Hierarchical Relational Graph Learning for Autonomous Multirobot Cooperative Navigation in Dynamic Environments

Ting Wang, *Senior Member, IEEE*, Xiao Du, *Student Member, IEEE*, Mingsong Chen, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—As a specific kind of cyber–physical systems (CPSs), autonomous robot clusters play an important role in various intelligent manufacturing fields. However, due to the increasing design complexity of robot clusters, it is becoming more and more challenging to guarantee the safety and efficiency for multirobot cooperative navigation in dynamic and complex environments. Although deep reinforcement learning (DRL) shows great potential in learning multirobot cooperative navigation policies, existing DRL-based approaches suffer from scalability issues and rarely consider the transferability of trained policies to new tasks. To address these problems, this article presents a novel DRL-based multirobot cooperative navigation approach named HRMR-Navi that equips each robot with both a two-layered hierarchical graph network model and an attention-based communication model. In our approach, the hierarchical graph network model can efficiently figure out hierarchical relations among all agents that either cooperate for efficiency or avoid obstacles for safety to derive more advanced strategies, and the communication model can accurately form a global view of the environment for a specific robot, thus, the multirobot cooperation efficiency can be further strengthened. Meanwhile, we propose an improved proximal policy optimization (PPO) algorithm based on the Maximum Entropy Reinforcement Learning, named MEPPO, to enhance the robot exploration ability. Comprehensive experimental results demonstrate that, compared with state-of-the-art approaches, HRMR-Navi can achieve more efficient cooperative navigation with less time cost, lower collision rate, higher scalability, and better knowledge transferability.**

***Index Terms*—Cyber–physical systems (CPSs), deep reinforcement learning (DRL), multirobot cooperation, robot navigation.**

I. Introduction

ALONG with the rapid development of artificial intelligence (AI), Internet of Things (IoT), and cyber–physical systems (CPSs) technologies, multirobot cooperative navigation problem (MRNP) [1], [2] becomes more and more important, since it enables the cooperation among robots to complete complex and cumbersome tasks in logistics systems and manufacturing industry [6]. Essentially, the objective of the MRNP is to ensure that multiple robots can jointly achieve optimal paths toward all respective targets in a quick manner without causing any collisions, where the targets are dynamically assigned in a highly dynamic environment. However, due to the lack of effective design automation methods [3], [4], modern robot cluster-based systems suffer from the inaccurate modeling of interactions between different entities, which strongly affects the quality of robot cooperation in practice [5]. Various approaches have been proposed to address the intractable MRNP, such as simultaneous localization and mapping (SLAM)-based planning [7] and velocity obstacles (VOs)-based velocity selection [8]. However, most of these methods [3], [4], [5], [8] are based on strong premises, such as requiring a priori global knowledge of the environment for path planning. Furthermore, as the number of robots increases, such methods will inevitably face various problems, including the nonstationarity of the environment, and increasingly growing spaces of robots' actions and states.

Deep reinforcement learning (DRL) has been recognized as a promising means to tackle the challenging MRNP [9], [10], [11], [12], [13], [14], [15], [16], [17]. Nevertheless, existing DRL-based solutions have their respective shortcomings. For example, some of them assume that robots make their decisions independently without considering any cooperation with other robots [11], [12], [13], [14]. Moreover, some DRL-based robot navigation approaches rely on simplified collaborative mechanisms, such as parameter sharing policy, to achieve basic cooperative navigation among correlated robots [9]. However, such methods do not allow optimally allocating target locations. Although various works [15], [16], [17] have been proposed recently to enable robots to select targets dynamically or avoid dynamic obstacles during navigation, due to the insufficient modeling of interactions and relations among obstacles and robots, so far their robot exploration ability is strongly limited.

Based on the above observations, this article aims to solve the following four critical problems in multirobot navigation. First, most existing approaches [9], [11], [12], [13], [17] cannot fully represent latent interactions between obstacles, where

Manuscript received 18 August 2022; revised 22 December 2022; accepted 18 March 2023. Date of publication 22 March 2023; date of current version 20 October 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0114600; in part by the Natural Science Foundation of China under Grant 62272170; and in part by the Shanghai Trusted Industry Internet Software Collaborative Innovation Center. This article was recommended by Associate Editor P. Bogdan. *(Corresponding author: Mingsong Chen.)*

Ting Wang, Xiao Du, and Mingsong Chen are with the MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200241, China (e-mail: twang@sei.ecnu.edu.cn; 71194501039@stu.ecnu.edu.cn; mschen@sei.ecnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New York, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TCAD.2023.3260710

they usually model the joint impact of obstacles by simply aggregating the pairwise interactions. Second, most methods fail to model the relations among heterogeneous agents (e.g., moving obstacles and robots) in a complex task [11], [12], [14], [16], [17]. As a result of the above two problems, these methods are limited for scaling up to model a larger number of agents owing to high complexity and potential instability of tasks. Third, most schemes [15], [16], [17], [24] suffer from poor knowledge transferability, where the trained policies are difficult to be transferred to different scenarios with different numbers of robots. Fourth, most existing multiagent RL-based methods are implemented based on the centralized training with decentralized execution (CTDE) framework, which depends on assumptions, such as centralized training and information sharing among all agents [15], [21], [23], [24], [25]. However, this is hard to meet in practical systems and may degrade the system robustness to attacks.

