

Bidirectional Task–Motion Planning Based on Hierarchical Reinforcement Learning for Strategic Confrontation

Qizhen Wu¹, Lei Chen², Kexin Liu³, and Jinhu Lü⁴

Abstract—In swarm robotics, confrontation scenarios, including strategic confrontations, require efficient decision-making that integrates discrete commands and continuous actions. Traditional task and motion planning methods separate decision-making into two layers, but their unidirectional structure fails to capture the interdependence between these layers, limiting adaptability in dynamic environments. Here, we propose a novel bidirectional approach based on hierarchical reinforcement learning, enabling dynamic interaction between the layers. This method effectively maps commands to task allocation and actions to path planning, while leveraging cross-training techniques to enhance learning across the hierarchical framework. Furthermore, we introduce a trajectory prediction model that bridges abstract task representations with actionable planning goals. In our experiments, it achieves over 80% in confrontation win rate and under 0.01 seconds in decision time, outperforming existing approaches. Demonstrations through large-scale tests and real-world robot experiments further emphasize the generalization capabilities and practical applicability of our method.

I. INTRODUCTION

Recent advances in artificial intelligence lead to significant progress in robotics [1], [2], with particular attention given to robotic swarm confrontations [3], [4]. In these scenarios, AI-driven decision-making empowers robots to engage in strategic tasks such as pursuit–evasion [5], defense–attack [6], and strategic confrontations [7]. These applications highlight the need for effective decision-making mechanisms capable of handling a hybrid process, including discrete commands and continuous actions [8], which are central to the success of modern robotic systems.

Traditional methods for solving such problems rely heavily on task and motion planning (TAMP) [9], which divides the problem into two distinct decision-making layers. Although conventional TAMP methods [8], [10] demonstrate effectiveness by organizing tasks within detailed planning frameworks, they suffer limitations due to their reliance on expert knowledge. The complexity of modeling and task specification renders these methods less accessible for non-expert users. Heuristic algorithms [11], [12] attempt to

*This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3305600, and in part by the National Natural Science Foundation of China under Grants 62141604, 62088101, and 62003015. (Corresponding author: Lei Chen)

¹Qizhen Wu, ³Kexin Liu, and ⁴Jinhu Lü are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. (wuqzh7@buaa.edu.cn, skxliu@163.com, jhlu@iss.ac.cn)

²Lei Chen is with the Advanced Research Institute of Multidisciplinary Sciences and State Key Laboratory of CNS/ATM, Beijing Institute of Technology, Beijing 100081, China. (bit_chen@bit.edu.cn)

The experiment video is available at https://www.bilibili.com/video/BV1JTwmeaEeN/?vd_source=9de61aecdd9fb684e546d032ef7fe7bf.

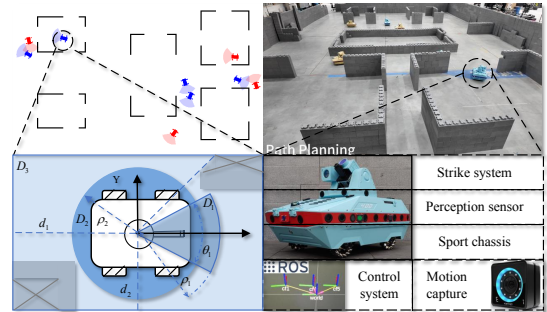


Fig. 1. Simulations and real-robot systems in a confrontation scenario.

address these challenges by adaptively searching for feasible solutions, but the independent nature of the hierarchical architecture prevents the upper layer from verifying the enforceability of commands transferred to the lower layer. Furthermore, TAMP methods struggle with generalization, as evidenced by the frequent need for repeated adjustments [13], [14] when adapting to dynamic environments or scaling to larger robot populations.

In contrast, deep reinforcement learning (DRL) [15], [16] emerges as a promising alternative by using a single-layer framework [17] that learns optimal decision-making strategies through end-to-end training. It aims to maximize cumulative rewards, fine-tuning its decision network to produce effective strategies. However, in complex environments [18], DRL encounters challenges due to its focus on global objectives, leading to sparse-reward scenarios [19]. This results in practical limitations when applied to intricate real-world tasks.

To address these limitations, hierarchical reinforcement learning (HRL) [20] provides a more flexible solution by decomposing the complex task into two independently trainable networks, similar to the TAMP approach. It fosters a dynamic feedback loop between the two layers, enabling them to communicate and refine their strategies as they learn. In HRL, the upper layer divides the task into temporal segments and assigns tailored rewards to guide the lower layer. The method allows the upper layer to refine its task decomposition based on the learning progress of the lower layer, offering a more adaptable framework to tackle complex problems [21], [22]. This hierarchical structure enables more detailed problem-solving, particularly in scenarios where discrete commands and continuous actions are closely intertwined.

In this paper, we propose an innovative HRL method to tackle the unique challenges of swarm confrontation.

Unlike previous approaches, our method introduces a two-layer interaction mechanism that enhances both the accuracy and stability of decision-making. Specifically, we integrate task allocation (for discrete commands) and path planning (for continuous actions) within a bidirectional closed-loop architecture. This structure enables effective communication between the two layers, ensuring the enforceability of assigned tasks. Additionally, we introduce a trajectory prediction model that bridges the gap between abstract task representations and executable planning targets, such as predicting the most likely location of enemies in an evasion task. Our experimental results demonstrate that our approach outperforms baseline methods, including traditional TAMP approaches and single-layer DRL, especially in terms of generalization across different scales. Through ablation studies, we highlight the critical importance of the interaction mechanism and show that our method exhibits superior adaptability in real-robot systems illustrated in Fig. 1.

We organize the rest of the paper as follows. Section II outlines the preliminaries of goal-conditioned HRL and formulates the strategic confrontation problem. Section III elaborates on the proposed HRL method and its implementation. Section IV presents the experimental evaluations of our approach. Finally, Section V concludes the paper.

