electronics

MDPI

Article

Multi-Agent Multi-Target Pursuit with Dynamic Target Allocation and Actor Network Optimization

Baoqiang Han, Lin Shi, Xueyuan Wang and Lihua Zhuang *

Aliyun School of Big Data, Changzhou University, Changzhou 213164, China; s21150812011@smail.cczu.edu.cn(B.H.); slcczu@cczu.edu.cn(L.S.); xywang@cczu.edu.cn(X.W.)
* Correspondence: zlh@cczu.edu.cn

Abstract: In this paper, we consider the cooperative decision-making problem for multi-target tracking in multi-agent systems using multi-agent deep reinforcement learning algorithms. Multi-agent multi-target pursuit has faced new challenges in practical applications, where pursuers need to plan collision-free paths and appropriate multi-target allocation strategies to determine which target to track at the current time for each pursuer. We design three feasible multi-target allocation strategies from different perspectives. We compare our allocation strategies in the multi-agent multi-target pursuit environment that models collision risk and verify the superiority of the allocation strategy marked as POLICY3, considering the overall perspective of agents and targets. We also find that there is a significant gap in the tracking policies learned by agents when using the multi-agent reinforcement learning algorithm MATD3. We propose an improved algorithm, DAO-MATD3, based on dynamic actor network optimization. The simulation results show that the proposed POLICY3-DAO-MATD3 method effectively improves the efficiency of completing multi-agent multi-target pursuit tasks.

Keywords: multi-agent; deep reinforcement learning; multi-target; allocation strategy

Citation: Han, B.; Shi, L.; Wang, X.; Zhuang, L. Multi-Agent Multi-Target Pursuit with Dynamic Target Allocation and Actor Network Optimization. *Electronics* **2023**, *12*, 4613. https://doi.org/10.3390/electronics12224613

Academic Editors: Junhua Ding, Haihua Chen, Yunhe Feng and Tozammel Hossain

Received: 8 October 2023
Revised: 2 November 2023
Accepted: 3 November 2023
Published: 11 November 2023

1. Introduction

Pursuit-evasion has been an extensively studied problem where multiple pursuers aim to capture multiple moving evaders. This problem has faced new challenges as the applications of robots and UAVs have become ubiquitous. In these scenarios, the pursuers are equipped with advanced control system hardware and algorithms, and assume a great degree of autonomy.

As a result, these pursuers are modeled as independent agents, and decentralized multi-agent pursuit eliminates single point failures while requiring sufficient and effective coordination to capture targets collaboratively. Constraints in the pursuers' maneuvers (e.g., non-holonomic constraints [1]) render some classical approaches based on omni-directional pursuers or simple geometry (such as [2–4]) ineffective because of their inability to adapt to the evader's changes in behavior. In addition, multiple evaders are modeled as targets that can move irregularly to simulate a more realistic environment. So, we need an appropriate allocation strategy [5] for all agents to determine which target to track at the current time.

Decentralized multi-agent pursuit benefits from recent advances in deep reinforcement learning (DRL) [6–10]. However, these works do not take into consideration some important real-world limitations in motion constraints. Souza et. al. address such issues in [1] by introducing non-holonomic pursuers in bounded arenas to capture an evader. Though their results based on the twin delayed deep deterministic policy gradient (TD3) are promising, the collision of agents was not investigated, and they did not continue to study the environment where the multiple targets existed. In [5], the authors introduced a suboptimal but scalable approach that assigned individual agents to individual targets and could dynamically re-compute such assignments. This is called the universal target

allocation method, but it has not yet been validated in a multi-agent reinforcement learning framework. In addition, when tasks are executed under the centralized evaluation and decentralized execution of a multi-agent deep reinforcement learning algorithm, there is a significant gap in the policies learned by each agent. This is also an urgent problem that needs to be solved.

Facing these difficulties, we considered a decentralized multi-agent multi target pursuit problem where multiple UAVs worked together to find more efficient collision-free paths to capture all evaders. Then, the problem was transformed into a Markov decision process (MDP) and we combined MATD3 with a novel reward function and artificial potential field (APF) to solve the collision avoidance problem. Next, we specifically studied how to determine the current target that needed to be tracked for each agent. We designed two general allocation strategies, POLICY1 and POLICY2, as well as an allocation strategy, POLICY3, that considered agents and targets as a whole. In addition, we proposed an improved method, DAO-MATD3, for addressing the imbalance problem of agents' policies in the MATD3 algorithm. At last, we analyzed the design space (number of agents and targets, evaders' behavior and speed) and demonstrated the effectiveness of the proposed approach.

We make the following contributions in this paper:

- We design different trajectories for multiple moving targets and devise various dynamic multi-target allocation strategies from different perspectives to guide multi-agent multi-target pursuit.
- We reveal the problem of the agents' policy gaps in existing solutions and propose the dynamic actor network optimization to effectively address this issue.
- We retain the conflicts between agents and conduct comprehensive experiments to compare our proposed methods to demonstrate their effectiveness.

This paper is organized as follows: Section 2 introduces related work. In Section 3, the problem of multi-agent multi-target pursuit is defined in a formal manner. Section 4 describes the multi-agent deep reinforcement learning algorithm with collision avoidance, allocation strategies and dynamic actor network optimization, in detail. The methodology of experiments is presented in Section 5. The experimental results are discussed in Section 6. Finally, the paper is concluded in Section 7.

2. Related Work

2.1. Multi-Agent Deep Reinforcement Learning

Multi-agent reinforcement learning is an important branch in the field of multi-agent system research. By applying reinforcement learning technology, game theory, etc., to multi-agent systems, multi-agents can complete more complex tasks via interactions and decision making in a higher dimensional and more realistic scenario. In recent years, researchers have combined multi-agent reinforcement learning with other technologies to improve and innovate the algorithm from different aspects, such as pursuit-evasion[1,11,12], multi-agent control [13–15], swarm systems[16–18], scalability[19,20] and collision avoidance [21–23]. These studies have provided us with sufficient inspiration for multi-agent mobility control, physical constraints and conflict avoidance, but further research is needed to solve the problem of tracking multiple targets.

2.2. Multi-Target Pursuit

Different methods have been developed in the literature for multi-agent cooperation to achieve multiple goals, such as the task removal inference strategy [24], a shared Q-table for all agents [25] and the cooperative co-evolutionary multi-agent system [26]. However, these methods mainly focus on the completion of multiple tasks rather than having multiple targets within a pursuing task.

Multi-target pursuit has also been taken into account in multi-UAV scenarios. For example, in [27], an algorithm combining heuristic particle swarm optimization and a decentralized cooperative auction was proposed. However, this research was not conducted

on the basis of deep reinforcement learning. The authors in [28] proposed a max-consensus protocol to guarantee the consistency of the joint multi-target probability distribution (JMTPD). Then, a distributed partially observable Markov decision algorithm was used to make tracking decisions. However, this method was considered from the perspective of the agent itself, rather than an overall consideration of agents and targets. The authors in [29] proposed a simultaneous solution to these problems based on the probabilistic navigation function (PNF). The PNF provides an analytic solution, and guarantees the simultaneous interception of all targets while limiting the risk of agents to a given value. However, this method was not implemented within the framework of multi-agent reinforcement learning algorithms, and its performance was not optimal. In contrast to these studies, our method compares multiple dynamic target allocation strategies and addresses the policy gaps of agents when tracking multiple targets under a multi-agent reinforcement learning algorithm.

