comparisons of different methods on the success rate, intercepted steps, and reward

Episode	Method	*succ %*	s	s'	$E(r)$	$D(r)$
	DQN	0	-	-	-0.7402	0.0102
	PPO	0	-	-	-0.7457	0.0057
	DDPG	0	-	-	-0.5914	0.0071
6,000	MADDPG	0	-	-	-0.5396	0.0040
	MATD3	0	-	-	-0.6466	0.0034
	MAAC	36	57.81	84.81	-0.3006	0.0200
	Our Method	36	66.00	87.76	-0.1410	0.0020
	DQN	0	-	-	-0.7407	0.0115
	PPO	0	-	-	-0.7560	0.0057
	DDPG	0	-	-	-0.6106	0.0095
12,000	MADDPG	0	-	-	-0.5123	0.0047
	MATD3	1	68.00	99.68	-0.6018	0.0042
	MAAC	72	52.51	65.81	-0.2334	0.0109
	Our Method	96	38.23	40.70	-0.1176	0.0014
	DQN	0	-	-	-0.7721	0.0141
	PPO	0	-	-	-0.7456	0.0061
	DDPG	0	-	-	-0.7148	0.0138
18,000	MADDPG	1	48.00	99.48	-0.5267	0.0060
	MATD3	1	54.00	99.54	-0.6328	0.0046
	MAAC	89	44.84	50.91	-0.1939	0.0044
	Our Method	100	32.17	32.17	-0.1184	0.0014

the success rate. The success rate of the model is 0% when using DPF or EA alone. However, when DA is combined with EA or DPF, the model can achieve a 9% or 34% higher success rate, respectively, than the model using DA alone.

Table 4 lists the influence of the value of the individual reward factor (λ) on experimental results. When $\lambda = 0.9$, the proposed method achieves a 100% success rate with a minimum number of interception steps (32.17). The model obtains a 100% success rate for $\lambda \in [0.4, 0.9]$, and within this interval, the number of interception steps gradually increases as the value of λ decreases.

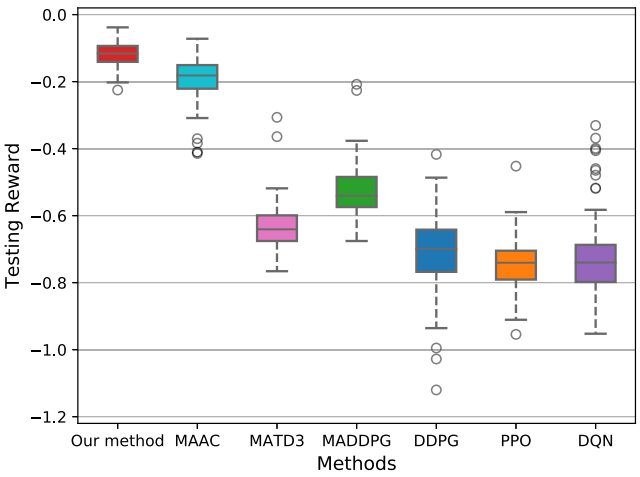


Fig. 9 Comparison of testing reward by the models saved in 18,000 episode

Figure 12 shows more visually the influence of the individual reward factor (λ) on success rate and mean intercepted step (s'). In general, when $\lambda \leq 0.9$, the success rate rises with the increase of the individual reward factor, as displayed in Fig. 12a. As λ grows from 0.0 to 0.1, the success rate climbs significantly from 71% to 97%. As λ grows from 0.9 to 1.0, the success rate drops from 100% to 98%. In terms of interception efficiency, when $\lambda \leq 0.9$, the mean intercepted step (s') decreases with the increase of the individual reward factor, as displayed in Fig. 12b. As λ grows from 0.9 to 1.0, the mean intercepted step (s') increases from 32.17 to 34.41. The proposed method can achieve 100% successful interception at an average of 32.17 steps when $\lambda = 0.9$. According to (9), when $\lambda = 0$, each agent only takes the shared reward into account. With the increase of λ, the proportion of individual reward to total reward increases. When $\lambda = 0.3$, though the upward trend of the success rate shown in Fig. 12a fluctuates slightly, it can be seen from Fig. 12b that the average number of tracking steps still maintains a decreasing trend. Since only the individual reward is considered when $\lambda = 1$, the interception success rate decreases.

