Subgoal-Based Hierarchical Reinforcement Learning for Multiagent Collaboration

Cheng Xu, *Member, IEEE*, Yuchen Shi, Changtian Zhang, Ran Wang, *Graduate Student Member, IEEE*, Shihong Duan, Yadong Wan, and Xiaotong Zhang, *Senior Member, IEEE*

***Abstract*— Recent advancements in reinforcement learning (RL) have driven progress across various domains; however, RL algorithms often struggle in complex multiagent environments due to challenges such as instability, low sample efficiency, and the curse of dimensionality. Hierarchical RL (HRL) provides a structured framework for decomposing complex tasks into more manageable subtasks, making it a promising approach for multiagent systems. In this article, we introduce a novel hierarchical architecture that autonomously generates effective subgoals without explicit constraints, thereby enhancing both training stability and adaptability. To further improve sample efficiency and adaptability, we propose a dynamic goal-generation strategy that adjusts subgoals in response to environmental changes. Additionally, we address the critical challenge of credit assignment in multiagent settings by integrating our hierarchical architecture with a modified QMIX network, thereby facilitating more effective strategy coordination. Extensive comparative experiments against state-of-the-art RL algorithms demonstrate that our approach achieves superior convergence speed and overall performance in multiagent environments. These results validate the effectiveness and flexibility of our method in handling complex coordination tasks. The implementation is publicly available at https://github.com/SICC-Group/GMAH**

***Index Terms*— Hierarchical learning, multiagent collaboration, reinforcement learning (RL), subgoal learning.**

I. Introduction

MULTIAGENT reinforcement learning (MARL) [1] has recently achieved remarkable success across a wide range of domains, including autonomous driving [2], robotic navigation [3], and strategic game-playing [4]. Despite these advancements, the practical deployment of RL remains limited by several fundamental challenges, particularly the *curse of dimensionality* [5], where increasing state and action space complexity significantly slows down learning. Moreover, nonstationarity and credit assignment in MARL further complicate training. Value-based approaches such as WQMIX [6] and QTRAN [7] employ value factorization to decompose the global Q value into individual utility functions, achieving strong performance in cooperative tasks. Conversely, policy-based methods like multiagent proximal policy optimization (MAPPO) [8] and deep deterministic policy gradient (DDPG) [9] have been adapted to MARL to manage continuous action spaces and complex policies.

However, the majority of prior work on MARL has not adequately tackled the challenge of sparse rewards. To mitigate sparse rewards, intrinsic-motivation techniques leverage environmental cues and task-specific goals to enhance the learning process. Methods such as density-based exploration [10] and curiosity-driven exploration [11] encourage broader state-space coverage, improving exploration efficiency. Additionally, hierarchical reinforcement learning (HRL), inspired by Sutton's *options* framework [12], has been proposed to decompose complex decision-making processes into high-level policies that guide macroactions.

In some HRL algorithms, techniques such as hindsight experience replay (HER) [13] and universal value function approximators (UVFA) [14] further improve sample efficiency by reframing past failures as successes under alternative goals. Although HRL is effective in single-agent scenarios, extending these methods to multiagent settings remains challenging due to the increased difficulty of coordinating exploration across agents. While hierarchical value decomposition (HAVEN) [15] extends HRL to multiagent settings, it cannot dynamically adjust the temporal spacing of subgoals or its intrinsic reward mechanism.

In response to these challenges, this article presents a novel approach that reshapes intrinsic rewards for multiagent RL. Building upon intrinsic-motivation methods such as exploration by random network distillation (RND) [16] and never give up (NGU) [17], our approach extends these techniques to facilitate *coordinated* and *efficient* exploration in hierarchical MARL. Unlike previous works that primarily focus on single-agent RL or struggle to scale intrinsic rewards effectively in multiagent contexts, our method dynamically adjusts intrinsic rewards based on interagent interactions and collective experiences, and aligns these signals with high-level subgoal structure. This fosters more comprehensive

Received 11 February 2025; revised 16 September 2025; accepted 16 December 2025. Date of publication 12 January 2026; Date of current version 26 January 2026. This work was supported in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515140071, in part by the National Natural Science Foundation of China (NSFC) under Grant 62101029, and in part by China Scholarship Council Award under Grant 202006465043 and Grant 202306460078. This article was recommended by Associate Editor B. Zhang. *(Corresponding authors: Shihong Duan; Yadong Wan.)*

The authors are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China, and also with Shunde Innovation School, University of Science and Technology Beijing, Foshan 528399, China (e-mail: xucheng@ustb.edu.cn; shiyuchen199@sina.com; changtizh14264@163.com; wangran423@foxmail.com; duansh@ustb.edu.cn; wyd@ustb.edu.cn; zxt@ies.ustb.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2025.3646451.

Digital Object Identifier 10.1109/TSMC.2025.3646451

exploration and improves coordination, ultimately leading to enhanced learning efficiency in high-dimensional environments.

This work introduces a task tree-based hierarchical framework that seamlessly integrates with multiagent systems. Through extensive empirical analysis, we demonstrate that our method significantly outperforms conventional MARL approaches, highlighting the potential of hierarchical learning architectures in complex collaborative tasks. The primary contributions of this article are as follows.

1) *Task Tree-Based Subgoal Generation Method:* We propose an innovative subgoal representation strategy that enhances comprehensibility and relevance for low-level policies, simplifying intrinsic reward function design and improving overall policy performance.
2) *Adaptive Subgoal Generation Policy:* We introduce a dynamic subgoal adjustment mechanism that responds to critical environmental changes, ensuring a more efficient and robust learning process.
3) *Goal Mixing Network Fine-Tuning:* We design a novel mixing network that refines high-level policy learning through a joint goal value function trained with global rewards. This extension of the hierarchical framework to multiagent environments addresses key challenges such as high-dimensional state spaces and reward allocation.

The remainder of this article is structured as follows. Section II reviews related research, positioning our work within the broader context of RL and MARL advancements. Section III introduces our subgoal-based hierarchical RL framework for multiagent collaboration, detailing its theoretical foundations and methodological components. Section IV presents experimental evaluations and an in-depth discussion of the findings, demonstrating the efficacy of our proposed approach. Finally, Section V concludes the article, summarizing key insights and outlining potential directions for future research.

II. Related Work

A. Multiagent Reinforcement Learning

MARL can be regarded as a partially observable Markov decision process (POMDP), generally abstracted as the tuple $G = < I, A, S, O, P, r, \gamma >$ [18], where $I = I_1, \ldots, I_n$ represents n agents, A is the action space with agents choosing an action $a_t \in A$ at any given time t, forming a joint action a. S denotes the state space, O denotes the observation space, $P : S \times A \to S$ denotes the state transition function $P(s_{t+1}|s_t, a_t)$ indicating the probability that agent I_i transitions to state s_{t+1} from state s_t by taking action a_t, r is the reward function $r(s, a) : S \times A \to R$, and $\gamma \in [0, 1)$ is the discount factor. Each agent i has an action-observation history (trajectory) $\tau^i = < o_0, a_0, \ldots, o_T, a_T >$, which is based on the agent's policy $\pi^i(a_t|\tau)$. The objective function is to optimize its accumulated discounted reward $J(\theta) = E_{s_t, a_t}\left[\sum_t \gamma^t r(s_t, a_t)\right]$.

In the field of MARL, learning frameworks are typically categorized as centralized or decentralized. Centralized approaches [19] utilize a unified policy to govern the collective actions of all agents, leveraging global information to optimize joint decision-making. In contrast, decentralized methods [20] allow each agent to independently optimize its own reward based solely on local observations, enhancing scalability but often at the cost of coordination efficiency. Bridging these two paradigms, centralized training with decentralized execution (CTDE) [21] has emerged as a widely adopted framework. CTDE enables agents to exploit global state information during training while maintaining independent decision-making at execution, thereby balancing coordination and scalability. Notable CTDE-based algorithms, such as COMA [22] and MADDPG [23], leverage an actor–critic architecture to train a centralized critic using global state inputs. Similarly, value decomposition methods, including QTRAN [7], VDN [24], QMIX [25], and WQMIX [6], decompose the joint Q-function into individual agent-specific Q-functions, setting performance benchmarks in MARL.

Despite its success in single-agent settings, proximal policy optimization (PPO) has seen limited adoption in MARL. Yu et al. [8] attributed this limitation to PPO's lower sample efficiency and the difficulty of directly adapting its single-agent tuning strategies to multiagent environments. To address these challenges, their work extends PPO to MARL by modifying the policy distribution and restructuring the centralized value function to rely on global rather than local state information. Several key factors influence PPO's effectiveness in MARL, including generalized advantage estimation (GAE) [26], observation normalization, gradient clipping, value function clipping, layer normalization, and ReLU activation functions with orthogonal initialization. These modifications collectively define MAPPO, an adaptation of PPO designed for cooperative multiagent learning.

Furthermore, QMIX [25] introduces an innovative approach to MARL by combining individual agents' local Q-functions through a mixing network that incorporates global state information during training. Unlike traditional decomposition-based methods, QMIX posits that complete factorization of the joint Q-function is unnecessary for effective decentralization. Instead, ensuring that the global arg max operation on the joint action–value function Q_{total} aligns with the individual arg max operations on each agent's Q-function Q_i is sufficient for optimal coordination.