3. Multi-Agent Multi-Target Pursuit: Problem Definition

The multi-agent multi-target pursuing problem is defined as follows (scenario based on [1]):

- Multiple homogeneous pursuers $i (i \in \{1, 2, \ldots, n_{max}\})$ chase multiple homogeneous evaders $t (t \in \{1, 2, \ldots, m_{max}\})$;
- The pursuers are subject to non-holonomic kinematic constraints, whereas the evaders face no constraints and are omni-directional;
- Each pursuer only tracks one target for a given moment;
- A successful capture is defined as one evader(t) being in close proximity to one of the pursuers (i.e., $d_{i,t} < D_{cap}$), and a successful task is defined as all evaders being successfully captured;
- The goal of the trials is to minimize the capture time;
- If the evaders are not completely captured within $T_{timeout}$, the trial is deemed unsuccessful;
- Based on a globally observable environment, each pursuer can obtain all information from the environment, and all information can be directly obtained by sensors, with no communication between pursuers. The introduction of sensors is detailed in Section 5.2.

Specifically, the pursuers are assumed to move in a way that follows a unicycle model:

$$x_i = v \cos \phi_i \tag{1}$$

$$y_i = v \sin \phi_i \tag{2}$$

$$\phi_i = \omega : (\omega_{min} \leq \omega \leq \omega_{max}) \tag{3}$$

where (x, y) is the position, ϕ is the heading angle, v is the linear velocity and w is the angular velocity bounded in range $[\omega_{min}, \omega_{max}]$. Note that w is the only controllable variable determined by the policy of an agent.

The arena is a circular area with radius R_{arena}. Neither the pursuer nor the evader can leave the arena. Actions that would lead them outside the arena cause them to stay close to the border.

3.1. Evader Policies

As shown in Figure 1 the evaders move along one of the three different combination paths (single circle, concentric circles or under the repulsive influence of the pursuers). In the path of a single circle, all evaders move in the same direction on a circle of the same radius, and the initial positions of the evaders on the path are randomly chosen. In the path of concentric circles, each evader moves in the same direction on circles of different radii, and the initial positions of the evaders are randomly chosen on their own path. In the case of a repulsive path, we apply the repulsive force from the pursuers onto each evader.

The scene boundary also applies a repulsive force to the evaders. As a result, the evader's path is determined by the combined forces from both the pursuers and the scene boundary. The force is modeled in Equation (4).

$$\vec{v} = \sum_j \left(\frac{\overrightarrow{pos_e} - \overrightarrow{pos_j}}{d_j^2} \right) + \eta \frac{\hat{\tau}/2 * R_{\text{scene}} - \overrightarrow{pos_e}}{d_w^2} \tag{4}$$

where pos_e and pos_j are the coordinates of an evader and agent j, respectively, d_j is the distance between them, d_w is the distance of the evader to the scene boundary, τ is the angular speed of one evader toward the closest point on the boundary and the value of η for each evader is not the same to keep their movement paths different. If an evader passes the boundary during a time step, we reset its coordinates to its prior location and its linear velocity direction toward a pursuer, so that any invalid computations by the neural network on the boundary point can be avoided.

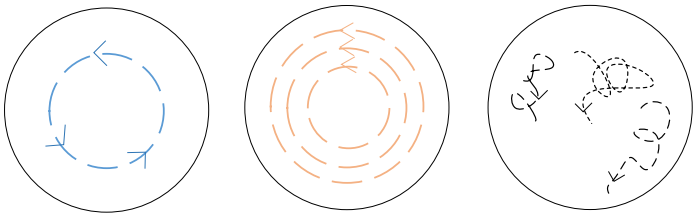

Figure 1. Three policies of the three evaders. (Blue: single circle; Orange: concentric circles; Black: under the repulsive influence of the pursuers).

3.2. Collision of Agents

Unlike evaders, the behaviors of pursuers are determined by the reinforcement learning strategy network, which assigns different angular velocities to each pursuer at a given moment, forming different individual and collective strategies and behaviors.

In the scenario of multi-agent multi-target tracking, we retain the possibility of collisions between agents. When the distance between pursuing agents is less than the collision threshold $D_{collide}$, the pursuing agents will collide. If there is a collision before all the evaders are captured, the trial is declared to be a "fail" and terminates. Although this assumption could be relaxed in some real-world scenarios, our work strives to completely avoid collisions. Our collision avoidance method is presented in next section.

Table 1 summarizes the parameters used to define the problem.

Table 1. Problem definition.

i	Pursuer agent
t	Evader target
n_{max}	Number of pursuer agents
m_{max}	Number of evader targets
D_{cap}	Distance threshold to capture
$D_{i,t}$	Distance between pursuer i and current target t
$D_{collide}$	Distance threshold for collisions between pursuers

4. Deep Reinforcement Learning with Multi-Agent Multi-Target Pursuit

In this section, we first convert the multi-UAV path planning problem into a Markov decision process (MDP), and then provide a detailed introduction to our collision avoidance method. Then, we present the multi-agent reinforcement learning algorithm used in this study. Finally, we introduce the relevant methods of the multi-agent multi-target pursuit problem in detail, including multi-target allocation strategies and dynamic actor network optimization.

4.1. MDP Formulation

The multi-UAV path planning problem, as a sequential decision-making problem, can be formulated as an MDP [30], which is described by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state-transition model, $\mathcal{R}$ is the reward function and $\gamma \in [0, 1]$ is a discount factor that trades off the importance of immediate vs. future rewards.

4.1.1. State Space $\mathcal{S}$

Assuming the ith pursuer is tracking the kth evader, $T \in t$. In contrast to when there is only a single evader, we added labels for different evaders to the state space S when there were multiple evaders. The state of the ith pursuer is defined as $s_i = [k, \psi_i, \dot{\psi}_i, s_{i,T}, s_{i,N}^{jn}]$, which consists of following four parts:

- k is the index of the evader tracked by the current pursuer, i. In order to reduce the instability of neural networks in training, the numerical magnitudes of k and other parameters in the state space should be kept consistent. So, we normalize k using this formula: $k = \dfrac{k+1}{m}$, where m is the maximum number of evaders that need to be tracked.
- ψ_i is the heading of the pursuer i with respect to a fixed world frame.
- $\dot{\psi}_i$ is the i^{th} pursuer's angular velocity w_i.
- $s_{i,T} = [d_{i,T}, \vec{d}_{i,T}, \alpha_{i,T}, \vec{\alpha}_{i,T}]$ is the relative state of the evader with respect to pursuer i, where $d_{i,T}$ is the distance between the evader and the pursuer, $\vec{d}_{i,T}$ is the vector from pursuer i to the evader, $\vec{\alpha}_{i,T}$ is the angular difference between the heading of pursuer i and the vector between pursuer i and the evader and $\alpha_{i,T}$ is the absolute value of $\vec{\alpha}_{i,T}$.
- $s_{i,N}^{jn} = [s_{i,j}, \forall j, j \neq i]$ is the joint vector of the relative state of the pursuer j with respect to pursuer i. In addition, $s_{i,j} = [d_{i,j}, \alpha_{i,j}]$, where $d_{i,j}$ is the Euclidean distance between pursuer i and j, and $\alpha_{i,j}$ is the heading error defined as the angle between the heading of pursuer i and the vector between pursuer i and j.

The representation of some notations in the state space is displayed in Figure 2.

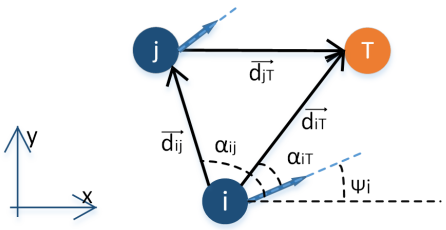

Figure 2. The state space for each pursuer i. T denotes the evader and j denotes one other agent, i.e., the pursuer j.