To verify the scalability of the proposed method, we also conduct extensive experiments in multiple scenarios,

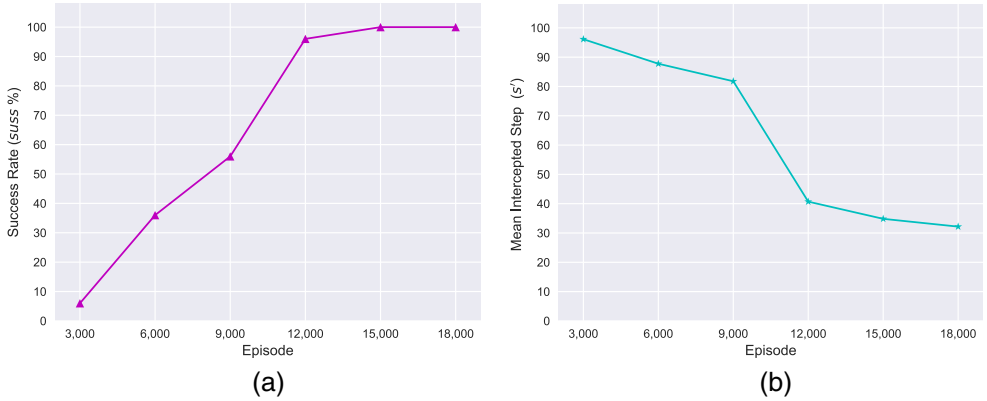


Fig. 10 Testing results of models in different episodes by the proposed method

including the effects of different relative velocities, different numbers of tracking agents, and different numbers of obstacles.

Table 5 shows the effect of different relative velocities between the tracking agents and the intruding agent on the performance of the method. When the value of κ is 1.3 or 1.6, the model can achieve 100% successful interception. When the value of κ is 1, the interception success rate of the model is 82%. As κ decreases, the number of tracking steps before interception increases.

The scenarios of experiments with different numbers of tracking agents are displayed in Fig. 13. Table 6 lists the test results obtained by the trained models with different numbers of tracking agents in the scenarios. When the number of tracking agents is 3, 4, or 5, 100% interception success rate can be achieved. As the number of tracking agents increases, the number of successful interception steps decreases gradually. The calculation time for the learning period increases with the number of tracking agents.

The scenarios of experiments with different numbers of obstacles are shown in Fig. 14. Table 7 reports the performance of the number of obstacles in the scenario on the experimental results. With more obstacles in the scenario, the agent encounters a greater probability of switching topology and a larger blind area of observations when interacting with the environment, resulting in an increase in the number of interception steps. When the number of obstacles is 25, the model success rate is 97%. As the number of obstacles increases, the computation of the learning period takes more time.

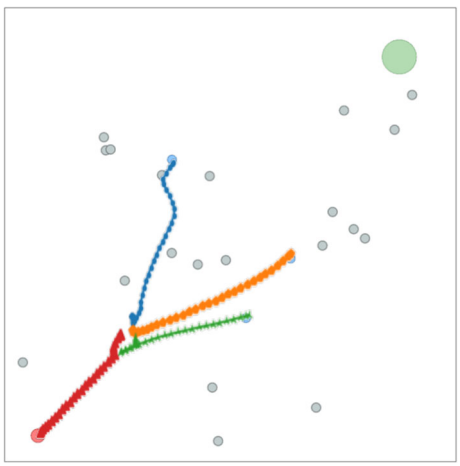


Fig. 11 Trajectories of an interception in testing

5.4 Discussion and limits

In the multi-agent interception task under the switching topology condition due to occlusion by multiple obstacles, our proposed method outperforms the comparison methods with a higher interception success rate and fewer interception steps. Furthermore, the training time of the proposed method is shorter than that of the best comparison methods. The ablation experiments prove that each of the proposed modules is effective, and the combination of the three can produce the best results. The results of parameter sensitivity experiments show the relationship between individual reward factors and algorithm performance. More extensive experiments demonstrate that our method has good robustness in scenarios with different numbers of agents and different numbers of obstacles. However, the proposed method is not optimized for scenarios with multiple opponents. In addition, the experiments assume that there is no delay in the observed information. These limitations should be considered and addressed in our future work.