II. PRELIMINARIES AND PROBLEM DESCRIPTION

A. Definition of strategic confrontation

This study models strategic confrontation as a dynamic process that involves task allocation and path planning in a static-obstacle environment. In this scenario, the blue and red agents compete against each other, with the one that eliminates all opponents being declared the winner. To simplify the agents' movement, we consider the confrontation in two-dimensional space. Let \mathbf{p} and \mathbf{v} denote the position and velocity vectors in two dimensions, respectively. Agents on both sides are subject to the same constraints on their abilities. For the i -th agent $\mathbf{u}_i = (\mathbf{p}_i, \mathbf{v}_i, f_i)$, we define $\mathbb{U}_a = \{u_{a,j} | j = 1, \dots, N_a\}$ and $\mathbb{U}_e = \{u_{e,k} | k = 1, \dots, N_e\}$ as the sets of N_a allies and N_e enemies, respectively. Here, f_i represents the task currently performed by the i -th agent, defined as searching ($f_i = -3$), escaping ($f_i = -2$), supporting ($f_i = -1$), or chasing ($f_i = 0$). At each time step, every agent assigns a task and then performs path planning based on the task goal to ensure obstacle avoidance.

As shown in Fig. 1, each agent has an attack range D_1 modeled as a sector with radius ρ_1 and angle θ_1 . When an opponent enters the attack range D_1 , the agent automatically launches an attack. At this point, the probability of the bullet hitting and causing the opponent to die is ϵ . By defining an avoidance range D_2 with radius ρ_2 , the agent plans feasible paths to avoid collisions with obstacles. In this work, the i -th agent gathers information about neighboring allies and enemies through communication and perception, respectively. We define the communication and perception ranges as a rectangle with length d_1 and width d_2 . While obstacles obstruct perception, they do not interfere with communication. When the j -th agent is within the communication and

perception ranges of the i -th agent, we label it as $j \in \mathbb{N}_i$, where \mathbb{N}_i denotes the set of agents observable by the i -th agent. This confrontation setting provides the foundational context for the goal-conditioned HRL framework.

B. Goal-conditioned HRL

Let \mathbb{R} denote the set of real numbers, \mathbb{R}^+ the set of positive real numbers, and \mathbb{Z} the set of integers. We formulate a long-horizon planning task for N agents as a goal-conditioned Markov Decision Process (goal-conditioned MDP), represented by the eight-tuple $(N, \mathbb{S}, \mathbb{G}, \mathbb{O}, \mathbb{A}, P, R, \gamma)$, where:

- $\mathbb{S} \subseteq \mathbb{R}^{m_s}$, $\mathbb{A} \subseteq \mathbb{R}^{m_a}$, $\mathbb{O} \subseteq \mathbb{R}^{m_o}$, and $\mathbb{G} \subseteq \mathbb{R}^{m_g}$ represent the state space, the action space, the observation space, and the goal space, respectively.
- m_s, m_a, m_o , and m_g denote the dimensions of the state, action, observation, and goal spaces, respectively.
- $\mathbf{s} \in \mathbb{S}$ represents the state of the environment.
- $\mathbf{z}_i \in \mathbb{O}$ represents the observation of the i -th agent and $\mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$ denotes the joint observations.
- $\mathbf{a}_i \in \mathbb{A}$ represents the action of the i -th agent and $\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_N)$ denotes the joint actions.
- $P(\mathbf{s}, \mathbf{a}) : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow [0, 1]$ denotes the state transition probabilities from $\mathbf{s}(t)$ to $\mathbf{s}(t+1)$ under $\mathbf{a}(t)$.
- $R(\mathbf{s}, \mathbf{a}, \mathbf{g}) : \mathbb{S} \times \mathbb{A} \times \mathbb{G} \rightarrow \mathbb{R}$ is the reward function.
- $\gamma \in (0, 1)$ represents the discount factor.

The subgoal \mathbf{q} and the goal \mathbf{g} are defined in the goal space \mathbb{G} , which is considered a subspace of \mathbb{S} , with a mapping function $\varphi : \mathbb{S} \rightarrow \mathbb{G}$. HRL typically involves two-layer policies: an upper-layer policy $\pi_u(\mathbf{q}_i(t) | \mathbf{z}_i(t), \mathbf{g})$ and a lower-layer policy $\pi_l(\mathbf{a}_i(t) | \mathbf{z}_i(t), \mathbf{q}_i(t))$. We assume that each subtask consists of h steps. The upper layer predicts the subgoal $\mathbf{q}_{i,m}$, $m = 0, 1, \dots, T_h$, where $T_h = T/h$ and T is the total time. Upon receiving the observation $\mathbf{z}_{i,m+n}$, the lower layer attempts to reach the subgoal by executing the appropriate action $\mathbf{a}_{i,m}$. Starting from the state \mathbf{s}_m , the state-action trajectory for the m -th subtask can be represented as $\tau_m = \{(\mathbf{s}_{m+n}, \mathbf{a}_n, \mathbf{s}_{m+n+1}) | \mathbf{s}_m, \mathbf{q}_m\}_{n=0}^{h-1}$.

To ensure the enforceability of the predicted subgoal, the upper-layer policy is guided by the reward feedback from the environment. The reward function and optimization objective in the upper layer can be described as follows:

$$R_u(\tau_m) = \sum_{n=0}^{h-1} r_{\text{env}}(\mathbf{s}_{m+n}, \mathbf{a}_n, \mathbf{g}), \quad (1)$$

$$\pi_u^* = \arg \max_{\pi_u} \sum_{m=0}^{T_h-1} \gamma^m R_u(\tau_m). \quad (2)$$

Here, $\arg \max_x F(x)$ represents the value of the variable x when $F(x)$ reaches its maximum value. The intrinsic reward function of the lower layer is designed to achieve the subgoal provided by the upper layer. The reward function and optimization objective in the lower layer can be described as

$$R_l(\mathbf{s}_{m+n}, \mathbf{a}_n, \mathbf{q}_m) = -D(\mathbf{s}_{m+n}, \mathbf{q}_m), \quad (3)$$

$$\pi_l^* = \arg \max_{\pi_l} \sum_{n=0}^{T-1} \gamma^n R_l(\mathbf{s}_n, \mathbf{a}_n, \mathbf{q}_{\lfloor n/h \rfloor}), \quad (4)$$