WQMIX [6] builds on QMIX by viewing learning as a projection of the Bellman target onto the *monotonic* value-factorization class and replacing the uniform TD objective with a weighted projection. Concretely, WQMIX assigns larger weights to joint actions that are *coordination-consistent* (i.e., near the decentralized greedy solution implied by the IGM condition) and downweights actions that conflict with decentralized execution. This yields a weighted TD loss and modified backup that reduce approximation error and mitigate over/under-estimation stemming from representational limits of monotonic mixing—without relaxing the monotonic constraint or sacrificing linear scalability in the number of agents. WQMIX improves stability and sample efficiency over QMIX on cooperative benchmarks.

Relation to MARL baselines. Value-factorization methods (e.g., WQMIX/QTRAN) and policy-gradient methods (IPPO/MAPPO) target cooperative MARL under CTDE but provide no mechanism for temporal abstraction or

interaction-aware intrinsic motivation. As a result, exploration often duplicates across agents and becomes nonstationary when single-agent bonuses are naively ported to MARL. By contrast, our framework couples interpretable, discrete subgoals with team-consistent intrinsic signals and a goal-level mixing network, aligning exploration with coordinated behavior while remaining compatible with standard CTDE training.

B. Hierarchical Reinforcement Learning

HRL is a structured extension of RL that decomposes complex decision-making processes into multiple levels, where each layer can be trained independently using value-based or policy gradient methods. By introducing hierarchical control, HRL enables agents to operate across different temporal scales, facilitating the decomposition of complex tasks into manageable subtasks or predefined skills. This hierarchical structure enhances sample efficiency, accelerates learning, and enables more effective policy generalization in long-horizon tasks. HRL is particularly advantageous in environments characterized by prolonged task durations and delayed rewards. However, training HRL models presents significant challenges due to the inherent nonstationarity of the environment. Since higher-level policies depend on the evolving strategies of lower layers, transition dynamics fluctuate, complicating the stabilization of optimal policies. To mitigate these issues, Huiling Wang [27] proposed probabilistic subgoal representations (PSR), which learn a posterior over subgoal representations via Gaussian processes, providing an uncertainty-aware latent goal space and adaptive memory that improves robustness and sample efficiency under stochastic dynamics. Empirically, PSR achieves strong gains across standard benchmarks and transfers low-level skills across tasks.

Another notable HRL approach, HRLfD-RbRS [28], improves hierarchical RL from demonstrations by introducing reachability-based reward shaping to reduce the negative impact of suboptimal expert data. This framework constrains high-level subgoals within a reachability region around demonstrations, thereby stabilizing training and accelerating convergence on complex robotic tasks, even when demonstrations are imperfect.

Beyond hierarchical subgoal selection, disentangled unsupervised skill discovery (DUSDi) [29] introduces a skill-based paradigm that learns disentangled, reusable skills purely from interaction, enabling efficient chaining and composition within a hierarchical controller to tackle sparse-reward, long-horizon tasks. Reported results show improved downstream performance over prior skill-discovery baselines.

In contrast, graph-structured HRL provides an alternative perspective by explicitly organizing subgoals through relational structure. G4RL [30] constructs state graphs and a graph encoder–decoder to generate subgoal representations that generalize to unseen states, yielding more reliable subgoal identification and improved performance in both dense- and sparse-reward settings.

Hierarchical value decomposition (HAVEN) [15] extends HRL to multiagent settings. HAVEN brings hierarchical RL into fully cooperative multiagent problems by coupling a two-level, QMIX-style value decomposition architecture with a dual coordination mechanism that links interlevel and interagent behaviors. At a slower high-level timescale, macroactions are selected; the high-level advantage is then redistributed as an intrinsic reward to low-level controllers over the next k steps, aligning temporal abstraction with decentralized coordination and stabilizing concurrent learning.

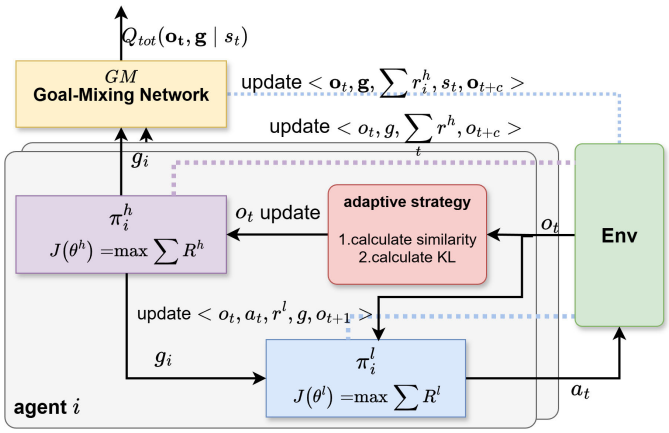


Fig. 1. Overall GMAH structure.

Limitations of the HRL algorithm in a multiagent field. Recent HRL advances (e.g., HRLfD-RbRS, DUSDi, and G4RL) improve subgoal representation, stability, and sample reuse in single-agent settings; however, they do not resolve interagent coordination or credit assignment inherent to MARL. Closer to our setting, hierarchical MARL approaches such as HAVEN extend temporal abstraction into cooperative tasks, yet typically rely on fixed subgoal cadences and do not provide an adaptive, interaction-aware intrinsic reward reshaping tied to team behavior. Our method differs by: 1) using a task-tree subgoal space for interpretability and ease of evaluation; 2) adapting subgoal update timing via state-feature change and policy-shift detection; and 3) introducing a goal mixing network that performs monotonic value composition at the high-level subgoal layer to address multiagent credit assignment.

III. Subgoal-Based HRL

This section delineates the communication-restricted multiagent cooperative environment explored in this study. The environment features N agents, each limited to its own local observations o_t^i. These agents operate concurrently, making independent decisions and interacting with the environment. They each receive individual local rewards r_t^i, but cannot communicate or access the rewards of others. The environment stops when all agents collectively complete the final task T, which is structured into a sequence of subtasks g_i. These subtasks can be accomplished individually or through cooperative efforts, presenting various routes to achieve the final objective.

A. General Architecture

We introduce an efficient subgoal-based multiagent HRL approach, designated as *GMAH*, which is visualized in Fig. 1.

This method segments each agent's policy into two levels: a high-level policy π^h and a low-level policy π^l. The high-level policy in the GMAH method decomposes the main task into simpler, attainable subgoals based on prior knowledge. These subgoals are crucial steps for completing the task and are executed by agents under the strategic direction of the high-level policy, which utilizes environmental rewards for navigation. Simultaneously, the low-level policy, motivated by intrinsic rewards, guides agents in achieving these subgoals.

The central component of the GMAH method is the training of a subgoal value function that utilizes global rewards to enhance the efficiency of the high-level policy in allocating subgoals among agents. This hierarchical architecture promotes better coordination and task execution across the agent collective. Further sections will detail the hierarchical architecture applied to each agent in the GMAH algorithm, highlighting advancements over conventional hierarchical models, such as the task-tree method for subgoal generation. Additionally, an adaptive goal generation policy is introduced to refine the hierarchical execution logic, alongside a method for fine-tuning the goal mixing network. This approach aids in expanding the sophisticated hierarchical framework to wider multiagent environments.

B. Task-Tree Style Subgoal Generation

HRL typically defines the goal space G as a subspace of the state space S or observation space O, that is, $G \subseteq S$ or $G \subseteq O$. The low-level reward is defined as some distance metric between the goal space and state space [31], [32], or as a binary function [10]. These definitions represent anticipated future states that agents aim to reach. However, due to the inherent uncertainty in neural network outputs, high-level policy networks often struggle to generate subgoal representations that align with environmental designs, causing issues in many scenarios. To address the complexities of abstract subgoal definitions, this article proposes a task-tree style subgoal generation method that eschews goal spaces in favor of imparting actual significance to subgoals, training the high-level policy to generate subgoals in a task-tree structure.

The core idea of the task-tree style subgoal generation method is to replace complex abstract goal spaces with simple, explicit sets. Assume the environment's ultimate task is T_E, which, upon analysis of the task content and environment, is deemed decomposable into N simple tasks. Completing T_E requires achieving N independent and distinct simple tasks or reaching certain intermediary states, referred to as subgoals T_i. These N distinct subgoals form a set T_N, with each element representing a subgoal. The high-level policy's decision-making objective is to select a subgoal from this set for the low-level policy to achieve. This task-tree structure is depicted in Fig. 2, starting from the root node (the environment's initial state) and proceeding through subgoals selected by the high-level policy, with the edges weighted by the extrinsic rewards obtained during the process, repeating until the terminal state s_T (environment termination or completion of T_E). The high-level policy's ultimate goal is to complete T_E, generally by maximizing environmental rewards, that is, $J(\pi^h) = \max R(\tau)$. This tree-structured approach

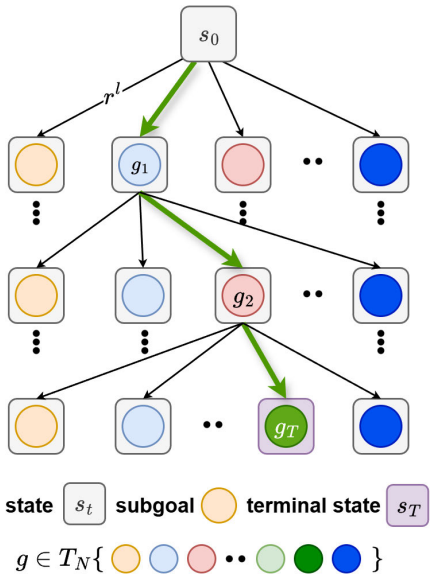


Fig. 2. Typical diagram of the task tree.

significantly simplifies high-level policy decision-making by reducing it to a problem of planning subgoals or solving for the optimal path in an N-ary tree.

