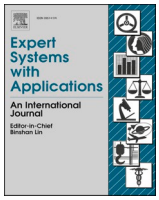
Expert Systems With Applications 231 (2023) 120643

Contents lists available at ScienceDirect

Expert Systems With Applications

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



A multi-agent reinforcement learning algorithm with the action preference selection strategy for massive target cooperative search mission planning

Xiaoyan Wang , Xi Fang *

School of Science, Wuhan University of Technology, Wuhan 430070, China

A R T I C L E I N F O

Keywords:
Reinforce algorithm
Multi-agent
Cooperative target search
Action selection strategy

A B S T R A C T

Target search is widely applied in military reconnaissance, geological exploration and personnel search and rescue. Most target search algorithms perform well in single target search but are inefficient or even ineffective in multi-target search. To solve the multi-target search problem in an uncertain environment, this paper constructs a multi-agent massive target cooperative search mission planning model and proposes an improved reinforcement learning algorithm using the action preference selection strategy. Based on the Reinforce algorithm, this algorithm solves the problem of invalid searches within the stochastic strategy by changing the preferred action selection method. The proposed method improves the efficiency of multiple agents in the search for targets without collision using a cooperative mechanism and reward rules based on the odor effect. Simulation experiments are conducted in three aspects to verify the effectiveness and robustness of the improved algorithm and compare it with other reinforcement learning algorithms in the field of multi-agent learning. The results demonstrate that the improved algorithm has obvious advantages in terms of mission success rate, target search rate and average search time, and the movement trajectory of multiple agents is more concise.

1. Introduction

With the rapid development of artificial intelligence, robotic systems are widely applied in many fields, such as warehousing, transportation, search and rescue (Fiorini & Botturi, 2008; Din et al., 2018; Ebel & Eberhard, 2019). In particular, the application of autonomous robots for cooperative target search in different environments has gradually become a popular research problem in the field of intelligence (Robin & Lacroix, 2016; Yan et al., 2019), including target detection and region exploration in some environments that are difficult for humans to approach or where unknown hazards exist, such as marine mineral resource exploration, mine detection, wreckage search after air crashes or fires, and missing person rescue in earthquakes or maritime environments (Acar et al., 2003; Luo et al., 2018; Mou et al., 2021). In complex and dangerous working environments and missions, a single robot has many deficiencies and performs poorly in terms of information acquisition and data processing (Senanayake et al., 2016). Compared to a single robot, multi-robot systems have obvious advantages, such as high efficiency, strong reliability, and fast operation (Xue & Zeng, 2008). Based on this, an increasing number of scholars have focused on multi-agent systems that accomplish large-scale complex missions by cooperating with each other.

Cooperative target search mission planning for multiple agents can generally be regarded as an optimization problem. The traditional methods to solve this problem are mainly heuristic algorithms, such as the ant colony algorithm (Purbolingga et al., 2019; Morin et al., 2023), particle swarm optimization algorithm (Doctor et al., 2004; Prasetya et al., 2013; Paez et al., 2021) and bee colony algorithm (Ataei et al., 2013; Cui et al., 2022). Most of these algorithms need to have certain prior information about the environment, and there are some problems in the search process, such as poor cooperation, invalid searches and easily falling into local optimization, so the search efficiency is not high. Reinforcement learning seeks approximate optimal solutions through constant trial-and-error and learning in an unknown environment. It is an algorithm that learns from the continuous interaction between environments and agents, and has been applied to several fields in the past few years (Sutton & Barto, 1998; Daoun et al., 2022). In recent years, neural networks (Woźniak et al., 2021; Połap & Woźniak, 2022) have also gradually become a research hotspot, which can solve the problem of large amounts of continuous spatial data in reinforcement learning. Therefore, scholars have gradually combined reinforcement learning algorithms with neural networks to solve cooperative target search

* Corresponding author.
E-mail addresses: 319581@whut.edu.cn (X. Wang), fangxi@whut.edu.cn (X. Fang).

https://doi.org/10.1016/j.eswa.2023.120643
Received 5 December 2022; Received in revised form 27 April 2023; Accepted 28 May 2023
Available online 12 June 2023

problems.

Both heuristic algorithms and reinforcement learning algorithms have the problem of randomly selecting strategies in the process of model solving (Tan & Karakose, 2020). On the one hand, the existence of a random strategy in the algorithm prevents the algorithm from falling into a local optimum (Zhou et al., 2020); on the other hand, invalid random searches will reduce the execution efficiency of the algorithm. In actual search and rescue environments, it is often necessary to find all potential targets in the shortest time, so it is very important to make the optimal decision in a limited time.

Based on the above problems, this study proposes an improved reinforcement learning algorithm with the action preference selection strategy to improve efficiency. It can not only solve the problems of poor cooperation among agents and invalid searches, but also has a good effect on the number of targets being greater than the number of agents in the search environment. This is different from single target search and multi-target search, where the number of targets is less than the number of agents. In this algorithm, the agents can acquire environmental information through sensors and make new action choices according to the updated information, which enables multiple agents to obtain better search results in the same number of episodes and improves the execution efficiency of the search mission. The main contributions of this study are as follows. (1) A multi-agent massive target cooperative search mission planning model is constructed. To solve the cooperative search problem where the number of targets is larger than the number of agents, this study details the search environment model and the motion model of agents, and sets the collision avoidance mechanism and the reward rules based on the odor effect in the search process, so that the agents can search as many targets as possible without collision. Finally, the execution effect of the multi-agent cooperative search mission is evaluated by the target discovery benefit and the reward benefit. (2) An improved reinforcement learning algorithm with the action preference selection strategy is proposed. Due to the existence of invalid random searches and poor cooperation in the Reinforce algorithm, this study reduces invalid searches and obtains better search results by the action selection strategy based on action preferences. (3) To verify the effectiveness and robustness of the improved Reinforce algorithm, the effects of parameter ω_2, the number of agents, the number of targets and search modes on the mission execution are considered, and the movement trajectory of agents is analyzed. Finally, the effectiveness of the improved algorithm is verified by comparison with other multi-agent reinforcement learning algorithms.

The rest of this paper is organized as follows: Section 2 briefly reviews the literature related to the multi-agent cooperative target search mission planning. Section 3 constructs a massive target cooperative search mission planning model for multiple agents. Section 4 introduces the multi-agent reinforcement learning algorithm and proposes an improved reinforcement learning algorithm with the action preference selection strategy. Section 5 sets the simulation environments, conducts simulation experiments, and carries out the comparison and analysis of the experimental results. Section 6 concludes the paper and provides an outlook on future work.

2. Related work

In recent years, cooperative target search mission planning based on intelligent optimization algorithms has become a very important and challenging research topic. Many scholars have studied the cooperative target search problem for multiple agents.

2.1. Brief review of traditional heuristic algorithms

At present, the study of cooperative target search problems is still dominated by heuristic algorithms. Some scholars have improved conventional mathematical methods and heuristic algorithms based on biological behavior to make the search algorithms meet the specific criteria, which improve the efficiency of robotic systems to search targets. For example, Sun and Cai (2014) proposed a cooperative target search method based on the multi-ant colony algorithm in view of the search probability graph and unmanned aerial vehicle (UAV) environment model. This method allows for effective cooperation among multiple UAVs by modeling a single UAV executing a search mission as an ant colony system. For the underwater environment, Zhang et al (2022) proposed the global-best difference-mutation brain storm optimization algorithm (GDBSO) for cooperative area target search of multiple attacking underwater unmanned vehicles (AUUVs), which can significantly improve the search efficiency in the contained unknown target area. Because the traditional particle swarm optimization algorithm easily converges to a specific position in the search environment, the algorithm was improved in prior studies (Dadgar et al., 2016; Tang et al., 2019; Garg et al., 2022). Compared with other methods, the improved algorithms not only shorten the search time, but also efficiently avoid falling into local optima. In addition, Cai and Yang (2013, 2016) took the cooperation rules of potential field functions as fitness functions to improve the particle swarm optimization algorithm, and applied the improved algorithms to a multi-robot cooperative target search mission in a complex and unknown environment.

Some scholars have applied certain heuristic algorithms based on biological behavior for the first time to the cooperative target search problem of multi-robot systems. For example, Tang et al. (2021) first used the cooperative multi-robot method based on the gray wolf optimization algorithm for static and dynamic target search in unknown environments. It is significantly better in terms of the number of iterations, success rate and efficiency compared to other methods. Additionally, Tang et al. (2020) used a multi-robot cooperative method based on the bat algorithm for the first time to address the target search problem. They improved the algorithm using an adaptive inertial weight strategy, the Doppler effect and a multi-swarm strategy, and they finally proposed the adaptive robotic bat algorithm (ARBA). The advantages of this algorithm are fewer iterations, smooth search trajectories and high success rates.

2.2. Brief review of reinforcement learning algorithms

Reinforcement learning is a kind of policy learning in artificial intelligence, and a unique branch of machine learning. The characteristics of self-learning and online learning make it adapt to complex and unknown environments, and it is an important direction for our research problems at present. Especially in recent years, many excellent achievements have been made in resolving problems such as chess game confrontation (Silver et al., 2016; Silver et al., 2018). Reinforcement learning algorithms have also gradually received increasing attention from scholars in the field of cooperative target search mission planning for multiple agents. In Cai et al. (2013), a cooperative strategy of a hierarchical reinforcement learning algorithm was proposed that automatically obtains the required parameters through learning. It can help robots accomplish cooperative target search missions in completely unknown environments. Zhou et al. (2022) proposed an experience-sharing reciprocal reward multi-agent actor-critic (MAAC-R) algorithm based on an experience-sharing training mechanism, which enhances the cooperation of large-scale UAV swarms by learning their cooperative sharing strategy. In Lai and Rui (2020), a curiosity-driven deep reinforcement learning algorithm with spatial location annotation was introduced. It uses a proximal policy optimization (PPO) algorithm to update neural network parameters during training, and achieves functions such as searching random targets in unknown regions, avoiding obstacles and adjusting flight altitude. The method enables UAVs to find the optimal search strategy faster, and improves search efficiency and accuracy. Further, Zhang et al. (2011) proposed a UAV target search algorithm based on Q-learning in unknown environments, and compared it with a search method based on D-S evidence theory. The results showed that the algorithm based on Q-learning has a certain

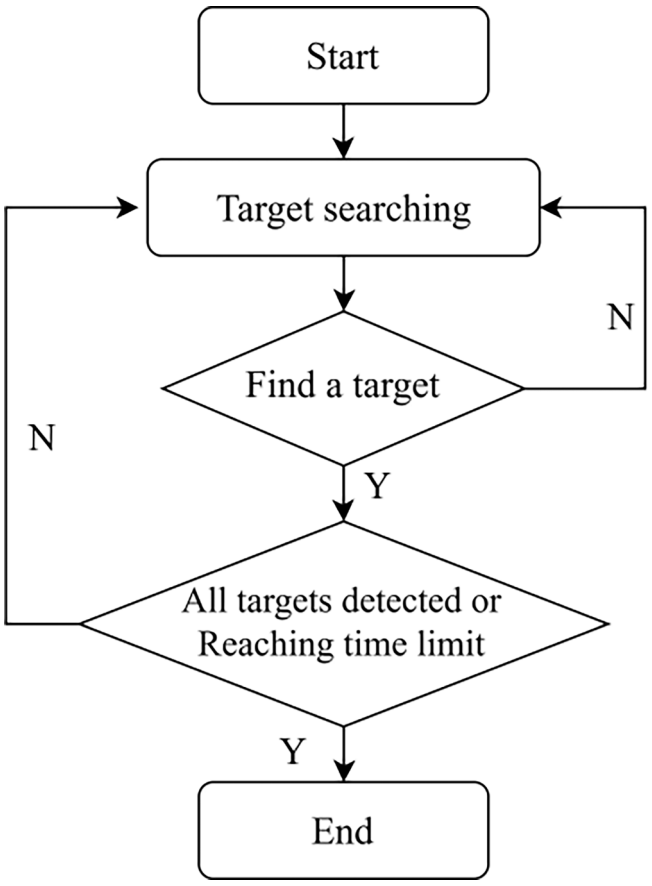


Fig. 1. Cooperative target search mission flow chart.

environmental adaptability, and can effectively carry out search missions in battlefield environments without any target information, which is more suitable for the target search problem in unknown environments.

2.3. Brief review of cooperative target search

Multi-agent cooperative target search is an important problem in both military and civilian fields. It involves technologies such as communication, trajectory optimization, obstacle avoidance, cooperative control, etc. It has recently attracted great attention with many research results. For example, Wang and Xiao (2019) used the improved pigeon swarm optimization algorithm and Markov chains to construct a cooperative multi-UAV search method. The method not only built a honeycomb environment model to reduce the repeat search rate, but also introduced the Cauchy disturbance into the basic pigeon swarm optimization algorithm, which effectively solved the problems of repetitive search and low search efficiency in the cooperative search process. Xing et al. (2019) introduced the artificial potential field (APF) into the ant colony algorithm and proposed a distributed ant colony optimization algorithm based on the artificial potential field (ACO-APF), which improved the state transfer rules of the algorithm and significantly increased the search coverage. In addition, Luo et al. (2023) proposed the concept of uncertainty to evaluate the search process, which reflects the reliability of the target search results. Then, they proposed a deep Q-network (DQN) based deep reinforcement learning architecture with a separate Q-network to jointly make optimal computation offloading decisions and flying orientation choices for multi-UAV cooperative target search. Aiming at the problem of a short time cycle and high real-time requirement of target search in the naval battle field, Yang et al. (2022) propose a path planning method based on deep reinforcement learning. Based on the Rainbow deep reinforcement learning algorithm, the state vector, neural network structure and algorithm framework of target search planning in naval battle fields are designed. Finally, a case is used to verify the feasibility and effectiveness of the proposed method.

Due to the high efficiency and strong flexibility, biological-inspired neural networks have also been widely used in multi-agent cooperative target search problems in recent years. For example, Cao et al. (2018) combined the Glasius bio-inspired neural network (GBNN) with bio-inspired cascade tracking control method, and proposed an integrated algorithm for a cooperative team of multiple autonomous underwater vehicles (multi-AUVs) to improve search efficiency. Yao and Zhao (2021) added a Gaussian mixture model (GMM) to a Glasius bio-inspired neural network, and verified the high efficiency and strong robustness of the GBNN-GMM algorithm through simulation experiments.

