applied sciences

MDPI

Article

Target-Oriented Multi-Agent Coordination with Hierarchical Reinforcement Learning

Yuekang Yu [1], Zhongyi Zhai [2,*], Weikun Li [2] and Jianyu Ma [1]

[1] School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China; yuyuekang@mails.guet.edu.cn (Y.Y.); 17771062323@163.com (J.M.)
[2] School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China; liweikun1105@outlook.com
* Correspondence: zhaizhongyi@guet.edu.cn; Tel.: +86-187-0773-5801

Abstract: In target-oriented multi-agent tasks, agents collaboratively achieve goals defined by specific objects, or targets, in their environment. The key to success is the effective coordination between agents and these targets, especially in dynamic environments where targets may shift. Agents must adeptly adjust to these changes and re-evaluate their target interactions. Inefficient coordination can lead to resource waste, extended task times, and lower overall performance. Addressing this challenge, we introduce the regulatory hierarchical multi-agent coordination (RHMC), a hierarchical reinforcement learning approach. RHMC divides the coordination task into two levels: a high-level policy, assigning targets based on environmental state, and a low-level policy, executing basic actions guided by individual target assignments and observations. Stabilizing RHMC's high-level policy is crucial for effective learning. This stability is achieved by reward regularization, reducing reliance on the dynamic low-level policy. Such regularization ensures the high-level policy remains focused on broad coordination, not overly dependent on specific agent actions. By minimizing low-level policy dependence, RHMC adapts more seamlessly to environmental changes, boosting learning efficiency. Testing demonstrates RHMC's superiority over existing methods in global reward and learning efficiency, highlighting its effectiveness in multi-agent coordination.

Keywords: multi-agent; target-oriented; hierarchical reinforcement learning; coordination

Citation: Yu, Y.; Zhai, Z.; Li, W.; Ma, J. Target-Oriented Multi-Agent Coordination with Hierarchical Reinforcement Learning. *Appl. Sci.* **2024**, *14*, 7084. https://doi.org/10.3390/app14167084

Academic Editor: André Sales Mendes

Received: 5 June 2024
Revised: 14 July 2024
Accepted: 17 July 2024
Published: 12 August 2024

1. Introduction

Coordination is crucial in real-world scenarios, as it permits individuals to collaborate to achieve shared goals. In certain tasks, such as collaborative drone control [1], coordination among autonomous vehicles [2], and the target coverage problem in directed sensor networks [3], better team performance can be achieved through coordination [4]. These issues can be modeled as cooperation between multi-agents to accomplish their own tasks while maximizing the global reward. Reinforcement learning is an effective approach for sequential decision-making, which improves its policy through continuous trial-and-error exploration and pursuit of reward maximization [5–7]. Additionally, reinforcement learning has shown tremendous improvements in various domains, including gaming [8], robot control [9,10], intelligent transportation [11,12], and industrial applications [13]. Despite the successful extension of single-agent algorithms such as DQN [10], DDPG [6], and PPO [7] to multi-agent systems, multi-agent coordination continues to face challenges [14,15].

Target-oriented multi-agent environments refer to task environments where agents collaborate in the presence of specific entities. A target can be identified as an entity in the environment, such as an object or a location that must be involved in completing a task. The agent needs to establish a correspondence with the target entity and use their personal skills to accomplish the task as efficiently as possible. Examples of target-oriented scenarios include warehouse robots managing cargo [14], drones tracking objects [15], and resource scheduling systems where server resources need to be allocated efficiently [16]. In the latter

case, coordination involves managing and optimizing server resources to handle various tasks or requests in a balanced and efficient manner [17,18]. Coordinating the relationships between agents and targets is crucial for improving the overall efficiency of collaborative tasks [19,20].

For one-to-one agent-target tasks, the IQL method treats the remaining agents directly as part of the environment, leading to instability as the policies of other agents change over time [17,21]. Since there is no guarantee of convergence, it is easy for the agent to become stuck in an endless exploration, limiting IQL to simple tasks such as the Pong game [22]. However, in cooperative tasks with multiple targets, such as cooperative navigation, where agents must reach multiple target locations, it is critical to coordinate the relationship between agents and targets to avoid internal conflicts [23,24]. MADDPG learns multiple policies for each agent and optimizes the overall effect of all policies to improve the stability and robustness of the algorithm [18]. COMA utilizes a global critic network to assess all current actions and states, enhancing the efficiency of information exchange and collaboration between agents during training [19]. QMIX employs a hybrid network to combine the single agent's local value function with global state information, facilitating training and learning procedures, and enhancing the algorithm's performance [20]. In multi-agent coordination, maximizing local rewards does not ensure global optimality. In contrast, prioritizing the global reward can cause entanglement between local and global rewards, leading to suboptimal solutions [21]. Balancing the relationship between the local and the whole is very important for multi-agent systems to accomplish the task better [22,24].

The regulatory hierarchical multi-agent coordination (RHMC) framework is proposed to tackle this problem. Our contribution is threefold: (1) A hierarchical reinforcement learning algorithm is proposed for the multi-agent coordination problem. This framework comprises two layers; the high-level policy governs target allocation, while the low-level policy guides agent actions. (2) RHMC implements a reward regularization mechanism to reduce the high-level policy's dependence on modifications in low-level behavior, providing stability to the policy training process. (3) The RHMC method is validated on target-oriented multi-agent cooperative tasks, and the experimental results show that our method outperforms the state-of-the-art algorithms.

2. Related Work

The target-oriented multi-agent coordination framework includes two categories of multi-agent reinforcement learning: local reward-based coordination and global reward-based coordination. Local reward-based approaches aim to optimize overall performance by maximizing the individual agent's local rewards. Early approaches, such as the IQL method [17], convert multi-agent-related tasks into single-agent tasks, with each agent learning its own value function separately. This method is not effective because all agents affect the overall environment, and the agents are also affected by other agents during the learning process, leading to unstable agent learning. The Centralized Training with Decentralized Execution (CTDE) framework is employed in multi-agent learning work such as intelligent transportation [21], smart home [22], recommendation algorithms [23–25], common-pool resource appropriation [26–28], and sequential social dilemmas [29,30]. This approach addresses the issue of non-stationarity in the environment that can arise when agents learn independently without taking into account the actions of other agents.

The MADDPG method [18] can be regarded as a deterministic policy gradient method that uses deep reinforcement learning to solve the continuous action space problem. Although MADDPG enhances stability and robustness, it still faces challenges in diverse environments. The M3DDPG method [31] enhances the generalization capability of the learned policies and accommodates the heterogeneity of reward functions across agents, but it does not explicitly address the coordination requirements in multi-agent systems. The MAPPO method [32] is a multi-agent reinforcement learning method based on the policy gradient algorithm of confidence region, which limits the size of the parameter

update step to stabilize the training process. The NC-HDQN method [33] analyzes the correlation between agents and their connected neighbors to measure the observed value and reward and uses a hysteretic DQN-weighted information optimization strategy. The LOLA method [34] enables agents to better understand the policies and motivations of other agents through opponent modeling, thus coordinating their actions more effectively.

Global reward-based approaches tend to have agents with one goal and a shared global reward. These approaches aim at maximizing the global reward while optimizing the local reward. The COMA method [19] uses a centralized critic network to collect information from all agents during training and employs counterfactual baselines to address the credit assignment problem. Value decomposition methods represent the joint action value function as a combination of the value functions of each agent, guiding agents to learn a better policy. The VDN method [35] uses a deep neural network to represent the value function of each agent and the whole joint action value function as a linear combination of these value functions. However, VDN overlooks the role of global state information in the learning process and the intricate nonlinear relationships between agents. To address these issues, the QMIX method [20] uses an additional network to fuse the values of each agent, including global information as auxiliary information when expressing the joint action value function. QMIX obligates the joint action value function to be monotonic with the value function of each agent to comply with the maximum consistency principle between corresponding value functions. The PMI-Q method [36] improves the generalization ability of the policy by preventing overfitting through a combination of pessimistic averaging and average invariance. Drawing inspiration from the collective behavior of ant colonies, the SIRL method [37] introduces the concept of pheromones into multi-agent systems to reduce local conflicts.

Hierarchical reinforcement learning (HRL) benefits structured exploration by learning sub-task policies over primitive actions. Exploration efficiency is essential in reinforcement learning. Although the majority of HRL methods use single-agent exploration methods at a low level, such as greedy [38] and intrinsic reward [39], some methods focus on high-level policy, such as diversity concepts [40] and coordinator cooperation, with well-performed executors [41]. The former requires a diversified set of low-level skills, while the latter requires a well-defined pre-training setting [42]. Despite these advancements, existing HRL methods often fail to address the complex coordination required in multi-agent settings [43–48]. Our work introduces a distinctive hierarchical structure that meticulously addresses both local and global coordination needs in multi-agent systems, effectively filling a significant gap in current methodologies and considerably enhancing task performance [49]. To underscore the rationale for our choice of comparative analysis, the following table, designated as Table 1, presents a detailed comparison of selected multi-agent reinforcement learning methods. These methods, including MADDPG, CTDE, and QMIX, are not only ubiquitous in the realm of multi-agent research but also represent pivotal advancements in addressing fundamental challenges such as coordination complexity, environmental non-stationarity, and policy scalability. Their widespread use and seminal contributions to the field substantiate their selection for our comparative analysis. This strategic choice allows us to vividly highlight how our RHMC framework leverages and transcends the foundational strengths of these established methods, thereby providing a profound insight into the significant enhancements and unique positioning of our approach. For a detailed comparison, see Table 1.