Despite the dynamic programming approach typically used for solving N-ary tree path problems, it is impractical in a reinforcement learning (RL) context due to the stochastic nature of environment initialization and the nonstationarity of any given state or observation within the task tree. Thus, neural networks are still employed to model this policy. The high-level policy network structure, as shown in Fig. 3(a), outputs a probability distribution over the set of subgoals, from which a subgoal g is sampled, that is, $g \sim \pi^h(o_t), g \in T_N$. This design transforms the high-level policy's decision-making from a state-space to subgoal-set mapping, drastically reducing the dimensionality of the policy's outputs and the overall learning difficulty.

The clarity of objectives set by the high-level policy enhances the execution capabilities of the low-level policy.

In the hierarchical architecture, the low-level policy is driven by intrinsic rewards that are conditioned on the agent's performance toward achieving these subgoals. We use rule-based evaluations to determine the achievement of subgoals, but with a significant enhancement. The task-tree style subgoal generation policy imparts concrete meaning to each subgoal, simplifying the evaluation rules while achieving greater accuracy than distance metrics. Our model modifies the standard indicator function by introducing a time-decaying factor to the intrinsic reward function for the low-level policy, which is defined as follows:

$$R_t^l(o_t, a_t|g) = \begin{cases} 1 - \beta \dfrac{t}{T_M}, & \text{if get } g \\ 0, & \text{else .} \end{cases} \tag{1}$$

Here, T_M represents the maximum time steps per episode, t is the current time step, and β is a discount factor that adjusts the reward based on the time taken to achieve the subgoal g.

The hierarchical structure applied to each agent aligns with traditional models, maintaining a consistent high-level to low-

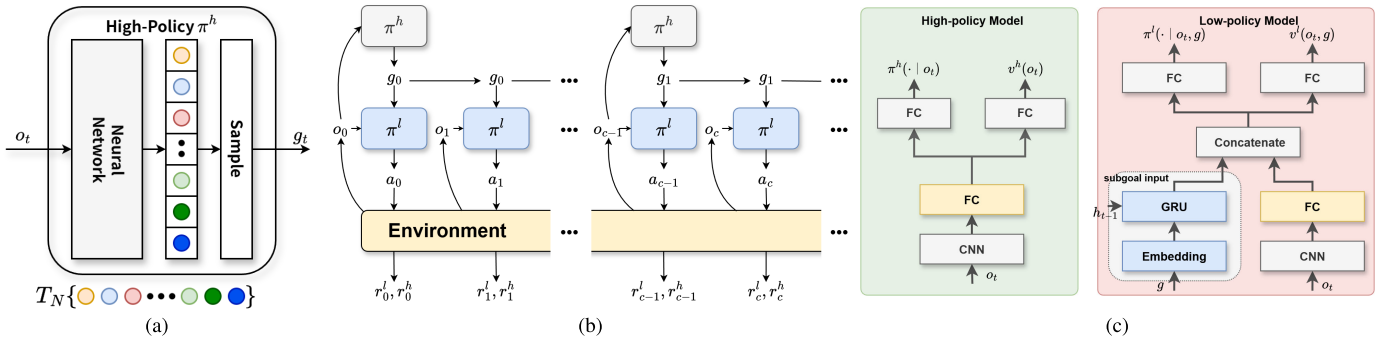


Fig. 3. (a) High-policy network model. (b) Hierarchical architecture applied by a single agent in the GMAH algorithm. (c) Network model of GMAH.

level policy framework as shown in Fig. 3(b) and (c). At each time step t, the high-level policy outputs probabilities for each subgoal based on the observation o_t and samples a subgoal g_t. Subsequently, at each time step t', the low-level policy generates actions $a_{t'}$ based on the current observation $o_{t'}$ and the subgoal $g_{t'}$, executing these actions. The maximum time steps between subgoal generations is defined as C. Once the subgoal is achieved or the action sequence exceeds C time steps, a new subgoal g_{t+c} is sampled, and transition data $\langle o_t, g_t, \sum R_{t:t+c}, o_{t+c}\rangle$ is collected for training the high-level policy, with $\langle o_t, g_t, r^l, o_{t+1}\rangle$ gathered for the low-level policy.

The approach leverages a task-tree style subgoal generation method that employs prior knowledge to construct a higher-level abstraction model. This model predetermines a set of subgoals, which both the high-level and low-level policies rely on for training, independent of each other. This decouples the training of the high and low-level policies, addressing a key instability issue in hierarchical learning. By training the low-level policy first to a satisfactory performance level before training the high-level policy, this approach simplifies the overall training process and significantly reduces the training complexity, laying a solid foundation for extending the hierarchical framework to multiagent environments.

C. Adaptive Goal Generation Policy

The GMAH algorithm incorporates a robust hierarchical architecture optimized for multiagent interactions. One significant challenge within this architecture is managing the goal update interval, which affects both the high-level and low-level policies. This interval encompasses the time steps between the decision points of the high-level policy to generate new subgoals and the duration required by the low-level policy to achieve these subgoals. In response, this article introduces an adaptive goal update policy for the GMAH algorithm that features flexible update intervals and proactive goal updates.

1) Flexible Update Intervals: Most methods, such as HIRO and HRAC [31], [33], adopt a fixed interval. HIRO defines the goal space as a subspace of the state space and treats the goal update interval as a hyperparameter c, where the high-level policy generates a subgoal g_t every c time steps. The subgoals in HIRO represent the expected relative distance between the target state the agent aims to reach and its current state. Additionally, HIRO designed a goal transition function $h(s_t, g_t, s_{t+1}) = s_t + g_t - s_{t+1}$, which updates the subgoal based on the agent's state transitions. Fundamentally, the agent's goal within c time steps is a fixed expected position. HIRO uses the Frobenius norm to measure the distance between the state and subgoal, with an intrinsic reward function being a binary function. By setting an appropriate threshold ε, a reward of 1 is assigned if the distance is less than this threshold; otherwise, a reward of 0 is given. However, since HIRO does not explicitly constrain the "distance" represented by subgoals generated by the high-level policy, nor does it ensure that the agent reaches the goal within the time interval c and receives the corresponding reward, the data produced by failed c-step interactions cannot be used for training. These ineffective data scenarios can be categorized as follows.

1) The agent nearly reaches the goal after c time steps but does not meet the threshold distance for goal completion.
2) The agent is farther from the goal after c time steps, even more than at the time before c steps.

Examples of these trajectories are shown in Fig. 4(a), where trajectory a represents the first scenario, and trajectories b, c, and d represent the second scenario. Trajectory a points out the direction for the agent to complete the goal, while trajectories b, c, and d are not. But for the RL method, the value of these four failed trajectories is the same. HRAC makes a critical enhancement over HIRO by imposing a K-step neighborhood constraint. This constraint limits the goals generated by the high-level policy to the space within a K Manhattan distance around the agent, thus making it easier for the agent to achieve the goal within c time steps. The constraints on the goal space by HRAC are illustrated in Fig. 4(b).

Besides neighborhood constraints, HRAC also utilizes an important technique to increase sample efficiency: HER. For failed exploration trajectories, it still samples their state transitions $\langle s_t, a_t, r_t^l, s_{t+1}, g_t\rangle$, but it relabels the state the agent actually reaches after c time steps as the subgoal for that batch of transition data $\widehat{g_t} = s_{t+c}$, and calculates the corresponding intrinsic reward $\widehat{r_t}$. The data $\langle s_t, a_t, \widehat{r_t^l}, s_{t+1}, \widehat{g_t}\rangle$ are then used for training the low-level policy. However, although HRAC imposes constraints on the range of subgoals, thereby enhancing the low-level policy's ability to achieve them and subsequently improving algorithm performance, it still lacks an effective policy to ensure the low-level policy can achieve the constrained subgoals. This article adopts a flexible goal update interval combined with the technique of HER. This