4.1.2. Action Space $\mathcal{A}$

Each pursuer's action $a \in \mathcal{A}$ is the angular velocity w, which is saturated to be in the interval $[w_{\min}, w_{\max}]$.

4.1.3. State-Transition Model $\mathcal{P}$

In an MDP, the state transition of an agent follows a Markov chain. Each agent takes action according to the current state, and then turns into the next state after interacting with the environment. The transition probability distribution is related to the applied algorithm.

4.1.4. Reward Function $\mathcal{R}$

As the evader moves continuously, leading to a dynamically changing environment, sparse rewards do not enable pursuers to learn the tracking strategy. Therefore, a dense reward function needed to be designed in this work, and the details are as follows:

$$r_i = \begin{cases} r_{\text{captor}}, & \text{if } d_{iT} \leq d_{\text{cap}} \\ r_{\text{helper}}, & \text{if } d_{jT} \leq d_{\text{cap}} \text{ and } d_{iT} > d_{\text{cap}}, \exists j \neq i \\ r_{\text{ins,i}} + r_{\text{rep,i}}, & \text{otherwise} \end{cases} \tag{5}$$

where d_{iT} and d_{jT} are the distances from pursuer i and pursuer j to the evader, respectively. d_{cap} is a distance threshold for capture, i.e., if $d_{iT} \leq d_{cap}$, pursuer i successfully catches the evader. r_{captor} is a large positive reward for successful evader capture, which is used to encourage each pursuer to catch the evader. r_{helper} is a positive reward for helping other pursuers capture the evader. In addition, $r_{\text{ins,i}}$ is designed to encourage the pursuer to move closer to the evader, and can be expressed as

$$r_{\text{ins,i}} = \begin{cases} (R_{\text{scene}} - d_{iT,t}), & \text{if } d_{iT,t} < d_{iT,t-1} \\ -d_{iT,t}, & \text{if } d_{iT,t} \geq d_{iT,t-1} \end{cases} \tag{6}$$

where R_{scene} is the width of the considered area, $d_{iT,t}$ and $d_{iT,t-1}$ are the distances between pursuer i and the evader at time step t and $t-1$, respectively. $r_{\text{rep,i}}$ is the reward term designed to avoid collision. More specifically, we include two distance thresholds, i.e, the sum of the radii of two pursuers' physical bodies, R_i, and the radius of the collision avoidance zone, D_{cz}, which can be found in Figure 3. If the distance between two pursuers, d_{ij} is smaller than R_i, the two pursuers will collide and pursuer i will receive a big penalty, r_{collide} . If $R_i < d_{ij} \leq D_{cz}$, pursuer i does not collide with pursuer j, but it steps into pursuer j's collision avoidance zone, and pursuer i will receive a d_{ij}-based penalty. Otherwise, the pursuer is safe, and $r_{\text{rep,i}} = 0$. This reward term can be mathematically expressed as

$$r_{\text{rep,i}} = \begin{cases} r_{\text{collide}}, & \text{if } d_{ij} \leq R_i, \exists j \neq i \\ r_{\text{collide}} \times \dfrac{(D_{cz} - d_{ij})}{(D_{cz} - R_i)}, & \text{if } R_i < d_{ij} \leq D_{cz}, \exists j \neq i \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

4.2. Multi-Agent Pursuit with Collision Avoidance

We designed a modified artificial potential field (APF) method to adapt to the multi-agent cooperative multiple moving evaders tracking scenario. Our goal was to plan collision-free paths for the pursuers to capture all the evaders. Therefore, to avoid collisions among agents, a potential field was added for each pursuer. The potential field was modeled as an effective area, and simplified to a circular region with radius D_{cz}. The potential field of all pursuers were designed to be the same size; an illustration can be found in Figure 3. From this figure, we observe that if the distance between two pursuers $d_{ij} > D_{cz}$, the pursuer is in a safe zone. If the pursuer enters into another agent's potential field (named the collision avoidance zone), i.e, $R_i < d_{ij} \leq D_{cz}$, this pursuer receives a force from the other pursuer and should perform collision avoidance actions. The influence of this force on the pursuer's action is denoted as $\hat{\vec{e}}_{ij}$. In addition, we designed an attractive potential field around the evader, the influence of which is denoted as $\hat{\vec{e}}_{Ti}$. Thus, the influence of all forces can be summarized as

$$\overrightarrow{rp}_i = \alpha \times \hat{\vec{e}}_{ij} + \beta \times \hat{\vec{e}}_{Ti} \tag{8}$$

where α and β are constants. For a pursuer, the action chosen by the reinforcement learning policy can be formulated as

$$\overrightarrow{A_q} = [\cos \varphi_{t+1}, \sin \varphi_{t+1}] \tag{9}$$

where $\varphi_{t+1} = \vec{w} \times dt + \varphi_t$ and φ_t is the orientation of the pursuer at time step t. $\vec{w}$ is the output of the policy. Influenced by the APF, the pursuer's final velocity is modified to

$$A = \left(\gamma \times \overrightarrow{A_q} + \widehat{\overrightarrow{rp}_i}\right) \times v_{\text{base}} \tag{10}$$

where v_{base} is a fixed speed and γ is a constant.

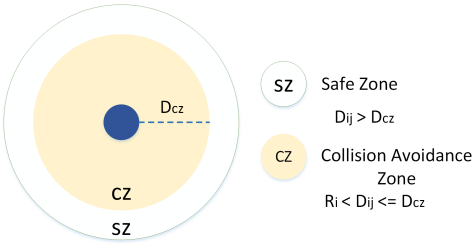

Figure 3. Illustration of the artificial potential field around a pursuer.

4.3. Multi-Agent Deep Reinforcement Learning Algorithm

A multi-agent twin delayed deep deterministic policy gradient (MATD3 [31]) was used to learn the policy via centralized learning and distributed execution. Particularly, the cooperation among multiple pursuers was encouraged through centralized evaluation, and each pursuer executed its own distributed action according to its own observations. Note that we treated all agents as homogeneous, and each agent's experience was used for the learning of its own policy.

4.4. Multi-Agent Multi-Target Pursuit

In this section, we explore the problems faced by multi-agent multi-target pursuit tasks and demonstrate our solutions. One of the problems is that when there are multiple targets, it is necessary to determine which target to track currently for each pursuer, as each pursuer can only track a single target at a time. Another problem is that during the experimental process of the cooperative task of multiple agents pursuing multiple targets, there is a significant gap in the performance of each pursuer's strategy. This is because the decentralized execution architecture of the current off-policy MADRL algorithm, MATD3, allows each pursuer to have its own action network. The details of our method are presented below.

4.4.1. Allocation Strategy of Multi-Target Pursuit

In this part, we provide various tracking methods for pursuers when multiple targets exist. We needed to find a suitable allocation strategy to determine which target should be tracked by each pursuer in a given moment, which would affect the efficiency of multi-agent multi-target task completion. We considered the current positions of agents and targets for this task, but not their potential future positions. For multiple pursuers and targets, we focused on the scenarios where the number of pursuers was equal to or greater than the number of targets. When the number of pursuers was less than the number of targets, our approach allowed for a simple solution. All pursuers focused on the portion of targets with an equal number. Then, if a target was captured, another target was assigned to the pursuer. We studied three different allocation strategies (POLICY1, POLICY2 and POLICY3) and the specific details are shown in Figure 4.