Table 1. Comparative analysis of common multi-agent reinforcement learning methods.

Method Name	Key Features	Addressed Issues	Application Scenarios	RHMC Advantages
MADDPG	Deep RL method with deterministic gradients; focuses on stability via centralized training and decentralized execution.	Enhances stability in dynamic, continuous action spaces.	Suitable for robotic collaboration and autonomous fleets.	Boosts robustness in continuous action environments.
CTDE	Combines centralized training with decentralized execution to tackle environmental non-stationarity.	Improves stability by addressing learning issues from independent agent actions.	Ideal for cooperative multi-agent systems like intelligent traffic.	Resolves non-stationarity in multi-agent learning effectively.
QMIX	Integrates agent value functions using a mixing network for consistent policy updates; includes global information.	Ensures consistent policies and cooperation, managing update conflicts.	Used in complex scenarios requiring consistent policies, such as multiplayer games.	Provides a novel method for enhancing policy consistency and cooperation.
RHMC	Introduces a clear hierarchical structure, explicitly handling both local and global coordination needs. Focuses on structured exploration and policy optimization.	Enhances coordination between local and global tasks in multi-agent systems, optimizing overall task performance.	Applicable to multi-agent environments requiring complex coordination and efficient execution.	

3. Preliminaries

3.1. Reinforcement Learning

A reinforcement learning problem can be modeled by a Markov decision process [48] (MDP), defined by the tuple $\langle S, A, P, r, \gamma \rangle$, in which

- $S \subseteq R$ is a n-dimensional state space;
- $A \subseteq R$ is a m-dimensional action space;
- $P : S \times A \times S \to R_+$ is a transition probability function;
- $r : S \to R$ is reward function;
- $\gamma \in (0, 1]$ is the discount f.

In Markov decision process, the agent performs an action $a_t \in A$ according to the current state $s_t \in S$ and its policy $\pi(a_t|s_t)$ at each timestep. The objective of the agent is to learn an optimal policy $\pi_{\theta^*} := argmax_{\pi\theta}\left[\sum_{i=0}^{T} \gamma^i r_{t+i}|s_t = s\right]$.

Hierarchical reinforcement learning uses a multi-layer policy to interact with the environment. For the target-oriented multi-agent coordination problem, the joint policy π_j oint consists of a high-level policy π_h and a low-level policy π_l. The high-level policy distributes targets g_t to coordinate the individual low-level agents. Then the low-level policy agents output the primitive action according to the received target assignment g_t and current state s_t. Differently, the high-level policy is used to coordinate agents to maximize global rewards, while the low-level policy is used to maximize individual local rewards under the high-level policy target assignment.

3.2. Actor–Critic Method

The actor–critic method is a reinforcement learning method combining a policy gradient and a value function. There are two main methods of reinforcement learning: the value-based method and the policy-based method [49]. The policy gradient method can easily select appropriate actions in continuous action space, while value-based Q-learning can only solve problems in discrete action space. A combination of the policy-gradient and the value-based method will form the actor–critic algorithm. The predecessor of the actor is the policy gradient, and the predecessor of the critic is the value-based method. The actor

selects behavior based on probability distribution, the critic evaluates a score based on the behavior generated by the actor, and the actor modifies the probability of selection behavior according to the score of the critic [44]. Directly optimizing a policy, policy gradient (PG) algorithms update policy parameters θ in the direction of the gradient of the expected return $\nabla_\theta E[G_{\pi\theta}]$, which is conveniently given by the policy gradient theorem as follows:

$$\nabla_\theta E[G_{\pi\theta}] = E_{h,a}\left[\nabla_\theta \log \pi_\theta(a|h) Q^{\pi\theta}(h,a)\right] \tag{1}$$

Actors use policy functions that generate actions and interact with the environment. Critic, on the other hand, uses a value function and is responsible for evaluating the performance of the actor and guiding the action of the actor in the next stage [45].

3.3. Multi-Agent Deep Deterministic Policy Gradient

Multi-agent deep deterministic policy gradient (MADDPG) is an adaptation of the deep deterministic policy gradient algorithm to multi-agent settings. It is an actor–critic, off-policy method that uses the paradigm of CTDE to learn deterministic policies. In MADDPG, a separate actor and critic is learned for each agent, such that arbitrary reward functions can be learned. Each agent, i, has a deterministic policy, $\pi_i(a_i||o_i)$, parameterized by θ_i (abbreviated as π_i), and let $\pi = \{\pi_i(a_i||o_i)\}_{i=1}^n$ be the set of all agent policies. A centralized and monolithic critic that estimates the joint action–value function $Q_i^\pi(s, a_1 \ldots, a_n)$ is learned for each agent separately. The critic is said to be centralized as it utilizes information only available to it during the centralized training phase, the global state and the actions of all agents, $a_1, \ldots, a_n$, to estimate the joint action–value function Q_i^μ, which is parameterized by ϕ_i. This joint action–value function is trained by minimizing the following loss:

$$L(\phi_i) = \mathbb{E}\left[(y_i - Q_i^\pi(s, a_1, \ldots, a_n))^2\right] \tag{2}$$

where,

$$y_i = r_i + \gamma Q_i^\pi\left(s', a_1', \ldots, a_n'\right)\Big|\Big|_{a_i' = \pi_i(o_i)} \tag{3}$$

Here r_i is the reward received by each agent i, $a_1, \ldots, a_n$ is the set of target policies with delayed parameters θ_i^-, and ϕ_i^- are the parameters of the target critic. The replay buffer $\mathcal{D}$ contains the transition tuples $(s, s\prime, a_1 \ldots, a_n, r_1 \ldots, r_n)$.

The following policy gradient can be calculated individually to update the policy of each agent, i:

$$\nabla_{\theta_i} J(\pi_i) = \mathbb{E}_\mathcal{D}\left[\nabla_{\theta_i} \pi_i(o_i) \nabla_{a_i} Q_i^\pi(s, a_1, \ldots, a_n) \Big\|_{a_i = \pi_i(o_i)}\right] \tag{4}$$

where the current agent i's action a_i is sampled from its current policy π_i when evaluating the joint action–value function Q_i^π, while all other agents' actions are sampled from the replay buffer $\mathcal{D}$.

Updating the target network parameters of each agent i is shown in the following formula:

$$\theta_\prime^i \leftarrow \tau\theta_\prime^i + (1-\tau)\theta_\prime^i \tag{5}$$

4. Method

The regulatory hierarchical multi-agent coordination (RHMC) method employs a two-level hierarchical approach that aims to maximize global rewards for all agents. At the high-level, the policy assigns targets based on the overall state of the environment. At the low-level, agents select primitive actions based on the assigned targets and their local observations.

The Regulatory hierarchical multi-agent coordination (RHMC) architecture is showed in Figure 1.

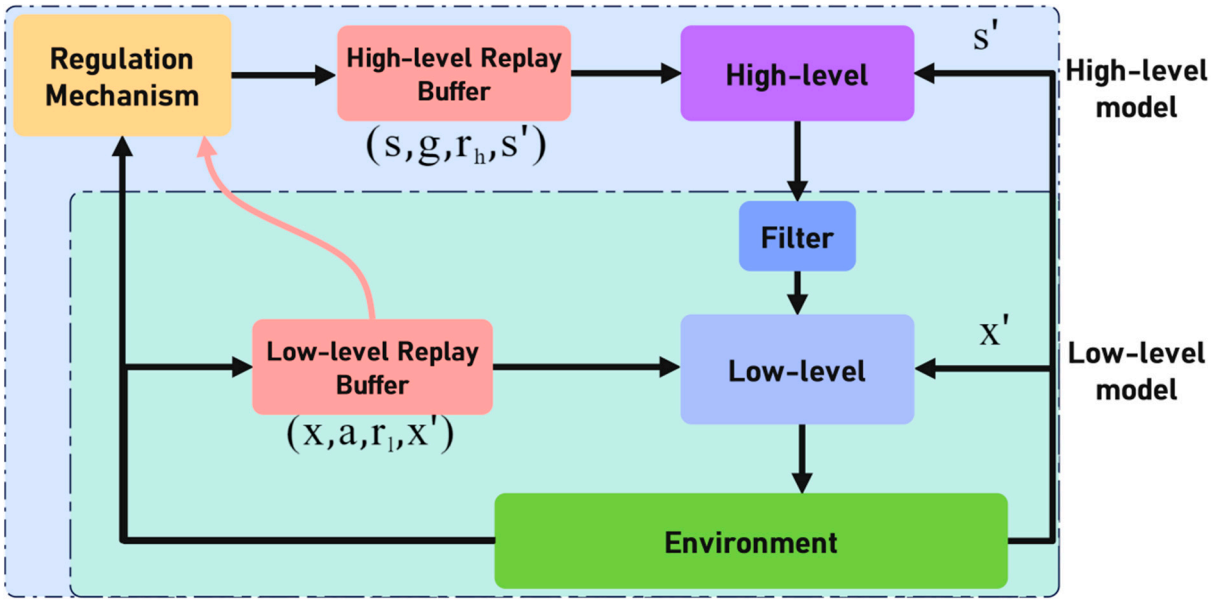

Figure 1. Regulatory hierarchical multi-agent coordination (RHMC) architecture.