As seen from the brief review of traditional heuristic algorithms and reinforcement learning algorithms, heuristic algorithms are still dominant in solving the multi-agent cooperative target search problem, reinforcement learning algorithms are few, and most of them suffer from long cooperative search times and poor algorithm stability. In some cases, the robots fail to find all targets, and the search process is prone to local optima and repeated searches. A brief review of cooperative target search shows that most of the target search problems have more agents than the number of targets at present. Few studies have been conducted for massive cooperative target search missions. Therefore, this study chooses the improved reinforcement learning algorithm to solve the massive target cooperative search problem.

3. Massive target cooperative search model construction

Multi-agent cooperative target search is the premise of resource exploitation, personnel rescue, target attack, etc. This section introduces the environment model, motion model, collision avoidance mechanism and objective function model in the multi-agent massive target cooperative search mission planning. This model is based on the premise that the search area is known and the number of targets is known. The purpose is to maximize the search efficiency in the unknown area, i.e., to obtain the target information in the entire search area and reduce the search time, so that the targets can be searched as much as possible within a limited timeframe.

3.1. Problem description

The cooperative target search refers to the N_R homogeneous agents and N_T targets to be searched that are randomly distributed in a search area of unknown size, and the agents search the targets through a certain cooperative mechanism without collision. At the beginning of the mission, the agents start from the designated positions to search for targets in the mission area. When a target is searched by one of the agents, it means that the mission of this target has been completed, and all agents will judge whether all targets have been searched. If so, the agents will stop moving; otherwise, they will continue searching until all targets have been searched or the time limit has been reached.

In this study, it is assumed that the agents only know the boundary information of the search area and the number of targets before executing the search mission, and know nothing about the distribution of targets, i.e., the agents do not know the specific location of the targets during the search process, and the detection information of sensors is the only information they obtain. At the same time, there is a certain randomness in the agents' search directions, leading to uncertainty in the search results, i.e., the search area is full of uncertainties for the agents. Therefore, in this study, a cooperative search mechanism is designed to study the massive target cooperative search problem, which enables multiple agents to effectively search all targets under specific parameters. The search process consists of three stages, and the flow

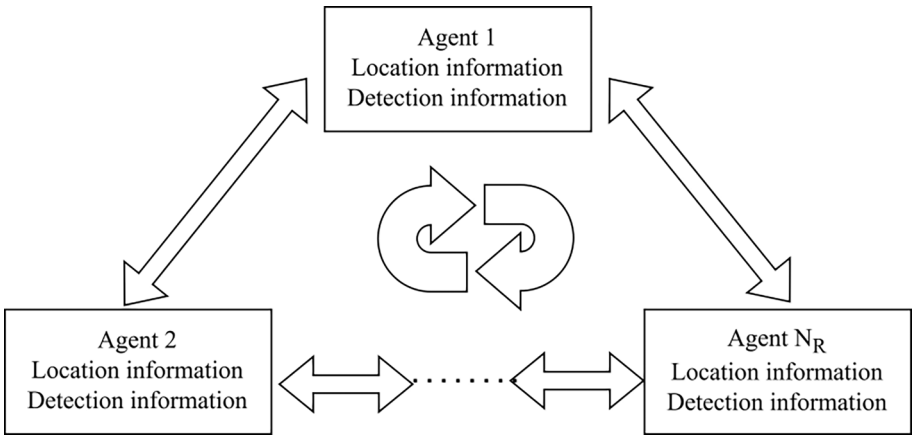


Fig. 2. Information sharing mechanism between agents.

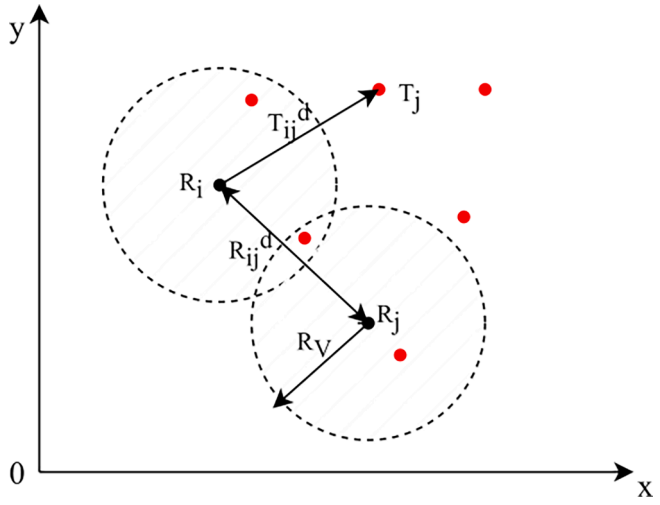


Fig. 3. Schematic diagram of cooperative target search.

chart is shown in Fig. 1.

3.2. Environment model

In massive target cooperative search missions, to optimize the model and solution space (Huang et al., 2020), the size of the search area in the unknown environment is scaled down to a rectangular region of $m \times n$. In this rectangular region, N_R agents carrying detection sensors are used to search N_T stationary targets, in which the agents move at a constant velocity over a two-dimensional region and their motion is the heading angular velocity. When the agents move near a target, they will judge whether they can search for the target according to the detection information of sensors, and their trajectories are determined by the heading angle. The heading angle is the angle between the direction of movement at the next moment and the direction of movement at the current moment. The position of agents at the next moment can be calculated by the heading angle and the movement velocity, and the movement trajectory of agents can be finally obtained.

In unknown environments, cooperative target search missions require collaborative sharing of location information and detection information among the agents to avoid collisions and repeated target searches. The information sharing mechanism between agents is shown in Fig. 2. Constrained by the communication distance of sensors, the agents can only obtain the state of neighboring agents and the location information of the targets that have been searched.

3.3. Motion model

Suppose that the detection radius of sensors carried by the agents is known and is R_V. In the mission area, the agents and targets are represented by particles; specifically, the agents are denoted by $R = \{R_i | i = 1, 2, \cdots, N_R\}$, whose position can be expressed as $R_i(x_i, y_i, \varphi_i)$, and the targets are $T = \{T_j | j = 1, 2, \cdots, N_T\}$, whose position can be expressed as $T_j(x_j, y_j)$. To simplify the motion model of agents, the following assumptions are made in this section.

(1) The agents only move uniformly in the two-dimensional search area, and both the agents and targets are considered particles.

(2) The agents communicate well with each other and share a common information network, which can ensure that multiple agents can search targets more effectively than a single agent, since the detection range of sensors seems to become wider with the synergistic effect.

(3) The agents are able to communicate freely within the mission area, i.e., the distance between any two positions in the mission area is less than the communication distance.

(4) There are no threats or obstacles in the mission area, and only collisions between agents should be considered during the search process.

(5) The targets are stationary and not capable of attack.

Let the state of agent R_i be $s_i = [x_i, y_i, \varphi_i]^T$, where x_i and y_i represent the x-axis coordinates and the y-axis coordinates respectively, and φ_i represents the heading angle. In this case, the second-order kinematic equation can be expressed as follows:

$$\begin{cases} x_i = v_i cos\varphi_i \\ y_i = v_i sin\varphi_i \end{cases} \tag{1}$$

where v_i is the velocity of agent R_i. From Eq. (1), we can obtain:

$$\begin{cases} x_i^{t+1} = x_i^t + v_i^t cos(\varphi_i^t + \Delta\varphi_i^t) \\ y_i^{t+1} = y_i^t + v_i^t sin(\varphi_i^t + \Delta\varphi_i^t) \\ \varphi_i^{t+1} = \varphi_i^t + \Delta\varphi_i^t \end{cases} \tag{2}$$

where $\Delta\varphi_i^t$ denotes the yaw angle of agent R_i at time t, where $\Delta\varphi_i \in \{0^\circ, 20^\circ, -20^\circ\}$.

During the execution of search missions, the agents will constantly update the mission execution progress based on the information detected by the sensors. Due to sensor measurement error, target misjudgment or target revisit may occur during the search process. Therefore, this study assumes that the detection probability p_d of sensors carried by the agents is 0.9, which means that the probability of sensors detecting the target in the presence of the target is 0.9, and the false alarm probability p_f is 0.1, which means that the probability of sensors

detecting the target in the absence of the target is 0.1.

In addition, to consider the spatial constraints between multiple agents, let the distance between agent R_i and agent R_j be R_{ij}^d, the distance between agent R_i and target T_j be T_{ij}^d, and the safe distance be R_s^d. According to assumption (3), the agents maintain good communication with each other during the mission execution process. Let the distance R_{ij}^d be greater than the safe distance to avoid collisions between agents. The schematic diagram of cooperative target search is shown in Fig. 3.

3.4. Collision avoidance mechanism

Considering the detection radius R_v and safety distance R_s^d, when $R_s^d \le R_{ij}^d \le R_v$, agent R_i and agent R_j can achieve collision avoidance under the condition of mutual communication. When $0 \le T_{ij}^d \le R_v$, agent R_i can detect target T_j and search for it. In addition, it is assumed that the movement velocity V_R of agents is constant at 1. There is a turning limit during movement, and the maximum yaw angle θ_{max} is $\pi/4$, i.e., the agents can only change their forward direction within $[-\pi/4, \pi/4]$ of the current movement direction.

To avoid collisions between agents during the cooperative search process, it is far from sufficient to consider only the safe distance. This paper introduces a collision avoidance mechanism in the model. The collision avoidance factor f_a and collision avoidance distance D_a are set in this mechanism. When the distance between agent R_i and another agent R_j is smaller than the collision avoidance distance, that is, $R_s^d < R_{ij}^d < D_a$, there is an incremental function:

$$\begin{cases} f_x = R_s^d \times f_a \times V_R \times \dfrac{x_i - x_j}{\left(R_{ij}^d\right)^2} \\ f_y = R_s^d \times f_a \times V_R \times \dfrac{y_i - y_j}{\left(R_{ij}^d\right)^2} \end{cases} \tag{3}$$

where (x_i, y_i) and $\left(x_j, y_j\right)$ are the coordinates of agent R_i and agent R_j at the current moment, respectively. This indicates that when there is a collision risk between agents, they will move in the opposite direction to move away from each other. The formula is as follows:

$$\begin{cases} x_i = f_x + x_i \\ y_i = f_y + y_i \end{cases} \tag{4}$$

where f_x and f_y are the values of the incremental function at the current moment.

3.5. Objective function model

The task of multiple agents is to search all targets in the area in a short time, while avoiding collisions with each other, i.e., the distance between agents during the search process should always be greater than the safe distance. When the distance between an agent and a target is less than or equal to the detection radius of the sensor, the target is regarded as having been searched. In the following, this section describes in detail the setting of the reward function and the evaluation of the search effect in the target search process.

3.5.1. Reward function

The reward function in the search process consists of two parts: one is the presence penalty $R_p = -t/T_{max}$, where t is the time required by the agents to search all targets and T_{max} is the maximum search time allowed for the agents in an episode, and the other is the search reward R_s. The presence penalty is used to speed up the detection process, and the search reward is used to guide the agents to detect targets in the right forward direction while avoiding collisions with other agents.

To further accelerate the search, this study sets different reward values around targets based on the odor effect. The odor effect can speed up the training process, but setting the odor range too low is not conducive to the acceleration process, and setting it too high is not suitable for multi-agent or multi-target training due to the mutual interference between odors. Therefore, it is important to choose the appropriate odor range and reward setting in the experiments. This study finally sets the odor range and reward values through many experiments as follows:

$$r = \begin{cases} 0, T_{ij}^d > 20 \\ 1, 18 < T_{ij}^d \le 20 \\ 2, 16 < T_{ij}^d \le 18 \\ 3, 14 < T_{ij}^d \le 16 \end{cases} \tag{5}$$

Combined with the reward rules based on the odor effect, the reward for finding a target is set as r_1 in the search reward, whose value is 5; the additional reward for searching all targets is r_2, whose value is 100. When the agent is at the boundary of the mission area and tries to leave this area in the next step, there is a boundary penalty r_p with a value of -3; the movement cost of agents is r_m, whose value is -1. When a collision occurs between agents, there is a collision penalty r_c. Finally, the search reward can be expressed as follows:

$$R_s = \begin{cases} r, odoreffect \\ r_1 = 5, findatarget \\ r_2 = 100, findalltargets \\ r_p = -3, boundarypenalty \\ r_m = -1, movementcost \\ r_c = -\alpha \times \left\| R_{ij}^d \right\| - \dfrac{\beta}{\pi} \times arccos \dfrac{\langle R_i, \overline{R} \rangle}{\|R_i\| \times \|\overline{R}\|}, collisionpenalty \end{cases} \tag{6}$$

where $\overline{R}$ is the standard direction vector, which is generally set to $(0, 1)$, and α and β are the weight coefficients of the distance adjustment penalty and direction adjustment penalty, whose values are 1 and 0.1 respectively. It should be noted that the search reward is for each agent, i.e., if a target is searched at the current moment, each agent will get the reward r_1. At the same time, the movement cost of agents is also set for an agent.

Eventually, the reward function can be set as follows:

$$r_{t+1} = \begin{cases} \gamma r_t + r_m \times N_R + r, targetnotfound \\ \gamma r_t + r_1 \times N_R + r_m \times N_R + r, foundatarget \\ \gamma r_t + r_2 + r_m \times N_R + r, foundalltargets \\ \gamma r_t + r_m \times N_R + r_c + r, agentscrashed \\ \gamma r_t + r_m \times N_R + r_p + r, agentsleftthearea \end{cases} \tag{7}$$

where γ is the discount factor with a value of 0.95. Of course, the reward at the last timestep is minus the presence penalty generated by the search process, which is $R_t^* = R_t + \rho R_p$, where ρ is the presence penalty factor with a value of 5.

3.5.2. Mission optimization model

In this paper, the purpose of constructing the massive target cooperative search model is to enable agents to search a larger area and find more targets. Therefore, this subsection defines the target discovery benefit expressed by the average number of targets and the reward benefit calculated by the total reward, and finally establishes the mission optimization model.