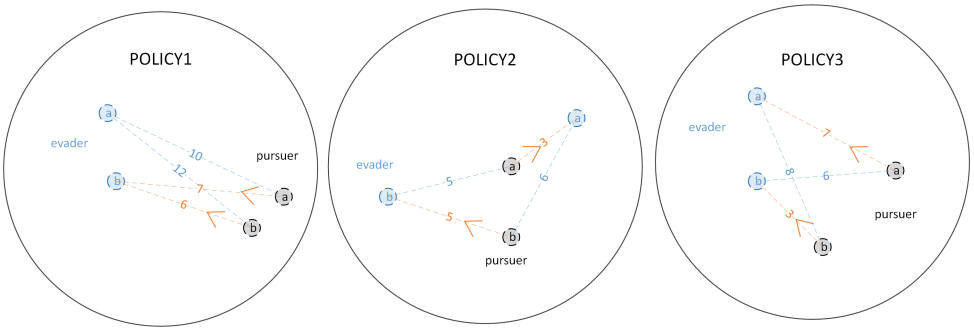

Figure 4. Presentation of three policies under the task of two agents pursuing two targets. (Black circle: the pursuer; Blue circle: the evader; a, b: the serial number of the pursuer or evader; Orange line: the actual target selection of the pursuer; Blue line: the possible target selection of the pursuer; Number: the distance cost).

POLICY1: This allocation strategy inherits its idea from the task of multiple agents tracking a single target. Even if there are multiple targets, all pursuers will only track one target at a given moment. When this target is captured, a new one will be selected from the remaining targets for all pursuers to track. Selecting a target requires calculating the average distance between each target and all pursuers at the current moment, then determining the target that needs to be tracked based on the shortest average distance. From Figure 4, we see that the average distance between evader a and pursuers a and b is $(10 + 12)/2$, which is greater than the average distance between evader b and pursuers a and b, which is $(6 + 7)/2$. So, at the moment shown in the figure, all pursuers choose evader b as the current target.

POLICY2: This allocation strategy originates from the general idea of a single agent tracking multiple targets. When multiple targets exist, from the perspective of each agent, each pursuers will choose to track the closest target to themselves. If a target has already been assigned to a pursuer, other pursuers cannot choose this target, and they will choose the next closest target from those remaining. When a target is captured, the redundant pursuer will be assigned to one of the selected targets based on the nearest distance criterion. From Figure 4, it can be clearly seen that the distances between pursuer a and evaders a and b are 3 and 5, respectively. So, pursuer a will track the closest evader a, and similarly, evader b will be assigned to pursuer b.

POLICY3: We observe that the pairing of a pursuing agent and target is limited to $n_{max} \times m_{max}$, given n_{max} agents and m_{max} targets. Therefore, by traversing, we can obtain all possible allocation combinations between agents and targets. Then, by evaluating the suitability of each allocation combination, the optimal allocation combination between agents and targets at the current moment can be obtained. From a general perspective, our method uses the distance between the agent and the target as the evaluation criterion, also known as cost. The distance between each agent and target is called individual cost, and the sum of individual costs included in an allocation combination is called the summation cost. The larger the distance between the agent and the target, the longer the tracking time, and the completion time of the entire task also increases. Thus, when comparing various allocation combinations, individual cost is the main basis of evaluation, and summation cost is the secondary factor. When calculating, it is necessary to find the maximum value of individual costs contained in each allocation combination, and then find the allocation combination corresponding to the minimum value from these values. If there is are multiple allocation combinations with a minimum value of equal size, then we would select the one with the smallest summation cost as the optimal allocation combination at the current time. In Figure 4, we define pursuer a tracking evader a and pursuer b tracking evader b as an allocation combination, denoted as $(a, a; b, b)$. The individual costs included in this allocation combination are recorded as $(7, 3)$. Obviously, the summation cost is 10. Similarly, the allocation combination $(a, b; b, a)$ has the individual costs $(6, 8)$ and a summation cost of 14. We can calculate the maximum individual costs for the two allocation combinations

as 7 and 8, respectively. Thus, we choose the combination $(a, a; b, b)$, corresponding to the minimum value of 7, as the optimal allocation combination.

4.4.2. Dynamic Actor Network Optimization of MATD3

During our study, we noticed that there was significant performance disparity among the policies of the agents during the training process, as illustrated in Figure 5.

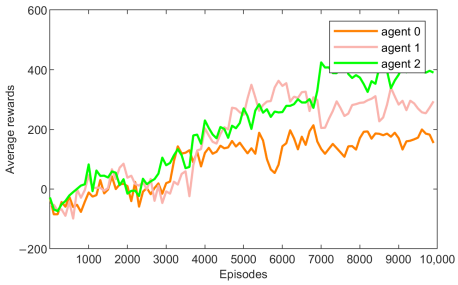

Figure 5. The average rewards of each pursuer in a task of three pursuers tracking three evaders.

It can be clearly seen from the figure that there is no significant difference in the average rewards received by the three pursuers at the beginning of the training phase. However, in the subsequent training process, the average reward received by pursuer 2 continued to increase, whereas the average reward received by pursuer 0 basically did not increase. The average reward received by pursuer 1 was between the other two pursuers. In other words, most of the targets in this task were captured by pursuer 2, and the other two pursuers' strategies were lackluster compared with the strategy of pursuer 2.

In collaborative tasks, an imbalance between the strategies of multiple agents can reduce the efficiency of completing the entire task. When using the MATD3 algorithm, this can be attributed to the imbalanced performance of each agent's actor network. We made adjustments to the execution process of the original algorithm, and the structure of our method is shown in Figure 6.

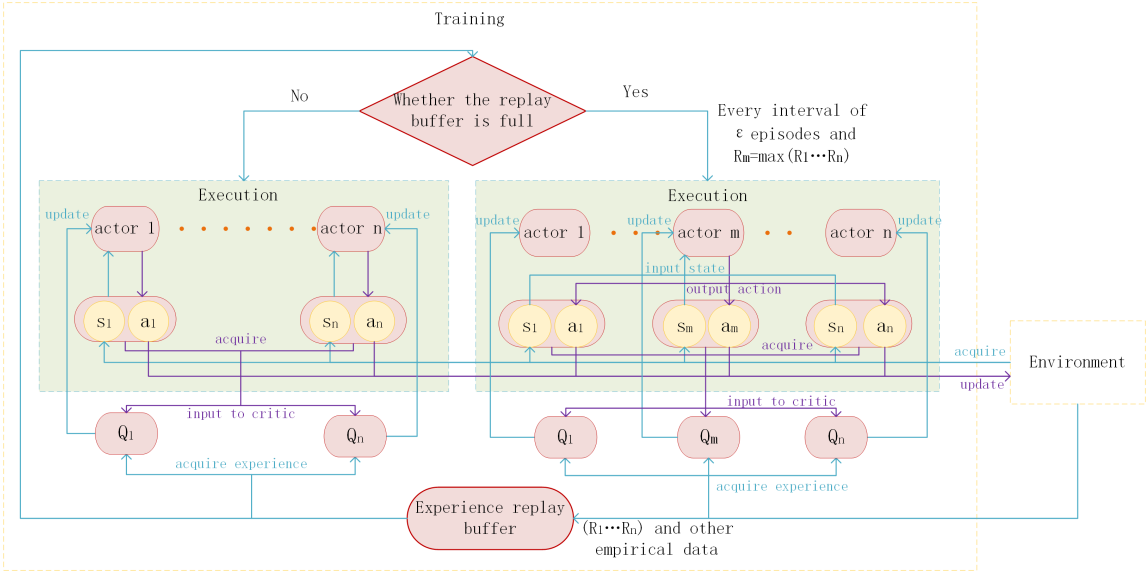

Figure 6. Overview of the proposed algorithm.