4.1. Mechanism Description

The regulatory hierarchical multi-agent coordination (RHMC) architecture is a two-level framework designed to enhance multi-agent coordination through hierarchical policies. At the high level, the policy collects joint state information from all agents and distributes target assignments to the low-level policy. The low-level policy then generates primitive actions based on both the assigned targets and the agents' observations, directly applying these actions to the environment. This hierarchical approach allows for more efficient and organized coordination among agents. The high-level actors focus on strategic target distribution, while the low-level actors handle tactical execution. Additionally, the high-level critic utilizes a regulation mechanism (RM) to calculate the value, ensuring robust and stable learning across both policy levels.

RHMC has two Markov decision processes (MDPs) for the high-level policy and the low-level policy, respectively:

$$MDP_h = \langle S_h, G, P_h, r_h, \gamma_h \rangle$$

$$MDP_i = \langle S_l, A_l, P_l, r_l, \gamma_l \rangle$$

(1) High-Level Policy (Symbol 1): The high-level policy is responsible for target allocation. It considers the overall state of the environment and assigns targets to agents. This level focuses on global coordination and strategic planning, ensuring that each agent is directed towards a target that optimizes the overall system performance.

(2) Low-Level Policy (Symbol 2): The low-level policy deals with the execution of actions. Based on the assigned targets and local observations, agents choose primitive actions. This level emphasizes local decision-making and immediate interactions with the environment.

The RHMC algorithm increases sample utilization efficiency via hierarchical policy training. However, the high-level policies in the hierarchical structure can be influenced by the low-level policies. When policies of all levels are trained simultaneously, the low-level policies are continually updated, leading to changes in the high-level transfer function. In such a non-stationary environment, it becomes challenging for high-level policy to learn the optimal policy.

While the low-level policies $\mu_\theta^l(s, g)$ interact directly with the environment, the high-level policy interacts indirectly. The non-stationarity emerges in the high-level policy $\mu_\theta^h(s)$

transition. The low-level policy makes primitive actions directly on the environment, and the same state transition function remains unchanged. In contrast, the high-level policy acts indirectly on the environment. The same action output by the high-level policy can produce different actions at the low-level during training, leading to dynamic changes in state transitions at the high-level. This uncertainty in transitions $(s, g) \rightarrow s'$ leads large changes in the high-level reward r_h.

To address these challenges, we propose a reward regulation mechanism and a target-conditioned filter:

(1) Both Regulation Mechanism: This mechanism aims to stabilize the training process by reducing the high-level policy's dependence on modifications in low-level behavior. By implementing a regularization term in the reward function, we ensure that the high-level policy remains robust despite the dynamic changes at the low-level. This helps maintain consistent learning and prevents the high-level policy from being adversely affected by the non-stationarity of the low-level policies.

(2) Target-Conditioned Filter: The target-conditioned filter helps in refining the action selection process by incorporating target-specific information into the decision-making framework. This filter ensures that the actions chosen by the low-level policy are more aligned with the high-level targets, thus improving coordination and efficiency. By conditioning on the target, the filter reduces the variability in low-level actions, leading to more predictable and stable transitions, which in turn enhances the performance of the high-level policy.

These components collectively improve the overall efficiency and robustness of the RHMC framework, allowing for better coordination and performance in multi-agent environments.

4.2. High-Level Target Assignment

The goal of the high-level policy $\mu_\beta(s)$ is to learn an optimal policy to maximize the global reward. The high-level policy determines the target assignment for each agent. It is implemented through the construction of a deep neural network, which consists of two components: actor and critic. According to the joint state $s = (s_1, s_2, \ldots, s_n, \ldots, s_{n+m})$ where m is the number of target and n is the low-level agent, the actor outputs the target assignment $g = (g_1, g_2, \ldots, g_n)$. An actor receives the joint state as an input and calculates the probability g_{ij} of each assignment through a fully connected layer where g_{ij} is a binary value indicating whether or not to let agent i track target j. Therefore, the actor will output the target assignment of all agents $g = (g_1, g_2, \ldots, g_n)$, where $g_i = (g_{i1}, g_{i2}, \ldots, g_{im})$ denotes the target assignment of agent i.

The high-level reward function is shown as follows:

$$r_h = \sum_{j=1}^{m} \left(\frac{p_j}{k_j} - \alpha_1 q_i \right) \Bigg|_{\frac{p_j}{k_j}=0 \ when \ k_j=0} \tag{6}$$

where m is the number of targets, p_j represents whether the assigned target task i is completed $(0 : No, 1 : Yes)$, $k_j = \sum_{i=1}^{n} g_{ij}$ represents the number of agents tracked to the assigned target j, q_j represents whether the assigned target i is not completed $(0 : No, 1 : Yes)$, and α_1 is a punishment factor that ranges from 0 to 1. p and q motivate the agents to complete more assigned target tasks, the larger the number of k the smaller the reward, avoiding targets conflict in agent coordination.

The critic network calculates loss function and updates actor network parameters. The critic calculates the optimal value of the state through the neural network, and the actor uses the optimal value to update the parameters of the policy function, then selects the

action and receives feedback and the next state. The loss function of the critic is calculated as follows:

$$L(\phi) = \underset{(s,g,r_h,s')\in D_h}{E} \left\| Q_\phi(s,g) - r_h + \gamma_h Q_{\phi target}\left(s', \mu_{\beta target}(s')\right) \right\|^2 \quad (7)$$

where the D_h is a subset of the replay buffer B_h at a high level. Since the Q network and policy function are constantly changing in the process of training, which will lead to the instability of the environment, we need to introduce the objective function. The β_{target} and ϕ_{target} are the parameter the target policy network and the Q network, respectively, which change slowly.

$$\beta_{target} = \rho\beta_{target} + (1-\rho)\beta\phi_{target} = \rho\phi_{target} + (1-\rho)\phi \quad (8)$$

where ρ is a number close to one. Therefore, the target network will update very slowly. By using the policy gradient theorem, the policy network update is as follows:

$$\beta = \beta + \alpha_2 \underset{s\in D_h}{E} \nabla\beta\mu\beta(s)\nabla_g Q_\phi(s,g) \quad (9)$$

The model of the high-level structure is showed in Figure 2.

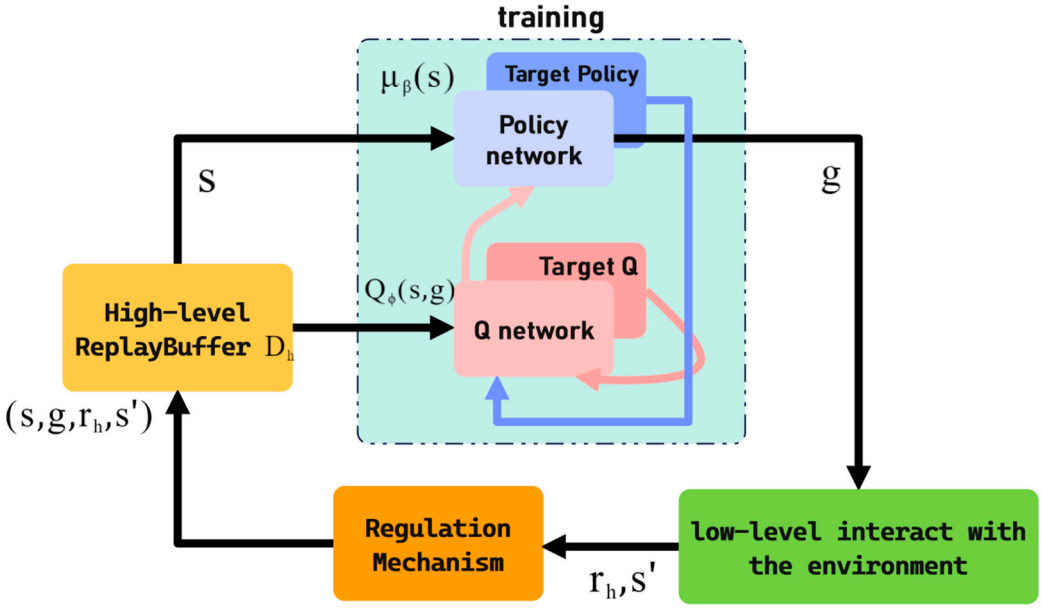

Figure 2. The frame of high-level.

4.3. Low-Level Primitive Action

Upon receiving the target assignment from the high-level policy, each agent performs the task independently based on the assigned target condition. The aim of the low-level policy is to maximize the local reward under the given target assignment. Therefore, while the high-level policy handles coordination, the low-level policy executes the assigned target task. The input to the low-level policy $\mu_{\theta^i}(o,g)$ in agent i consists of two parts: one is its observation of states $o_i = (o_{i1}, o_{i2}, \ldots, o_{in}, \ldots, o_{in+m})$, and the other is the assigned target g_i transmitted by the high-level policy. Note that the observation states of agents are different in different environments, so we use the most observation states in the model.