The target discovery benefit is denoted by J_t, where the average number of targets n_T refers to the average number of targets searched by agents in the given number of episodes, i.e.,

$$J_t = n_T = \frac{\sum_{k=1}^{N} T_k}{N} \tag{8}$$

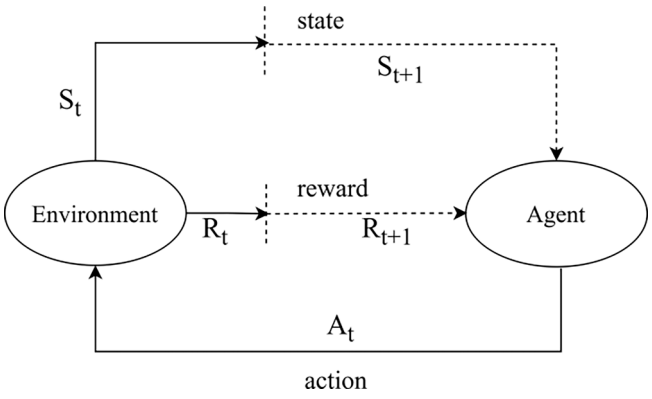


Fig. 4. The agent and environment interaction process.

where N is the total number of episodes, and T_k is the number of targets searched in the k episode.

The reward benefit is denoted by J_r, where the total reward r_T is the average reward for the given number of episodes, i.e.,

$$J_r = r_T = \frac{\sum_{k=1}^{N} r_k}{100 \times N} \tag{9}$$

where r_k is the reward obtained by agents in the k episode.

Finally, the mission optimization model established with the goal of maximizing the target discovery benefit and the reward benefit can be expressed as follows:

$$J^* = argmax_J(\omega_1 \times J_t + (1-\omega_1) \times J_r) \tag{10}$$

where J^* is the total benefit, and ω_1 is the weight coefficient, which is 0.75.

4. Cooperative target search algorithm

Reinforcement learning is a common tool for addressing a wide range of complex problems, and its model consists of two parts: the environment and the agent. The agent continuously interacts with the environment through trial-and-error and learning, and the interaction process is shown in Fig. 4. The agent takes an action A_t in the current state S_t and then transfers to the next state S_{t+1}; the environment receives the action A_t and gives the agent the corresponding reward R_t. Finally, the agent changes its actions to adapt to the environment based on feedback from the environment. When the environment provides a positive reward, the agent will strengthen the behavior strategy by increasing the probability of selecting the current action. Otherwise, the agent will decrease the probability of executing the behavior strategy. According to the interaction characteristics, the above learning process can be described as the Markov decision process, which is an unsupervised learning method.

Reinforcement learning consists of five basic elements: state S_t, action A_t, state transfer probability $P(s_{t+1}|s_t, a_t)$, reward R_t, and policy π. These five elements are described as follows:

State S_t represents the state of the agent at time t. It is a constantly changing quantity that can be either discrete or continuous. The set of states $\{S_t\}$ represents all possible states in the environment.

Action A_t denotes the actions that the agent can take at the current moment. It is a description of the behavior of the agent, which can be discrete or continuous. The action set $\{A_t\}$ represents all actions that can be taken by the agent.

State transfer probability $P(s_{t+1}|s_t, a_t)$ represents the probability that the agent takes action a_t in state s_t, and the state changes to s_{t+1} at the next moment.

Reward R_t denotes the positive or negative feedback signals obtained by the agent after taking action a_t in state s_t, and its value r_t is related to state s_{t+1} at the next moment. The reward set $\{R_t\}$ is all the feedback information that the agent can obtain.

Policy π refers to how the agent selects action a_t according to the current state s_t, and the expression is $a_t = \pi(s_t)$. Generally, there are two kinds of policies, namely, the stochastic policy and the deterministic policy. The stochastic policy means that the output is a distribution of actions for a given state s_t, where each action is selected based on probability, and the expression is $a_t \sim \pi(\bullet|s_t)$. The deterministic policy means that the policy gives a deterministic action a_t for a given state s_t, which can be expressed as $a_t = \mu(s_t)$.

Generally, reinforcement learning algorithms can be divided into two categories: one is based on value function updates, such as Q-learning and DQN, and the other is based on policy updates, including Reinforce algorithm with Monte Carlo policy gradients and the actor-critic algorithm based on temporal differences (TDs). Because the reinforcement learning algorithm based on value function updates cannot handle the stochastic policy problem and continuous action space problem well, this paper chooses the reinforcement learning algorithm based on policy updates. The Reinforce algorithm has better convergence, fast and direct processing of large-scale state and action space, and it is easier to calculate rewards in complex environments. Therefore, the Reinforce algorithm is ultimately chosen in this paper.

4.1. Reinforce algorithm with Monte Carlo policy gradients

In general, reinforcement learning algorithms guide actions through feedback signals from environments during the learning process for the maximum reward. Therefore, we need to define an objective function to obtain the maximum expected cumulative reward by following the policy π. In a finite timestep, the agents try to maximize the expected cumulative reward for T timesteps, and the cumulative reward is as follows:

$$V_\pi(s_t) = E[r_{t+1} + r_{t+2} + \cdots + r_{t+T}] = E\left[\sum_{i=1}^{T} r_{t+i}\right] \tag{11}$$

The expected cumulative reward is affected by the discount factor in infinite timesteps, which is:

$$V_\pi(s_t) = E\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \cdots\right] = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right] \tag{12}$$

where γ is the discount factor, $\gamma \in [0, 1]$. Finally, to obtain the optimal policy π^*, the expected cumulative reward in Eq. (11) or (12) is maximized:

$$V^*(s_t) = max_\pi V_\pi(s_t), \forall s_t \tag{13}$$

In some cases, we cannot directly maximize the expected cumulative reward $V(s_t)$, in which case we can obtain the optimal policy π^* by processing the state-action value $Q(s_t, a_t)$, i.e.,

$$\begin{aligned} V^*(s_t) &= max_{a_t} Q^*(s_t, a_t) \\ &= max_{a_t} E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right] \\ &= max_{a_t} E\left[r_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i+1}\right] \\ &= max_{a_t} E[r_{t+1} + \gamma V^*(s_{t+1})] \end{aligned} \tag{14}$$

Furthermore, by introducing the state transfer probability $p(s_{t+1}|s_t, a_t)$ into Eq. (14), the following can be obtained:

$$V^*(s_t) = max_{a_t}\left(E(r_{t+1}) + \gamma\sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)V^*(s_{t+1})\right) \tag{15}$$

Similarly, through mathematical derivation, we can obtain the Bellman equation for $Q(s_t, a_t)$, i.e.,

$$Q^*(s_t, a_t) = E(r_{t+1}) + \gamma\sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)max_{a_{t+1}}Q^*(s_{t+1}, a_{t+1}) \tag{16}$$

The Reinforce algorithm maximizes the expected cumulative reward under the given policy π_θ by continuously optimizing the policy parameters θ to obtain the optimal policy. Among them, the update method of policy parameters is an episode-by-episode exploratory method, which specifically means that the update of policy parameters is performed only after each completed episode. In general, the algorithm uses the stochastic gradient ascent method to update parameters according to the predetermined evaluation indices, so that the evaluation indices are continuously optimized. The stochastic gradient ascent takes all actions into account in the gradient update, and its formula is as follows:

$$\theta_{t+1} = \theta_t + \alpha\sum_{a_t}\widehat{Q}(s_t, a_t)\nabla\pi(a_t|s_t, \theta) \tag{17}$$

where $\widehat{Q}(s_t, a_t)$ is the approximation of $Q_\pi(s_t, a_t)$, and α is the step size. Because the essence of the Monte Carlo process is to take the average after multiple samples as an approximation of the expected cumulative reward, only one actual taken action is considered for each update in the Reinforce algorithm, i.e., the sum of actions is converted to expectations, and the total future revenue $G_t = \sum_{i=t+1}^{T}\gamma^{i-t}r_i$ is used as $Q_\pi(s_t, a_t)$. The derivation is as follows:

$$\nabla J(\theta) = E_{s\sim\ \pi_\theta}\left[\sum_{a_t}\pi(a_t|s_t, \theta)Q_\pi(s_t, a_t)\frac{\nabla\pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta)}\right]$$

$$= E_{s,a\sim\ \pi_\theta}\left[Q_\pi\left(s_t, a_t^*\right)\frac{\nabla\pi\left(a_t^*|s_t, \theta\right)}{\pi\left(a_t^*|s_t, \theta\right)}\right]$$

$$= E_{s,a\sim\ \pi_\theta}\left[G_t\frac{\nabla\pi\left(a_t^*|s_t, \theta\right)}{\pi\left(a_t^*|s_t, \theta\right)}\right] \tag{18}$$

Finally, the expression for the parameter update is obtained as follows:

$$\theta_{t+1} = \theta_t + \alpha G_t\frac{\nabla\pi\left(a_t^*|s_t, \theta_t\right)}{\pi\left(a_t^*|s_t, \theta_t\right)}$$

$$= \theta_t + \alpha G_t\nabla ln\pi\left(a_t^*|s_t, \theta_t\right) \tag{19}$$

If the future total revenue $G_t > 0$, the update direction of parameter θ_t will increase the probability of agents being in the current state, i.e., if the revenue is favorable, the probability of the current action will be increased. The greater the revenue, the greater the gradient update magnitude, and the greater the probability increase. Algorithm 1 presents the pseudo code for this method.

Algorithm 1: Reinforce Algorithm

Input: A differentiable policy parameterization $\pi(a_t|s_t, \theta)$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\theta \in R^d$ (e.g., to 0)
Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \cdots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\bullet|\bullet, \theta)$
  Loop for each step of the episode $t = 0, 1, 2, \cdots, T-1$:
  $G \leftarrow \sum_{i=t+1}^{T}\gamma^{i-t-1}R_i$
  $\theta \leftarrow \theta + \alpha\gamma^t G\nabla ln\pi(A_t|S_t, \theta)$

4.2. Improved Reinforce algorithm

In general, reinforcement learning algorithms use the greedy strategy to select actions, i.e., the agents always select actions with the maximum action value function $Q^*(s_t, a_t)$ as the optimal action at the current moment. This strategy has the obvious disadvantage that the agent takes the same action each time it faces the same state, which cannot represent the effect of other actions and has no self-learning ability. Therefore, some scholars have applied the ε-greedy strategy to reinforcement learning, which selects actions by comparing ε and r on the basis of the greedy strategy. The formula is as follows:

$$a_t = \begin{cases} argmax_{a_t}Q(s_t, a_t), r \geq \varepsilon \\ randomfrom\{A_t\}, r < \varepsilon \end{cases} \tag{20}$$

where ε is the exploration rate, $\varepsilon \in [0, 1]$, and r is a random number in $[0, 1]$. When the agent adopts the ε-greedy strategy, the actions are randomly selected with a certain probability, which may lead to undesirable benefits in the current episode, but the agent can gain more benefits by taking nonoptimal actions in the long run.

In Section 4.1, we mentioned the Reinforce algorithm with Monte Carlo policy gradients, which can select actions without considering the action value function. The action value function can be used to learn the policy parameter, but it is not necessary for action selection. Thus, the algorithm is not suitable for selecting actions via the ε-greedy strategy.

To reduce invalid random searches, this paper proposes an improved action selection strategy based on action preference, and applies it to the multi-agent massive target cooperative search mission planning model in Section 3. This strategy forms the parameterized numerical preference $h(s_t, a_t, \theta)$ for each pair of state-action according to the output values of the policy network, and normalizes $h(s_t, a_t, \theta)$ by the softmax distribution. A random perturbation λ is added to the normalized numerical preference $\dot{h}(s_t, a_t, \theta)$, and different weights are assigned to $\dot{h}(s_t, a_t, \theta)$ and λ to balance exploration and exploitation. Finally, the action with the highest action preference value in each state is selected. The formula is as follows.

$$\pi(a_t|s_t, \theta, \lambda) = \omega_2\dot{h}(s_t, a_t, \theta) + (1 - \omega_2)\lambda_t \tag{21}$$

where $\dot{h}(s_t, a_t, \theta) = h(s_t, a_t, \theta)/\sum h(s_t, a_t, \theta)$, and the sum of the action preferences in each state is 1.

In the action preference function, λ_t follows a beta distribution with the parameters of α and β, and the initial values of α and β are set to 1 in each episode, i.e., $\lambda_0 \sim \beta(1, 1)$. In each subsequent timestep, the values of α and β are updated by comparing the random number η_t generated in the uniform distribution of $[0, 1]$ with λ_t. When the random number η_t is less than λ_t, α is added to 1 in the next timestep; otherwise, β is added to 1. By updating α and β, the action preference function $\pi(a_t|s_t, \theta, \lambda)$ reaches a stable state by iterating and updating λ_t constantly.

To solve the problem of the large amount of continuous spatial data in the algorithm, this study uses a recurrent neural network when building the policy network. The recurrent neural network can handle the information of any length input sequence, and it uses the history information of all previous timesteps when calculating the result of each timestep in each episode, which has the outstanding advantages of memorability and parameter sharing.