6 Conclusions

In this paper, a new multi-agent interception environment under switching topology is designed and its reward function related to individual as well as common goal is studied. According to the multi-agent switching interception problem, a novel method named multi-agent level-fusion actor-critic (MALFAC) is proposed, which is composed

Table 3 Comparisons of the model with different components

DA	DPF	EA	*succ %*	s	s'	$E(r)$	$D(r)$
✓	✓	✓	100	32.17	32.17	-0.1184	0.0014
✓	✓		100	33.69	33.69	-0.1206	0.0019
✓		✓	75	80.47	85.35	-0.1244	0.0016
	✓	✓	1	41.00	99.41	-0.4559	0.0052
✓			66	71.39	81.12	-0.1198	0.0017
	✓		0	-	-	-0.4976	0.0053
		✓	0	-	-	-0.4717	0.0050

Table 4 Comparisons of the model with different individual reward factors

λ	*succ %*	s	s'	$E(r)$	$D(r)$
0.0	71	55.66	68.52	-0.1216	0.0016
0.1	97	56.81	58.11	-0.1155	0.0015
0.2	98	50.19	51.19	-0.1127	0.0014
0.3	97	40.97	42.74	-0.1169	0.0014
0.4	100	38.31	38.31	-0.1159	0.0014
0.5	100	34.96	34.96	-0.1186	0.0015
0.6	100	34.71	34.71	-0.1169	0.0014
0.7	100	34.02	34.02	-0.1195	0.0015
0.8	100	32.73	32.73	-0.1196	0.0014
0.9	100	32.17	32.17	-0.1184	0.0014
1.0	98	33.07	34.41	-0.1211	0.0016

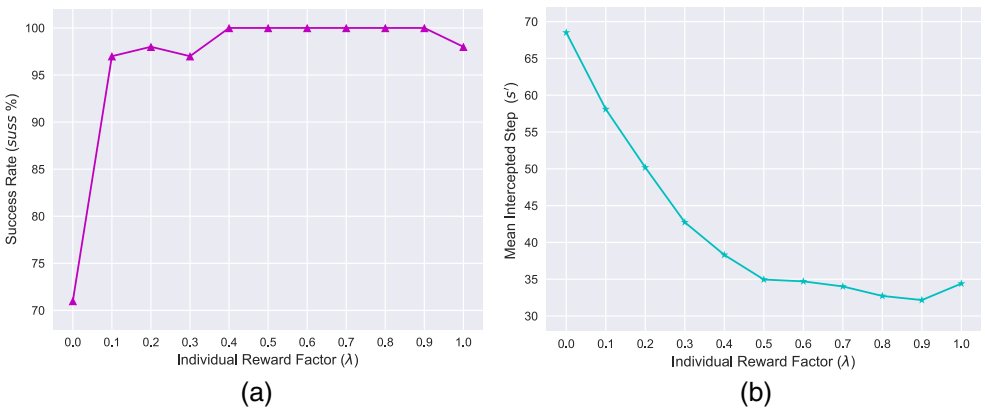
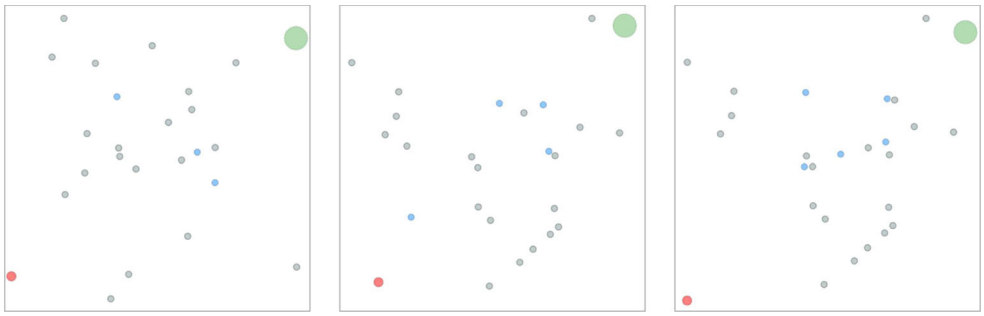


Fig. 12 The influence of the individual reward factor on success rate and mean intercepted step

Table 5 Comparisons of the proposed method for the scenarios with different ratios of relative maximum velocity (κ)

κ	*succ %*	s	s'	$E(r)$	$D(r)$	$t(h)$
1.0	82	42.26	52.65	-0.1362	0.0021	4.23
1.3	100	32.17	32.17	-0.1184	0.0014	4.26
1.6	100	30.43	30.43	-0.1054	0.0011	4.29