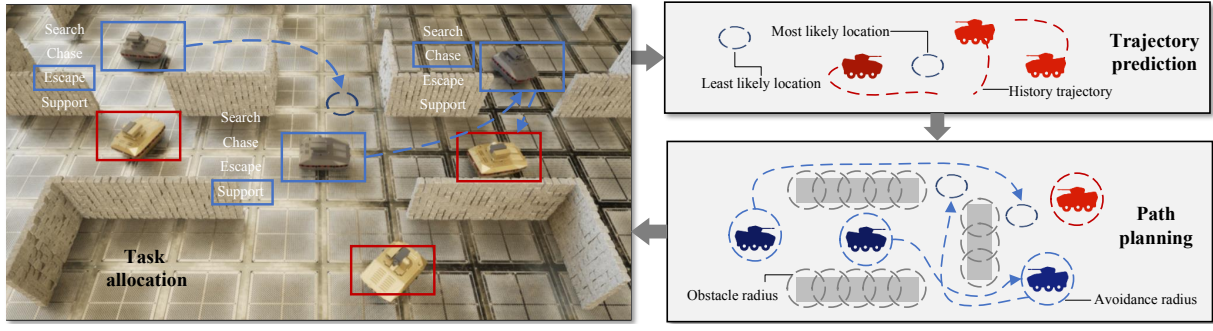


Fig. 2. Overview of the proposed framework integrating task allocation (left), trajectory prediction (top right), and path planning (bottom right) modules.

where $D : \mathbb{S} \times \mathbb{G} \rightarrow \mathbb{R}^+$ is a continuous distance used to characterize the difference between the state s and subgoal q . Additionally, $\lfloor x \rfloor = \max \{m \in \mathbb{Z} | m \leq x\}$, where $x \in \mathbb{R}$.

III. METHODOLOGY

In this section, we present the HRL method to address the strategic confrontation problem in a static-obstacle environment. An overview of our method is shown in Fig. 2. The lower layer utilizes a multi-agent deep deterministic policy gradient (MADDPG) algorithm [5] for path planning based on subgoals. The upper layer employs a decentralized deep Q-learning (DQN) algorithm [7] for task allocation. This method integrates the cumulative rewards from the lower layer into the upper layer, establishing an interaction mechanism between the two layers. In addition, we employ a trajectory prediction model to bridge abstract task representations with concrete planning goals. Finally, we introduce a cross-training strategy to enhance the training efficiency and stability of the proposed method.

A. Lower Layer for Path Planning

We enclose all rectangular obstacles with N_o circles of radius ρ_3 , where the position of the center of each circle is denoted as $\mathbf{p}_{o,c}$, with $c = 1, \dots, N_o$. Moreover, we consider allies within the communication range and enemies within the perception range as neighboring agents \mathbf{u}_j , where $j \in \mathbb{N}_i$. Based on this, we create the following observation:

$$\mathbf{z}_{l,i} = (\mathbf{p}_i, \mathbf{p}_j, \mathbf{v}_i, \{\mathbf{v}_j\}_{j \in \mathbb{N}_i}, \{\mathbf{p}_{o,c}\}_{c \in \mathbb{N}_i}). \quad (5)$$

The action consists of the velocity magnitude $\|\mathbf{v}\| \in [0, \|\mathbf{v}\|_{\max}]$ and velocity direction $\psi \in [-\pi, \pi]$ of the i -th agent, denoted as

$$\mathbf{a} = (\|\mathbf{v}_i\|, \psi_i), \quad (6)$$

where $\|\cdot\|$ denotes the Euclidean norm and $\|\mathbf{v}\|_{\max}$ denotes the maximum velocity magnitude. Based on the current state and actions, we can calculate the position at the next time step using the first-order dynamics model

$$\begin{aligned} p_x(t+1) &= p_x(t) + \delta t \|\mathbf{v}\| \cos \psi, \\ p_y(t+1) &= p_y(t) + \delta t \|\mathbf{v}\| \sin \psi, \end{aligned} \quad (7)$$

where p_x and p_y denote the horizontal and vertical coordinates of the agent's position, respectively. The symbol δt is the length of each time step.

We design an avoidance reward to guide agents in avoiding collisions with obstacles and other agents. Specifically, if the i -th agent collides ($\|\mathbf{p}_i - \mathbf{p}_j\| < 2\rho_2$ or $\|\mathbf{p}_i - \mathbf{p}_{o,c}\| < \rho_2 + \rho_3$), the avoidance reward $r_{i,a} = -1$. Otherwise, $r_{i,a} = 0$. This reward, therefore, actively promotes collision avoidance and safe navigation. Although avoiding collisions is crucial, agent pathfinding must also prioritize minimizing path length to ensure efficiency. The upper layer introduces an intrinsic reward $r_{i,b}$, which is defined as follows:

$$r_{i,b} = -\frac{\|\mathbf{q}_i(\lfloor t/h \rfloor) - \mathbf{p}_i(t)\|}{\|\mathbf{q}_i(\lfloor t/h \rfloor) - \mathbf{p}_i(\lfloor t/h \rfloor)\|}. \quad (8)$$

To combine both the avoidance and path minimization objectives, we define the total reward in the lower layer as the sum of the avoidance and intrinsic rewards:

$$R_l(\mathbf{s}_{m+n}, \mathbf{a}_n, \mathbf{q}_m) = \sum_i^N (\epsilon_1 r_{i,a} + (1 - \epsilon_1) r_{i,b}). \quad (9)$$

where ϵ_1 is the weighting factor that balances avoidance reward and intrinsic reward. It encourages the agent to navigate safely while minimizing the path length, facilitating efficient completion of its target-reaching task.

B. Upper Layer for Task Allocation

The goal \mathbf{g} of the confrontation is to eliminate as many enemies as possible. Given the limited observations in the confrontation scenario, the local observation of the i -th agent includes information about observable allies and enemies, expressed as

$$\mathbf{z}_{u,i} = (\mathbf{u}_i, \mathbf{u}_{a,j}, \mathbf{u}_{e,k}), \quad (10)$$

where $j, k \in \mathbb{N}_i$. Furthermore, the tasks of enemies $\mathbf{u}_{e,k}$ are unknown to the i -th agent.