In the two-level architecture, however, irrelevant information can impede efficient learning of the low-level policies. For example, for agent i, the targets not assigned to it are not supposed to be concerned, but there is a large amount of state information of irrelevant targets during the training process that affects its learning efficiency. The target condition filter sets the unassigned target information to 0 to reduce its disturbance. That is, for the

assigned target $g_i = (g_{i1}, g_{i2}, \ldots, g_{im})$ and o_i, if g_{ij} is 0, then the corresponding $o_i(n+j)$ information will be changed to 0 before input. The filter layer can be represented as the following function:

$$filter(o_{ij}) = \begin{cases} o_{ij}, j \in [1, n] \\ 0, g_i(j-n) \quad = 0 \\ o_{ij}, other \end{cases} \tag{10}$$

For each low-level agent, it adopts an actor–critic architecture, where the actor network μ_θ outputs primitive actions, and the critic network Q_φ will collect all the state x in the environment including target and agent to calculates the value and updates the network parameters to optimize the actor network. During each episode, the agent interacts with the environment to produce trajectory data. Therefore, in agent i, the loss function of φ^i is calculated as follows:

$$L\left(\varphi^i\right) = \underset{(x,a,r_l,x') \in D_l}{(E)} \left\| Q_{\varphi^i}(x,a) - r_l^i - \gamma_l Q_{\varphi^i_{target}}\left(x', a'\right)\Big|_{a_i' = \mu_{\theta^i_{target}}(o_i')} \right\|^2 \tag{11}$$

where D_l is a subset of replay buffer B_l at a low level, and the $\mu_{\theta^i_{target}}$ and $Q_{\varphi^i_{target}}$ target are the same as the target network at a high level. Noted that the $a = (a_1, a_2, \ldots, a_n)$ is the actions performed by all agents and $r_l = \left(r_l^1, r_l^2, \ldots, r_l^n\right)$ is the rewards for all agents.

$$\begin{aligned} \theta^i_{target} &= \rho\theta^i_{target} + (1-\rho)\theta^i \\ \varphi^i_{target} &= \rho\varphi^i_{target} + (1-\rho)\varphi^i \end{aligned} \tag{12}$$

Also, the loss function of the parameter θ^i of the policy network in agent i is:

$$L\left(\theta^i\right) = \underset{x \in D_l}{E} Q_{\varphi^i}(x,a)|_{a_i = \mu_{\theta^i}(o_i)} \tag{13}$$

Below, we give the model diagram of the low-level network.

4.4. Regulation Mechanism

High-level exploration has a considerable impact on the overall exploration process. However, the non-stationarity of high-level transitions affects the overall learning efficiency. The high-level policy has the responsibility of outputting target assignments and receiving reward feedback, where the reward measures the value of high-level actions. The high-level policy expects the maximum reward that the low-level policy can receive under the given target assignment, representing the upper limit of the assignment and its true value. Nevertheless, during the training process, the low-level policy may not initially perform well under the given target assignment due to its ongoing learning exploration process and its model that has not yet been trained. This discrepancy can result in a mismatch between the reward feedback received by the high-level policy and the true value of its actions. Therefore, a good assignment of high-level action may yield poor rewards, which is not conducive to the high-level policy's learning.

The regulation mechanism lessens the impact of the low-level policy through reward regulation. The high-level problem is modeled as follows: under the specified target assignment g, there are n agents, and each agent can take action a. For achieving their respective tasks as per the target assignment, every agent is rewarded. The main challenge is to coordinate the actions of all the agents to maximize the total reward. The regulatory mechanism compares the dynamic programming's evaluation reward with the low-level execution reward, and the higher reward serves as feedback for the high-level action.

In the initialization setting, n is the number of agents, d is the number of agent actions, and r_h is the reward feedback under the action of the low-level policy. The action function, $act(.)$, can calculate the states of assigned target after the action taken by the

taking agent. The reward function, $r(.)$, can calculate the reward value corresponding to the states of assigned target using the function 6. The DP matrix $dp[n][d]$ has n rows and d columns. $dp[i][j]$ is a binary sequence, indicating the maximum number of targets that can be accomplished by the first i agents taking action, and the selected action of agent i is j. In the binary sequence of $dp[i][j]$, there is a 1 in position t which indicates that the target t is tracked. The $cnt(.)$ function calculates the number of 1 in the binary sequence, which coincides with the number of assigned targets tracked.

We give the proof of the optimality of the reward regulation algorithm below.

Theorem 1. *Both in the case of discrete and continuous action space, Algorithm 1 can calculate the optimal high-level reward value.*

Algorithm 1 Regulation Mechanism

Input: the state information s and target assignation g
Output: estimated high-level reward r_h
Initialize the number of the action d, reward r_h
Initialize the dynamic programming array dp Given the action function $act(.)$, counting function $cnt(.)$ and reward function $r(.)$
for $i := 1 \rightarrow n$ **do**
  for $j := 1 \rightarrow d$ **do**
    Compute the tracking status mask in $act(i,j)$
    for $k := 1 \rightarrow d$ **do**
      if $i = 0$ **then**
        $dp[i][j] \Leftarrow mask$
      else
        if $cnt(dp[i][j]) < cnt(dp[i-1][k]mask)$**then**
          $dp[i][j] \Leftarrow [i-1][k]|mask$
        end if
      end if
    end for
  end for
end for
for $i := 1 \rightarrow n$ **do**
  $r_n \Leftarrow max(r_h, r(dp[n][i]))$
end for
return rh

Proof of Theorem 1. We first give the proof in discrete action space. In the high-level reward function (6), when the target assignment policy g is given, the k value is determined because of it definition $k_j = \sum_{i=1}^{n} g_{ir}$. Therefore, we can deduce that as long as we allow the agent to accomplish as many tasks as possible, the more high-level rewards will be received. We use the $\Delta cnt_k(act(i,j))$ to indicate the amount of change that will occur if the i agent adopts action j, assuming that the first $i-1$ agents have reached the maximum target number and the $i-1$-th agent choose action k. Therefore, we can describe the above problem as the following: In this model, we transform the problem of regulating the reward into the problem of finding the maximum path in the abovementioned DAG. The path length before two nodes is defined as the amount of change in the completed targets resulting from the action taken to reach the end of the current path. For example, the length of the path between two nodes of agent (i,d) and agent $(i+1,j)$ is $\Delta cnt_d(act(i+1,j))$, where (i,d) means agent i takes action d. Each point in the model represents the maximum number of acquisition targets that can be achieved if the current layer performs this action. For example, the node (i,d) means $Max(cnt(dp[i][d]))$. Note that once the action of the first $i-1$ agent is determined, the amount of change of the i-th agent is also determined.

Therefore, each path in the diagram has been determined. In this model, the following optimized substructures exist

$$Max(cnt(dp[i][j])) = Max(Max(cnt(dp[i-1][k])) + \Delta cnt_k(act(i,j))) \tag{14}$$

where $k = 1, 2, \ldots, d$. Note that in different k, the $\Delta cnt_k(act(i+1,j))$ is different because of its definition. Therefore, we find the optimal substructure of this problem, which proves that the optimal solution can be obtained by using dynamic programming algorithm.

As for the continuous action space, we can find that every For every action, we ultimately examine the condition of its target. In other words, for a continuous action space Ω, the function $act(.)$ can be mapped.

$$act : \Omega \times S \rightarrow B \tag{15}$$

where B is the binary sequence, whose length is the number of targets. Therefore, the continuous action space Ω can be divided into intervals that map to different binary sequences B. For example, for cases where the number of targets is 3, all actions in a subinterval Ω_1 of the continuous action space achieve a target capture of 010, while all actions in another subinterval Ω_2 of the continuous action space achieve a target capture of 110. Consequently, the continuous action space can be divided into at most 2^m subspaces and can be treated as a case of discrete action space.

We start by considering the discrete action space. In the high-level reward function (6), when the target assignment policy π is given, the value of k is determined by its definition. Therefore, we can deduce that the more tasks an agent completes, the higher the reward received by the high-level policy. Let Δ_{ij} denote the change in the number of completed tasks when agent iii adopts action j, assuming the first $i-1$ agents have already maximized their task completions and agent $i-1$ chose action k.

This problem can be modeled as finding the maximum path in a directed acyclic graph (DAG), where each node represents the maximum number of completed tasks if the current agent performs a specific action. The path length between two nodes indicates the change in the number of completed tasks resulting from the action taken. For example, the path length between nodes (i,d) and $(i+1,j)$ is Δ_{ij}. The goal is to find the path that maximizes the total reward, which can be achieved through dynamic programming.

Define the optimal substructure for the problem as follows:

$$\Delta_{ij} = \max\left(\sum_{k=1}^{d} \Delta_{ij}\right) \tag{16}$$

where, $j = 1, 2, \ldots, d$.

Here, the optimal path is determined by the maximum number of tasks completed by the agents.