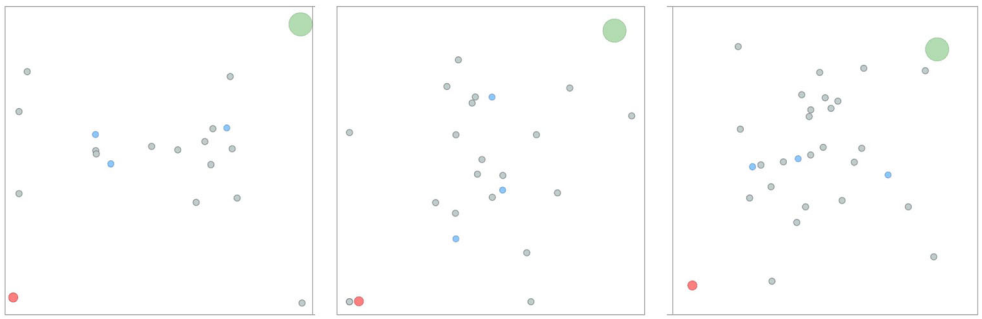


(a) $N = 3$ (b) $N = 4$ (c) $N = 5$

Fig. 13 The scenarios with different number of tracking agents (N)

Table 6 Comparisons of the proposed method for the scenarios with different number of tracking agents (N)

N	$succ\ \%$	s	s'	$E(r)$	$D(r)$	$t(h)$
3	100	32.17	32.17	-0.1184	0.0014	4.26
4	100	29.42	29.42	-0.1169	0.0008	6.55
5	100	26.73	26.73	-0.1240	0.0010	9.07



(a) $M = 15$ (b) $M = 20$ (c) $M = 25$

Fig. 14 The scenarios with different number of obstacles (M)

Table 7 Comparisons of the proposed method for the scenarios with different number of obstacles (M)

M	$succ\ \%$	s	s'	$E(r)$	$D(r)$	$t(h)$
15	100	31.96	31.96	-0.1128	0.0013	3.56
20	100	32.17	32.17	-0.1184	0.0014	4.26
25	97	34.96	36.92	-0.1228	0.0023	4.90