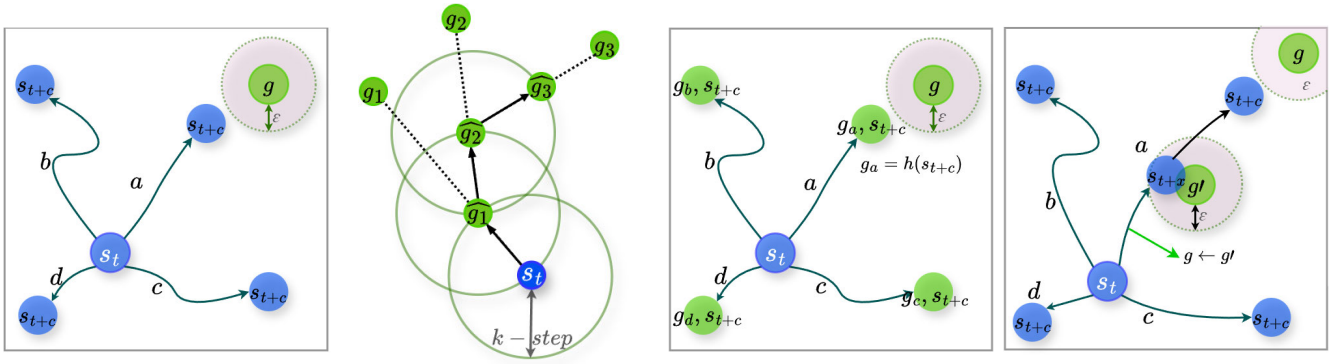


Fig. 4. (a) Trajectory example of agent c-step interaction. (b) Adjacency constraint on the goal space of HRAC. Goal relabel on the trajectory of agent c-step interaction: (c) relabel the abstract subgoal and (d) relabel the subgoal of GMAH.

approach discards the fixed goal update interval, assessing whether the subgoal is achieved with each step of interaction after the low-level policy receives a subgoal. If the subgoal is achieved, the high-level policy is prompted to update the subgoal. If it is not achieved, the subgoal is proactively updated after reaching the maximum update interval c.

In addition, this article also uses HER to improve sample efficiency. However, due to the different definition of the target space, the logic of re-marking the agent trajectory of the GMAH algorithm is different from that of HIRO, HRAC, and other algorithms. In order to use the failed trajectory to learn, it is necessary to judge whether the agent has completed other subtargets in the interaction process between the agent and the environment in advance. Prerecord the subgoals actually achieved and calculate the corresponding intrinsic rewards. The difference between methods such as GMAH and HIRO using HER for remarking is shown in Fig. 4(c) and (d).

2) Proactive Goal Updating: This article explores the adaptive generation of subgoals within a hierarchical RL framework, advocating that high-level policy should dynamically update subgoals not only upon the completion of existing goals or reaching predetermined intervals but also in reaction to significant environmental changes. A key challenge is determining the precise moments for the high-level policy to update subgoals.

Thus, we propose a dual-stage decision-making process. The initial phase employs an Auto-Encoder that is considerably more compact than the high-level policy model. This Auto-Encoder consists of an Encoder, which condenses the input into a low-dimensional feature vector f, and a Decoder, which reconstructs the input from this vector. Training is based on minimizing the differences between actual inputs and their reconstructions, guided by the loss function detailed in the following equation:

$$\ell\left(\theta_e, \theta_d\right) = \left\| o_t - \phi_{\theta_d}\left(\phi_{\theta_e}\left(o_t\right)\right)\right\|^2 \tag{2}$$

where ϕ_{θ_e} and ϕ_{θ_d} represent the encoder and decoder, respectively.

Inspired by DSR, the feature vector f output by the encoder network is input into a fully connected layer parameterized by ω, outputting a mapping from state to reward. This mapping represents the estimated value based on the state. Through this

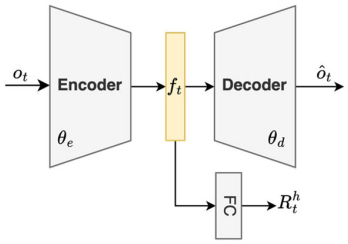


Fig. 5. Autoencoder with successor feature correction.

design, the encoder network's output feature vectors are used with the agent's state transition data to train the joint network $\langle\theta_e, w\rangle$ using the loss function shown in (3), with the complete model structure illustrated in Fig. 5

$$\ell\left(\theta_e, \omega\right) = \left\|\phi_{\theta_e}\left(o_t\right)\cdot\omega - R_t^h\right\|^2 \tag{3}$$

where R_t^h represents high-level reward.

The feature vectors output by the dual-trained encoder network represent the low-dimensional embedding of states. In the actual GMAH training, this type of autoencoder is pretrained. During the high-level policy training of the agent, the agent's observations at each moment are input into the encoder, recording the feature vectors $f_t = \phi_{\theta_e}(o_t)$, $f_{t+1} = \phi_{\theta_e}(o_{t+1})$ for consecutive time steps. The cosine similarity between these feature vectors is calculated as shown in (4). Cosine similarity, a measure of the degree of similarity between the directions of two vectors, is determined by the cosine of the angle between the vectors

$$s = \frac{f_t \cdot f_{t+1}}{\|f_t\| \cdot \|f_{t+1}\|}. \tag{4}$$

When s falls below a threshold ε_1, it is deemed that the encoder network has captured a significant state change, prompting the second stage of the process. During this stage, the observations at the current and previous moments, o_t and o_{t+1}, are input into the high-level policy network to obtain two subgoal probability distributions p_t and p_{t+1}. The KL divergence between these distributions is calculated as shown in the following equation:

$$d_{KL} = H\left(p_t\left(g\right), p_{t+1}\left(g\right)\right) - H\left(p_t\left(g\right)\right)$$

Algorithm 1 Adaptive Subgoal Update Policy

Input: Encoder ϕ_{θ_e}, Decoder ϕ_{θ_d}, high-level policy π^h, low-level policy π^l, Agent set I_N, Target update interval c
Output: Subgoal distribution p

```
Initialize θ_e, θ_d, ε_1, ε_2
for t in 0, ..., T − 1 do
    for agent i in I_N do
        if t mod c ≡ 0 or agent achieve g then
            Subgoals are obtained from observations, that is, g ← π^h(o_t)
        end if
        Perform the action a_t ← π_l(o_t, g), obtain new observation o_{t+1}, reward r_t
        Collect data ⟨o_t, a_t, g, o_{t+1}, r_t⟩
        Calculate the similarity of state feature vectors
        s = φ_{θ_e}(o_t)·φ_{θ_e}(o_{t+1}) / (‖φ_{θ_e}(o_t)‖·‖φ_{θ_e}(o_{t+1})‖)
        if similarity s < ε_1 then
            Calculate the KL divergence of the high-level policy distribution
            d_KL(p_t ‖ p_{t+1}) = Σ_g p_t(g) log(p_t(g) / p_{t+1}(g))
            if d_KL > ε_2 then
                update g' ← π^h(o_t, I_i)
            end if
        end if
        Update Encoder-Decoder
    end for
end for
```

$$= \sum_{g} p_t(g) \log\left(\frac{p_t(g)}{p_{t+1}(g)}\right) \tag{5}$$

where $H(p_t(g), p_{t+1}(g))$ represents the cross-entropy of $p_t(g)$ and $p_{t+1}(g)$ and $H(p_t(g))$ represents the information entropy of $p_t(g)$.

Then, a new subgoal is sampled from the current probability distribution and communicated to the low-level policy.

This setup allows the high-level policy to accurately detect environmental shifts and dynamically adjust subgoals based on state successor features, thus greatly enhancing the adaptability of the hierarchical architecture and reducing potential subgoal conflicts in multiagent environments. The comprehensive algorithmic workflow is detailed in Algorithm 1.

D. Fine-Tuning of the Goal Mixing Network

In communication-limited multiagent cooperation scenarios, a critical issue is the credit assignment problem, which concerns quantifying an individual agent's contribution to the collective outcome. Typically, agents develop an action value function or policy function, selecting actions that maximize either the action value or cumulative reward. Each agent operates based on local observations to secure individual rewards, while the overarching aim is to maximize global rewards. Opting for actions that enhance individual gains may, however, compromise the total global rewards. In scenarios where only individual rewards are accessible, it becomes challenging for agents to make decisions that optimize global outcomes. This dilemma is a central focus of research in MARL.

This article seeks to integrate the proposed hierarchical architecture with established MARL policies to formulate a comprehensive GMAH algorithm, thereby extending the advantages of hierarchical designs to the multiagent context. We integrate the concept of QMIX with a hierarchical architecture. To implement this idea, QMIX uses a special mixing network to ensure that Q_{tot} has a monotonic constraint over Q_a. The general approach is to apply a DRQN network to each agent, constructing a special mixing network that receives the outputs of all DRQN networks as inputs and outputs the joint action value function Q_{tot}.

The mixing network is a two-layer feedforward neural network that takes the output from each agent's network as input and monotonically mixes them to produce the joint action value function Q_{tot}. To ensure the monotonicity constraint, the weights of the mixing network (excluding biases) are restricted to be nonnegative. This allows the mixing network to arbitrarily closely approximate any monotonic function [34], with the weights of the mixing network generated by separate hypernetworks. Each hypernetwork takes the state s_t as input and generates the weights for one layer of the mixing network. Each hypernetwork consists of a single linear layer followed by an absolute value activation function to ensure the output vector is nonnegative, which is then reshaped into an appropriately sized matrix to serve as the weight matrix of the mixing network. The biases for the first layer of the mixing network are generated in the same manner, but are not restricted to be nonnegative. The biases for the last layer are generated by a hypernetwork with a ReLU activation function. It is noted that the state s_t information is only used by the hypernetworks and not directly passed to the mixing network. Typically, for individual agents, we use methods such as temporal difference, as shown in (6) to fit their action value functions $Q(s, a)$ based on rewards

$$\text{Loss} = \underset{s,a}{E}\left[\left(y^{dqn} - Q(s, a; \phi)\right)^2\right] \tag{6}$$

where $y^{dqn} = r + \gamma \max_{a'} Q(s', a'; \phi^-)$, ϕ^- is the parameter of the target network. It is periodically copied from ϕ.