Use dynamic programming to solve the optimization problem. Let $f(i,j)$ be the maximum number of tasks completed if the first iii agents choose actions that lead to node j. Then, the optimal solution can be obtained by the following:

$$f(i,j) = \max_k\left(f(i-1,j) + \Delta_{ij}\right)$$

For continuous action spaces, we transform the continuous space into discrete intervals. Each action in the continuous space can be mapped to a binary sequence representing the target capture status. For example, if there are three targets, actions in one subinterval may correspond to capturing target 2 (010), while actions in another subinterval may correspond to capturing targets 1 and 2 (110). Thus, the continuous action space is divided into at most 2^m subspaces, where m is the number of targets.

The mapping function for continuous actions is defined as follows:

$$act(a) \rightarrow B \tag{17}$$

where B is a binary sequence of length m. □

Each subinterval of the continuous action space corresponds to a unique binary sequence, effectively converting the continuous space into a discrete space.

By treating the continuous action space as a collection of discrete subspaces, we apply the same dynamic programming approach to find the optimal high-level reward.

In conclusion, both in discrete and continuous action spaces, Algorithm 2 guarantees the optimal high-level reward by leveraging dynamic programming and appropriate mapping of actions to target statuses. This proof ensures rigorous handling of both types of action spaces, providing a clear and detailed derivation of the optimal solution.

Algorithm 2 RHMC Method

for *episode* := 1 → M **do**
Initialize a random process ℵ for action exploration
Receive initial state x
for t := 1 → *max-episode-length* **do** do
The highlevel outputs target assignment policy g
For each agent i, select action $a_i = \mu_{\theta^i}(o^i, g^i) + \aleph_t$ w.r.t the current policy and exploration after filtering treatment g by using 0.
Execute action $a = (a_1, \ldots, a_N)$ and observe lowlevel reward r_l, high-level reward r_h and new state x'
Store (x, a, r_l, x') in replay buffer B_l
Regulate high-level rewards r_h by using Algorithm 1
Store (x, a, r_l, x') in replay buffer B_h
$x \Leftarrow x'$
for *agent* := 1 → N **do**
Sample a random minibatch of D_l samples (x, a, r_l, x') from B_l
Update critic by minimizing the 0
Update actor using the sampled policy gradient in 0
Update target network parameters for agent 0
end for
Sample a random minibatch of D_h samples (s, g, r_h, s') from B_h
Update high-level critic by minimizing the 0
Update low-level actor by 0
Update target network parameters using 0
end for
end for
return

5. Experiments

The experiments were performed in two target-oriented, multi-agent cooperative environments. Specifically, the first experiment was conducted in the cooperative navigation [16] scenario, which involves static targets, and the second experiment was conducted in the multi-sensor multi-target coverage task [41] scenario, which involves dynamic targets.

5.1. Environmental Settings

Cooperative Navigation. In this experiment, a group of n agents are required to reach n landmarks (5). The agents receive rewards based on the negative distance to their nearest landmark. In the event that an agent collides with another agent, it receives a penalty of -1. Therefore, the agents must learn to cover all the landmarks while avoiding collisions. To sum up, when the agent k moves from state S_i to state S_{i+1}, the reward that the environment gives back to the agent is as follows:

$$r_k = -d_{k,s_{i+1}} - \sum_{j=1}^{n} P(k,j)\Big|_{k \neq j} \tag{18}$$

where $P(i,j)$ represents whether the two agents collided $(0:No, 1:Yes)$, and $d_{k,s_{i+1}}$ represents distance to their nearest landmark. Finally, the reward of each agent will be summarized as the cooperation reward of the whole agent system. Each agent will use the global reward for training.

Multi-sensor Multi-target Coverage Task. In this experiment, a group of n agents (sensors) need to adjust their angles to cover as many as possible of the m targets present in the environment. The sensors are fixed in place and must take an action at each time step to adjust their angle. The targets moved randomly in the environment, making it difficult for the sensors to effectively track them. In every time step, the agent obtains the observed state $s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,m}).s_{i,j}$ describes the relative position information of the agent i and the target $j.s_{i,j} = (i, j, \alpha_{i,j}, d_{i,j})$, where i and j represent the identification number of the agent and the target, respectively. $\alpha_{i,j}$ and $d_{i,j}$ represent the relative angle and absolute distance of the agent i and the target j, respectively. Each agent took a primitive action at every time step, selecting from a discrete action space comprising turn right, turn left, and stay. The agent's rotation angle was fixed at 5 degrees. In RHMC, the action of the high-level action is the assigned target g, which is a vector of length $n.g_{i,j} = 1$, which means target j is assigned to agent i and $g_{i,j} = 0$ means it is not.

As agents' policies exhibit similar functionality, sharing parameters can reduce the computational complexity of learning, thereby improving learning efficiency and performance. To encourage varied behavior among the agents, the policy incorporates each agent's identity as a one-hot vector input. If the input sizes of agents vary, padding of zeros is incorporated to maintain consistent input dimensionality. If agents have varying numbers of operations, the probability of selecting invalid operations is set to zero. For each agent, losses are determined, and the shared parameters are adjusted accordingly. Random seeds were used for the experiments, with each experiment comprising a maximum of 300 episodes. The learning rate and discount factor used for the experiments were 0.0005 and 0.99, respectively.

The following baselines were used to compare our results: HiT-MAC [37] is a hierarchical coordination architecture that uses pre-trained, maladaptive low-level policies. QMIX [20] is a multi-agent algorithm based on value decomposition. It uses two neural networks to deduce the global value by combining the Q value of each agent and the global state. IQL [17] regarded other agents as environments without considering their changes, while COMA [8] adopted a counterfactual baseline to solve the problem of credit allocation. MAPPO [32] uses an adaptive KL divergence to ensure stable policy updates during optimization. MADDPG [18] is a deep reinforcement learning-based multi-agent algorithm that uses experience replay and target networks to reduce the variance of the Q-value function.

5.2. Results and Comparisons

In this experiment, different reinforcement learning algorithms were employed to tackle the cooperative navigation task, and their effectiveness was evaluated by comparing their reward averages. The results of the experiment, as shown in Figure 3, demonstrate that the RHMC algorithm performed the best and obtained the highest reward average for this task. This is because the RHMC algorithm emphasizes the coordination between high-level strategies, effectively coordinating agents and goals, and thus improving the global reward.

By contrast, optimization methods based on global rewards, such as the QMIX and COMA algorithms, did not perform as expected. Although these methods can handle collaboration among different agents, they face the problem of conflict between local and global rewards. Therefore, these methods even performed worse than the IQL algorithm, which only ignores other agents. This suggests that global reward optimization methods like QMIX and COMA require better solutions for the conflict between local and global rewards.

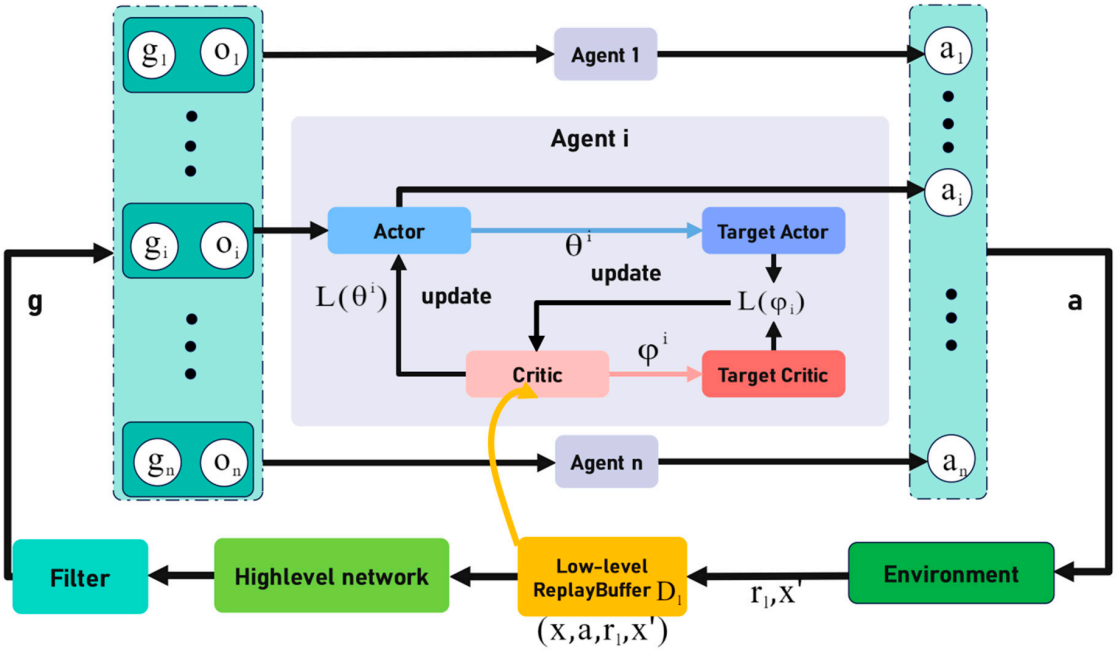

Figure 3. The frame of low-level.