There are n agents and each agent has its own actor network and critic network. At the beginning of training, each agent obtains the current state of the environment as input to the actor network and returns the action to be executed. Then, through the interaction of action information with the environment, the agent obtains experience information, such as the next moment's state and the reward, which is stored in the experience replay buffer.

The critic network takes not only the current agent experience as input, but also obtains experience from all other agents. This is the classic centralized evaluation decentralized execution architecture. When the experience replay buffer does not meet the training requirements, our method is the same as the original algorithm, MATD3, and the network does not change at this stage. After the experience replay buffer is full, every ε episode interval, we compare the average reward received by each agent in the previous ε episode, and in this paper, we used the experience value $\varepsilon = 100$. We assume that agent m, with the highest average reward, has the optimal actor network strategy, and then the states of all agents in the current ε episode will be inputted to actor m to select actions for all agents, which becomes a centralized execution architecture. However, we do not remove the dispersed actor networks, and all actor networks are still updated, dynamically selected and optimized during the training process. We called this method the dynamic actor network optimization of MATD3 (DAO-MATD3). And the pseudo code of Algorithm 1 is as follows.

Algorithm 1: Dynamic actor network optimization of MATD3 (DAO-MATD3)

Initailize. Actor network parameters θ, critic network parameters ϕ, target network parameters $\phi^{'}$ and experience replay buffer D;
for *episodes* $\leftarrow 1$ **to** M **do**
  Reset environment;
  Receive all agents' joint local state s;
  for *timestep* $\leftarrow 1$ **to** $M^{'}$ **do**
    while *replay buffer D is not full* **do**
      For each agent i, select action a_i from the $actor_i(s_i)$;
      Execute joint action a to receive joint environmental immediate reward r and the next moment joint state s^{t+1};
      Store (s, a, r, s^{t+1}) into replay buffer D;
    end
    if *episodes* % ε = *0* **then**
      For each agent i, calculate the average reward obtained from the previous ε episodes R_i;
      Select the maximum one from R and mark it as R_m;
    end
    For each agent i, select action a_i from the $actor_m(s_i)$;
    Execute joint action a to receive joint environmental immediate reward r and the next moment joint state s^{t+1};
    Replace experience in replay buffer D with new (s, a, r, s^{t+1});
    //Update actor and critic network;
    Randomly sample B samples $(s_k, a_k, r_k, s_k^{t+1})$ from buffer D;
    Set $y_{k1} = r_k + \gamma Q^{'}_{\phi_1^{'}}(s_k^{'}, a_k^{'})|_{a_k^{'}=\pi_\theta(s_k^{'})}$;
    Set $y_{k2} = r_k + \gamma Q^{'}_{\phi_2^{'}}(s_k^{'}, a_k^{'})|_{a_k^{'}=\pi_\theta(s_k^{'})}$;
    Set $y_{min} = Min(y_{k1}, y_{k2})$;
    Update ϕ by minimizing $\frac{1}{B}\sum_k \left(y_{min} - Q_\phi(s_k, a_k)\right)^2$;
    Update θ by the sampled policy gradient $\frac{1}{B}\sum_k \nabla_\theta \pi_\theta(s_k) \nabla_{a_k} Q_\phi(s_k, a_k)$;
    if *update target network* **then**
      $\phi^{'} \leftarrow \tau\phi + (1-\tau)\phi^{'}$
    end
  end
end

5. Experiments

In this section, we introduce the specific parameters of our environment and the relevant parameters of the neural network. We also provide some explanations on how to achieve this environment in practical scenarios of UAVs. Finally, we specify the various methods that are presented in Section 6.

5.1. Environments and Parameters

We designed a framework to simulate the environments of multiple pursuers tracking multiple evaders using the MADRL algorithm being studied. We modeled the position and speed (both linear and angular) of the agents and evaders at each time step, implemented the MADRL algorithm and our improved algorithm, and evaluated performance in the same framework. The performance indicators used mainly included the success rate, average time steps and average rewards.

Specifically, we adopted the following environmental parameters: All the environments were episodic. When the number of agents and targets was between 2 and 6, the maximum number of episodes was 10,000, and the maximum sustainable time steps for each episode was 500. The radius of the collision avoidance zone (D_{cz}) was 0.14 and the parameter γ in Equation (10) was set as $1/2$. As the number of agents and targets continues to increase to between 7 and 9, in order to maintain convergence in training, we adjusted the maximum number of episodes and maximum sustainable time steps for each episode to 15,000 and 1000, respectively. At the same time, in order to reduce the collision risk between pursuers and ensure the effectiveness of the collision avoidance method, we adjusted parameters D_{cz} and γ to 0.1 and $1/10$, respectively. The parameters, r_{captor} and r_{helper} in the reward function (Equation (5)) were set as 300 and 30, respectively. The parameter, $r_{collide}$ in the reward function (Equation (7)) was set to a smaller value of -0.1 to reduce the impact on tracking strategy convergence. We assumed the pursuers and evaders were the same size of 0.04, and the diameter of the scene was 2. The capture radius was 0.07. The speed of pursuers v_p was 0.116/time step. We varied the evaders' speed between 0.093 and 0.139 per time step (0.8× through 1.2× the pursuers' speed). The pursuers' maximum angular speed was set as $\pi/10$ per time step. The number of pursuers varied from 2 to 9, and their locations were initialized randomly in a circular area with a radius of 0.35. The evaders were initialized at a random location between the scene boundary and a circle with a radius of 0.8.

We adopted the following network hyperparameters: When the number of agents and targets was between 2 and 6, the length of the replay buffer was 1×10^5. As the number of agents and targets continued to increase to between 7 and 9, the length of the replay buffer was adjusted to 2×10^5. The discount factor, batch size and learning rate were set as 0.95, 1024 and 1×10^{-3}, respectively. In addition, the higher the updating frequency of the allocation strategy, the closer it was to the performance of the strategy itself; however, it also increased the computational workload. To balance performance with the computational workload of the allocation strategy, we performed reassignment every 10 time steps, which was an experience value. Due to the complexity of the allocation strategy and adjustment of the network structure, our method usually required twice as much computation and training time as the general MATD3 method. On the RTX3090 graphics card, the program running through our algorithm for three agents pursuing three targets condition typically took 200 to 300 s per 100 episodes.

5.2. Practical Scenario Application

Our approach simulates a scenario where three autonomous four-axis UAVs attempt to capture three moving target UAVs. The UAVs fly in a circular area with a radius of 5 m, and both pursuer and evader UAVs are limited to the same altitude. The pursuer UAVs are provided with RTK (real-time kinematic) technology to achieve real-time position information of all UAVs. Different from ordinary GPS positioning, RTK positioning can reach an accuracy of less than 1 cm and the measurement has delays at the millisecond

level. While in flight, the pursuer UAVs obtain their respective motion directions with an on-board IMU tilt angle sensor, which has a three-axis accelerometer and a three-axis gyroscope module and supports a feedback rate of milliseconds. Therefore, the UAV can obtain the current angle and angular velocity in real-time.

5.3. Method Statements

In summary, we conducted extensive experiments to compare the performance of the following approaches. The basic algorithm for all experiments was MATD3, so its labeling is omitted.

- POLICY1 and POLICY2: These two allocation strategies were general methods and we mainly used them as the baseline methods for POLICY3.
- POLICY3-optimal and POLICY3-sample: Assuming the number of pursuers and evaders at the current moment were n and m, respectively, when $m = n$, we paired pursuers and evaders one to one. So, the number of allocation combinations could be calculated as the factorial of n (denoted as $n!$).
  However, when $m < n$, in order to compare the most combinations, we allowed multiple pursuers to track the same target. The number of allocation combinations could be calculated as m^n. If we chose to traverse all allocation combinations, we called this approach POLICY3-optimal.
  As n and m increased, the computational workload required to compare all combinations also greatly increased. When $n > 6$, we randomly selected a portion of the obtained combinations to participate in the calculation to reduce computational workload. We called this approach POLICY3-sample. In next section, the number of combinations we sample is 500.
- POLICY3-no: When $n > 6$, we still used POLICY3-sample, but with the collision avoidance parameter when $n \leq 6$.
- POLICY1-DAO and POLICY3-optimal-DAO: These two methods refer to a combination of an allocation strategy and our improved algorithm.