Based on the goal and observation, the agent chooses to either support an ally that is escaping or chasing, or to chase an enemy at a relative disadvantage. The upper layer generates the subgoal for the i -th agent by determining the target position defined as follows:

$$\mathbf{q}_i = \mathbf{p}_{a,j} \text{ or } \mathbf{p}_{e,k}, \quad (11)$$

where $f_{a,j} = -2$ or $f_{a,j} = 0$, and $Adv(\mathbf{u}_i, \mathbf{u}_{e,k}) > 0$. The advantage function Adv depends on the positions and velocities of \mathbf{u}_i and $\mathbf{u}_{e,k}$, denoted as $Adv(\mathbf{u}_i, \mathbf{u}_{e,k}) = \cos \theta_{i,k}$,

where $\theta_{i,k}$ represents the angle between the vectors \mathbf{v}_i and $\mathbf{p}_{e,k} - \mathbf{p}_i$. The selection of enemies and allies determines whether the agent is engaged in a chasing or supporting task. If neither allies nor enemies, as described above, are within the communication and perception ranges, the agent will perform a searching or escaping task and invoke the trajectory prediction model. After the lower layer executes h steps, we feed the cumulative rewards to the upper layer, which adjusts its strategy based on the performance of the lower layer through bidirectional feedback. Therefore, the reward function of the upper layer in (1) is improved as

$$R_u(\tau_m) = r_{\text{env}}(\mathbf{s}_{m+n}, \mathbf{q}_m, \mathbf{g}) + \sum_{n=0}^{h-1} R_l(\mathbf{s}_{m+n}, \mathbf{a}_n, \mathbf{q}_m), \quad (12)$$

where r_{env} is the confrontation reward, defined as the total number of surviving allies and defeated enemies during these h steps. The interaction mechanism between the lower and upper layers is illustrated in Fig. 3.

C. Embedded Trajectory Prediction Model

When the agent performs a searching or escaping task, it invokes a trajectory prediction model to compute the subgoal. In this paper, we assume that a global planner stores the historical trajectories of all enemies $\mathbf{p}_{e,k}(t)$, $t = 0, \dots, T_n$, where T_n represents the current time step. The planner then makes a preliminary trajectory prediction using a second-order dynamic model

$$\mathbf{p}_{e,k}(T_n + 1) = \mathbf{p}_{e,k}(T_n) + \delta t \mathbf{v}_{e,k}(T_n) + \frac{1}{2} \delta t^2 \dot{\mathbf{v}}_{e,k}(T_n), \quad (13)$$

where $\dot{\mathbf{v}}_{e,k}(T_n) = (\mathbf{v}_{e,k}(T_n) - \mathbf{v}_{e,k}(T_n - 1)) / \delta t$. To avoid collisions with obstacles in the confrontation scenario, the planner performs collision detection for each trajectory and further refines the trajectory through the potential field method. We consider an obstacle circle $c \in \mathbb{N}_k$ if it satisfies $\|\mathbf{p}_{e,k}(T_n + 1) - \mathbf{p}_{o,c}\| < \rho_2 + \rho_3$, where \mathbb{N}_k denotes the set of obstacles that collide with the enemy k . The predicted position is then adjusted as follows:

$$\mathbf{p}_{e,k}(T_n + 1) = \mathbf{p}_{e,k}(T_n + 1) + \delta t \sum_{c \in \mathbb{N}_k} \frac{\mathbf{p}_{e,k}(T_n + 1) - \mathbf{p}_{o,c}}{\|\mathbf{p}_{e,k}(T_n + 1) - \mathbf{p}_{o,c}\|} d_3, \quad (14)$$

where $d_3 = \rho_2 + \rho_3 - \|\mathbf{p}_{e,k}(T_n + 1) - \mathbf{p}_{o,c}\|$. After iterating until time step $T_n + T_p$, the planner employs a hierarchical clustering algorithm [23] to group all predicted trajectory points. To generate trajectories for clustering, we repeatedly roll out the prediction model from each enemy's most recent position. The resulting positions are then used for hierarchical clustering. The center of the b_1 -th cluster, representing one of the most likely locations of enemies, is given by

$$\mathbf{p}_{\text{cluster}, b_1} = \frac{1}{N_p} \sum_{n=1}^{N_p} \mathbf{p}_n, \quad (15)$$

where \mathbf{p}_n is the position of each point in the cluster, and N_p is the number of points in the cluster. The symbol T_p

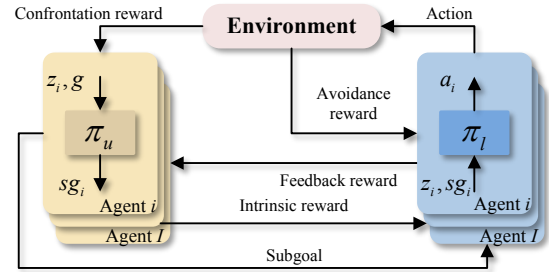


Fig. 3. Interaction mechanism of two-layer networks.

is a constant that indicates the length of the prediction time step. If there are no enemies within the perception range, and no allies in chasing or escaping within the communication range, the i -th agent will switch to a searching task, selecting the nearest center as its subgoal

$$\mathbf{q}_i = \arg \min_{\mathbf{p}_{\text{cluster}, b_1}} \|\mathbf{p}_{\text{cluster}, b_1} - \mathbf{p}_i\|. \quad (16)$$

Otherwise, if an enemy is within the perception range of the i -th agent and satisfies $\text{Adv}(\mathbf{u}_i, \mathbf{u}_{e,k}) < 0$, the agent will perform an escaping task. To compute the least likely locations of enemies, the planner randomly generates several candidate points that satisfy the following conditions:

$$\|\mathbf{p}_{\text{candidate}, b_2} - \mathbf{p}_{\text{cluster}, b_1}\| > d_4, \forall b_1, \forall b_2, \quad (17)$$

where d_4 is a constant indicating the distance of the candidate points from the most likely locations. The subgoal of the escaping agent can be expressed as

$$\mathbf{q}_i = \arg \min_{\mathbf{p}_{\text{candidate}, b_2}} \|\mathbf{p}_{\text{candidate}, b_2} - \mathbf{p}_i\|. \quad (18)$$

The subgoal is then transmitted from the global planner to the i -th agent.