The time complexity of the improved Reinforce algorithm depends on the size of the training data and the complexity of the model. At each timestep, the algorithm needs to calculate the outputs of the model and the corresponding gradients, and update the model parameters. For recurrent neural networks, $h_t = f(Ux_t + Wh_{t-1})$, where x_t is the current input with dimension $m \times 1$, U is the weight matrix with dimension $d \times m$, h_{t-1} is the value of the last state hidden layer with dimension $d \times 1$, and W is the weight matrix with dimension $d \times d$. Therefore, the time complexity of one operation is $O\left(d^2\right)$, and the total time complexity after n sequential operations is $O\left(nd^2\right)$. Subsequently, through the analysis of the algorithm, we obtained $T(episode, t) = O(n_s n_e)$, where n_s is the timestep in each episode, and n_e is the number of episodes. In the

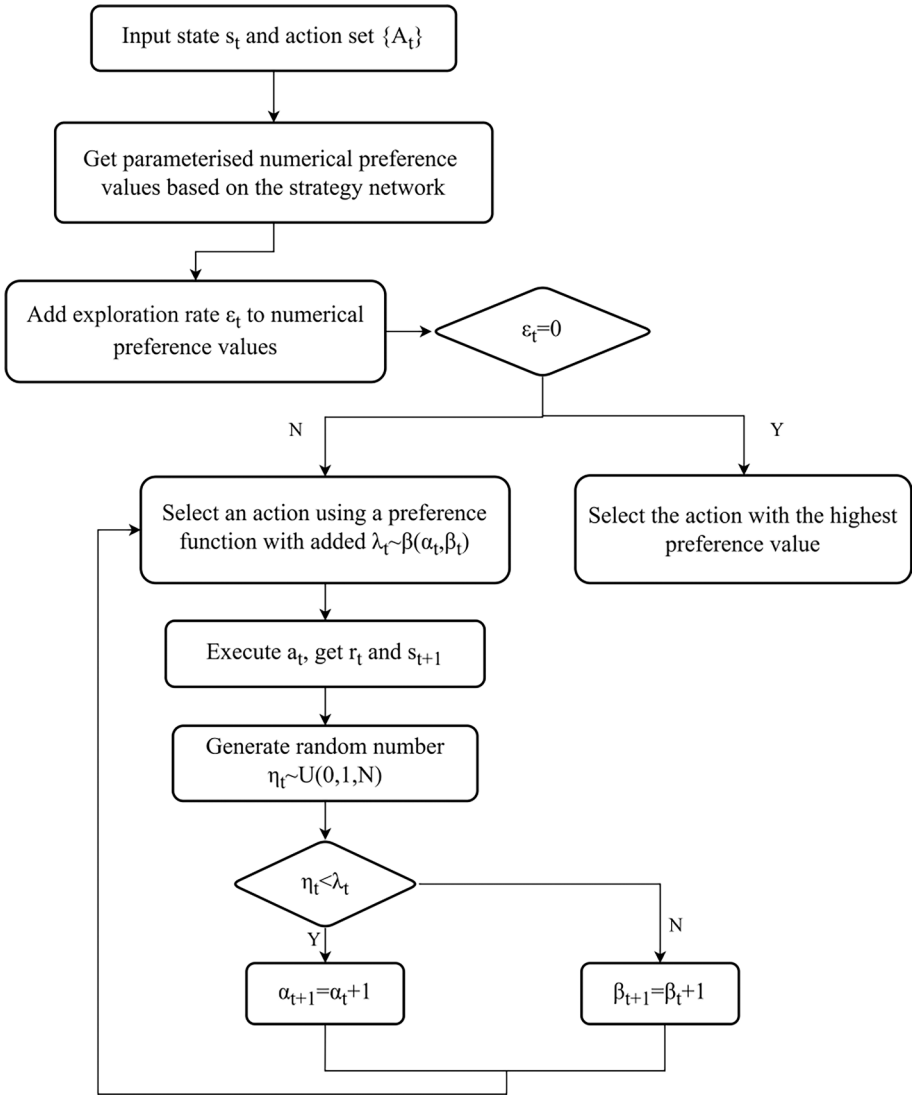


Fig. 5. Flow chart of the improved action selection strategy.

solution process, some variables have been eliminated by combining constant terms. Finally, the time complexity of the improved algorithm is $O\left(n_s n_e d^2\right)$.

The flowchart of the action preference selection strategy is given in Fig. 5. The pseudocode of the improved part of the algorithm is shown in Algorithm 2.

Algorithm 2: The action preference selection strategy

Input: Action set $\{A_t\}$, current state s_t, ε, λ_0, α_0, β_0
Output: Action a_t
①Get output values of all actions
②Convert output values to action preference values
③Add exploration rate ε:
$p = (1-\varepsilon)p + \frac{\varepsilon}{N}$, N is number of action
④$\eta_t \sim U(0,1,N), \lambda_t \sim \beta(\alpha,\beta,N)$
⑤If $\varepsilon = 0$,
Select the action with the highest preference values
else:
$\pi(a_t|s_t,\theta,\lambda) = \omega_2 \dot{h}(s_t,a_t,\theta) + (1-\omega_2)\lambda_t$
⑥Execute a_t, get r_t and s_{t+1}
⑦The timestep is increased by 1, so $t = t+1$
If $\eta_{t-1} < \lambda_{t-1}$,
$\alpha = \alpha + 1$
else:
$\beta = \beta + 1$
Determine if termination is reached, and if not, execute ①.

The parameterization of the policy network based on the action preference function can make the approximate policy close to the deterministic policy and solve the invalid search problem in stochastic policies. In addition, the policy network parameterization method usually learns faster and generates better asymptotic policies more easily. Furthermore, if the optimal policy is deterministic, the selection of the optimal action will be driven to be infinitely better than the suboptimal action as far as the parameters allow.

In the improved Reinforce algorithm, the agents select actions based on the action preference selection strategy, which increases the exploration degree of the algorithm and reduces invalid searches. In the multi-agent massive target cooperative search mission planning model, the invalid searches of multiple agents increase the search time of the current episode, so that the agents fail to search all targets, which reduces the probability of the algorithm obtaining a reasonable solution under the same number of episodes. The improved algorithm reduces the invalid random searches and increases the probability of obtaining a reasonable solution. The specific process of the above algorithm is shown in Fig. 6, where N is the total number of episodes.

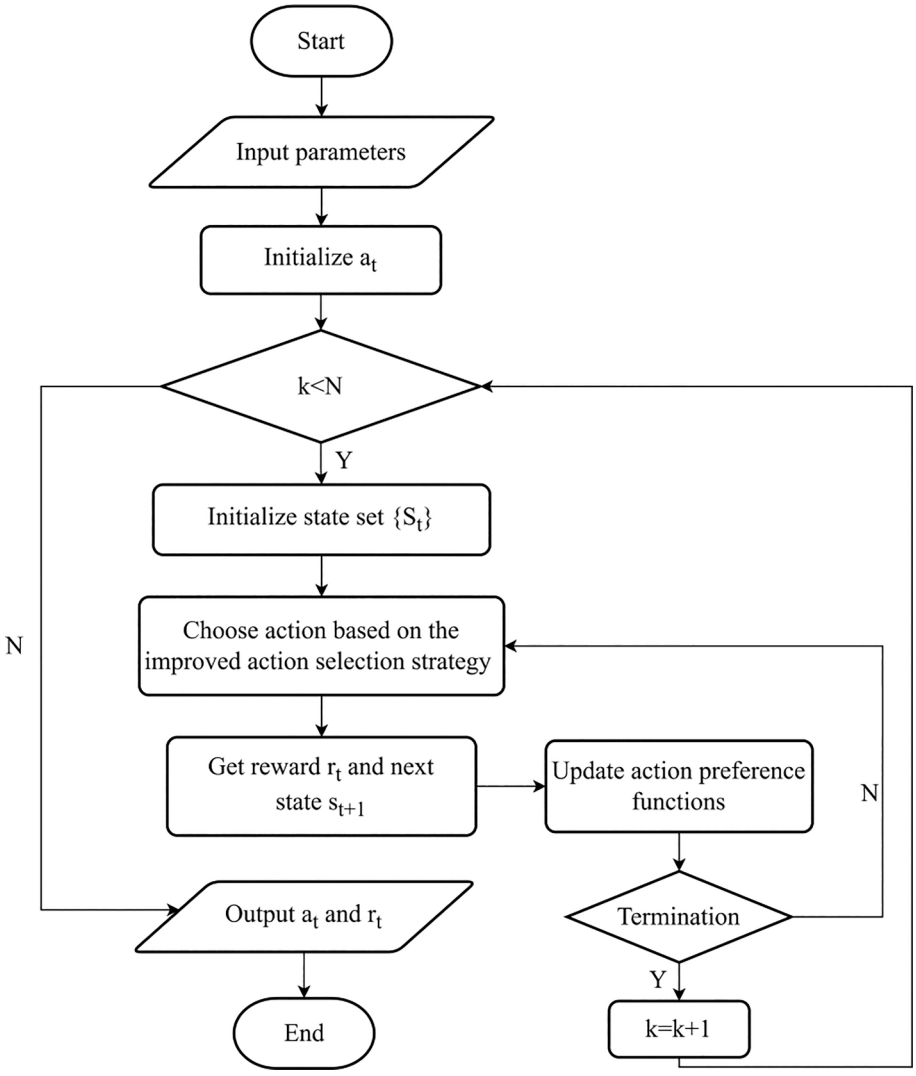
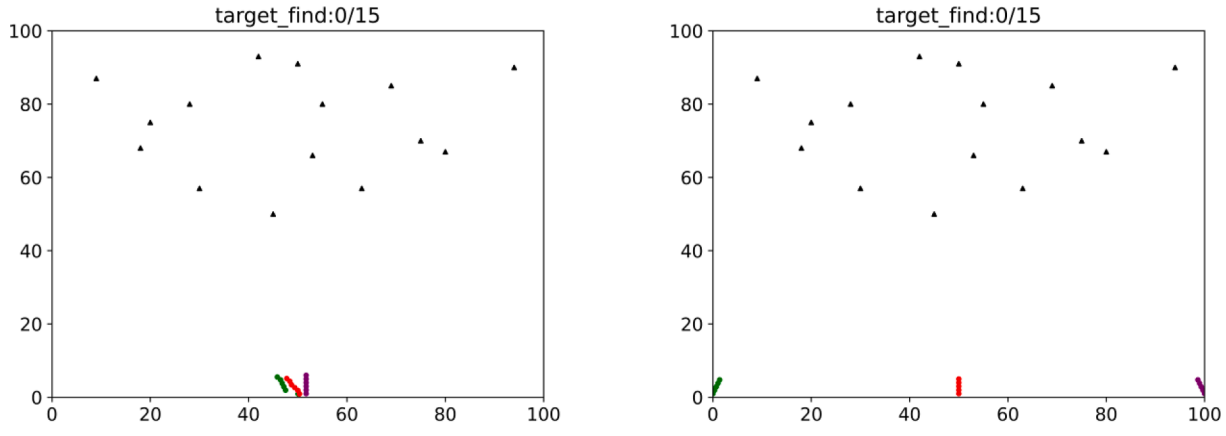


Fig. 6. Flow chart of the improved Reinforce algorithm.



(a) Centralized search mode (b) Distributed search mode

Fig. 7. Cooperative target search experimental environment.

5. Simulation and discussion

To test the effectiveness of the improved Reinforce algorithm, this section constructs a massive target cooperative search simulation environment and conducts many simulations for different targets by training multiple agents. These simulations help us not only validate the efficacy of the improved algorithm but also compare the performance with other multi-agent reinforcement learning algorithms. In the different simula-

Table 1
The model parameter settings.

Parameter	Mean	Value	Parameter	Mean	Value
R_v	Detection radius	14	α	Distance adjustment weight	1
R_s	Safe distance	1	β	Direction adjustment weight	0.10
D_a	Avoidance distance	3	ρ	Presence penalty factor	5
f_a	Collision avoidance factor	0.80	ε	Exploration rate	[0,0.5]
θ_{max}	Maximum yaw angle	$\pi/4$	*maxstep*	Maximum iteration step	400
p_d	Detection probability	0.90	M_e	Iterative episodes	10,000
p_f	False alarm probability	0.10	ω_1	Mission optimization model weight	[0,1]
V_R	Search velocity	1	ω_2	Action preference function weight	[0,1]
γ	Discount factor	0.95	φ	Heading angle	$[-\pi/4, \pi/4]$

tions, several sets of experiments are set up by changing the parameter ω_2, search modes, number of agents and number of targets, and the performance, scalability and robustness of the algorithm are compared and analyzed. The experimental system configuration is Intel (R) Core (TM) i5-7200U CPU @ 2.50 GHz, 2.71 GHz, 8.0 GB RAM, and the experimental environment is Python3.7.

5.1. Experimental environment and parameter setting

The massive target cooperative search environment model and the motion model of agents are detailed in Section 3, in which the agents perform a search mission from the specified starting positions and search the current coverage area at every moment. Once a target is searched, it is marked as found so that this target cannot be repeatedly searched by other agents. According to the detection results of sensors, the agents can determine the current state, the targets that have been searched and the targets to be searched. Fig. 7 depicts the basic environment for searching 15 targets using 3 agents in centralized search mode and distributed search mode, with the agents and targets located in a 100 × 100 two-dimensional mission area. The green, red, and purple dots represent three agents, the black triangles represent targets to be searched, and the black triangle turns into the orange triangle when the target is searched.

To ensure that the agents can search for targets in an unknown environment, the mode of the target position is defined in the simulation experiments. When the mode of the target position is 0, the parameter *deter* is introduced, $deter = t$ indicates that the position of the targets in each episode is deterministic and unchanged, and $deter = f$ indicates that the position of the targets in each episode would change with certain uncertainty. When the mode of the target position is 1, the position of the targets is fully random; when the mode of the target position is 2, the position of the targets is deterministic. In subsequent experiments, the mode for setting the target position is randomly selected between 0 and 1, and all experiments use the same environment configuration. The values of the parameters are taken in a floating up and down manner after several adjustment experiments, and the model parameters used for training are determined as shown in Table 1. Among them, a simulated annealing mechanism is used to balance exploration and exploitation. Specifically, setting $\varepsilon = 0.5$ at the beginning of training encourages the agents to randomly select different actions in the action space to collect various experiences. As the training progresses, ε gradually anneals to 0, and the agents try to use the learned strategy to select actions. In the experiments, the current search mission is terminated to enter the next episode if the agents search all targets or reach the maximum iteration step.

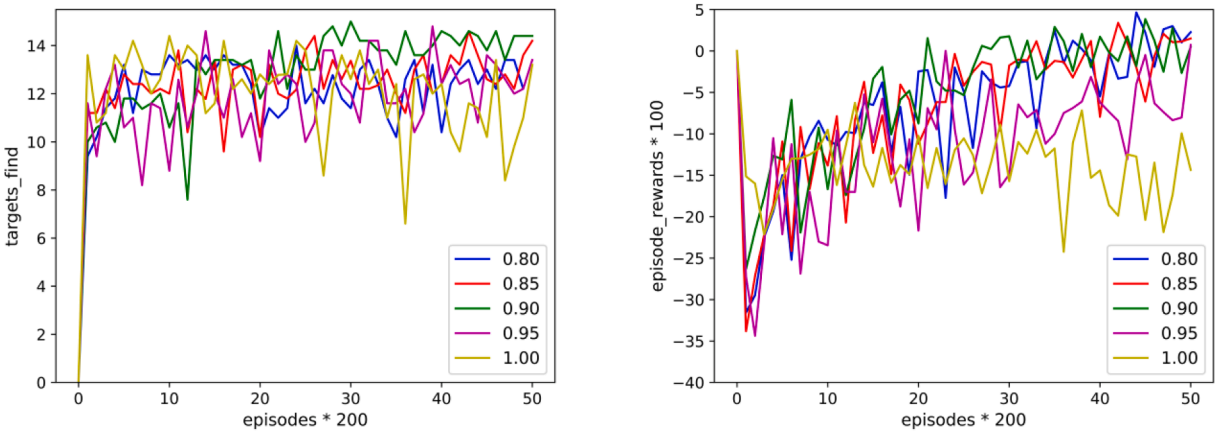
5.2. Effectiveness analysis of the improved Reinforce algorithm

5.2.1. The effectiveness analysis for parameter ω_2

This subsection discusses the influence of the weight coefficient ω_2 of action preferences in the action selection strategy on the effectiveness of target search, where the values of ω_2 are 0.80, 0.85, 0.90, 0.95 and 1.00 for the simulations. For each value, the simulation experiment is independently repeated five times, with 10,000 episodes each time. Meanwhile, the mission area for all experiments is set to 100 × 100, and 15 targets are searched using 3 agents in centralized search mode and distributed search mode. Finally, by analyzing the impact of various values of ω_2 on the search efficiency, the values of ω_2 for different search modes are determined.