In the hierarchical architecture proposed in this article, the behavior of the subgoals output by the high-level policy can be viewed as the agent's abstract action under high-dimensional temporal sequences. It has a fixed action space (the size of the subgoal set is fixed) and its interaction with the environment conforms to the definition of a Markov decision process. We directly use the high-level policy network to replace the DRQN network in QMIX, changing the training target of the mixing network to the joint goal value function $Q_{\text{tot}}(\mathbf{o_t}, \mathbf{g}|s_t)$, adopting a network design consistent with QMIX to ensure the monotonicity constraint of the joint goal value function, as shown in the following equation:

$$\frac{\partial Q_{\text{tot}}(\mathbf{o_t}, \mathbf{g}|s_t)}{\partial Q_a} \geq 0 \quad \forall a \sim \pi_i^h(o_t). \tag{7}$$

The overall architecture of this goal-mixing network is shown in Fig. 6, where each agent's high-level policy receives local observations and outputs a subgoal. The goal mixing

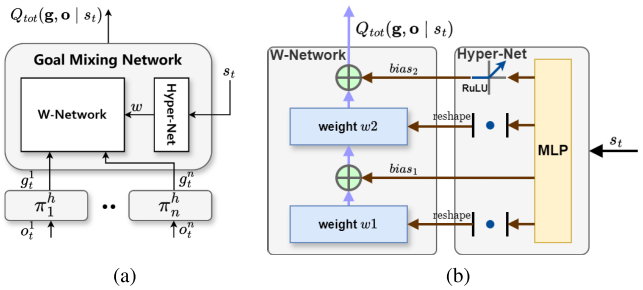


Fig. 6. (a) Goal-mixing network. (b) Hyper-Net and W-network.

network receives the subgoals output by all agents' high-level policy, along with the global state as input, and outputs the joint goal value.

Fig. 6(a) shows the goal mixing network, where the Hyper-Net represents the hypernetwork that generates the weight vector based on the state s_t, and the W-Network represents the weight network composed of reconstructed weight matrices and offsets. The structure of the hypernetwork and weight network is shown in Fig. 6(b). Compared to QMIX, which uses four hypernetworks to separately generate the weight network and offsets for each layer of the mixing network, this article uses only one hypernetwork connected to different fully connected layers with different activation functions to generate the corresponding information. The final goal mixing network is trained according to the loss function shown in the following equation to learn the joint goal value function:

$$\ell(\theta) = \left\| \sum_i r^h + \gamma \max_{\mathbf{g}'} Q_{\text{tot}}^{\theta}\left(\mathbf{o_{t+1}}, \mathbf{g}' | s_{t+1}\right) - Q_{\text{tot}}^{\theta}\left(\mathbf{o_t}, \mathbf{g} | s_t\right) \right\|^2. \tag{8}$$

In summary, the complete GMAH algorithm follows the framework shown in Fig. 1 and is trained according to the following steps.

1) Train the low-level policy π_i^l based on a predefined set of subgoals. Each episode randomly samples a subgoal g from the subgoal set and fixes it, sampling the state transition data of each agent's interactions $\langle o_t, a_t, r_t^l, g, o_{t+1}\rangle$ and training according to the objective function $J(\theta^l) = \max \sum R^l$.
2) Once the low-level policy converges, train the high-level policy π_i^h. Collect data during the interval between two outputs of subgoals by the high-level policy $\langle o_t, g_t, \sum r_t^h, o_{t+c}\rangle$ and train according to the objective function $J(\theta^h) = \max \sum R^h$. Control the subgoal update process using the adaptive goal generation policy.
3) Once the high-level policy tends toward stability, collect joint data at each step $\langle \mathbf{o_t}, \mathbf{g_t}, s_t, \sum r_i^h, s_{t+1}, \mathbf{o_{t+1}}\rangle$ and train according to the loss function (8) to fine-tune the high-level policy.

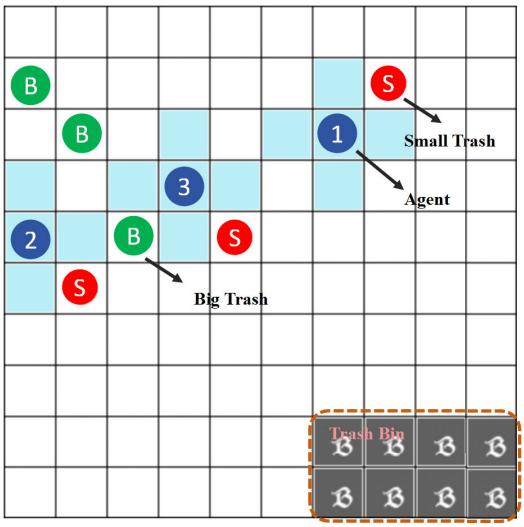


Fig. 7. Tras-Grid environment diagram.

IV. Experiments and Discussion

To verify the performance of the GMAH algorithm, this article selects to conduct experiments in the Trash-Grid [35] environment and the StarCraft II multiagent challenge (SMAC) [36] environment, respectively, to verify the effect of GMAH. The experiments were conducted on a high-performance computational platform equipped with an Intel[1] Xeon[1] Gold 6240R processor (24 cores), 128GB DDR4 RAM, an NVIDIA GeForce RTX 3090 GPU, and running Ubuntu 20.04 LTS, with PyTorch 2.2 as the primary deep learning framework. Our code is open-sourced at https://github.com/SICC-Group/GMAH

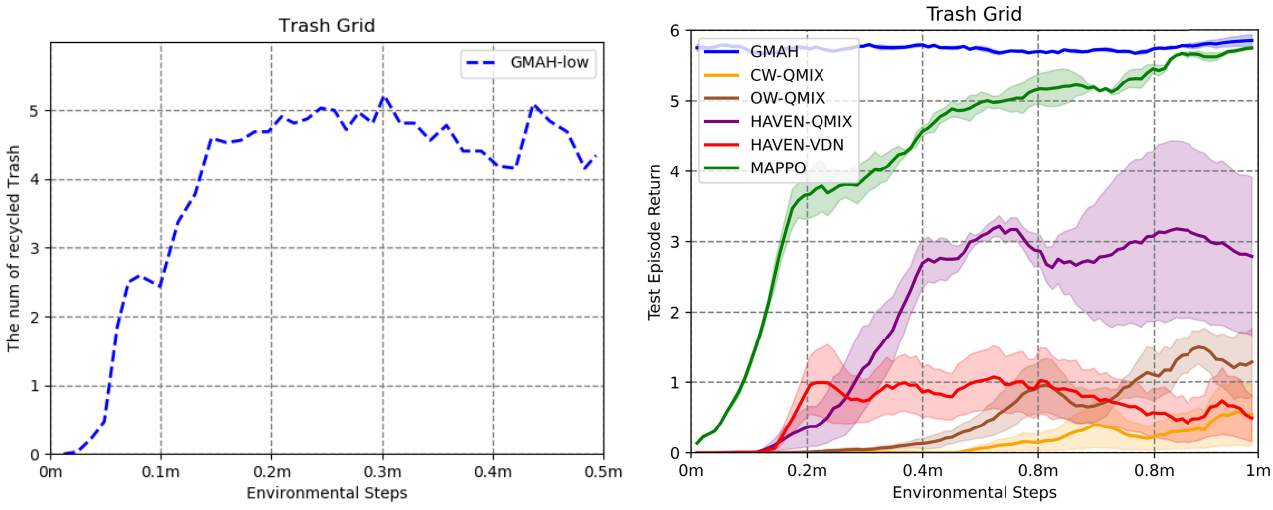
A. Trash-Grid

For multiagent studies involving the GMAH algorithm, we designed a multirobot trash collection environment called Trash-Grid, based on the PettingZoo [37] and Mini-Grid rendering frameworks, which enables sequential execution of multiple agents and updates state information in each cycle.

1) Environmental Setup: The Trash-Grid, as shown in Fig. 7, features a 10 × 10 grid with $N = 3$ agents (robots) oriented downward, randomly distributed with $K1 = 5$ small "trash" items and $K2 = 5$ large "trash" items. A 2 × 4 grid area in the bottom right corner serves as the "recycling station." The agents' task is to transport all trash to the recycling station in the shortest time possible. Each small trash item weighs 1, and an agent can carry up to 3 small trash items simultaneously. Large trash items cannot be carried but must be split into 1 small trash item by the agent. The agents' action space includes behaviors as the sets {0 : Up}, {1 : Down}, {2 : Left}, {3 : Right}, {4 : Pickup}, {5 : Putdown}, and {6 : Split}. The subgoal space for the GMAH algorithm is predefined as shown in the sets {0 : FindTrash}, {1 : PickupsTrash}, {2 : SplitBTrash}, and {3 : PutTrash}.