D. Training Methods for Two Layers

Combining the goal-conditioned MDP framework, we adopt double DQN for training in the upper layer. In this approach, each agent estimates the optimal state-action value function $Q_i^* : \mathbb{S} \times \mathbb{A} \times \mathbb{G} \rightarrow \mathbb{R}$ through a parameterized neural network $Q_{\phi_{u,i}}(\mathbf{s}_i(t), \mathbf{a}_i(t), \mathbf{g})$. The subscript $\phi_u = (\phi_{u,1}, \dots, \phi_{u,N})$ represents the weight parameters of the value-function network in the upper layer. For $\gamma \approx 1$, Q^* estimates the discounted returns of the optimal strategy over an infinite horizon. The method approximates Q^* using the network $Q_{\phi_{u,i}}$, and the loss function is formulated as

$$L(\phi_{u,i}) = E_{\mathbf{s}, \mathbf{a}, \mathbf{g}} \|Q_{\phi_{u,i}}(\mathbf{s}_i(t), \mathbf{a}_i(t), \mathbf{g}) - y_i\|^2, \quad (19)$$

where E denotes the mathematical expectation. The Q -target y_i is defined as

$$y_i = R_u(\mathbf{s}_i(t), \mathbf{a}_i(t), \mathbf{g}) + \gamma \max Q_{\phi_{u,i}}(\mathbf{s}_i(t+h), \mathbf{a}_i(t+h), \mathbf{g}). \quad (20)$$

To enable agents to make decisions based on local observations while collaborating with each other to achieve subgoals in the lower layer, we use MADDPG, a method that employs

Algorithm 1 Cross-Training Method

Input: the number of training steps T ; the number of cross-training episodes E_c ; the length of each subtask h ;

Initialization: the parameters of the networks in the upper layer ϕ_u and lower layer ϕ_l, ω_l ;

```
1: for all  $e = 1 \rightarrow E_c$  do
2:    $t_0 \leftarrow 0$ 
3:   for all  $t = 0 \rightarrow T$  do
4:      $R_l \leftarrow \text{Eq. (9)}$ ;
5:      $L(\phi_l) \leftarrow \text{Eq. (21)}$ ,  $L(\omega_l) \leftarrow \text{Eq. (23)}$ ;
6:     Refresh networks  $Q_{\phi_l}, \pi_{\omega_l}$ ;  $\triangleright$  Lower layer training
7:     if  $t = t_0 + h$  then
8:        $R_u \leftarrow \text{Eq. (12)}$ ;
9:        $L(\phi_u) \leftarrow \text{Eq. (19)}$ ;
10:      Refresh network  $Q_{\phi_u}$ ;  $\triangleright$  Upper layer training
11:       $t_0 \leftarrow t$ 
12:    end if
13:  end for
14: end for
```

centralized training with decentralized execution. It designs a separate critic network $Q_{\phi_{l,i}}(\mathbf{s}(t), \mathbf{a}(t), \mathbf{q}(\lfloor t/h \rfloor))$ for each agent, which is updated as follows:

$$L(\phi_{l,i}) = E_{\mathbf{s}, \mathbf{a}, \mathbf{q}} \|Q_{\phi_{l,i}}(\mathbf{s}(t), \mathbf{a}(t), \mathbf{q}(\lfloor t/h \rfloor)) - y_i\|^2, \quad (21)$$

where the subscript $\phi_l = (\phi_{l,1}, \dots, \phi_{l,N})$ represents the weight parameters of the critic network in the lower layer. The Q -target is revised as

$$y_i = R_l(\mathbf{s}_i(t), \mathbf{a}_i(t), \mathbf{q}_i(\lfloor t/h \rfloor)) + \gamma \max Q_{\phi_{l,i}}(\mathbf{s}(t+1), \mathbf{a}(t+1), \mathbf{q}(\lfloor t/h \rfloor + 1)). \quad (22)$$

In addition, each agent maintains a policy network $\pi_{\omega_{l,i}}(\mathbf{z}_i(t), \mathbf{q}_i(\lfloor t/h \rfloor))$ to determine an action based on its local observation, with the following loss function:

$$L(\omega_{l,i}) = E_{\mathbf{s}, \mathbf{a}, \mathbf{q}} (-Q_{\phi_{l,i}}(\mathbf{s}(t), \mathbf{a}(t), \mathbf{q}(\lfloor t/h \rfloor))). \quad (23)$$

The subscript $\omega_l = (\omega_{l,1}, \dots, \omega_{l,N})$ represents the weight parameters of the policy network in the lower layer.

To ensure stable policy convergence under the strong coupling between the two layers, we introduce a cross-training method. The training alternates between updating the lower-layer policies every h steps and subsequently refining the upper-layer policy using the accumulated feedback. Details are outlined in Algorithm 1. This method updates the upper and lower networks simultaneously, enabling both layers to adjust their policies in real time.

IV. EXPERIMENTS

In this section, extensive experiments are conducted to evaluate the performance of our HRL method in solving the strategic confrontation problem at different sizes. In comparative experiments, we adopt various baselines, including the expert system, heuristic algorithm, and single-layer DRL. We assess the impact of the interaction mechanism through

ablation studies. Moreover, we test the model trained on a small scale to solve larger-scale problems, investigating the generalization of our method. Finally, we deploy the cross-trained model from simulations to the real-robot system.

A. Setting Up

Before analyzing the comparative results, we outline the detailed settings used in the simulations. Following the conventions in [8], [22], we consider an equal number of red and blue agents, randomly distributed in a 30 m \times 20 m scenario with static obstacles. In our experiment, three scenarios are considered: five agents versus five agents (denoted V5), seven agents versus seven agents (denoted V7), and nine agents versus nine agents (denoted V9). The attack radius ρ_1 , angle θ_1 , and probability ϵ are set to 1 m, $\pi/2$, and 0.5, respectively. The avoidance radius ρ_2 and the obstacle radius ρ_3 are set to 0.3 m. The observation length d_1 and width d_2 are set to 5 m. The maximum velocity magnitude $\|\mathbf{v}\|_{\max}$ is 2 m/s, and the step length δt is 0.1 s.

For algorithm training, we cross-train the two layers for $E_c = 60$ episodes and $T = 300$ steps in 100 randomly generated instances. The networks in both layers are implemented as three-layer MLPs with hidden units [128, 64, 32]. All networks use ReLU activations and Adam optimizer. The replay buffer size is 10^5 , and mini-batch size is 64. The discount factor γ and the learning rate α are set to 0.99 and 0.001, respectively. To plot the experimental curves, we use solid curves to represent the mean values across all instances, with shaded regions indicating the standard deviation. In our experiments, we apply the HRL method and baselines to the blue agents, while the red agents use traditional TAMP methods, including the expert system [8] and heuristic algorithm [11].

B. Comparative Analysis

In cross-training, we employ the HRL method to enhance the performance of both layers. We compare this method with baselines, including the expert system, heuristic approach, and single-layer DRL. Additionally, to investigate the influence of interactions between the two layers, we conduct an ablation study by removing the feedback reward in HRL. The baselines are briefly introduced as follows.