(1) Effectiveness analysis in the centralized search mode.

To analyze the influence of parameter ω_2 on the search efficiency in the centralized search mode, Fig. 8 shows the search effect of the algorithm under five different parameter values, where Fig. 8(a) shows the average number of targets searched by the agents, and Fig. 8(b) shows the average reward obtained by the agents. From the left figure, we can see that the yellow curve fluctuates greatly in the late iterations, and the results of the other curves are obviously better than the yellow curve, so



(a) Average number of targets (b) Average reward

Fig. 8. Effectiveness analysis of ω_2 in the centralized search mode.

Table 2
Influence of ω_2 in the centralized search mode.

ω_2	$succ(\%)$	avg_{tgt}	avg_{rew}	avg_{time}	J^*
0.80	14.51	12.21	−745.50	65.85	7.30
0.85	15.29	12.28	−758.66	62.38	7.31
0.90	32.94	13.01	−598.65	60.22	8.26
0.95	25.49	11.74	−1244.45	58.52	5.69
1.00	25.49	12.14	−1445.54	56.60	5.49

we can judge that adding λ to the action selection strategy can improve the efficiency of the agents searching for targets, that is, the improved action selection strategy is better than the original strategy. In the process of increasing ω_2 from 0.80 to 0.90, the average number of targets found increases gradually, and the algorithm performance improves gradually. When $\omega_2 = 0.95$, the curve greatly fluctuates, and the search effectiveness is unstable. Therefore, we can roughly judge that the effect is better when $\omega_2 = 0.90$. As seen from the right figure, except that the curve fluctuates greatly when $\omega_2 = 1.00$, other values of ω_2 have no obvious influence on the average reward.

In the centralized search mode, the influence of ω_2 on the search efficiency of the algorithm is further quantitatively analyzed. Table 2 summarizes the effect of ω_2 on the target search effectiveness by various statistical indicators, where *succ* denotes the mission success rate, which is the average ratio between the number of episodes in which all targets are searched and the total number of episodes in the five experiments; avg_{tgt} represents the average number of targets searched by the agents in each episode, avg_{rew} denotes the average reward, avg_{time} represents the average search time, and J^* denotes the total benefit. As shown in Table 2, the action preference selection strategy enables the agents to search for more targets in the same number of episodes except for $\omega_2 = 0.95$, and the total benefit is higher than the original strategy regardless of the value of ω_2. As ω_2 increases, the performance of $\omega_2 = 0.90$ is the best when 3 agents are used to search 15 targets, with a mission success rate of 32.94% and a total benefit of 8.26. Therefore, this study chooses $\omega_2 = 0.90$ in the centralized search mode.

(2) Effectiveness analysis in the distributed search mode.

Similarly, to study the influence of parameter ω_2 on the search efficiency in the distributed search mode, Fig. 9 shows the search effect of the algorithm under five different values of ω_2, where Fig. 9(a) represents the average number of targets and Fig. 9(b) represents the average reward. From the left figure, we can see that all results except the blue curve are better than that of the yellow curve, so we can judge that the action preference selection strategy mostly outperforms the original strategy in the distributed search mode. In the process of increasing ω_2 from 0.80 to 0.95, the average number of targets searched by the agents increases gradually, and the performance of the algorithm improves gradually. The average number of targets of $\omega_2 = 0.90$ and $\omega_2 = 0.95$ in the late iterations are close to each other, so the search effect cannot be judged and needs further analysis. As seen from the right figure, the influence of different values of ω_2 in the action preference selection strategy on the average reward is not obvious.

To further analyze the effect of ω_2 on the search efficiency of the algorithm, we perform a quantitative analysis of the search results. Table 3 summarizes the statistical indicators, such as the average number of targets, the average reward and the average search time under the different parameter values. As shown in Table 3, in addition to $\omega_2 = 0.80$, the action preference selection strategy can make the agents search more targets with the same number of episodes and improve the search efficiency. When 15 targets are searched using 3 agents, the performance of the algorithm is also improved with increasing ω_2. Overall, the improved Reinforce algorithm shows the best performance when $\omega_2 = 0.95$. The mission success rate, the average number of targets, and the total benefit are the highest at 69.41%, 13.77 and 9.10, respectively. Therefore, in this study, $\omega_2 = 0.95$ is chosen for the distributed search mode.

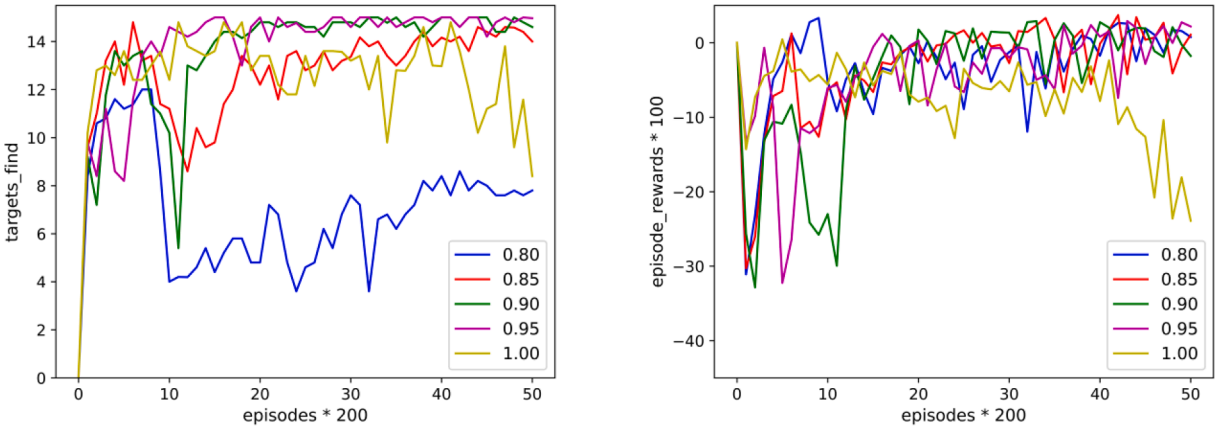
5.2.2. *The effectiveness analysis for different numbers of agents*

To evaluate the impact of the number of agents on the performance of the improved algorithm, this subsection sets the number of agents to 1, 3, and 5 (the number of targets is set to 1, 3, 5, 10, and 15) in the distributed search mode. In theory, the number of agents can be larger, but the maximum number of agents is set to 5 considering the limitations of the size of the area and the maximum iterative step. Meanwhile, to ensure the reliability of the experimental results, we use different random seeds to conduct three independent repeated experiments in each case. The number of training episodes in each experiment is 10000, and the maximum iteration step is 400. Finally, the average values of three experimental results are taken for analysis.

As shown in Fig. 10, the histograms of statistical indicators verify that the action preference selection strategy in the improved algorithm is applicable to different scenarios, i.e., the algorithm shows good performance for different numbers of agents. Meanwhile, we find that with

Table 3
Influence of ω_2 in the distributed search mode.

ω_2	$succ(\%)$	avg_{tgt}	avg_{rew}	avg_{time}	J^*
0.80	10.59	6.98	−410.53	73.09	4.21
0.85	35.29	12.81	−419.07	67.52	8.63
0.90	60.39	13.63	−557.13	62.05	8.83
0.95	69.41	13.77	−493.28	58.71	9.10
1.00	43.92	12.68	−801.433	56.63	7.51



(a) Average number of targets (b) Average reward

Fig. 9. Effectiveness analysis of ω_2 in the distributed search mode.

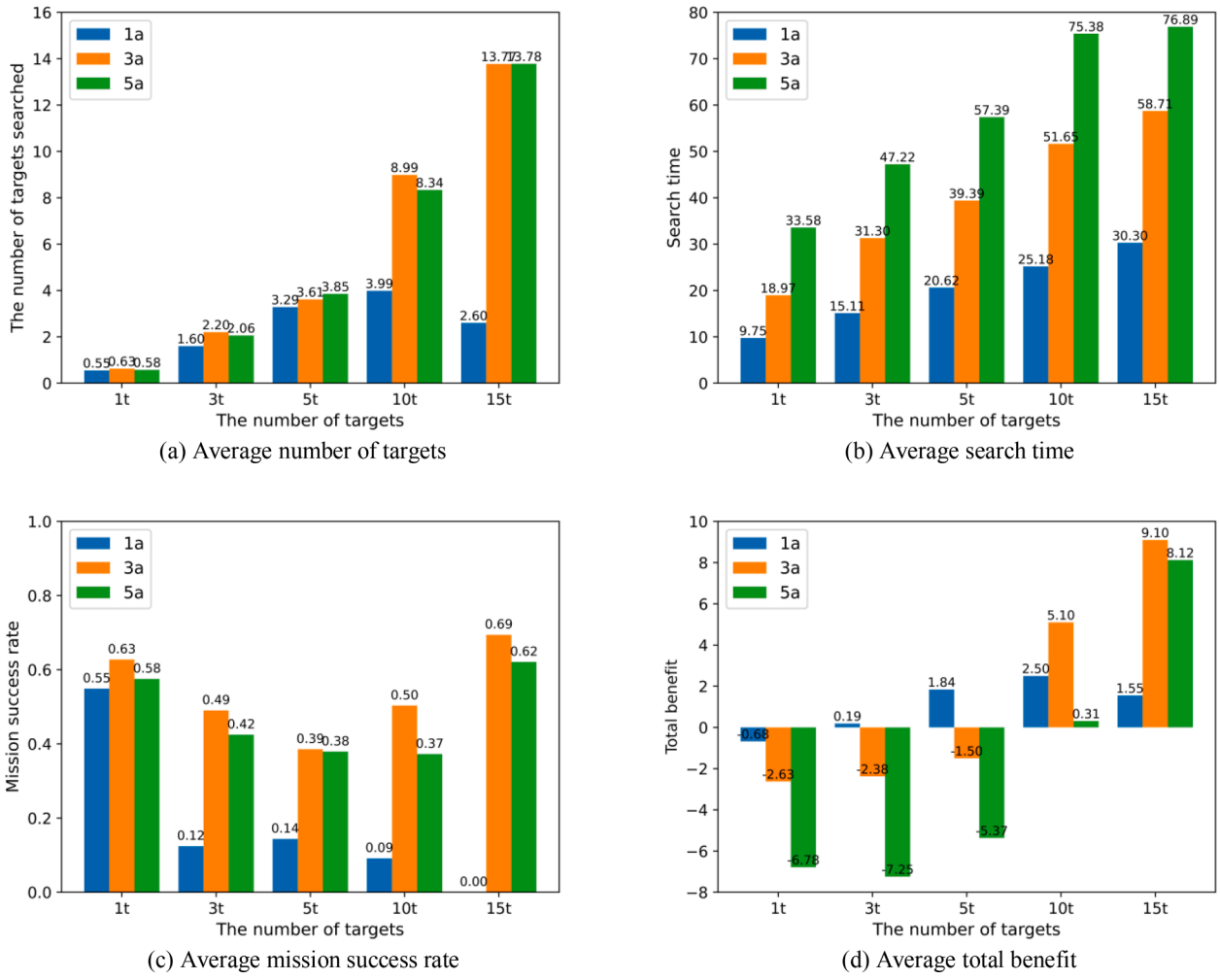


Fig. 10. Statistical indicators of the search results under different numbers of agents.

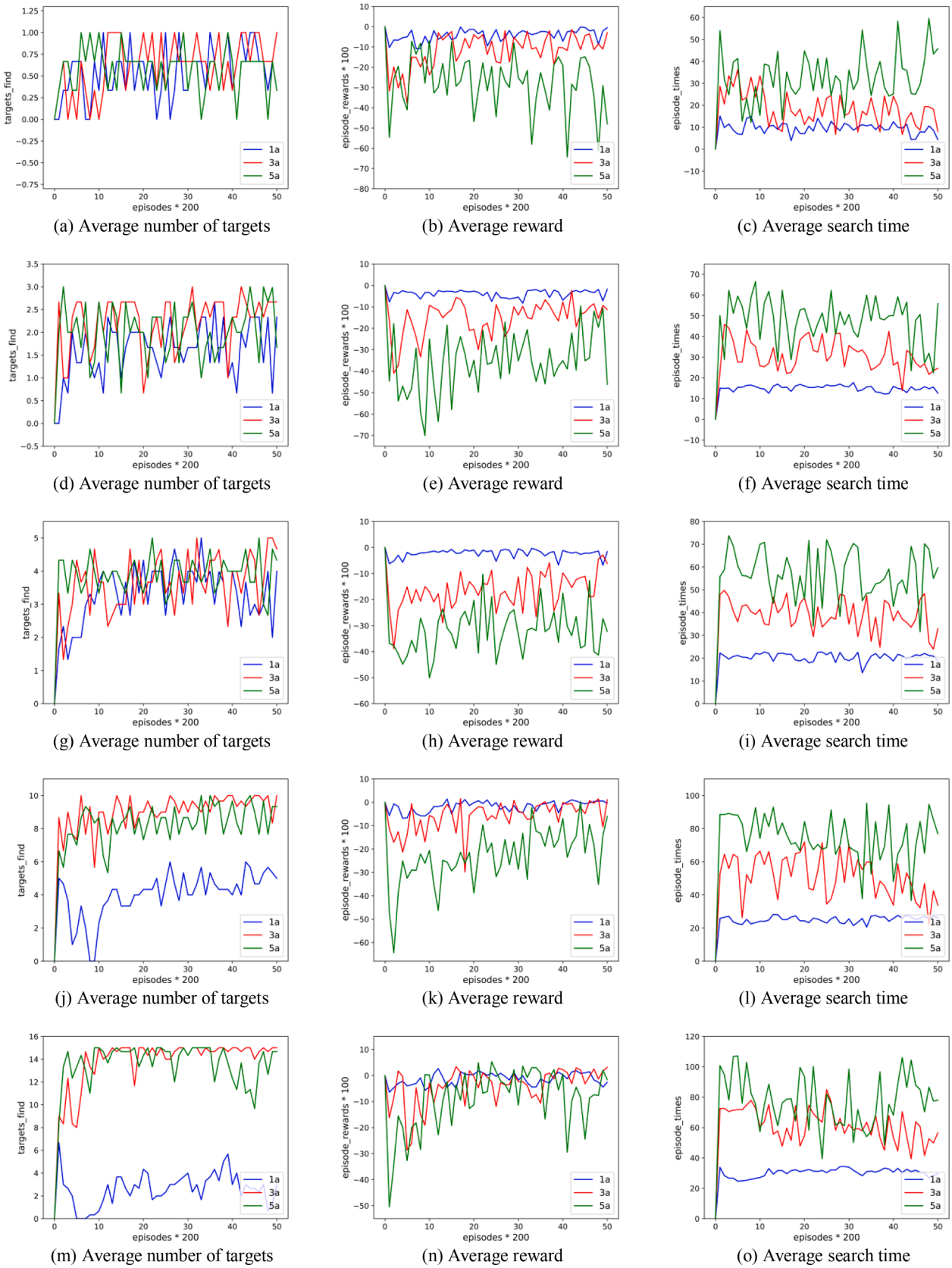
increasing numbers of targets in the mission area, the average number of targets and the mission success rate are both higher compared with a single agent, which indicates that the cooperative search among multiple agents can increase the search efficiency and improve the search results to a certain extent. Of course, a comprehensive analysis of the average number of targets, average search time and mission success rate shows that the efficiency of using 3 agents to search for targets cooperatively is the highest, regardless of the number of targets in the environments.