Overall, this experiment demonstrates the importance of coordination between high-level strategies for the cooperative navigation task. The RHMC algorithm performs well in coordinating agents and goals to improve global rewards. However, for global reward optimization methods like QMIX and COMA, it is necessary to resolve the conflict between local and global rewards to enhance their performance.

The RHMC algorithm is not limited to cooperative navigation experiments with static targets but also performs well in multi-sensor multi-target coverage experiments with dynamic targets. As shown in Figure 4, after 5 million time steps of training in an environment with four agents and five targets, the RHMC algorithm obtained the highest global reward. This indicates that the high-level coordination of the RHMC algorithm is also applicable in multi-sensor multi-target coverage experiments and can improve global rewards.

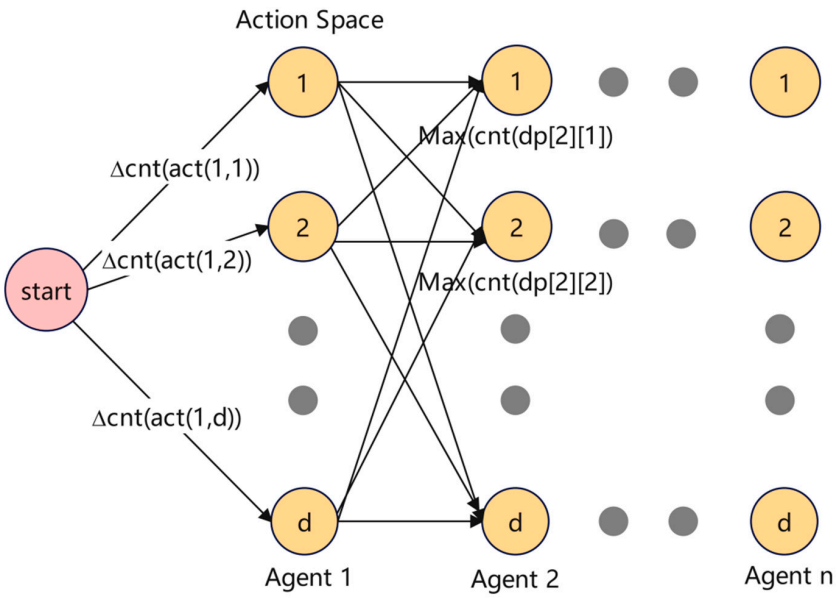

Figure 4. The model of regulation algorithm.

Some methods based on local rewards, such as the MADDPG and MAPPO algorithms, did not perform well in this task. Although they used global information to maximize the local rewards of each agent, they still could not reach the performance level of the RHMC algorithm. In contrast, the HIT-MAC and QMIX algorithms ultimately converged to around 60 rewards, while the MADDPG and MAPPO algorithms reached the highest point of their learning curve after 1.5 million time steps. These results suggest that global reward optimization methods are more suitable than local reward optimization methods for solving the task in multi-sensor multi-target coverage experiments.

During the converging phase, the RHMC algorithm achieved better global rewards than other algorithms through the coordination of high-level strategies. This is because the agents performed well in initial target allocation, but in the later stages of agent training, the strategies were bottlenecked by internal target conflicts. Therefore, the coordination of the RHMC algorithm largely improved the performance of the agents and achieved higher global rewards. This indicates that the coordination of high-level strategies is important for optimizing global rewards in multi-agent systems.

As shown in Figure 5, we investigated the scalability and adaptability of the RHMC algorithm in the multi-target multi-sensor target coverage task with respect to the number of agents. We conducted experiments with different numbers of agents, compared the RHMC algorithm with other baseline algorithms, and observed the performance of the RHMC algorithm in different scenarios.

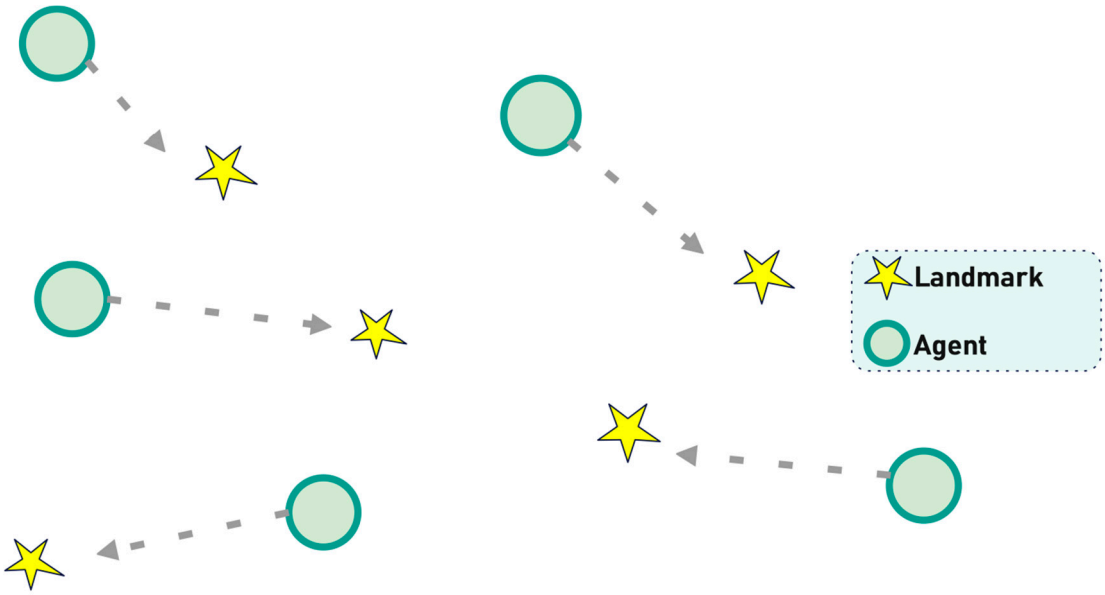

Figure 5. Target-oriented multi-agent cooperative environments: cooperative navigation.

The experimental results in Figure 6 demonstrate that the RHMC algorithm exhibits good scalability and adaptability in the presence of different numbers of agents. Regardless of an increase or decrease in the number of agents, the RHMC algorithm can adapt to different situations and achieve the highest global rewards. Notably, the performance of the RHMC algorithm remains stable when the number of agents increases, indicating its ability to scale up in multi-agent collaborative tasks. Moreover, the RHMC algorithm also shows excellent performance in multi-sensor multi-target coverage experiments, further proving its effectiveness in coordinating agents and targets to increase global rewards.

In the multi-sensor multi-target coverage task, the experimental environment was set up with four agents and five targets. Figure 7 shows the comparison of learning curves between our method (RHMC) and baseline methods. It is evident that the RHMC method outperforms other baseline methods, such as HiT-MAC, QMIX, MAPPO, and MADDPG, both in terms of the convergence speed and the final reward value.

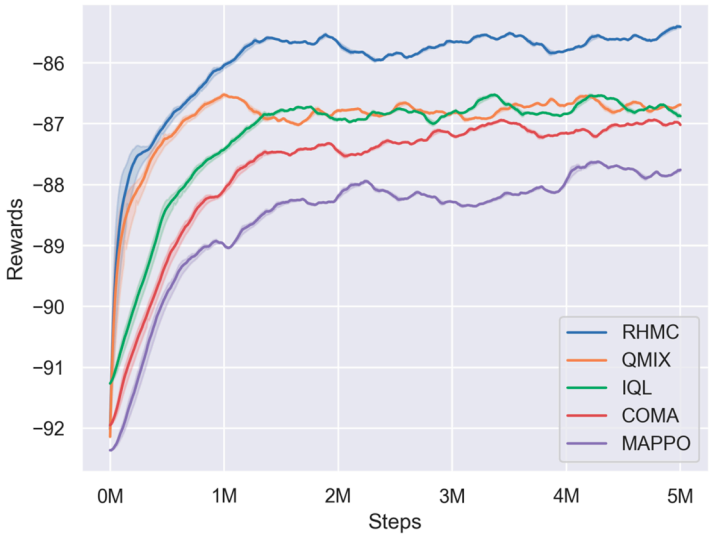

Figure 6. The learning curve of our method with baselines in cooperative navigation.

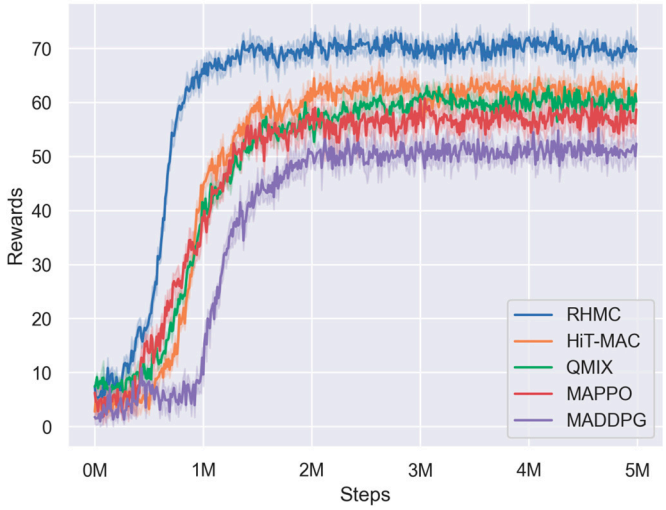

Figure 7. Learning Curves of RHMC and Baselines in Multi-Sensor Multi-Target Coverage Task.