Agents cannot pass through other agents; collisions result in a negative reward as a penalty. Agents must face the "trash"

[1]Registered trademark.



(a) (b)

Fig. 8. Training result of GMAH, CW-QMIX, OW-QMIX, HAVEN-QMIX, HAVEN-VDN, and MAPPO in Trash-Grid environment with $T = 256$. (a) Result of training the low-level policy of GMAH for 400 000 steps. (b) Experimental results of training 1 000 000 steps by the five methods.

to pick it up. If an agent's load is full, it can still move or split large trash, but cannot carry more. Agents can drop all carried "trash" at any position within the recycling station area and receive a discounted reward r based on the weight of the trash dropped, defined as

$$r_t = \left(1 - \beta \frac{t}{T}\right) \times R \tag{9}$$

where t is the time step of activation, T is the maximum number of time steps, β is the discount factor set to 0.5, and R is fixed at 1, with r_t ranging from 0 to just below 1. Additionally, agents receive a negative reward at each time step as a penalty.

The observation space for agents is a discrete vector of length $N \times 4 + (\mathrm{K1} + \mathrm{K2}) \times 3 + 8 \times 2$, representing the current agent's position, load, orientation, relative positions, loads, and orientations of other agents, and the position and type of all "trash" items. The state space is a 10×10 discrete matrix. Agents do not have access to other agents' actions and goals. The observation space is normalized during actual training.

2) Comparative Experiments: The final set of experiments is conducted at $T = 256$. The GMAH algorithm trains the low-level policy for 400 000 steps, during which subgoals are randomly selected based on prior probabilities, and low-level rewards are assigned upon successful completion of these subgoals. Once the low-level policy is trained, GMAH optimizes its high-level policy to effectively allocate subgoals.

To evaluate the effectiveness of the hierarchical architecture, we compare GMAH against CW-QMIX/OW-QMIX [6], MAPPO [8], and HAVEN-QMIX/HAVEN-VDN [15]. These baseline algorithms use the same reward function as defined in (9) for training. HAVEN-QMIX and HAVEN-VDN, like GMAH, are hierarchical multiagent algorithms. The remaining methods differ from GMAH in that they do not allocate intermediate rewards; hyperparameters are kept as consistent as possible across all implementations. Additionally, CW-QMIX, OW-QMIX, and MAPPO employ a network structure without a goal input module, ensuring comparability with the high-level model of GMAH. The experimental results at $T = 256$ are illustrated in Fig. 8.

As shown in Fig. 8, each algorithm is evaluated using three random seeds, and the mean and standard error for each seed are computed. The low-level policy of GMAH (GMAH-low) successfully recycles more than four pieces of trash on average within 400 000 steps. Building upon this trained low-level policy, the high-level policy of GMAH effectively allocates subgoals, enabling the algorithm to recycle nearly all trash within one million steps. This demonstrates that hierarchical subgoal allocation significantly enhances both performance and efficiency.

Among the comparison algorithms, MAPPO achieves the best performance and is the only method that approaches GMAH's performance at one million steps. The QMIX-based variant of HAVEN outperforms the VDN-based variant, indicating that QMIX's mixing architecture affords greater representational capacity than VDN. Moreover, both CW-QMIX and OW-QMIX underperform HAVEN-QMIX, underscoring the inherent advantage of hierarchical approaches under sparse rewards. HAVEN-VDN exhibits the poorest performance, likely due to VDN's overly simplistic additive decomposition, which severely limits representational capacity and degrades performance.

3) Demonstrating Results: While the final convergence of reward values during training serves as an indicator of algorithm performance, the random placement and varying quantity of trash in the Trash-Grid environment, coupled with time-discounted rewards, introduce significant uncertainty. This variability complicates the analysis of agent behavior based solely on reward data. Moreover, evaluating performance exclusively through final reward values fails to provide insights into the effectiveness of the high-level policy.

To better illustrate the role of the high-level policy, we visualize the intermediate execution process of the GMAH algorithm. Qualitative analysis is conducted based on results from a representative experimental run. Fig. 9 presents a

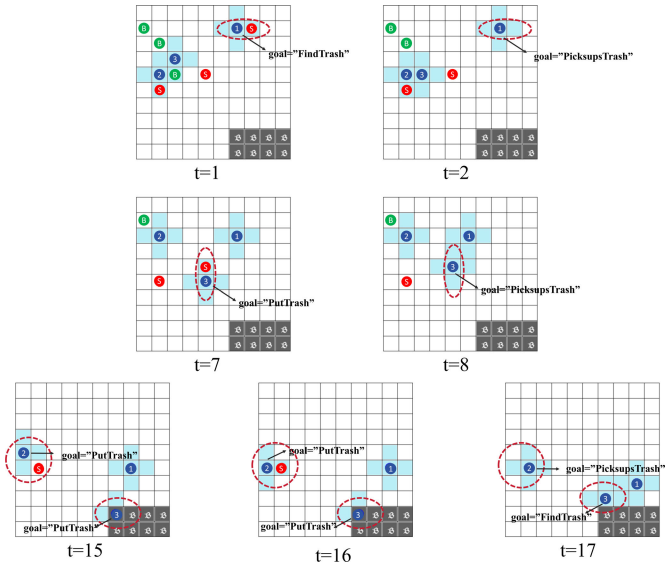


Fig. 9. Example of test results of GMAH in Trash-Grid environment. The figure shows the environment states at seven intermediate moments in a test round. It also shows the real-time subgoal changes of some agents (output of high-level policies.

sequence of seven key moments from a test round, depicting the GMAH algorithm in action within the Trash-Grid environment. These snapshots, spanning from $t = 1$ to $t = 17$, capture the dynamic evolution of the environment's state, providing deeper insights into the operational process of GMAH and the impact of its hierarchical decision-making strategy.

Initially, three agents randomly appear on the map. The goal is to segment the trash and recycle it to the recycling bin in the lower right corner, which is represented in gray. At $t = 1$, the subgoal of agent 1 is FindTrash. When $t = 2$, agent 1 successfully finds the trash. At this time, the high-level policy switches the subgoal to PickupsTrash and guides the low-level policy to successfully pick up the trash. At $t = 7$ and $t = 8$, the same thing happens to agent 3. From $t = 15$ to $t = 17$, the subgoal of agent 2 is originally PutTrash. However, when agent 2 encounters trash on the way to the recycling bin, the high-level policy switches the subgoal of agent 2 to PickupsTrash and makes it successfully pick up the trash. At the same time, under the guidance of the goal of PutTrash, agent 3 successfully reaches the recycling bin and puts down the trash. Then the high-level policy switches the subgoal of agent 3 to FindTrash.

4) Analysis and Discussion: The low-level policy training phase in the Trash-Grid experiments using the GMAH algorithm exhibits a distinct learning curve. In Trash-Grid, the final task necessitates the repeated achievement of subgoals, placing a significant demand on the high-level policy's planning capabilities. The policy that emphasizes iterative planning for trash collection and recycling proves to be more effective than those that directly focus on recycling alone.

However, the low-level policy in the GMAH algorithm demonstrates limited performance in the Trash-Grid environment, managing to recycle an average of only four pieces of trash. This indicates substantial room for improvement. One key challenge arises from the unstable environment, which complicates subgoal completion. Additionally, the low-level policy is solely focused on achieving subgoals, which, while beneficial for structured learning, can hinder overall policy effectiveness by restricting adaptability. This limitation underscores a fundamental challenge of hierarchical architectures: unlike traditional nonhierarchical approaches, the low-level policy must learn specialized behaviors for multiple subgoals, which can be further complicated by the complexity of subgoal inputs. If the subgoals closely resemble the final task, the learning process becomes even more challenging.

In conclusion, the design of an effective subgoal space is crucial to the success of the GMAH method. A thorough analysis of the environment is essential to decompose the main task into well-structured, independent subgoals, ensuring successful task completion while maintaining flexibility in policy execution.

B. SMAC

To further evaluate the GMAH algorithm, we conduct experiments using the StarCraft Multiagent Challenge (SMAC) Benchmark, which presents a more complex and dynamic testing environment. Performance is assessed empirically across different scenarios using first-in–first-out (FIFO) buffers. We select two maps for the experiments: (a) 3m and (b) 2s3z, with the opponent's AI difficulty set to hard in both cases.

In the Trash-Grid experiments, the primary focus was to highlight the advantages of GMAH's hierarchical architecture, specifically how its high-level policy assigns subgoals to guide the low-level policy in completing tasks. However, these experiments did not emphasize multiagent cooperation. To address this limitation, we employ the SMAC benchmark to demonstrate the performance of GMAH in multiagent cooperative tasks.

1) Environmental Setup: We consider the combat scenario of StarCraft II unit micromanagement tasks. In this situation, the built-in AI controls the enemy units, while an RL algorithm controls each allied unit. Units in two groups can consist of different types of soldiers, but the soldiers in the same group should belong to the same race. At each time step, every agent selects an action from the discrete action space, which includes the following actions: no-op, move [direction], attack [enemy id], and stop.