To intuitively analyze the influence of the number of agents on the search results, the results for the average number of targets, average reward and average search time under different scenarios are given in Fig. 11, where (a), (b), and (c) are the results of searching for 1 target, (d), (e), and (f) are the results of searching for 3 targets, (g), (h), and (i) are the results of searching for 5 targets, (j), (k), and (l) are the results of searching for 10 targets, and (m), (n), and (o) are the results of searching for 15 targets. As seen from the change in the average number of targets in Fig. 11, the gap between the average number of targets searched by multiple agents and that of a single agent widens gradually as the number of targets in the mission area increases. This shows that a single agent is suitable for search missions with a small number of targets, and the cooperative search of multi-agents is suitable for massive target search missions. From the average search time, the greater the number of agents, the longer the average search time, regardless of the number of targets in the mission area. Because the reward in each episode is related to the number of agents, the number of targets and the search time, the average reward shows irregular and large fluctuations. Combined with Fig. 10, the total benefit of a single agent is higher than that of multiple agents when the number of targets is 1, 3 and 5. In contrast, when the number of targets is large, the benefit of multi-agent cooperative search will be higher than that of a single agent, where the total benefit of three agents is the highest.

5.2.3. *The effectiveness analysis for different numbers of targets*

To evaluate and analyze the effect of the number of targets on the search efficiency, this subsection tests the effectiveness of the improved Reinforce algorithm by changing the number of targets in a mission area of 100×100. In the experiments of the distributed search model, the number of agents is set to 3 and 5, and the number of targets is selected between 1 and 20. Theoretically, the number of targets can be larger, but considering the constraints of the mission area and search time, the maximum number of targets is set to 20. Similarly, to ensure the reliability of the experimental results, three independent repetition experiments are conducted for each case by the different random seeds. The number of training episodes in each experiment is 10,000, the maximum iteration step is 400, and the average values of three experimental results are finally taken for analysis.

Fig. 12 shows the results of three and five agents performing search missions with different numbers of targets, where (a), (b), and (c) are the results for the average number of targets, average reward and average search time of three agents performing search missions, respectively, and (d), (e), and (f) are for five agents. As seen from Fig. 12, whether it is three agents or five agents, as the number of targets in the search mission increases, the average number of targets that the agents can search



(a) Average number of targets (b) Average reward (c) Average search time

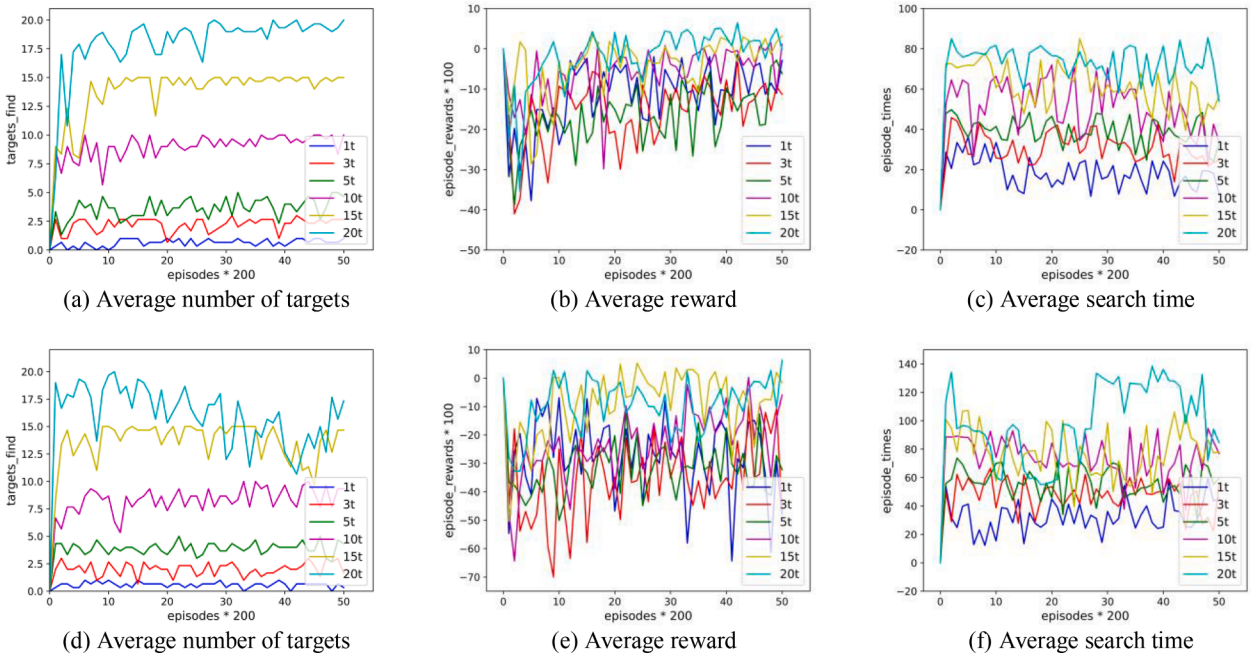
(d) Average number of targets (e) Average reward (f) Average search time

(g) Average number of targets (h) Average reward (i) Average search time

(j) Average number of targets (k) Average reward (l) Average search time

(m) Average number of targets (n) Average reward (o) Average search time

Fig. 11. Effectiveness analysis of the number of agents under different target numbers.



(a) Average number of targets (b) Average reward (c) Average search time

(d) Average number of targets (e) Average reward (f) Average search time

Fig. 12. Effectiveness analysis of the number of targets under different agent numbers.

Table 4
Statistical indicators of search results under different numbers of targets.

N_R	N_T	$succ(\%)$	avg_{tgt}	avg_{rew}	avg_{time}	$rate_{tgt}(\%)$	J^*
3	1	62.75	0.63	−1238.92	18.97	62.75	−2.63
	3	49.02	2.20	−1611.81	31.30	73.42	−2.38
	5	38.56	3.61	−1684.27	39.39	72.28	−1.50
	10	50.33	8.99	−656.68	51.65	89.85	5.10
	15	69.41	13.77	−493.28	58.71	91.81	9.10
	20	35.29	17.94	−260.34	73.74	89.68	12.80
5	1	57.52	0.58	−2886.23	33.58	57.52	−6.78
	3	42.48	2.06	−3516.27	47.22	68.62	−7.25
	5	37.91	3.85	−3301.68	57.39	76.99	−5.37
	10	37.25	8.34	−2377.68	75.38	83.37	0.31
	15	62.09	13.78	−885.32	76.89	91.85	8.12
	20	32.68	15.94	−1075.24	100.08	79.70	9.27

increases gradually, and the average time required to search for targets takes longer. When 20 targets are searched using 5 agents, the average number of targets decreases, and the search time becomes longer after 5000 episodes, which leads to poorer search results. In Fig. 12(b) and (e), the average reward shows irregular and large fluctuations. The reason is that on the one hand, increasing the number of targets increases the search reward during the mission execution, and on the other hand, the lengthening of the search time increases the presence penalty.

To further analyze the influence of the number of targets on the search efficiency in the distributed search mode. Table 4 summarizes the search results for three and five agents, $rate_{tgt}$ denotes the target search rate, which is the ratio of the average number of targets searched by agents to the number of targets contained in the mission area. The results of each statistical indicator in Table 4 verify the scalability of the improved Reinforce algorithm in different scenarios. In terms of the target search rate, the target search rate is above 50% regardless of the number of targets in the mission area, which indicates that the algorithm can perform well under different numbers of targets. With increasing numbers of targets in the search area, whether three or five agents, both the mission success rate and the target search rate tend to increase gradually, and the maximum in the number of targets is 15, where the mission success rate is above 60% and the target search rate reaches above 90%. This shows that the algorithm performs best when the number of targets is 15.

5.3. Trajectory analysis

In this section, the effectiveness of the improved Reinforce algorithm is further verified by comparing and analyzing the trajectories of agents in the two search modes (centralized search and distributed search) introduced in Section 5.2.1.

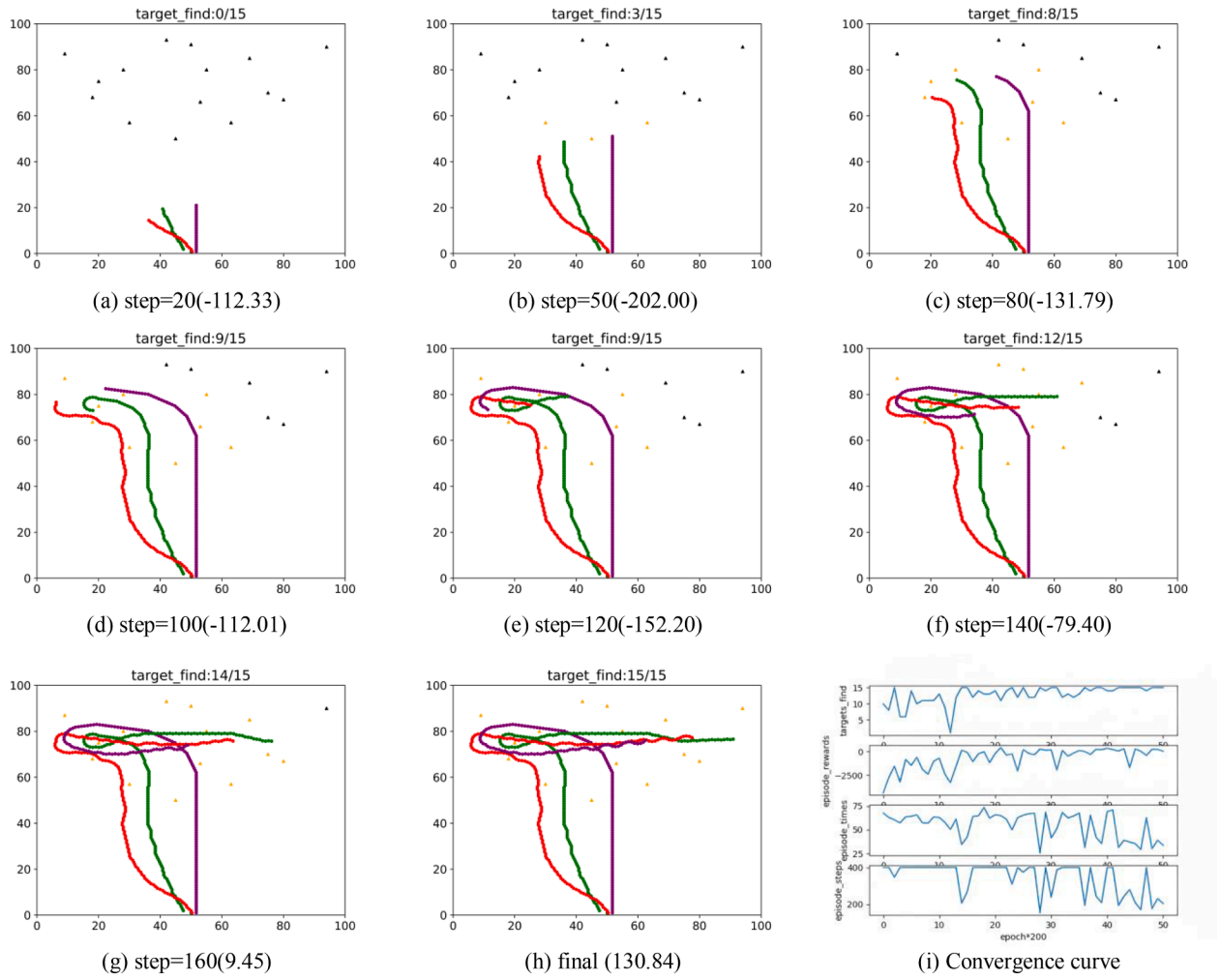
5.3.1. Centralized search trajectory

Fig. 13 shows the multi-agent cooperative target search trajectories in the centralized search mode, where the number of agents is 3, the number of targets is 15 and $\omega_2 = 0.90$. In this scenario, three agents start from the same location of the search environment using the proposed strategy. To analyze the trajectory of agents more effectively, we intercepted the trajectory plots of agents at the 20th, 50th, 80th, 100th, 120th, 140th and 160th iterations, and the values in brackets are the rewards at the current iteration. The experimental results show that the agents maintained good communication and fast search during iterations 0 to 100. During iterations 100 to 140, the three agents successfully achieved turns and continued to search for targets in the environment. After the 177th iteration, the agents searched all targets and the search mission ended.