6. Results

In this section, we compare the various methods introduced in Section 5 through extensive experiments. The evaluation metrics mainly included the success rate, average time steps and average rewards. Due to the fact that the evader policy of the repulsive path was more general and the model trained in this environment could migrate well to the fixed path environments, we will mainly demonstrate the results obtained from the repulsive path environment.

We provide details of the evaluation results in the following subsections: Specifically, in Section 6.1, we evaluate the pros and cons of the three allocation strategies in the environment where the evaders are repulsive to pursuers. We evaluate with both a small and larger number of agents and targets with the relative speed of 0.8. In addition, we also provide the performance of the three allocation strategies under different relative speeds with three agents and three targets. In Section 6.2, we demonstrate the performance of our improved algorithm under the same allocation strategy to address the problem of imbalanced learning strategies among multiple pursuers, and we also demonstrate the impact of our improved algorithm on task completion efficiency. Finally, in Section 6.3, we validate the effectiveness of each part of our proposed improvement method in the same experimental environment. Furthermore, if $n < m$ at the initial stage, we only need to randomly and dynamically select n targets from m targets to execute our method until all targets are captured.

6.1. Performance of Allocation Strategy

The first set of experiments showed how the allocation strategies performed when the number and relative speed of the agents and targets changed. For the repulsive path environment, we studied the success rate and average time steps spent on successful tasks.

When the number of agents and targets n is between 2 and 6, the collision avoidance parameters of the environment remain unchanged. Figure 7 shows that when $n > 3$, the success rate of POLICY1 significantly decreases. The success rate of POLICY2 also decreases, but the magnitude is smaller than that of POLICY1. Through our experiments, 90 and 80% of the declining success rates of POLICY1 and POLICY2, respectively, were due to collisions. POLICY1 had the greatest risk of collisions between agents, whereas POLICY2 also had a certain risk of collision between agents. POLICY3-optimal maintained the lowest risk of collisions between agents and achieved the highest success rate; As the number of agents and targets increased, the agents under POLICY1 maintained a relatively large task completion time, but the overall change in their time spent was stable, indicating a certain balance between the number of agents and task difficulty under this method. When $n \leq 4$, the task completion time of POLICY2 was close to that of POLICY3-optimal. However, when $n > 4$, the time spent by POLICY2 increased significantly, indicating that as the number of agents increased, the effectiveness of each agent using a locally optimal strategy to select one target decreased. By leveraging a strategy that considers both agents and targets as a whole, POLICY3-optimal always maintained the lowest task completion time under changes in the number of agents and targets.

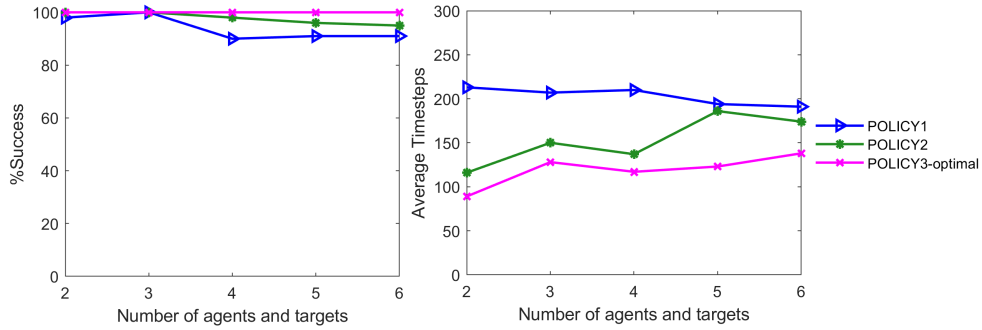

Figure 7. Success rate and average time steps for different numbers of agents and targets (when evaders followed repulsive path).

When $n > 6$, we adjusted the collision avoidance parameters of the environment to eliminate the risk of collision between agents. Figure 8 shows that when $n = 7$, all the three allocation strategies can achieve the highest success rate. However, as n continues to increase, the success rates of POLICY1 and POLICY2 decrease to a certain extent, and their task completion time also increases significantly. This indicates that as task complexity increases, the astringency of the two general allocation strategies continues to decrease. Although we adopted a random selection approach to implement POLICY3 while maintaining a small computational cost, POLICY3-sample still maintained the highest success rate and a stable task completion time when there were a large number of agents and targets. POLICY3-no, with no modifying collision avoidance parameters, could not solve direct conflicts between agents in more complex environments. Its success rate decreased with the increase in the number of agents and always maintained a larger task completion time.

Figure 9 shows that (a) As the relative speed increases, the success rates of all the three allocation strategies show a downward trend, while POLICY3-optimal shows a slightly smaller decline compared to the other two methods, indicating that POLICY3-optimal has better stability at different relative speeds; (b) As the relative speed increases, the task completion time of the three allocation strategies increases to varying degrees, indicating that the difficulty of the task is constantly increasing. At the same time, the difference in time spent by the three strategies is gradually narrowing. This indicates that the performance improvement of individual allocation strategy cannot effectively combat the increase in difficulty caused by the increase in relative speed.

These results demonstrate that regardless of the number of agents and targets and different relative speeds, the performance of the POLICY3 method in tasks of multi-agent pursuing multi-target surpasses the other two general methods.

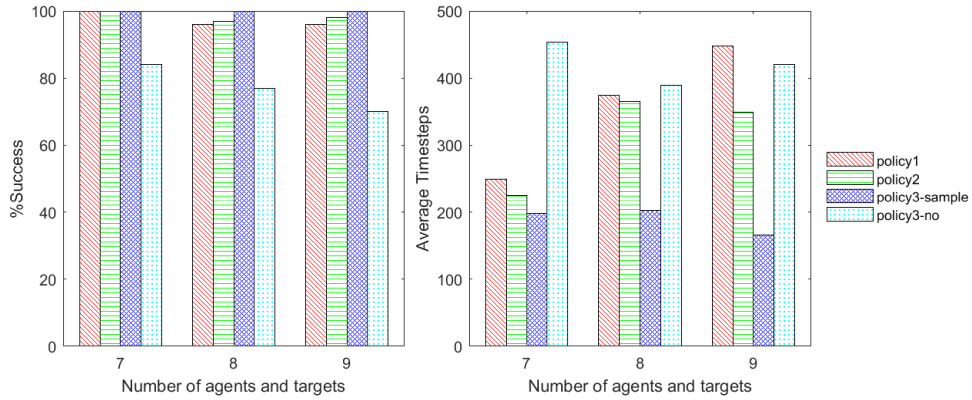

Figure 8. Success rate and average time steps for more agents and targets (when evaders follow repulsive path).