As shown in Figures 8 and 9, the experimental environment for the multi-sensor multi-target coverage task includes five targets with varying numbers of sensors, and the results of our method are compared to the baselines under these conditions.

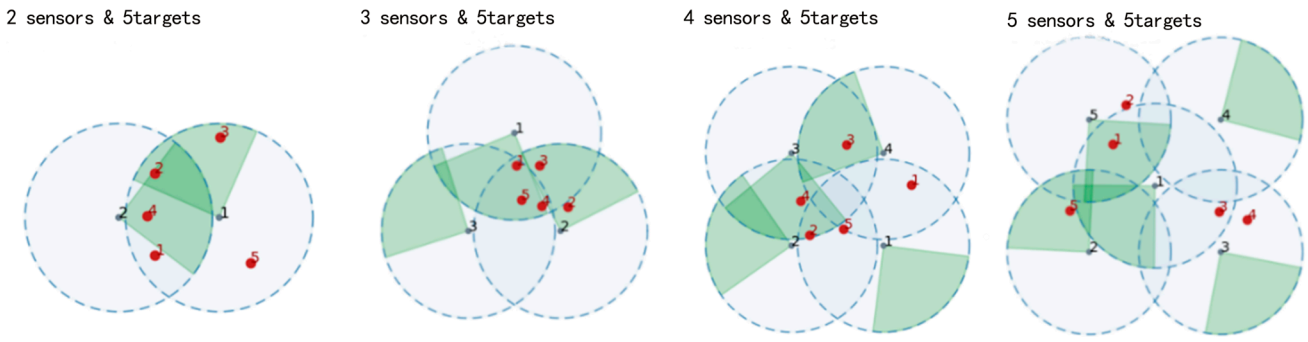

Figure 8. Experimental environment with different number of sensors with five targets in multi-sensor multi-target coverage Task.

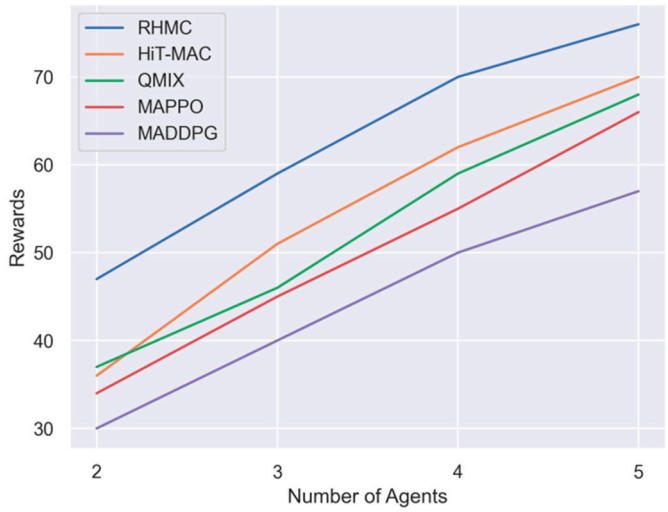

Figure 9. The result of our method with baselines in different number of sensors with five targets.

5.3. Ablation Studies

To better understand the individual contributions of each component within the RHMC framework, we conducted detailed ablation studies. These studies further validate the importance of the regulation mechanism and the target-conditioned filter in enhancing the RHMC's performance. As illustrated in Figures 2 and 10, the full RHMC method, incorporating both the regulation mechanism and the target-conditioned filter, achieved a final reward of 70. In contrast, the variants without these components achieved rewards of 55 and 35, respectively. This indicates that the regulation mechanism contributes to a 50% performance improvement, while the target-conditioned filter contributes to a 21.4% improvement.

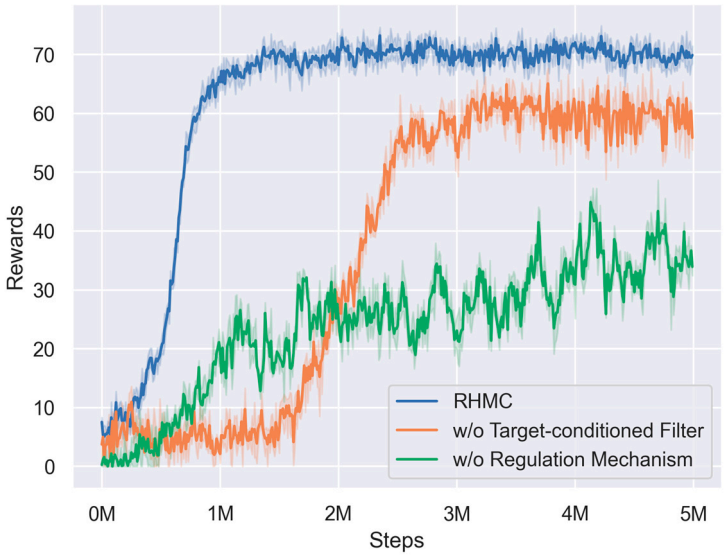

Figure 10. The learning curve of our method with its ablations in multi-sensor multi-target coverage Task.

Table 2 presents ablation study results showing final rewards and performance improvements across different methods.

Table 2. Ablation study results.

Method	Final Reward	Performance Improvement
Full RHMC	70.5	N/A
Without regulation mechanism	35.6	50.9%
Without target-conditioned filter	55.1	21.4%

The regulation mechanism plays a critical role in stabilizing the high-level policy learning process and mitigating the non-stationarity introduced by dynamic changes in low-level policies. Meanwhile, the target-conditioned filter reduces the noise from irrelevant information, accelerating the learning process and improving overall efficiency. Without the target-conditioned filter, the system is more susceptible to noise from extraneous data, leading to lower final rewards, slower learning rates, and diminished efficiency. Notably, after 1 million training steps, the RHMC method with both components demonstrates superior policy learning and gradual convergence, whereas the variant without the target-conditioned filter requires 3 million steps to achieve convergence.

These results underscore the significant roles these components play in enhancing the performance of the RHMC method. By stabilizing the training process and reducing the interference from irrelevant information, the regulation mechanism and target-conditioned filter collectively improve the overall efficiency and robustness of the RHMC framework, allowing for better coordination and performance in multi-agent environments.

6. Conclusions

In this paper, we propose the regulatory hierarchical multi-agent coordination (RHMC) framework to address coordination challenges in target-oriented multi-agent tasks. The RHMC framework introduces a two-level hierarchical reinforcement learning approach, where the high-level policy is responsible for target assignment and the low-level policy focuses on executing primitive actions.

Our extensive experimental results in both static and dynamic target-oriented environments demonstrate the significant advantages of the RHMC framework over existing methods. Specifically, in the cooperative navigation task, the RHMC algorithm achieved an average reward of 70.5, significantly outperforming QMIX and COMA, which achieved average rewards of 55.1 and 35.6, respectively. As illustrated in Figure 6, RHMC maintained a higher reward trajectory throughout the training process, demonstrating its superior high-level strategy coordination and target assignment capabilities.

In the dynamic multi-sensor multi-target coverage task, RHMC achieved a peak global reward of 92.1 after 5 million time steps, compared to 78.6 for QMIX and 75.2 for COMA, highlighting its adaptability to changing target dynamics. Figure 7 shows that RHMC's reward increased steadily, reaching higher rewards faster than other methods, thus proving its efficiency and robustness in real-time adaptation.

Ablation studies further validate the importance of the regulation mechanism and target-conditioned filter in enhancing the RHMC's performance. The full RHMC method, incorporating both the regulation mechanism and the target-conditioned filter, achieved a final reward of 70.5, while the variants without these components achieved 55.1 and 35.6, respectively. This indicates that the regulation mechanism and target-conditioned filter contribute to a 50.9% and 21.4% performance improvement, respectively, as shown in Figure 10.

The regulation mechanism effectively stabilized the high-level policy learning process, mitigating the non-stationarity introduced by dynamic changes in low-level policies. This stability, coupled with the target-conditioned filter's ability to reduce noise from irrelevant information, significantly enhanced coordination efficiency and overall system performance.

In conclusion, the RHMC framework represents a substantial advancement in multi-agent reinforcement learning, particularly in target-oriented environments. By addressing

the coordination problem through a structured hierarchical approach, RHMC not only improves learning efficiency and robustness but also achieves superior performance metrics in both static and dynamic tasks. Future research could further explore the application of RHMC in more complex multi-agent systems, potentially extending its scalability and effectiveness. By integrating these innovations, RHMC sets a new benchmark for multi-agent coordination tasks, achieving unprecedented results in terms of efficiency and effectiveness compared to state-of-the-art methods.