of a direction assisted (DA) actor and a dimensional pyramid fusion (DPF) critic. Besides, there is a loss function that acts as an experience adviser (EA) added to the learning process of the DA actor. Experimental results show that the proposed method has better training performance than recent approaches. In addition, our method takes the least number of steps to achieve the highest success rate for interceptions during the test. Subsequently, ablation studies are conducted to verify the effectiveness of each innovation component in the proposed method. It is shown in the ablations that each component is indispensable for achieving good interception performance, including obtained reward, success rate, and interception steps. The extensive experimental results of different scenarios, including different numbers of agents, different numbers of obstacles, and different ratios of relative maximum velocity, show the scalability of the proposed method. In future work, we intend to address the problem of multi-agent systems for multiple opponents with time delay.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Nguyen TT, Nguyen ND, Nahavandi S (2020) Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE Trans Cybern 50(9):3826–3839
2. Mahmoud MS (2020) Multiagent systems: Introduction and coordination control. CRC Press, Boca Raton, FL, USA
3. Ji G, Yan J, Du J, Yan W, Chen J, Lu Y, Rojas J, Cheng SS (2021) Towards safe control of continuum manipulator using shielded multiagent reinforcement learning. IEEE Robot Autom Lett 6(4):7461–7468
4. Perrusqu'ia A, Yu W, Li X (2021) Multi-agent reinforcement learning for redundant robot control in task-space. Int J Mach Learn Cybern 12:231–241
5. Kim H, Kim D, Kim H, Shin JU, Myung H (2016) An extended any-angle path planning algorithm for maintaining formation of multi-agent jellyfish elimination robot system. Int J Control Autom Syst 14(2):598–607
6. Zhou W, Liu Z, Li J, Xu X, Shen L (2021) Multi-target tracking for unmanned aerial vehicle swarms using deep reinforcement learning. Neurocomputing 466:285–297
7. Kim J (2020) Cooperative localization and unknown currents estimation using multiple autonomous underwater vehicles. IEEE Robot Autom Lett 5(2):2365–2371
8. Chen Y-J, Chang D-K, Zhang C (2020) Autonomous tracking using a swarm of uavs: A constrained multi-agent reinforcement learning approach. IEEE Trans Veh Technol 69(11):13702–13717
9. Shi Y, Hu Q (2021) Observer-based spacecraft formation coordinated control via a unified event-triggered communication. IEEE Trans Aerosp Electron Syst 57(5):3307–3319
10. Dong Y, Chen J (2021) Nonlinear observer-based approach for cooperative control of networked rigid spacecraft systems. Automatica 128:109552
11. Zhang C, Wang J, Zhang D, Shao X (2018) Fault-tolerant adaptive finite-time attitude synchronization and tracking control for multi-spacecraft formation. Aerosp Sci Technol 73:197–209
12. Duan P, Liu K, Huang N, Duan Z (2020) Event-based distributed tracking control for second-order multiagent systems with switching networks. IEEE Trans Syst Man Cybern Syst 50(9):3220–3230
13. Dong L, Yu D, Yan H (2020) Stability analysis of nonlinear multi-agent relay tracking systems over a finite time interval. Int J Control 93(3):519–527
14. Wang Y-W, Lei Y, Bian T, Guan Z-H (2020) Distributed control of nonlinear multiagent systems with unknown and nonidentical control directions via event-triggered communication. IEEE Trans Cybern 50(5):1820–1832
15. Liu C, Jiang B, Zhang K, Patton RJ (2021) Distributed fault-tolerant consensus tracking control of multi-agent systems under fixed and switching topologies. IEEE Transactions on Circuits and Systems I: Regular Papers 68(4):1646–1658
16. Zou W, Shi P, Xiang Z, Shi Y (2020) Finite-time consensus of second-order switched nonlinear multi-agent systems. IEEE Trans Neural Netw Learn Syst 31(5):1757–1762
17. Jiang J, Jiang Y (2020) Leader-following consensus of linear time-varying multi-agent systems under fixed and switching topologies. Automatica 113:108804
18. Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT Press, Cambridge, MA, USA
19. Hernandez-Leal P, Kartal B, Taylor ME (2019) A survey and critique of multiagent deep reinforcement learning. Auton Agent Multi-Agent Syst 33(6):750–797
20. Zhang K, Yang Z, Başar T (2021) Multi-agent reinforcement learning: A selective overview of theories and algorithms. pp 321–384
21. Gronauer S, Diepold K (2021) Multi-agent deep reinforcement learning: a survey. Artif Intell Rev, pp 1–49
22. Gupta S, Singal G, Garg D (2021) Deep reinforcement learning techniques in diversified domains: A survey. Archives of Computational Methods in Engineering, pp 4715–4754
23. Shang M, Zhou Y, Fujita H (2021) Deep reinforcement learning with reference system to handle constraints for energy-efficient train control. Inf Sci 570:708–721
24. Le N, Rathour VS, Yamazaki K, Luu K, Savvides M (2021) Deep reinforcement learning in computer vision: a comprehensive survey. Artif Intell Rev
25. Zhou SK, Le HN, Luu K, V Nguyen H, Ayache N (2021) Deep reinforcement learning in medical imaging: A literature review. Med Image Anal 73:102193
26. Silver D, Schrittwieser J, Simonyan K et al (2017) Mastering the game of go without human knowledge. Nature 550:354–359
27. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv:1707.06347v2
28. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2016) Continuous control with deep reinforcement learning. In: 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, May 2-4, 2016
29. Silver D, Huang A, Maddison C et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529:484–489
30. Mnih V, Kavukcuoglu K, Silver D et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
31. Shoham Y, Leyton-Brown K (2009) Multiagent systems - algorithmic, game-theoretic, and logical foundations. Cambridge University Press, Cambridge, England