- 1) Expert system: Based on the confrontation rules constructed in [8], this algorithm uses a hierarchical decision-making framework to match optimal task allocation and path planning.
- 2) Heuristic algorithm: The algorithm in [11] designs an evolutionary-based attack strategy for swarms, evaluating the potential benefits or threats of the next perceptible attack positions of an agent. It also incorporates a collision-avoidance method for feasible path planning.
- 3) Single-layer DRL: The algorithm in [17] treats confrontation as an MDP process and employs single-layer DRL to output the agents' strategies end-to-end.
- 4) HRL/FB: This approach adopts our HRL method but updates the upper layer without the feedback reward transferred from the lower layer.

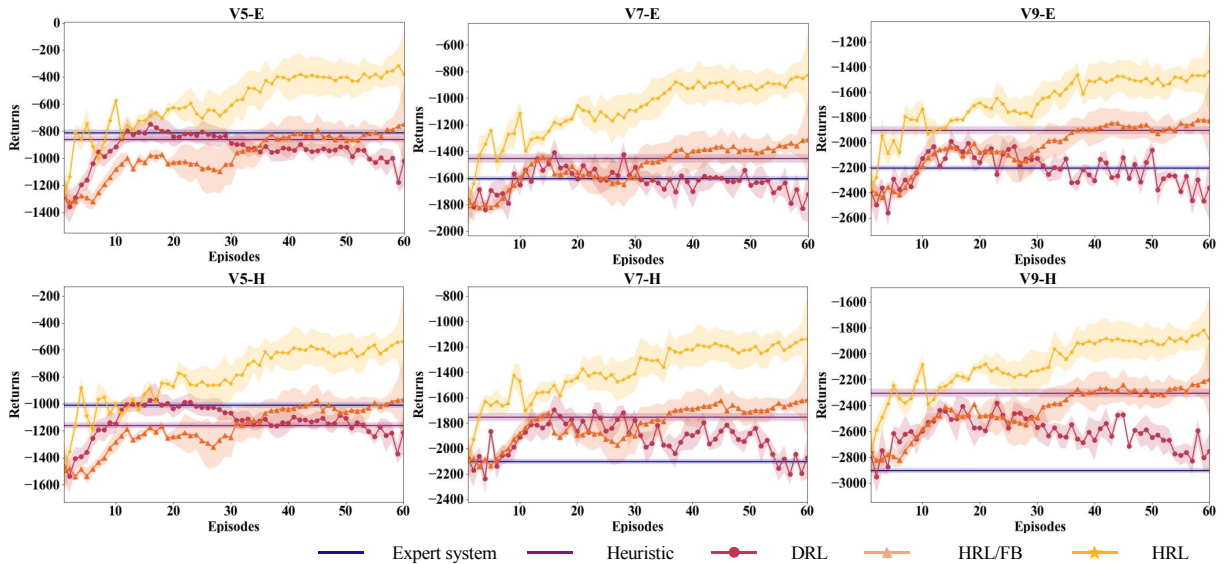


Fig. 4. Learning curves of baselines and our method for different swarm sizes.

TABLE I
EXPERIMENT RESULTS OF BASELINES AND OUR METHOD FOR DIFFERENT SWARM SIZES.

Algorithm of enemies	Method	V5			V7			V9		
		Re.	Ti.($\times 10^{-2}$ s)	W.R.(%)	Re.	Ti.($\times 10^{-2}$ s)	W.R.(%)	Re.	Ti.($\times 10^{-2}$ s)	W.R.(%)
Expert	Expert	-1009	3.21	51	-1602	3.41	50	-2202	3.70	51
	Heuristic	-1159	2.67	48	-1452	4.26	52	-1902	6.07	54
	DRL	-1214	0.26	46	-1723	0.32	47	-2362	0.45	50
	HRL/FB	-967	0.53	53	-1309	0.69	54	-1825	0.95	56
	HRL	-538		82	-827		84	-1433		85
Heuristic	Expert	-809	3.22	52	-2102	3.40	48	-2902	3.73	46
	Heuristic	-859	2.66	49	-1752	4.28	51	-2302	6.06	50
	DRL	-1018	0.27	46	-2072	0.30	49	-2754	0.47	47
	HRL/FB	-747	0.52	59	-1617	0.71	54	-2196	0.99	52
	HRL	-381		84	-1138		82	-1881		81

To ensure a fair comparison between our method and the baselines, we evaluate all methods under the same experimental conditions, using identical input data and performance metrics. The HRL learning curves and the baselines for different swarm sizes are shown in Fig. 4, where "VX-E" and "VX-H" represent the use of the expert system and heuristic algorithm as enemies' strategies in the VX scenario, respectively. Both the expert system and heuristic algorithm can develop effective strategies for agents across different instances. Between these two conventional TAMP algorithms, the heuristic algorithm outperforms the expert system in large-scale confrontations.

Initially, learning-based algorithms underperform due to the randomness of their starting strategies. With continuous exploration and training, their performance gradually surpasses that of the non-learning algorithms. Due to the hybrid decision-making process in the confrontation, the performance of single-layer DRL deteriorates over time during training. In contrast, HRL consistently finds better strategies than non-learning algorithms across various swarm sizes. Furthermore, when the feedback reward is omitted, the performance curve converges to a suboptimal value due to the lack of effective interaction between the two layers.

In addition, we deploy the policy networks trained using learning-based methods across another 100 instances and compare their performance with conventional TAMP algorithms. To provide a more comprehensive assessment, we introduce two additional evaluation metrics: decision time and confrontation win rate. Decision time refers to the average time that all agents take to allocate tasks and plan the path during each time step. Confrontation win rate measures the ratio of successful outcomes to the total number of instances. The average experimental results across 100 instances for different swarm sizes are summarized in Table I, which presents the episode returns (Re.), decision time (Ti.), and confrontation win rate (W.R.) for all methods.

Notably, all methods support real-time decision-making at each step, with decision time below the time interval δt . We observe a clear relationship between the win rate and episode returns, which are calculated using the reward function in this study. Agents train their strategies to maximize returns, which in turn leads to a higher win rate.