In traditional MARL algorithms, agents execute these actions to move and attack on continuous maps. At every time step, MARL agents receive a global reward equal to the total damage inflicted on enemy units. Additional rewards are granted for key events: killing an enemy unit provides a bonus of ten points; winning the combat yields an additional 200-point reward.

In SMAC, we set the subgoals as the numbers corresponding to the enemies. Taking map 3m as an example, there are three enemies. Then, the subgoal space is predefined as shown in the sets $\{0 : \text{enemy1}\}$, $\{1 : \text{enemy2}\}$, and $\{2 : \text{enemy3}\}$.

For the low-level policy, the reward is set as follows: when the subgoal of an agent is exactly the enemy being attacked by the corresponding policy, the agent will obtain the reward corresponding to the damage dealt to the enemy and the reward for killing the corresponding enemy. For the high-level policy, the reward for an agent is the damage received by

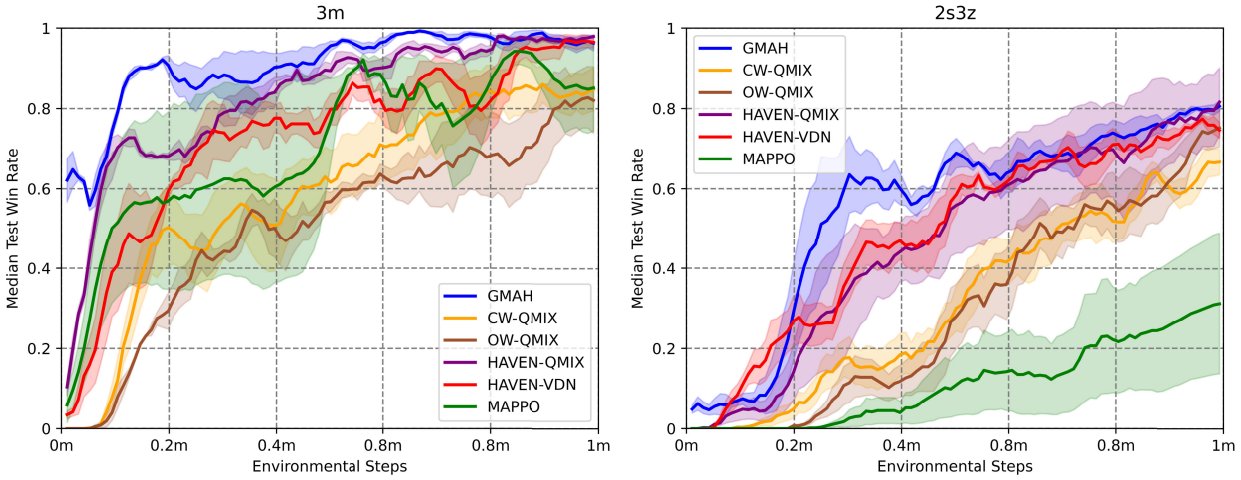


Fig. 10. Training result of GMAH, CW-QMIX, OW-QMIX, HAVEN-QMIX, HAVEN-VDN, and MAPPO in SMAC environment. (a) Median test win rate percentage for the 3m. (b) Median test win rate percentage for the 2s3z.

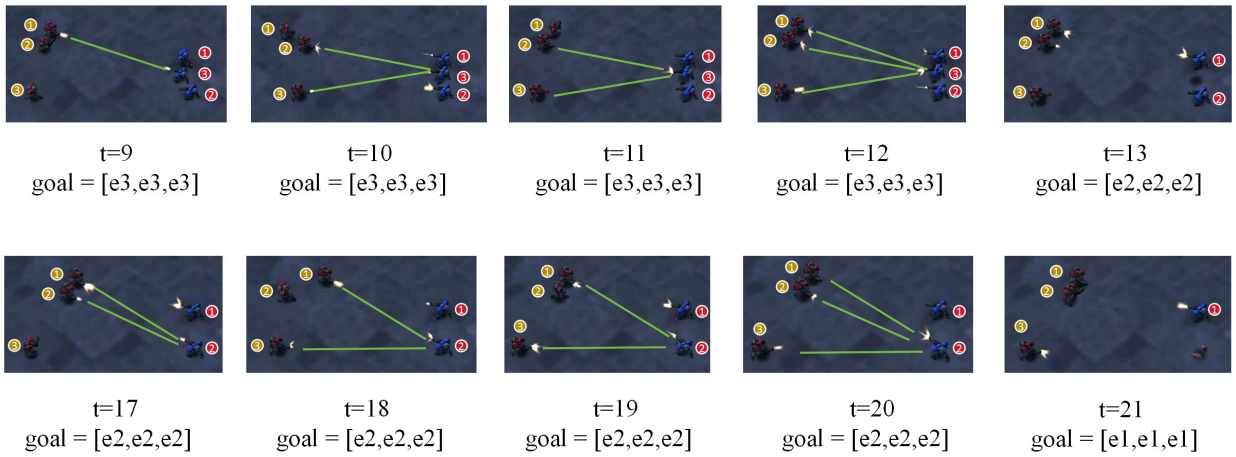


Fig. 11. Visualization of the execution process of GMAH in the StarCraft II mission (3m map). Among them, yellow nodes represent agents, and red nodes correspond to enemies. In 3m, there are three agents and three enemies, and the numbers are correspondingly 1, 2, and 3. The subgoals at the bottom of the figure represent the subgoals output by the high-level policy of GMAH before execution at the corresponding time. $e1$, $e2$, and $e3$ correspond to enemies no. 1, 2, and 3, respectively.

the enemy represented by the corresponding subgoal, and the reward for killing it, regardless of whether the attacker is the corresponding agent. At the same time, the agent can obtain additional rewards for winning.

2) Comparative Experiments: In the SMAC experiments, GMAH trains the low-level policy for 400 000 steps, during which subgoals are randomly selected based on prior probabilities, and rewards are assigned upon subgoal completion. Once the low-level policy is trained, GMAH optimizes its high-level policy to allocate subgoals effectively. The baseline algorithms used in the SMAC experiments are the same as those in Trash-Grid, including CW-QMIX, OW-QMIX, HAVEN-QMIX, HAVEN-VDN, and MAPPO. Except for HAVEN, all other algorithms employ a network structure without a goal input module, ensuring comparability with the high-level models of GMAH. All SMAC experiments are conducted using SC2.4.10, with a discount factor of $\gamma = 0.97$. Every 10 000 steps, we evaluate ten greedy test trajectories with $\epsilon = 0$. Each algorithm is tested using three random seeds, and we compute the mean and standard error for each seed.

The experimental results are presented in Fig. 10. On the 3m map, GMAH ultimately outperforms all other baselines. On the 2s3z map, GMAH performs on par with HAVEN-QMIX and surpasses the remaining baselines. In both scenarios, GMAH consistently demonstrates the highest overall performance, highlighting the algorithm's robustness. While MAPPO exhibits strong winning rates in the 3m scenario, its performance degrades significantly in 2s3z. HAVEN-QMIX and HAVEN-VDN achieve strong win rates on both maps, whereas CW-QMIX and OW-QMIX underperform HAVEN on both, further underscoring the advantages of hierarchical algorithms and the importance of intrinsic rewards. Moreover, HAVEN-QMIX consistently outperforms HAVEN-VDN on both maps, which is consistent with the results in the Trash-Grid environment, validating the effectiveness of the performance gains brought by the QMIX architecture itself. These findings highlight the limited generalization of conventional MARL methods.

Notably, during the early training stages, GMAH significantly outperforms all other algorithms, indicating the

effectiveness of its pretrained low-level policy. Moreover, throughout the high-level training phase, GMAH continues to improve, demonstrating that the high-level policy effectively refines decision-making through continuous interaction with the environment.

3) Demonstrating Results: Unlike the Trash-Grid experiment, where the effect of the high-level policy was less pronounced, its impact is clearly evident in SMAC, as shown in Fig. 10. Regardless of whether the scenario is 3m or 2s3z, training the high-level policy further enhances the algorithm's winning rate, validating its effectiveness. However, visualizing the execution process further clarifies the role of high-level policies and subgoal assignments in multiagent cooperation.

In Fig. 11, we visualized the game screen at some critical moments in the 3m map. In 3m, there are three enemies, so the subgoal space of the agent is $\{\text{enemy1}, \text{enemy2}, \text{enemy3}\}$ (abbreviated as $\{e1, e2, e3\}$). From $t = 9$ to $t = 12$, the high-level policies of the three agents all output $e3$, guiding the low-level policies to focus on attacking $e3$. Eventually, $e3$ is killed at $t = 12$. At $t = 13$, the subgoals of the three agents all become $e2$. In the following five moments, the three agents successfully focus fire and kill $e2$. Finally, at $t = 20$, the high-level policy makes all agents focus on attacking the last enemy.

This analysis demonstrates that the GMAH high-level policy learns to coordinate by achieving consensus in subgoal assignments, thereby improving collaborative execution among agents. Such behavior was not explicitly observed in the Trash-Grid environment, further showcasing the strength of hierarchical decision-making in complex multiagent interactions.