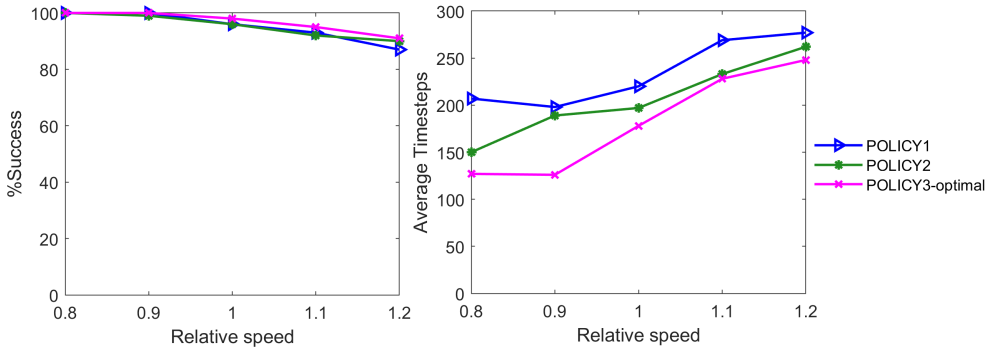

Figure 9. Success rate and average time steps with respect to relative speed (when three agents and three targets follow repulsive path).

6.2. Performance of DAO-MATD3

This set of experiments show how our improved algorithm performs when all experimental parameters are the same except for the algorithm used. The allocation strategies used was POLICY3-optimal. For the repulsive path environment, we studied the average rewards of each agent and average time steps spent on successful tasks.

Figures 10 and 11 illustrate that: (a) The method supported by only one allocation strategy always exhibits a phenomenon where there is a significant gap in the policies learned by each agent during training. According to the average rewards of each agent, it can be seen that the pursuing enthusiasm of the agents varies. After adding our improved algorithm, DAO-MATD3, the average rewards received by each agent were basically the same, indicating that the gap in the pursuing policies of agents had basically disappeared. (b) From the average rewards received by each agent, agent 2 received the most rewards, indicating that it had the highest pursuing motivation and longest total distance traveled in the task. However, the longest moving distance of all agents determines the completion time of the entire task. So, when using the DAO-MATD3 algorithm, the average time steps spent on tasks of multiple agent pursuing multiple targets can converge to within 100, whereas the average time steps spent by the MATD3 method are stable at over 200.

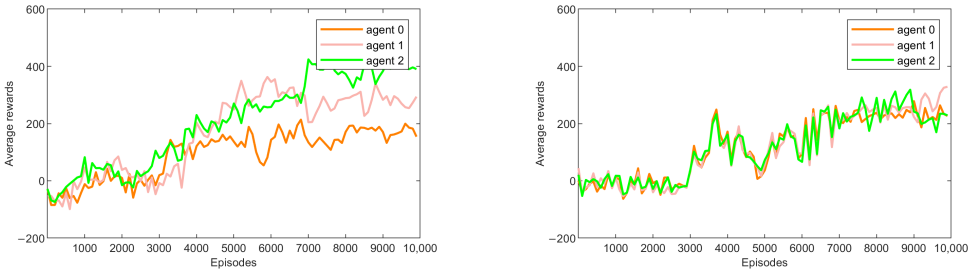

Figure 10. Reward comparison between our algorithm and MATD3 in task of three agents tracking three targets (when relative speed is 1): (**a**) Average rewards of each agent using MATD3; (**b**) Average rewards of each agent using DAO-MATD3.

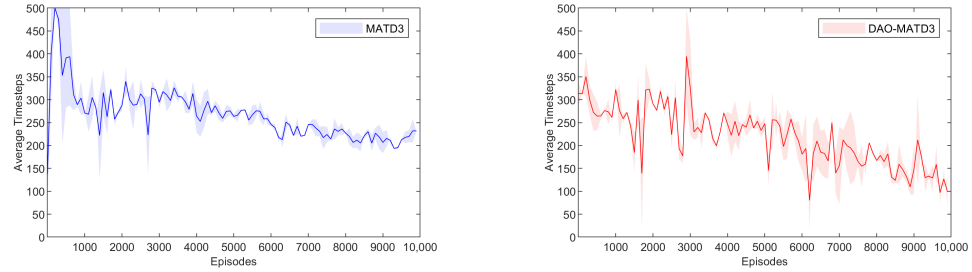

Figure 11. Time step comparison between our algorithm and MATD3 in task of three agents tracking three targets (when relative speed is 1): (**a**) Average time steps spent using MATD3; (**b**) Average time steps spent using DAO-MATD3.

6.3. Performance of Ablation

This set of experiments shows how the various parts of our complete method perform when other conditions are the same. For the repulsive path environment, we studied the success rate and average time steps spent on successful tasks.

Figure 12 shows that: (a) At most relative speeds, our method POLICY3-optimal-DAO achieved the highest success rate, and after replacing POLICY3-optimal with the weaker performance of POLICY1 in our method, the success rate only showed a slight decrease. When we did not retain our improved algorithm and only use the allocation strategy and MATD3 algorithm to execute tasks, the success rate was significantly reduced. This indicates that our improved algorithm could have a positive effect on the success rate of tasks of multiple agents pursuing multiple targets. (b) At all relative speeds, our method took significantly lower average time steps than other methods. After replacing the best allocation strategy, the average time steps taken increased to a certain extent, but were still much lower than when DAO-MATD3 was not used.

These results demonstrate that both the allocation strategy POLICY3, which considers the overall cost, and the DAO-MATD3 algorithm, which reduces the policy gap between agents, have important and positive effects on the efficiency of multi-agent multi-target pursuit tasks.

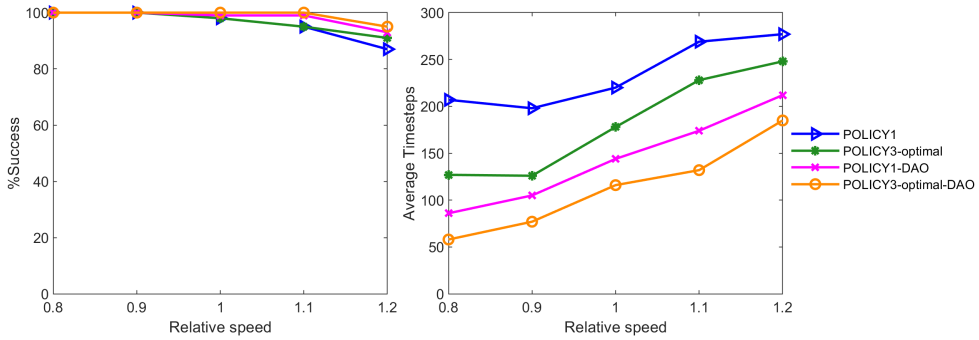

Figure 12. Ablation study on success rate and average time steps with respect to relative speed (when three agents and three targets follow repulsive path).

7. Conclusions

In this paper, we have proposed a deep reinforcement learning method called POLICY3-DAO-MATD3 to solve the problem of target allocation and imbalanced agent policies in multi-agent multi-target pursuit tasks. This approach leverages a strategy that considers agents and targets as a whole, as well as dynamic actor network optimization, effectively improving the task execution efficiency when multiple agents track multiple targets. The extensive evaluation results have demonstrated the performance advantages of POLICY3-DAO-MATD3, including reducing the risk of collision between agents, improving the success rate of pursuit, and effectively reducing the time spent on pursuing tasks. These are essential for real-world pursuit-evasion applications. In addition, our research can be further developed by further optimizing algorithms to reduce computational costs when there are more agents and targets, determining how to validate more multi-agent algorithms in more complex environments, etc.