Author Contributions: Conceptualization, Y.Y. and W.L.; Methodology, Z.Z.; Validation, Y.Y. and J.M.; Formal analysis, W.L. and J.M.; Investigation, Y.Y.; Resources, W.L. and J.M.; Data curation, W.L.; Writing—original draft, Y.Y.; Writing—review & editing, Z.Z.; Supervision, Y.Y. and Z.Z.; Project administration, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Guangxi Natural Science Foundation of China, grant number 2023GXNSFAA026270, the Guangxi Science and Technology Project, grant number AD20159034, and the National Natural Science Foundation of China, grant numbers 62262009, 61902086, and 6202780103. The APC was funded by these grants.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yun, W.J.; Ha, Y.J.; Jung, S.; Kim, J. Autonomous aerial mobility learning for drone-taxi flight control. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 20–22 October 2021; IEEE: Piscataway, NJ, USA; pp. 329–332.
2. Wang, X.; Krasowski, H.; Althoff, M. Commonroad-rl: A configurable reinforcement learning environment for motion planning of autonomous vehicles. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 11–14 July 2021; pp. 466–472.
3. Peng, S.; Xiong, Y. A new sensing direction rotation approach to area coverage optimization in directional sensor network. *J. Adv. Comput. Intell. Intell. Inform.* **2020**, *24*, 206–213. [CrossRef]
4. Mason, F.; Chiariotti, F.; Zanella, A.; Popovski, P. Multi-agent reinforcement learning for coordinating communication and control. *IEEE Trans. Cogn. Commun. Netw.* **2024**. [CrossRef]
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
6. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; PMLR: Breckenridge, CO, USA; pp. 387–395.
7. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
8. Ye, D.; Liu, Z.; Sun, M.; Shi, B.; Zhao, P.; Wu, H.; Yu, H.; Yang, S.; Wu, X.; Guo, Q.; et al. Mastering complex control in MOBA games with deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 6672–6679.
9. Chebotar, Y.; Hausman, K.; Lu, Y.; Xiao, T.; Kalashnikov, D.; Varley, J.; Irpan, A.; Eysenbach, B.; Julian, R.; Finn, C.; et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In Proceedings of the 38th International Conference on Machine Learning (ICML), Virtual Event, 18–24 July 2021.
10. Chen, H. Robotic manipulation with reinforcement learning, state representation learning, and imitation learning (student abstract). In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual Event, 2–9 February 2021.
11. Gao, H.; Huang, W.; Liu, T.; Yin, Y.; Li, Y. PPO2: Location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems. *IEEE Trans. Intell. Transp. Syst.* **2022**, *24*, 7599–7612. [CrossRef]
12. Wang, X.; Hu, J.; Lin, H.; Garg, S.; Kaddoum, G.; Jalilpiran, M.; Hossain, M.S. QoS and privacy-aware routing for 5G enabled industrial internet of things: A federated reinforcement learning approach. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4189–4197. [CrossRef]
13. Zhong, K.; Yang, Z.; Xiao, G.; Li, X.; Yang, W.; Li, K. An efficient parallel reinforcement learning approach to cross-layer defense mechanism in industrial control systems. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 2979–2990. [CrossRef]
14. Krnjaic, A.; Thomas, J.D.; Papoudakis, G.; Schäfer, L.; Børsting, P.; Albrecht, S.V. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. *arXiv* **2022**, arXiv:2212.11498.

15. Zhang, R.; Zong, Q.; Zhang, X.; Dou, L.; Tian, B. Game of drones: Multi-UAV pursuit-evasion game with online motion planning by deep reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 7900–7909. [CrossRef]
16. Yun, J.; Goh, Y.; Yoo, W.; Chung, J.-M. 5G multi-RAT URLLC and eMBB dynamic task offloading with MEC resource allocation using distributed deep reinforcement learning. *IEEE Internet Things J.* **2022**, *9*, 20733–20749. [CrossRef]
17. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE* **2017**, *12*, e0172395. [CrossRef] [PubMed]
18. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
19. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
20. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 7234–7284.
21. Ma, T.; Peng, K.; Rong, H.; Qian, Y.; Al-Nabhan, N. Hierarchical coordination multi-agent reinforcement learning with spatio-temporal abstraction. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 533–545. [CrossRef]
22. Gui, Y.; Zhang, Z.; Tang, D.; Zhu, H.; Zhang, Y. Collaborative dynamic scheduling in a self-organizing manufacturing system using multi-agent reinforcement learning. *Adv. Eng. Inform.* **2024**, *62*, 102646. [CrossRef]
23. Xie, S.; Li, Y.; Wang, X.; Zhang, H.; Zhang, Z.; Luo, X.; Yu, H. Hierarchical relationship modeling in multi-agent reinforcement learning for mixed cooperative–competitive environments. *Inf. Fusion* **2024**, *108*, 102318. [CrossRef]
24. Geng, M.; Pateria, S.; Subagdja, B.; Tan, A.-H. HiSOMA: A hierarchical multi-agent model integrating self-organizing neural networks with multi-agent deep reinforcement learning. *Expert. Syst. Appl.* **2024**, *252*, 124117. [CrossRef]
25. Tang, Y.; Sun, J.; Wang, H.; Deng, J.; Tong, L.; Xu, W. A method of network attack-defense game and collaborative defense decision-making based on hierarchical multi-agent reinforcement learning. *Comput. Secur.* **2024**, *142*, 103871. [CrossRef]
26. Xi, J.; Wang, C.; Yang, X.; Yang, B. Limited-budget output consensus for descriptor multiagent systems with energy constraints. *IEEE Trans. Cybern.* **2020**, *50*, 4585–4598. [CrossRef]
27. Zhang, Y.; Tian, G.; Zhang, S.; Li, C. A knowledge-based approach for multiagent collaboration in smart home: From activity recognition to guidance service. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 317–329. [CrossRef]
28. Tian, Y.; Zheng, B.; Wang, Y.; Zhang, Y.; Wu, Q. College library personalized recommendation system based on hybrid recommendation algorithm. *Procedia CIRP* **2019**, *83*, 490–494. [CrossRef]
29. Pérolat, J.; Leibo, J.Z.; Zambaldi, V.F.; Beattie, C.; Tuyls, K.; Graepel, T. A multi-agent reinforcement learning model of common-pool resource appropriation. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
30. Feehan, G.; Fatima, S.S. Augmenting reinforcement learning to enhance cooperation in the iterated prisoner's dilemma. In Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART), Vienna, Austria, 3–5 February 2022.
31. Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; Russell, S.J. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
32. Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of MAPPO in cooperative multi-agent games. In Proceedings of the 38th International Conference on Machine Learning (ICML), Virtual Event, 18–24 July 2021.
33. Zhang, C.; Tian, Y.; Zhang, Z.; Xue, W.; Xie, X.; Yang, T.; Ge, X.; Chen, R. Neighborhood cooperative multiagent reinforcement learning for adaptive traffic signal control in epidemic regions. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 25157–25168. [CrossRef]
34. Foerster, J.N.; Chen, R.Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; Mordatch, I. Learning with opponent-learning awareness. *arXiv* **2017**, arXiv:1709.04326.
35. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), Stockholm, Sweden, 10–15 July 2018.
36. Chen, M.; Li, Y.; Wang, E.; Yang, Z.; Wang, Z.; Zhao, T. Pessimism meets invariance: Provably efficient offline mean-field multi-agent RL. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 17913–17926.
37. Xu, X.; Li, R.; Zhao, Z.; Zhang, H. Stigmergic independent reinforcement learning for multiagent collaboration. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 4285–4299. [CrossRef] [PubMed]
38. Kim, Y.; Nam, W.; Kim, H.; Kim, J.-H.; Kim, G. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, CA, USA, 10–15 June 2019; PMLR: Breckenridge, CO, USA; pp. 3379–3388.
39. Kulkarni, T.D.; Narasimhan, K.; Saeedi, A.; Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016.
40. Eysenbach, B.; Gupta, A.; Ibarz, J.; Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv* **2018**, arXiv:1802.06070.
41. Xu, J.; Zhong, F.; Wang, Y. Learning multi-agent coordination for enhancing target coverage in directional sensor networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10053–10064.

42. Van Otterlo, M.; Wiering, M. Reinforcement learning and Markov decision processes. In *Reinforcement Learning: State-of-the-Art*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 3–42.
43. Grondman, I.; Busoniu, L.; Lopes, G.A.D.; Babuska, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2012**, *42*, 1291–1307. [CrossRef]
44. Nachum, O.; Norouzi, M.; Xu, K.; Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
45. Fontanesi, L.; Gluth, S.; Spektor, M.S.; Rieskamp, J. A reinforcement learning diffusion decision model for value-based decisions. *Psychon. Bull. Rev.* **2019**, *26*, 1099–1121. [CrossRef] [PubMed]
46. Ghavamzadeh, M.; Mahadevan, S.; Makar, R. Hierarchical multi-agent reinforcement learning. *Auton. Agents Multi-Agent. Syst.* **2006**, *13*, 197–229. [CrossRef]
47. Yang, J.; Borovikov, I.; Zha, H. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Auckland, New Zealand, 9–13 May 2020.
48. Loo, Y.; Gong, C.; Meghjani, M. A Hierarchical Approach to Population Training for Human-AI Collaboration. In Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI), Macau, China, 19–25 August 2023.
49. Ibrahim, M.; Fayad, A. Hierarchical Strategies for Cooperative Multi-Agent Reinforcement Learning. *arXiv* **2022**, arXiv:2206.12345.