In Fig. 13(h), we find that the movement trajectories of the three agents have overlapping parts. From the perspective of their complete movement process, three agents do not reach the same position at the same time in the overlapping parts of the trajectories, so there will be no collision between agents. Meanwhile, combined with the reward rules based on the odor effect, this is the inevitable result of the agents chasing the reward maximization.

5.3.2. Distributed search trajectory

Fig. 14 shows the movement trajectories of multi-target search with three agents in the distributed search mode, where the number of targets is 15 and $\omega_2 = 0.95$. In this scenario, the starting positions of the agents are in the lower-left, lower-middle and lower-right corners of the search area. Similarly, to analyze the trajectories of the agents more effectively, we intercepted the trajectory plots of agents at the 30th, 50th, 90th, 120th, 150th, 180th, and 200th iterations, and the values in brackets are



(a) step=20(-112.33) (b) step=50(-202.00) (c) step=80(-131.79)

(d) step=100(-112.01) (e) step=120(-152.20) (f) step=140(-79.40)

(g) step=160(9.45) (h) final (130.84) (i) Convergence curve

Fig. 13. Multi-target search movement trajectory in the centralized search mode (step = 177). Note: The green, red, and purple dots in the figure represent agents, the black triangles represent targets to be searched, and the black triangle becomes an orange triangle when the target is searched.

the rewards. The experimental results show that the algorithm progressed slowly during iterations 0 to 50. Before the 90th iteration, although agent 1 and agent 2 were close, the distance between them was greater than the safe distance, and the collision avoidance mechanism was not triggered. In the 120th iteration, agent 1 and agent 3 moved to the mission area boundary and tried to leave the search area. There was a boundary penalty at this moment, which prompted them to turn the movement directions. Finally, after the 214th iteration, the three agents searched all targets in the mission area, and the search mission ended.

In Fig. 13 and Fig. 14, the targets are randomly placed in the search area, and multiple agents search for targets in a cooperative manner. During the search process, when an agent detects a target, it sends the detection information to other agents to avoid the target being searched repeatedly and continues to detect the remaining targets. According to Eq. (21), combined with the search trajectory of agents, adding λ in the action preference selection strategy can not only balance exploration and exploitation and improve the diversity of search, but also effectively solve the problems of invalid searches and local optima. In the multi-agent cooperative target search, we also further accelerate the search process through reward rules based on the odor effect to achieve a more effective search.

5.4. Comparison with other algorithms

In this section, to better illustrate the effectiveness of the improved algorithm, five other multi-agent reinforcement learning algorithms are selected to compare with the algorithm proposed in this study, including independent Q-learning (IQL) (Tampuu et al., 2017), value decomposition networks (VDN) (Sunehag et al., 2018), QMIX (Rashid et al., 2020), QTRAN (Son et al., 2019) and multi-agent variational exploration (MAVEN) (Mahajan et al., 2020). In a multi-agent environment, the state transfer and reward function are affected by the joint actions of all agents. For a given agent in the multi-agent system, its action value function needs to be determined based on the actions taken by other agents. IQL is a method to treat the remaining agents as a part of the environment, which means that each agent is solving a single agent mission. VDN is method for cooperative multi-agent learning that decomposes reward signals to each agent through back propagation. QMIX is a multi-agent reinforcement learning algorithm based on monotonic value function decomposition that can train decentralized strategies in a centralized end-to-end manner. The update method of its evaluation network is similar to that of DQN, which uses TD-Error for network self-updates. QTRAN is a new value decomposition algorithm designed to eliminate structural constraints in VDN and QMIX for the purpose of decomposing any multi-agent reinforcement learning mission. MAVEN hybridizes value and policy-based methods by introducing a latent space for hierarchical control, which can address complex multi-agent missions.

The above six algorithms are applied in a distributed search scenario where 3 agents search 15 targets, and IQL, VDN, QMIX, QTRAN and MAVEN use the ε-greedy strategy to select actions. Each algorithm is trained independently five times using random seeds, and then the

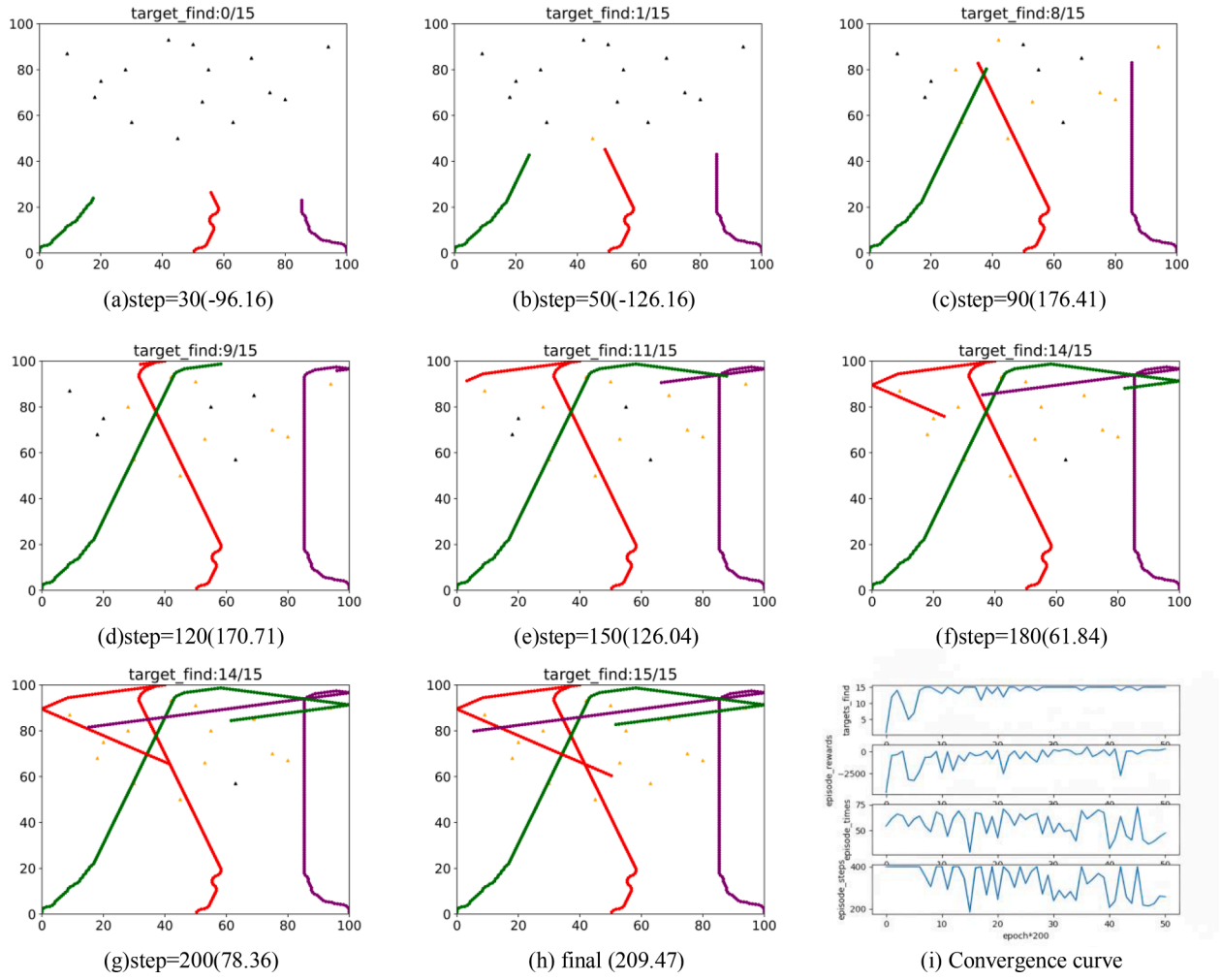


Fig. 14. Multi-target search movement trajectory in the distributed search mode (step = 214) Note: The green, red, and purple dots in the figure represent agents, the black triangles represent targets to be searched, and the black triangle becomes an orange triangle when the target is searched.

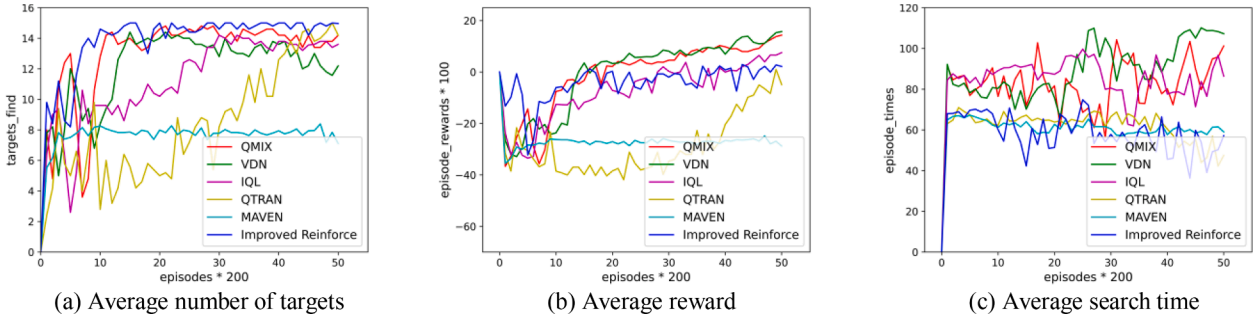


Fig. 15. Convergence curves of different multi-agent reinforcement learning algorithms.

Table 5
Statistical indicators of search results under different algorithms.

Algorithm	$succ$(%)	avg_{tgt}	avg_{rew}	avg_{time}	$rate_{tgt}$(%)	J^*
Improved Reinforce	69.41	13.77	−493.28	58.71	91.81	9.10
QMIX	37.25	12.87	−175.90	83.31	85.77	9.21
IQL	20.78	11.21	−809.93	85.81	74.73	6.38
VDN	20.39	11.97	−110.61	89.87	79.83	8.70
QTRAN	16.08	8.30	−2918.55	62.31	55.34	−1.07
MAVEN	2.75	7.62	−2783.74	61.37	50.77	−1.25

performance of the algorithms is evaluated from four aspects, namely, mission success rate, average number of targets, average reward and average search time. Among them, the mission success rate and average number of targets can be used to directly compare the performance of the algorithms; the higher the mission success rate and the larger the average number of targets, the better the performance of the algorithms. The average search time reflects the efficiency of algorithms to perform cooperative search missions; the shorter the average search time, the better the performance. Because the larger the average number of targets, the larger the total reward, the longer the search time and iteration steps, the smaller the total reward, it is difficult to judge the performance

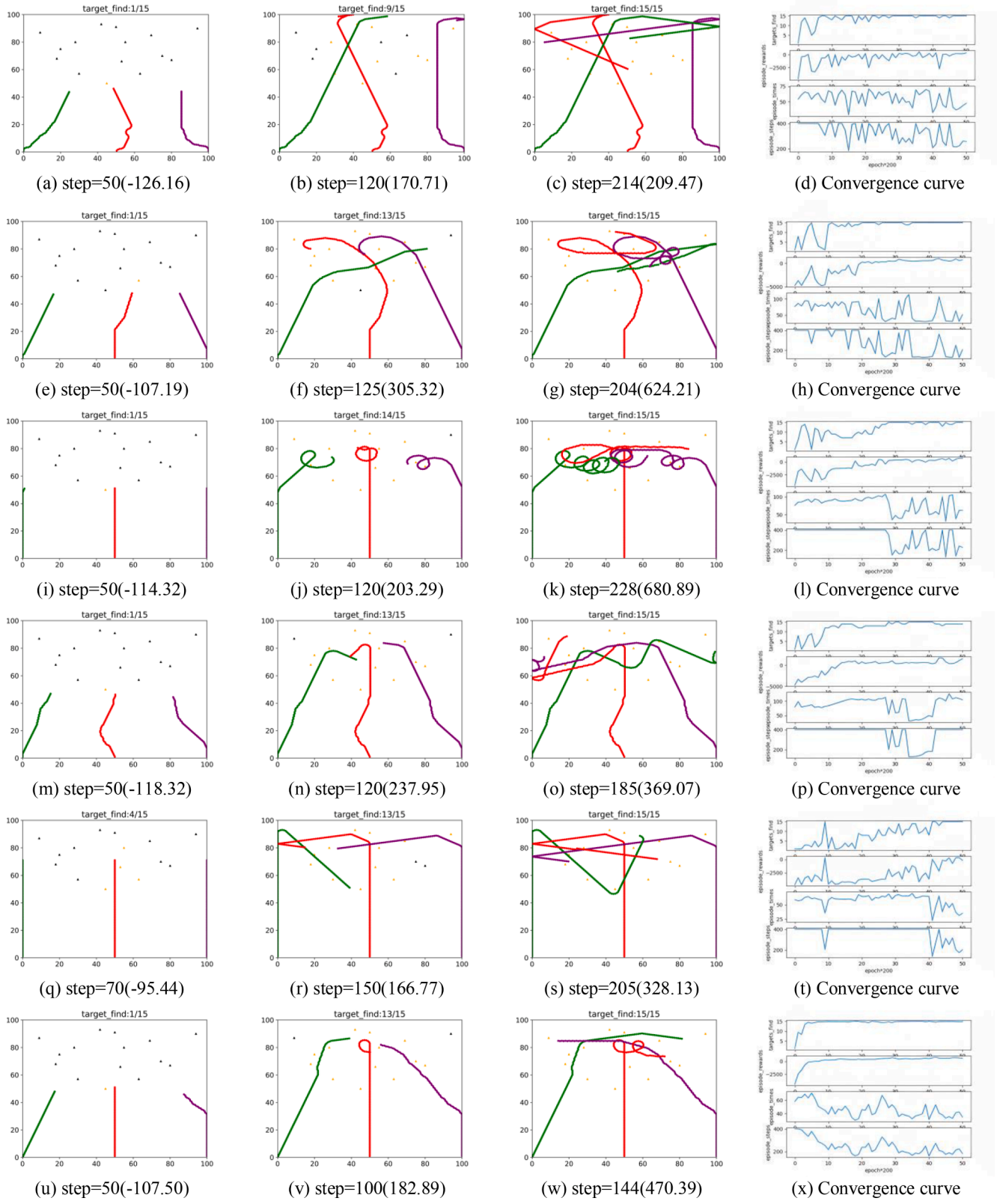


Fig. 16. Movement trajectory of agents under different algorithms Note: The green, red, and purple dots in the figure represent agents, the black triangles represent targets to be searched, and the black triangle becomes an orange triangle when the target is searched.

of algorithms by the average reward.