Author Contributions: Material preparation, data collection and analyses were performed by B.H. The first draft of the manuscript was written by B.H. The review and editing of the manuscript were performed by X.W. The supervision of the project was performed by L.S. and L.Z. Project administration was performed by L.S., L.Z. and X.W. All authors commented on previous versions of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Changzhou Municipal Advanced Technologies Research Center program (CM20193007) and the Changzhou Sci & Tech Program (No. CJ20220241).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors are affiliated with Changzhou University and the Changzhou Key Laboratory of Urban Big Data Technology and Applications.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. De Souza, C.; Newbury, R.; Cosgun, A.; Castillo, P.; Vidolov, B.; Kulić, D. Decentralized Multi-Agent Pursuit Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4552–4559. [CrossRef]
2. Janosov, M.; Virágh, C.; Vásárhelyi, G.; Vicsek, T. Group chasing tactics: How to catch a faster prey. *New J. Phys.* **2017**, *19*, 053003. [CrossRef]
3. Li, Y.; Jin, D.; Hui, P.; Han, Z. Optimal Base Station Scheduling for Device-to-Device Communication Underlaying Cellular Networks. *IEEE J. Sel. Areas Commun.* **2015**, *34*, 27–40. [CrossRef]
4. Angelani, L. Collective Predation and Escape Strategies. *Phys. Rev. Lett.* **2012**, *109*, 118104. [CrossRef] [PubMed]
5. Xie, F.; Botea, A.; Kishimoto, A. A Scalable Approach to Chasing Multiple Moving Targets with Multiple Agents. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, VIC, Australia 19–25 August 2017.

6. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30.
7. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 8–12 May 2017; pp. 66–83.
8. Hüttenrauch, M.; Šošić, A.; Neumann, G. Deep Reinforcement Learning for Swarm Systems. *J. Mach. Learn. Res.* **2019**, *20*, 1966–1996.
9. Xu, L.; Hu, B.; Guan, Z.; Cheng, X.; Li, T.; Xiao, J. Multi-agent Deep Reinforcement Learning for Pursuit-Evasion Game Scalability. In Proceedings of the 2019 Chinese Intelligent Systems Conference, Shanghai, China, 29–31 May 2019; pp. 658–669.
10. Yan, F.; Wang, J.; Du, C.; Hua, M. Multi-Objective Energy Management Strategy for Hybrid Electric Vehicles Based on TD3 with Non-Parametric Reward Function. *Energies* **2023**, *16*, 74. [CrossRef]
11. Zhang, R.; Zong, Q.; Zhang, X.; Dou, L.; Tian, B. Game of Drones: Multi-UAV Pursuit-Evasion Game With Online Motion Planning by Deep Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 7900. [CrossRef] [PubMed]
12. Wang, S.; Wang, B.; Han, Z.; Lin, Z. Local Sensing based Multi-agent Pursuit-evasion with Deep Reinforcement Learning. In Proceedings of the 2022 China Automation Congress (CAC), Xiamen, China, 25–27 November 2022; pp. 6748–6752. [CrossRef]
13. Bai, W.; Cao, L.; Dong, G.; Li, H. Adaptive Reinforcement Learning Tracking Control for Second-Order Multi-Agent Systems. In Proceedings of the 2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS), Dali, China, 24–27 May 2019; pp. 202–207. [CrossRef]
14. Luo, Z.; Zhang, P.; Ding, X.; Tang, Z.; Wang, C.; Wang, J. Adaptive Affine Formation Maneuver Control of Second-Order Multi-Agent Systems with Disturbances. In Proceedings of the 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), Shenzhen, China, 13–15 December 2020; pp. 1071–1076. [CrossRef]
15. Wang, L.; Li, J.; Liu, X.; Fang, Y. Event-Triggered Fault-tolerant Model Predictive Control of Nonlinear Multi-agent System with Time Delay and Parameter Uncertainty. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; pp. 5350–5355. [CrossRef]
16. Zuo, J.; Liu, Z.; Chen, J.; Li, Z.; Li, C. A Multi-agent Cluster Cooperative Confrontation Method Based on Swarm Intelligence Optimization. In Proceedings of the 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Nanchang, China, 26–28 March 2021; pp. 668–672. [CrossRef]
17. Biswas, S.; Anavatti, S.G.; Garratt, M.A. Particle swarm optimization based co-operative task assignment and path planning for multi-agent system. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–6. [CrossRef]
18. Tahifa, M.; Boumhidi, J.; Yahyaouy, A. Swarm reinforcement learning for traffic signal control based on cooperative multi-agent framework. In Proceedings of the 2015 Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 25–26 March 2015; pp. 1–6. [CrossRef]
19. Andrade, C.; Garrido, C.; Peters, A.; Vargas, F. A low cost experimental platform for the study of scalability issues in multi-agent systems. In Proceedings of the 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), Valparaiso, Chile, 13–27 November 2019; pp. 1–6. [CrossRef]
20. Yang, N.; Ding, B.; Shi, P.; Feng, D. Improving scalability of multi-agent reinforcement learning with parameters sharing. In Proceedings of the 2022 IEEE International Conference on Joint Cloud Computing (JCC), Fremont, CA, USA, 15–18 August 2022, pp. 37–42. [CrossRef]
21. Basjaruddin, N.C.; Noor, Z.I.R.; Widyantoro, D.H. Multi Agent Protocol for Cooperative Rear-end Collision Avoidance System. In Proceedings of the 2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI), Denpasar, Indonesia, 21–22 September 2019; pp. 23–26. [CrossRef]
22. Liu, J.; Zhang, C.; Huang, C.; Zhang, H.; Wang, Z.; Kong, D. Formation Control Strategy of Multi-agent Systems with Obstacle Avoidance. In Proceedings of the 2020 12th International Conference on Advanced Computational Intelligence (ICACI), Dali, China, 14–16 August 2020; pp. 138–143. [CrossRef]
23. Lu, M.; Zou, Y.; Li, S. Multi-agent formation control with obstacle avoidance based on receding horizon strategy. In Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019, pp. 1361–1366. [CrossRef]
24. Li, J.; Chen, F.; Wang, C.; Chen, Y.; Huang, Y.; Wnag, X. A performance-impact based multi-task distributed scheduling algorithm with task removal inference and deadlock avoidance. *Auton. Agents Multi-Agent Syst.* **2023**, *37*, 30. [CrossRef]
25. Daavarani Asl, Z.; Derhami, V.; Yazdian-Dehkordi, M. A new approach on multi-agent Multi-Objective Reinforcement Learning based on agents' preferences. In Proceedings of the 2017 Artificial Intelligence and Signal Processing Conference (AISP), Shiraz, Iran, 25–27 October 2017; pp. 75–79. [CrossRef]
26. Zhang, Z.; Sun, X.; Hou, L.; Chen, W.; Shi, Y.; Cao, X. A cooperative co-evolutionary multi-agent system for multi-objective layout optimization of satellite module. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 147–151. [CrossRef]

27. Liu, B.; Qin, Z.; Wang, R.; Gao, Y.b.; ping Shao, L. A hybrid heuristic particle swarm optimization for coordinated multi-target assignment. In Proceedings of the 2009 4th IEEE Conference on Industrial Electronics and Applications, Xi'an, China, 25–27 May 2009; pp. 1929–1934. [CrossRef]
28. Zhao, Y.; Wang, X.; Wang, C.; Cong, Y.; Shen, L. Systemic design of distributed multi-UAV cooperative decision-making for multi-target tracking. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 132–158. [CrossRef]
29. Hacohen, S.; Shoval, S.; Shvalb, N. Multi agents' multi targets mission under uncertainty using probability navigation function. In Proceedings of the 2017 13th IEEE International Conference on Control and Automation (ICCA), Ohrid, North Macedonia, 3–6 July 2017, pp. 845–850. [CrossRef]
30. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
31. Ackermann, J.; Gabler, V.; Osa, T.; Sugiyama, M. Reducing Overestimation Bias in Multi-Agent Domains Using Double Centralized Critics. *arXiv* **2019**, arXiv:1910.01465.