Between the two non-learning algorithms, the heuristic algorithm has a longer decision time, though its returns and win rate improve as the problem size increases. Due to its end-to-end nature, DRL exhibits the shortest decision time,

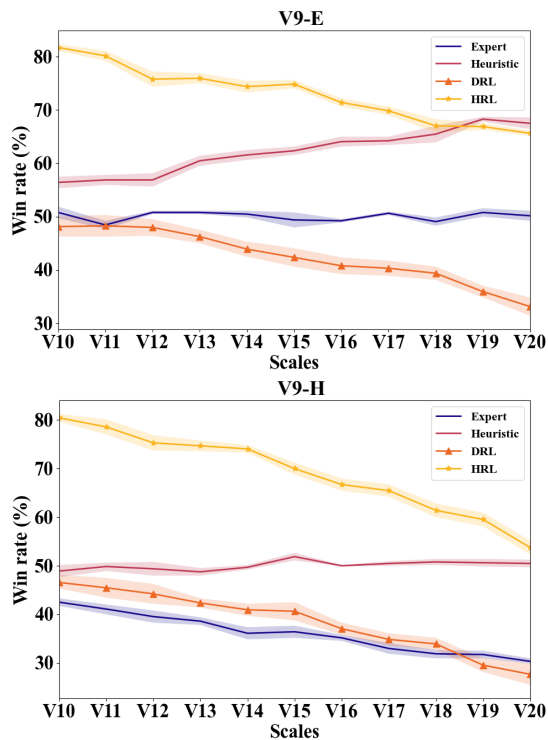


Fig. 5. Generalization to larger-scale instances.

but its win rate is significantly lower than that of HRL. The results demonstrate that our method outperforms the baselines in terms of episode returns, decision time, and confrontation win rate, especially in large-scale instances. This is attributed to the effectiveness of our approach in managing interactions between the upper and lower layers by decoupling confrontation into allocation and planning.

C. Generalization to Larger-Size Instances

In large-scale confrontation scenarios, retraining strategies can lead to significant consumption of computational resources. Therefore, our goal is for the trained models to exhibit strong generalization performance. In our method, the decentralized policies of agents can be applied directly to additional agents without the need for retraining. To assess the generalization capability of our approach, we apply a set of neural network parameters, trained on the V9 scenario, to instances involving a higher number of agents (ranging from 10 to 20). As the scenario size increases, agents face greater challenges and experience lower win rates. However, our method continues to maintain favorable solution quality compared to the baselines.

The generalization results are shown in Fig. 5, which presents the win rate across different instance scales. Due to the limited generalization capabilities of traditional TAMP algorithms, they require repeated adjustments for larger-scale scenarios. As shown, our method outperforms expert system, heuristic algorithm, and single-layer DRL in most instances. Our approach demonstrates high scalability in larger scenarios compared to the baselines. In particular, the model trained on the V9 scenario can generalize to V20,

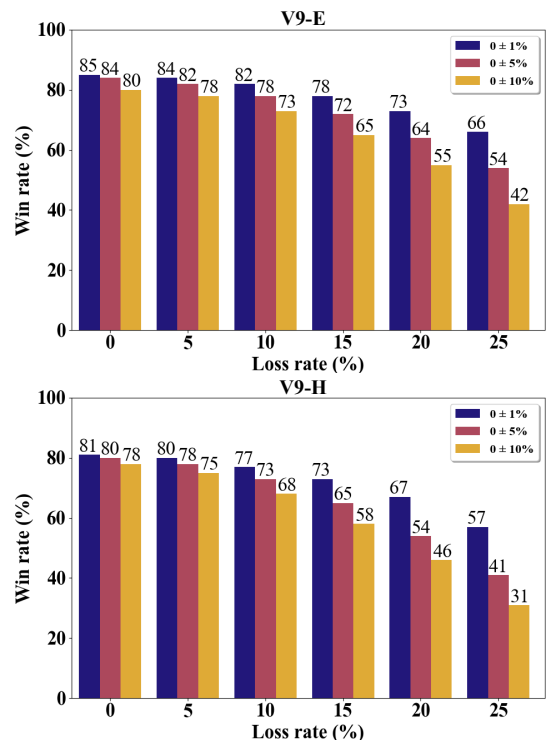


Fig. 6. Assessment in instances involving communication and sensor uncertainties.

achieving win rates consistent with those achieved by the heuristic algorithm reiterated on V20. These results verify that policies trained on small-scale instances can effectively make decisions in scenarios of varying scales. Thus, our method exhibits favorable generalization capability for the strategic confrontation problem.

D. Deployment in Real-Robot System

Before presenting the real-robot experiments, we evaluate the performance of our method through simulations that account for communication and sensor uncertainties, including message loss between neighbors and sensor noise error. The results are shown in Fig. 6, where the horizontal axis represents the message loss rate, the vertical axis indicates the win rate, and the legend represents different levels of Gaussian noise error. Although the success rate decreases as the uncertainty increases, it remains above 73% when the loss rate is below 10% and the noise error is within $\pm 5\%$. These results demonstrate that our method is suitable for real-world applications.

We then conduct experiments using an actual robotic system, as depicted in Fig. 7. A V3 confrontation is considered in a static-obstacle scenario, with experimental settings consistent with those defined in Section II-A. By performing cross-training in simulations, we obtain two-layer networks for the agents in this scenario. During the experiment, we control the agents via a ground control system, with motion capture provided by Optitrack. Our algorithm receives data from Optitrack that includes the positions and velocities of all agents. The algorithm subsequently calculates observations

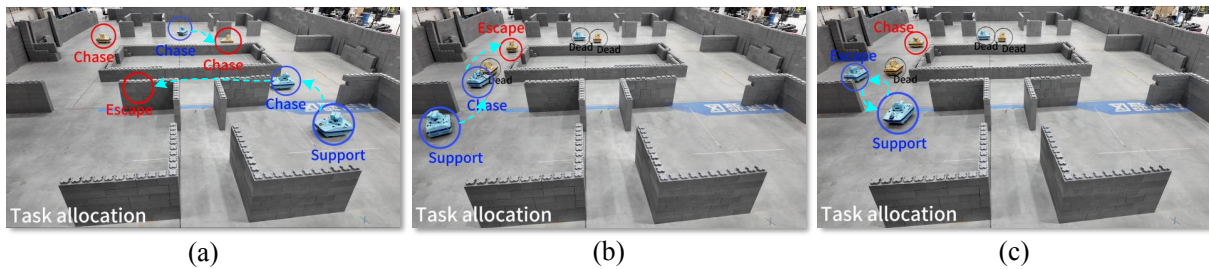


Fig. 7. The decision-making process for V3 scenario in the real-robot system.

from perception sensors and generates position commands based on the two-layer networks, which are sent to the ground control system. Using these networks, the agents successfully defeat all enemies during the experiments while avoiding collisions. After sufficient training with our method, the agents exhibit coordinated confrontation behaviors, such as prioritizing support for teammates in critical situations (Fig. 7(a)), using static obstacles for cover (Fig. 7(b)), and luring opponents into teammates' encirclement (Fig. 7(c)).