Fig. 15 shows the average number of targets, average reward and average search time of the improved Reinforce algorithm, IQL, VDN, QMIX, QTRAN and MAVEN. Based on the average number of targets, the improved Reinforce algorithm searches the most targets with the best performance in the middle and late iterations, QMIX is the second and MAVEN is the least effective. QTRAN has a good effect in the late iterations, but it requires a long learning time; MAVEN combines the value-

based method and policy-based method, resulting in low efficiency. From the perspective of the average search time, the improved Reinforce algorithm has the shortest search time, the highest search efficiency and the best performance.

Furthermore, Table 5 gives the specific values of different statistical indicators for the six algorithms. As seen from the table, the mission success rate of the improved algorithm is the highest, which is 69.41%, while the mission success rates of the other algorithms are all below 40%. Meanwhile, the improved algorithm has the largest target search rate, which is 91.87%, QMIX is the second with 85.77%, and the target search rates of the other four algorithms are below 80%. In addition, the improved algorithm has the shortest average search time and the fastest search process. Therefore, combined with Fig. 15 and Table 5, the improved algorithm in this paper performs the best in the multi-agent massive target cooperative search mission.

Finally, this section verifies the effectiveness of the improved algorithm by comparing the movement trajectory of the six algorithms in the scenario of 15 targets with 3 agents. As shown in Fig. 16, we selected the one with better search results among five independent repeated experiments under different algorithms, where (a), (b), (c), and (d) are the search trajectories and convergence curves of agents under the improved Reinforce algorithm, (e), (f), (g), and (h) are the search trajectories and convergence curves of the QMIX algorithm, (i), (j), (k), and (l) are the search trajectories and convergence curves of the IQL algorithm, (m), (n), (o), and (p) are the search trajectories and convergence curves of the VDN algorithm, (q), (r), (s), and (t) are the search trajectories and convergence curves of the QTRAN algorithm, and (u), (v), (w), and (x) are the search trajectories and convergence curves of the MAVEN algorithm. By comparing the movement trajectory of the agents, it can be found that the search trajectory of the improved Reinforce algorithm is the most concise. Other algorithms have better results in the early stage of the search, but there are invalid searches in the middle and late iterations, resulting in poor results.

6. Conclusion

The purpose of this paper is to discuss the improved Reinforce learning algorithm with the action preference selection strategy, which enables agents to search targets from different locations through cooperation, and expands the search scope through the information sharing mechanism to improve the search efficiency. The algorithm improves the action selection strategy by adding λ to the action preference function and assigning weights to them, which increases the exploration degree of the algorithm and solves the problem of local optima and invalid searches. Through simulation experiments, this paper evaluates the effects of parameter ω_2, the number of agents, the number of targets and search modes on the search efficiency, and compares the results with those of other multi-agent reinforcement learning algorithms. The simulation results show that the cooperative search of multiple agents can improve the mission success rate and target search rate compared with a single agent.

However, there are some limitations in this research. For example, only static target search problems have been studied, the search area is clear of obstacles, and the search diversity of agents is relatively poor. Future research work will focus on dynamic target hunting and complexity environments. Unlike mine detection and oil exploration, targets can move in dynamic target capture problems such as maritime personnel rescue and military strikes. The cooperative search problem is transformed into a tracking and hunting problem, that is, the agent needs to continuously track targets in all directions until all targets are captured or preset conditions are met. The improved reinforcement algorithm proposed in this paper is simple and efficient, but its theory and application are still in the exploration stage and are not sufficiently mature. However, this method has good application prospects and is worth further promotion for dynamic target hunting.

CRediT authorship contribution statement

Xiaoyan Wang: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft. **Xi Fang:** Conceptualization, Supervision, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

This work was supported by Equipment Pre-Research Ministry of Education Joint Fund [grant number 6141A02033703].

References

Acar, E. U., Choset, H., Zhang, Y., & Schervish, M. (2003). Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *International Journal of Robotics Research, 22*, 441–466. https://doi.org/10.1177/02783649030227002

Ataei, H. N., Ziarati, K., & Eghtesad, M. (2013). A BSO-based algorithm for multi-robot and multi-target search. *Lecture Notes in Computer Science, 7906*(1), 312–321. https://doi.org/10.1007/978-3-642-38577-3_32

Cai, Y., & Yang, S. X. (2013). An improved PSO-based approach with dynamic parameter tuning for cooperative multi-robot target searching in complex unknown environments. *International Journal of Control, 86*(10), 1720–1732. https://doi.org/10.1080/00207179.2013.794920

Cai, Y., & Yang, S. X. (2016). A PSO-based approach with fuzzy obstacle avoidance for cooperative multi-robots in unknown environments. *International Journal of Computational Intelligence & Applications, 15*(1), 1386–1391. https://doi.org/10.1142/s1469026816500012

Cai, Y., Yang, S. X., & Xu, X. (2013). A combined hierarchical reinforcement learning based approach for multi-robot cooperative target searching in complex unknown environments. 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Singapore, Singapore.

Cao, X., Sun, H., & Jan, G. E. (2018). Multi-AUV cooperative target search and tracking in unknown underwater environment. *Ocean Engineering, 150*, 1–11. https://doi.org/10.1016/j.oceaneng.2017.12.037

Cui, Y., Hu, W., & Rahmani, A. (2022). A reinforcement learning based artificial bee colony algorithm with application in robot path planning. *Expert Systems with Applications, 203*, Article 117389. https://doi.org/10.1016/j.eswa.2022.117389

Dadgar, M., Jafari, S., & Hamzeh, A. (2016). A PSO-based multi-robot cooperation method for target searching in unknown environments. *Neurocomputing, 177*, 62–74. https://doi.org/10.1016/j.neucom.2015.11.007

Daoun, D., Ibnat, F., Alom, Z., Aung, Z., & Azim, M. A. (2022). Reinforcement learning: A friendly introduction. *Lecture Notes in Networks and Systems, 309*, 134–146. https://doi.org/10.1007/978-3-030-84337-3_11

Din, A., Jabeen, M., Zia, K., Khalid, A., & Saini, D. K. (2018). Behavior-based swarm robotic search and rescue using fuzzy controller. *Computers & Electrical Engineering, 70*, 53–65. https://doi.org/10.1016/j.compeleceng.2018.06.003

Doctor, S., Venayagamoorthy, G. K., & Gudise, V. G. (2004). Optimal PSO for collective robotic search applications. Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA.

Ebel, H., & Eberhard, P. (2019). Optimization-driven control and organization of a robot swarm for cooperative transportation. *IFAC-PapersOnLin, 52*(15), 115–120. https://doi.org/10.1016/j.ifacol.2019.11.660

Fiorini, P., & Botturi, D. (2008). Introducing service robotics to the pharmaceutical industry. *Intelligent Service Robotics, 1*(4), 267–280. https://doi.org/10.1007/s11370-008-0019-2

Garg, V., Shukla, A., & Tiwari, R. (2022). AERPSO — An adaptive exploration robotic PSO based cooperative algorithm for multiple target searching. *Expert Systems with Applications, 209*, Article 118245. https://doi.org/10.1016/j.eswa.2022.118245

Huang, J., Sun, W., & Gao, Y. (2020). Cooperative searching for the multi-UAVs based on dual-attribute probability model optimization. *Systems Engineering and Electronics, 42*(1), 118–127.

Lai, J., & Rui, R. (2020). Application of deep reinforcement learning in indoor UAV target search. *Computer Engineering and Applications, 56*(17), 156–160.

Luo, J., Han, Y., & Fan, L. (2018). Underwater acoustic target tracking: A review. *Sensors, 18*(1), 112. https://doi.org/10.3390/s18010112

Luo, Q., Luan, T. H., Shi, W., & Fan, P. (2023). Deep reinforcement learning based computation offloading and trajectory planning for multi-UAV cooperative target

search. *IEEE Journal on Selected Areas in Communications, 41*(2), 504–520. https://doi.org/10.1109/jsac.2022.3228558
Mahajan, A., Rashid, T., Samvelyan, M., & Whiteson, S. MAVEN: Multi-agent variational exploration. arXiv:1910.07483v2 [Preprint], Jan 20, 2020 [cited 2023 Mar 16]. https://doi.org/10.48550/arXiv.1910.07483.
Morin, M., Abi-Zeid, I., & Quimper, C.-G. (2023). Ant colony optimization for path planning in search and rescue operations. *European Journal of Operational Research, 305*(1), 53–63. https://doi.org/10.1016/j.ejor.2022.06.019
Mou, J., Hu, T., Chen, P., & Chen, L. (2021). Cooperative MASS path planning for marine man overboard search. *Ocean Engineering, 235*, Article 109376. https://doi.org/10.1016/j.oceaneng.2021.109376
Paez, D., Romero, J. P., Noriega, B., Cardona, G. A., & Calderon, J. M. (2021). Distributed particle swarm optimization for multi-robot system in search and rescue operations. *IFAC-PapersOnLine, 54*(4), 1–6. https://doi.org/10.1016/j.ifacol.2021.10.001
Połap, D., & Woźniak, M. (2022). A hybridization of distributed policy and heuristic augmentation for improving federated learning approach. *Neural Networks, 146*, 130–140. https://doi.org/10.1016/j.neunet.2021.11.018
Prasetya, D. A., Yasuno, T., Suzuki, H., & Kuwahara, A. (2013). Cooperative control system of multiple mobile robots using particle swarm optimization with obstacle avoidance for tracking target. *Journal of Signal Processing, 17*(5), 199–206. https://doi.org/10.2299/jsp.17.199
Purbolingga, Y., Jazidie, A., & Effendi, R. (2019). Modified ant colony algorithm for swarm multi agent exploration on target searching in unknown environment. 2019 International Conference of Artificial Intelligence and Information Technology, Yogyakarta, Indonesia.
Rashid, T., Samvelyan, M., Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research, 21*, 7234–7284.
Robin, C., & Lacroix, S. (2016). Multi-robot target detection and tracking: Taxonomy and survey. *Autonomous Robots, 40*(4), 729–760. https://doi.org/10.1007/s10514-015-9491-7
Senanayake, M., Senthooran, I., Barca, J. C., Chung, H., Kamruzzaman, J., & Murshed, M. (2016). Search and tracking algorithms for swarms of robots: A survey. *Robotics & Autonomous Systems, 75*, 422–434. https://doi.org/10.1016/j.robot.2015.08.010
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G., … Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961
Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 363*(6419), 1–5.
Son, K., Kim, D., Kang, W. J., Hostallero, D. E., & Yi, Y. (2019). QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *Statistics*, 5887–5896.
Sun, X., & Cai, C. (2014). A cooperative target searching method based on multiple ant colony optimization algorithm. *Tactical Missile Technology, 6*(6), 26–31.
Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., ... Graepel, T. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. AAMAS '18: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden.
Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT press.
Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., … Aru, J. (2017). Multi-agent cooperation and competition with deep reinforcement learning. *PLoS One, 12*(4), 1–15. https://doi.org/10.1371/journal.pone.0172395
Tan, Z., & Karakose, M. (2020). Optimized deep reinforcement learning approach for dynamic system. 2020 IEEE International Symposium on Systems Engineering, Vienna, Austria.
Tang, H., Sun, W., Lin, A., Xue, M., & Zhang, X. (2021). A GWO-based multi-robot cooperation method for target searching in unknown environments. *Expert Systems with Applications, 186*, Article 115795. https://doi.org/10.1016/j.eswa.2021.115795
Tang, H., Sun, W., Yu, H., Lin, A., & Xue, M. (2020). A multirobot target searching method based on bat algorithm in unknown environments. *Expert Systems with Applications, 141*, Article 112945. https://doi.org/10.1016/j.eswa.2019.112945
Tang, H., Sun, W., Yu, H., Lin, A., Xue, M., & Song, Y. (2019). A novel hybrid algorithm based on PSO and FOA for target searching in unknown environments. *Applied Intelligence, 49*(7), 2603–2622. https://doi.org/10.1007/s10489-018-1390-0
Wang, R., & Xiao, B. (2019). Cooperative search for multi-UAVs via an improved pigeon-inspired optimization and Markov chain approach. *Chinese Journal of Engineering, 41*(10), 1342–1350. https://doi.org/10.13374/j.issn2095-9389.2018.09.02.002
Woźniak, M., Siłka, J., & Wieczorek, M. (2021). Deep neural network correlation learning mechanism for CT brain tumor detection. *Neural Computing and Applications, 1–16*. https://doi.org/10.1007/s00521-021-05841-x
Xing, D., Zhen, Z., Zhou, C., & Gong, H. (2019). Cooperative search of UAV swarm based on ant colony optimization with artificial potential field. *Transactions of Nanjing University of Aeronautics and Astronautics, 36*(6), 912–918. https://doi.org/10.16356/j.1005-1120.2019.06.004
Xue, S., & Zeng, J. (2008). Swarm robotics: A survey. *Pattern Recognition & Artificial Intelligence, 21*(2), 177–186.
Yan, F., Di, K., Jiang, J., Jiang, Y., & Fan, H. (2019). Efficient decision-making for multiagent target searching and occupancy in an unknown environment. *Robotics and Autonomous Systems, 114*, 41–56. https://doi.org/10.1016/j.robot.2019.01.017
Yang, Q., Gao, Y., Guo, Y., Xia, B., & Yang, K. (2022). Target search path planning for naval battle field based on deep reinforcement learning. *Systems Engineering and Electronics, 44*(11), 3486–3495. https://doi.org/10.12305/j.issn.1001-506X.2022.11.24
Yao, P., & Zhao, Z. (2021). Improved Glasius bio-inspired neural network for target search by multi-agents. *Information Sciences, 568*, 40–53. https://doi.org/10.1016/j.ins.2021.03.056
Zhang, H., Sun, Y., Wang, P., & Zhang, H. (2022). Multiple AUUV cooperative area target searching method based on distributed model. *AIP Advances, 12*(6), 1–7. https://doi.org/10.1063/5.0098293
Zhang, J., Zhou, D., & Zhang, K. (2011). Algorithm based on reinforcement learning for UAV search. *Application Research of Computers, 28*(10), 3659–3661.
Zhou, J., Zhao, X., Zhang, X., Zhao, D., & Li, H. (2020). Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm. *IEEE Access, 8*, 19306–19318. https://doi.org/10.1109/access.2020.2967061
Zhou, W., Li, J., Liu, Z., & Shen, L. (2022). Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning. *Chinese Journal of Aeronautics, 35*(7), 100–112.