32. Sadhu AK, Konar A (2020) Multi-agent coordination: A reinforcement learning approach. John Wiley & Sons
33. Zhang Y, Zhou Y, Lu H, Fujita H (2021) Cooperative multi-agent actor-critic control of traffic network flow based on edge computing. Futur Gener Comput Syst 123:128–141
34. Ye Z, Chen Y, Jiang X, Song G, Yang B, Fan S (2021) Improving sample efficiency in multi-agent actor-critic methods. Appl Intell
35. Cao D, Zhao J, Hu W, Ding F, Huang Q, Chen Z, Blaabjerg F (2021) Data-driven multi-agent deep reinforcement learning for distribution system decentralized voltage control with high penetration of pvs. IEEE Trans on Smart Grid 12(5):4137–4150
36. Du W, Ding S, Zhang C, Du S (2021) Modified action decoder using bayesian reasoning for multi-agent deep reinforcement learning. Int J Mach Learn Cybern 12(10):2947–2961
37. Xu C, Liu S, Zhang C, Huang Y, Lu Z, Yang L (2021) Multi-agent reinforcement learning based distributed transmission in collaborative cloud-edge systems. IEEE Trans Veh Technol 70(2):1658–1672
38. Sunehag P, Lever G, Gruslys A et al (2018) Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems (AAMAS), Stockholm, Sweden, July 10-15, 2018, pp 2085–2087
39. Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S (2018) QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: Proceedings of the 35th international conference on machine learning (ICML), Stockholm Sweden, 10-15 Jul, 2018, vol 80, pp 4295–4304
40. Son K, Kim D, Kang WJ, Hostallero D, Yi Y (2019) QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In: Proceedings of the 36th international conference on machine learning (ICML), Long Beach, California, USA, 9-15 june 2019, vol 97, pp 5887–5896
41. Foerster JN, Farquhar G, Afouras T, Nardelli N, Whiteson S (2018) Counterfactual multi-agent policy gradients. In: Proceedings of the 32nd AAAI conference on artificial intelligence (AAAI), New Orleans, Louisiana, USA, February 2-7, 2018, pp 2974–2982
42. Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in neural information processing systems 30 (NIPS), Long Beach, CA, USA, 4-9 december 2017, pp 6379–6390
43. Huang L, Fu M, Qu H, Wang S, Hu S (2021) A deep reinforcement learning-based method applied for solving multi-agent defense and attack problems. Expert Syst Appl 176:114896
44. Chen X, Liu G (2021) Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks. IEEE Internet of Things Journal 8(13):10843–10856
45. Yang Y, Li B, Zhang S, Zhao W, Zhang H (2021) Cooperative proactive eavesdropping based on deep reinforcement learning. IEEE Wirel Commun Lett 10(9):1857–1861
46. Wang L, Wang K, Pan C, Xu W, Aslam N, Hanzo L (2021) Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing. IEEE Trans Cogn Commun Netw 7(1):73–84
47. Wu T, Zhou P, Wang B, Li A, Tang X, Xu Z, Chen K, Ding X (2021) Joint traffic control and multi-channel reassignment for core backbone network in sdn-iot: A multi-agent deep reinforcement learning approach. IEEE Trans Netw Sci Eng 8(1):231–245
48. Gao A, Du C, Ng SX, Liang W (2021) A cooperative spectrum sensing with multi-agent reinforcement learning approach in cognitive radio networks. IEEE Commun Lett 25(8):2604–2608
49. Sun X, Qiu J (2021) Two-stage volt/var control in active distribution networks with multi-agent deep reinforcement learning method. IEEE Trans on Smart Grid 12(4):2903–2912
50. Zhang F, Li J, Li Z (2020) A TD3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment. Neurocomputing 411:206–215
51. Chaudhuri K, Salakhutdinov R (2019) Actor-attention-critic for multi-agent reinforcement learning. In: Proceedings of the 36th international conference on machine learning (ICML), 9-15 June 2019, Long Beach, California, USA
52. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, NV, USA, june 27-30, 2016
53. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: 3rd international conference on learning representations (ICLR 2015). ICLR, San Diego, CA, USA

Dongyu Fan received her B.E. and M.E. degrees from the School of Mechanical and Equipment Engineering, Hebei University of Engineering, China, in 2014 and 2017, respectively. She was a visiting scholar at the University of Malaya, Malaysia, and Tianjin University, China, in 2013 and 2016, respectively. Currently, she is pursuing her Ph.D. degree at the School of Mechanical, Electronic, and Control Engineering, Beijing Jiaotong University, China. Her research interests include deep neural networks, reinforcement learning, and multi-agent systems.

Haikuo Shen received his B.E. and Ph.D. degrees from Zhejiang University, China, in 2002 and 2007, respectively. From 2015 to 2016, he was a visiting scholar at Carleton University, Canada. He is currently a professor at the School of Mechanical Electronic and Control Engineering, Beijing Jiaotong University. His research interests include intelligent detection and deep learning.

Lijing Dong received the B.E. and Ph.D. degrees from the School of Automation, Beijing Institute of Technology, China, in 2010 and 2016, respectively. From 2013 to 2014, she was a visiting scholar at the University of Auckland. Currently, she is an Associate Professor at the School of Mechanical, Electronic, and Control Engineering, Beijing Jiaotong University. Her current research interests include multi-agent systems and distributed control systems.