4) Analysis and Discussion: The SMAC experiments reveal a distinct learning curve in the training of GMAH's high-level policy. The low-level policy prioritizes subgoal completion rather than global victory, making the training effect of the high-level policy more evident. This behavior is largely influenced by our reward setting, where global victory rewards are not incorporated at the low level. By decomposing the global objective into hierarchical subgoals, GMAH achieves a higher winning rate through structured goal allocation.

Unlike in trash-grid, where multiagent cooperation was not emphasized, the SMAC experiments clearly illustrate the role of interagent coordination in hierarchical decision-making. The high-level policy effectively directs agents toward a common goal, ensuring strategic target selection and synchronized attacks, ultimately leading to victory. This further highlights the advantages of HRL in complex, multiagent environments.

V. Conclusion

This article introduces GMAH, an HRL algorithm designed for multiagent environments with limited communication. By employing a subgoal-based learning framework and a task-based subgoal generation strategy, GMAH effectively structures decision-making across multiple levels. The integration of a goal mixing network enhances its adaptability to multiagent scenarios, while an adaptive goal generation policy ensures dynamic and flexible subgoal assignment.

Empirical evaluations in Trash-Grid and SMAC environments validate the effectiveness of GMAH. In Trash-Grid, GMAH demonstrates its ability to decompose complex tasks and guide low-level policies efficiently. In SMAC, it further showcases multiagent collaboration, outperforming traditional RL methods in learning efficiency, coordination, and convergence speed, even with a larger model size.

This work establishes a foundation for scalable hierarchical learning in cooperative multiagent systems, providing insights into structured decision-making, adaptive goal allocation, and efficient collaboration. Future research will focus on optimizing low-level policy execution, refining adaptive subgoal updates, and stabilizing high-level training to further improve hierarchical coordination in MARL.

References

[1] J. Hao et al., "Exploration in deep reinforcement learning: From single-agent to multiagent domain," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 8762–8782, Jul. 2024.

[2] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 103–118, Jan. 2024.

[3] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A general navigation model to drive any robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 7226–7233.

[4] A. K. Shakya, G. Pillai, and S. Chakrabarty, "Reinforcement learning algorithms: A brief survey," *Expert Syst. Appl.*, vol. 231, Nov. 2023, Art. no. 120495.

[5] Z. Hu, K. Shukla, G. E. Karniadakis, and K. Kawaguchi, "Tackling the curse of dimensionality with physics-informed neural networks," *Neural Netw.*, vol. 176, Aug. 2024, Art. no. 106369.

[6] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 10199–10210.

[7] K. Son, D. W. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5887–5896.

[8] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative, multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 24611–24624.

[9] T. P. Lillicrap, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[10] D. Yang, H. Zhang, X. Lan, and J. Ding, "Density-based curriculum for multi-goal reinforcement learning with sparse rewards," 2021, *arXiv:2109.08903*.

[11] L. Zheng et al., "Episodic multi-agent reinforcement learning with curiosity-driven exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 3757–3769.

[12] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.

[13] M. Andrychowicz et al., "Hindsight experience replay," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5048–5058.

[14] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[15] Z. Xu, Y. Bai, B. Zhang, D. Li, and G. Fan, "HAVEN: Hierarchical cooperative multi-agent reinforcement learning with dual coordination mechanism," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 10, pp. 11735–11743.

[16] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[17] A. P. Badia et al., "Never give up: Learning directed exploration strategies," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1258–1285.

[18] Y. Chen and H. Kurniawati, "Pomdp planning for object search in partially unknown environment," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 53146–53157.

[19] C. Lu, Q. Bao, S. Xia, and C. Qu, "Centralized reinforcement learning for multi-agent cooperative environments," *Evol. Intell.*, vol. 17, no. 1, pp. 267–273, Feb. 2024.
[20] W. Li, B. Jin, X. Wang, J. Yan, and H. Zha, "F2A2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 24, no. 178, pp. 1–75, 2020.
[21] R. Azzam, I. Boiko, and Y. Zweiri, "Swarm cooperative navigation using centralized training and decentralized execution," *Drones*, vol. 7, no. 3, p. 193, Mar. 2023.
[22] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 2974–2982.
[23] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6379–6390.
[24] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.
[25] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 178, pp. 1–51, 2020.
[26] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.
[27] V. Huiling Wang, T. Wang, W. Yang, J.-K. Kämäräinen, and J. Pajarinen, "Probabilistic subgoal representations for hierarchical reinforcement learning," 2024, *arXiv:2406.16707*.
[28] X. Gao, J. Liu, B. Wan, and L. An, "Hierarchical reinforcement learning from demonstration via reachability-based reward shaping," *Neural Process. Lett.*, vol. 56, no. 3, p. 184, May 2024.
[29] J. Hu, Z. Wang, P. Stone, and R. Martín-Martín, "Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 76529–76552.
[30] S. Zhang, Z. Wang, X.-W. Chang, and D. Precup, "Incorporating spatial information into goal-conditioned hierarchical reinforcement learning via graph representations," *Trans. Mach. Learn. Res.*, Oct. 2025.
[31] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3307–3317.
[32] S. Li, J. Zhang, J. Wang, Y. Yu, and C. Zhang, "Active hierarchical exploration with stable subgoal representation learning," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 5729–5746.
[33] T. Zhang, S. Guo, T. Tan, X. Hu, and F. Chen, "Generating adjacency-constrained subgoals in hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21579–21590.
[34] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. García, "Incorporating functional knowledge in neural networks," *J. Mach. Learn. Res.*, vol. 10, no. 42, pp. 1239–1262, 2009.
[35] D. Furelos-Blanco, M. E. Law, A. Jönsson, K. Broda, and A. Russo, "Hierarchies of reward machines," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 10494–10541.
[36] M. Samvelyan et al., "The starcraft multi-agent challenge," in *Proc. 18th Int. Conf. Auto. Agents MultiAgent Syst.*, 2019, pp. 2186–2188.
[37] J. K. Terry et al., "PettingZoo: Gym for multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 15032–15043.

Cheng Xu (Member, IEEE) received the B.E. degree in swarm intelligence, the M.S. degree in multirobot networks, and the Ph.D. degree in wireless localization and the Internet of Things from the University of Science and Technology Beijing (USTB), Beijing, China, in 2012, 2015, and 2019, respectively.

He was a Visiting Scholar at Université libre de Bruxelles (ULB), Brussels, Belgium, and Nanyang Technological University (NTU), Singapore, from 2023 to 2025. He is currently an Associate Professor at the Data and Cyber-Physical System Laboratory (DCPS), USTB. He was supported by the Post-Doctoral Innovative Talent Support Program from the Chinese Government in 2019. His research interests include swarm intelligence, multirobot networks, wireless localization, and the Internet of Things.

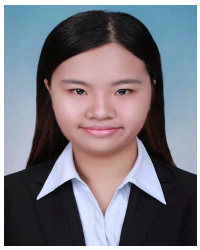
Dr. Xu is an Associate Editor of *International Journal of Wireless Information Networks*.

Yuchen Shi received the B.E. degree in multi-agent reinforcement learning, and the M.S. degree in patter recognition from the University of Science and Technology Beijing (USTB), Beijing, China, in 2021 and 2024, respectively, where he is currently pursuing the Ph.D. degree in the Internet of Things.

His research interests include multiagent reinforcement learning, pattern recognition, and the Internet of Things.

Changtian Zhang received the B.E. degree in multi-agent reinforcement learning, and the M.S. degree in patter recognition and the Internet of Things from the University of Science and Technology Beijing (USTB), Beijing, China, in 2021 and 2024, respectively.

His research interests include multiagent systems, reinforcement learning, and the Internet of Things.



Ran Wang (Graduate Student Member, IEEE) received the B.E. degree in multi-robots network from Beijing Information Science and Technology University, Beijing, China, in 2013, the M.S. degree in quantum optimization, and the Ph.D. degree in distributed security and the Internet of Things from the University of Science and Technology Beijing (USTB), Beijing, in 2016 and 2025, respectively.

She is currently a Lecturer at Beijing Union University, Beijing. Her research interests include multirobot networks, quantum optimization, distributed security, and the Internet of Things.

Shihong Duan received the Ph.D. degree in computer science from the University of Science and Technology Beijing (USTB), Beijing, China, in 2012.

She is an Associate Professor with the School of Computer and Communication Engineering, USTB. Her research interests include wireless indoor positioning, multirobot networks, and the Internet of Things.

Yadong Wan received the B.E. and Ph.D. degrees in computer science from the University of Science and Technology Beijing (USTB), Beijing, China, in 2003 and 2010, respectively.

He is a Professor with the School of Computer and Communication Engineering, USTB. His research interests include wireless sensor networks and computer architecture.

Xiaotong Zhang (Senior Member, IEEE) received the M.S. degree in wireless sensor networks and the Ph.D. degree in computer architecture from the University of Science and Technology Beijing, Beijing, China, in 1997 and 2000, respectively.

He was a Professor at the Department of Computer Science and Technology, University of Science and Technology Beijing. His research interests include wireless sensor networks and computer architecture.