V. CONCLUSIONS

This paper has presented a bidirectional HRL approach to address the hybrid decision-making process in strategic confrontations. It has developed two-layer DRL networks that associate commands and actions with task allocation and path planning, respectively. Furthermore, the method has refined the interaction mechanism and introduced a cross-training method to enhance learning between the two layers. We also have incorporated a trajectory prediction model to convert abstract command representations into concrete action objectives. Experimental results have demonstrated that our method outperforms the baselines in terms of episode returns, decision time, and confrontation win rate. Finally, the generalization and adaptability of our method have been validated by applying the trained model to larger instances and real-robot systems, respectively.

REFERENCES

- [1] X. Guo, W. Tang, K. Qin, Y. Zhong, H. Xu, Y. Qu, Z. Li, and Q. Sheng, "Powerful uav manipulation via bioinspired self-adaptive soft self-contained gripper," *Sci. Adv.*, vol. 10, no. 19, p. eadn6642, 2024.
- [2] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smart-llm: Smart multi-agent robot task planning using large language models," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2024, pp. 12 140–12 147.
- [3] S. Fan, H. Piao, Y. Hu, F. Jiang, and R. Yang, "Deep ad-hoc sub-team partition learning for multi-agent air combat cooperation," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2024, pp. 3403–3408.
- [4] H. Piao, Y. Han, S. He, C. Yu, S. Fan, Y. Hou, C. Bai, and L. Mo, "Spatio-temporal relationship cognitive learning for multi-robot air combat," *IEEE Trans. Cognit. Dev. Syst.*, vol. 15, no. 4, pp. 2254–2268, 2023.
- [5] Q. Wu, K. Liu, L. Chen, and J. Lü, "Hierarchical reinforcement learning for swarm confrontation with high uncertainty," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 8630–8644, 2025.
- [6] D. Liu, Q. Zong, X. Zhang, R. Zhang, L. Dou, and B. Tian, "Game of drones: Intelligent online decision making of multi-uav confrontation," *IEEE Trans. Emerging Top. Comput. Intell.*, vol. 8, no. 2, pp. 2086–2100, 2024.
- [7] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [8] Y. Hou, X. Liang, J. Zhang, M. Lv, and A. Yang, "Hierarchical decision-making framework for multiple ucavs autonomous confrontation," *IEEE Trans. Veh. Technol.*, vol. 72, no. 11, pp. 13 953–13 968, 2023.
- [9] T. S. Vaquero, G. Daddi, R. Thakker, M. Paton, A. Jasour, M. P. Strub, R. M. Swan, R. Royce, M. Gildner, P. Tosi *et al.*, "Eels: Autonomous snake-like robot with task and motion planning capabilities for ice world exploration," *Sci. Rob.*, vol. 9, no. 88, p. eadh8332, 2024.
- [10] L. Wang, T. Qiu, Z. Pu, J. Yi, J. Zhu, and Y. Zhao, "A decision-making method for swarm agents in attack-defense confrontation," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 7858–7864, 2023.
- [11] H. Liu, J. Zhang, P. Zu, and M. Zhou, "Evolutionary algorithm-based attack strategy with swarm robots in denied environments," *IEEE Trans. Evol. Comput.*, vol. 27, no. 6, pp. 1562–1574, 2023.
- [12] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "Rhh-igp: Receding horizon and heuristics-based logic-geometric programming for task and motion planning," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2022, pp. 13 761–13 768.
- [13] T. Xue, A. Razmjoo, and S. Calinon, "D-igp: Dynamic logic-geometric program for reactive task and motion planning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2024, pp. 14 888–14 894.
- [14] T. Ren, G. Chalvatzaki, and J. Peters, "Extended tree search for robot task and motion planning," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2024, pp. 12 048–12 055.
- [15] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.
- [16] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humpalik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Sci. Rob.*, vol. 9, no. 89, p. eadi8022, 2024.
- [17] X. Qu, W. Gan, D. Song, and L. Zhou, "Pursuit-evasion game strategy of usv based on deep reinforcement learning in complex multi-obstacle environment," *Ocean Eng.*, vol. 273, p. 114016, 2023.
- [18] M. Eppe, C. Gumbsch, M. Kerzel, P. D. Nguyen, M. V. Butz, and S. Wermter, "Intelligent problem-solving as integrated hierarchical reinforcement learning," *Nat. Mach. Intell.*, vol. 4, no. 1, pp. 11–20, 2022.
- [19] M. Dawood, N. Dengler, J. de Heuvel, and M. Bennewitz, "Handling sparse rewards in reinforcement learning using model predictive control," in *Proc. Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2023, pp. 879–885.
- [20] X. Huang, Q. Liao, Y. Ni, Z. Li, L. Smith, S. Levine, X. B. Peng, and K. Sreenath, "Hilma-res: A general hierarchical framework via residual rl for combining quadrupedal locomotion and manipulation," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2024, pp. 9050–9057.
- [21] W. Kong, D. Zhou, Y. Du, Y. Zhou, and Y. Zhao, "Hierarchical multi-agent reinforcement learning for multi-aircraft close-range air combat," *IET Control Theory & Appl.*, vol. 17, no. 13, pp. 1840–1862, 2023.
- [22] X. Nian, M. Li, H. Wang, Y. Gong, and H. Xiong, "Large-scale uav swarm confrontation based on hierarchical attention actor-critic algorithm," *Appl. Intell.*, vol. 54, pp. 3279–3294, 2024.
- [23] Z. Pang, C. Wang, L. Zhao, Y. Liu, and G. Sharma, "Cross-modality hierarchical clustering and refinement for unsupervised visible-infrared person re-identification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 4, pp. 2706–2718, 2024.