To address the above issues, this article proposes a novel multirobot cooperative navigation framework called HRMR-Navi that focuses on the mobility of obstacles in dynamic environments. In our approach, HRMR-Navi implements both representation learning and policy learning of an agent on top of our proposed hierarchical graph neural network (GNN) model and attention-based communication model. The hierarchical GNN consists of two layers: 1) the interagent layer based on a graph convolutional network (GCN) to effectively model the relations between robots and 2) the intergroup layer based on a graph attention network (GAT) to capture the relations between robots and obstacles. Note that a whole navigation involves a series of cooperation steps among robots. At the beginning of a cooperation step, each robot needs to use its hierarchical GNN to figure out a local state representation by encoding its local observation (i.e., the robot state, and the observed states of neighboring robots and obstacles) into an encoder vector. Then, within the cooperation step, all the robots need to iteratively exchange their local state representations with their neighboring agents to form a global view of the surrounding environment. Note that the above navigation process allows policies to be trained and executed in a fully distributed manner. This article makes the following three major contributions.

1) We propose a novel hierarchical GNN, with which HRMR-Navi can effectively extract the interagent relations based on a GCN, and adaptively reason the intergroup relations based on a GAT to help policies adjust their high-level strategies.
2) We design an attention-based inter-Robot communication model, which can help robots to obtain global information to further facilitate the multirobot cooperation.
3) We propose a novel training algorithm named MEPPO, which is an augmented PPO algorithm with entropy maximization, to enhance the exploration ability of the robots, as well as to enable the robots to obtain a more efficient cooperative navigation path without collisions in complex dynamic environments.

Comprehensive experimental results show that our approach can achieve more efficient navigation policies with better scalability and knowledge transferability compared with state-of-the-art approaches.

The remainder of this article is organized as follows. The related works are briefly reviewed in Section II. Section III provides some background knowledge about GAT and presents the problem formulation. Section IV details the design of our HRMR-Navi approach. Section V presents the performance evaluation results. Finally, Section VI concludes this article.

II. Related Work

A considerable number of methods have been proposed to solve the robot navigation problem, which can be mainly classified into three categories: 1) traditional schemes; 2) DRL-based schemes; and 3) DRL+GNN-based schemes.

A. Traditional Schemes

As one of the representative traditional methods, the Social Force Model [18] has been successfully used in dealing with the robot navigation problem [19]. The work [20] proposes a new concept, named reciprocal VOs (RVOs), for multirobot navigation, which assumes that each robot navigates independently without any communications with other robots, and other robots will make similar collision avoidance behaviors. The work [8] proposes a velocity-based approach for reciprocal n-body collision avoidance, named ORCA, which enables multiple robots to navigate safely in a complex environment without any communications among robots. These model-based methods inevitably suffer from a series of limitations that may lead to freezing behaviors and tedious parameter selection.

B. DRL-Based Schemes

DRL has shown significant potentials in multirobot navigation systems in recent studies. The work [15] employs a multiagent deep deterministic policy gradient (MADDPG) algorithm utilizing CTDE framework to enable multiagent clusters to accomplish a cooperative or competitive task. The work [21] presents counterfactual multiagent (COMA) to address the challenges of multiagent credit assignment, which uses a counterfactual baseline that marginalizes the action of a single agent while keeping the actions of other agents fixed. However, the policies trained with MADDPG and COMA lack the ability of knowledge transferring, and dynamic obstacles are also not considered. Mean Field [22] utilizes the state and mean action of neighboring agents to make decisions. However, it ignores various impacts of neighboring agents. To deal with this shortcoming, the work [23] proposes an attentional communication model, which allows the agent to decide when to communicate with which neighboring agents, aiming to achieve potential cooperation. The work [24] proposes QMIX based on the joint value function, which uses a neural network to estimate the complex relation between the joint value function and the value function of each agent, so that each agent can obtain a cooperative policy with only local observation. The work [25] proposes an actor–critic algorithm that trains decentralized policies with a

soft attention mechanism, which can dynamically select relevant information coming from other agents at every time step. Overall, the above works usually consider the environment as a black box and the learned knowledge cannot be transferred to different scenarios. Recently, the work [26] proposes a method, which combines imitation learning and multiagent RL to learn a policy of decentralization with local observable environments. However, this method needs an expert obtained through centralized planning as the precondition.

C. DRL + GNN-Based Schemes

GNN has attracted great attention in the field of robot navigation due to its capability to model non-Euclidean relations among agents [27]. HAMA [28] employs a hierarchical GAT to obtain information-condensed state representation. However, this method fails to take into account the latent interactions between noncooperative dynamically moving obstacles. Besides, HAMA does not consider the communications between robots, making it difficult to obtain cooperative behaviors among robots. The work [29] uses human maze data to train a GCN structure, so that the robot can obtain similar human attention to obstacles. However, it only considers the path planning of a single robot, and neglects the cooperation between robots. Meanwhile, it also does not consider the attention between obstacles. The work [30] proposes an algorithm called GPS, which fully utilizes the latent graphic symmetry between robots in multirobot scenario. The work [31] constructs an aggregate GNN with time-varying signals and time-varying networks to learn a local controller that only needs local observation and local communication. The work [32] presents a general learning model, which utilizes GNN to extract the local interaction of the robots, so as to learn the distributed policies for the robots. At the same time, imitation experts are also used in this learning method. The work [33] proposes a GNN-based multirobot perception method, which improves the robots' abilities of reasoning perception and fault recovery by nesting the spatial relation between robots in the communication information and adjusting the communication weights with the cross-attention mechanism. The work [34] considers a scenario, where the robots have a limited field of view and cannot access environments beyond their limited field of view. The authors propose a multirobot navigation model, where a convolutional neural network is used to extract the local spatial features of robots, GNN is employed to communicate with neighbors, and long short-term memory (LSTM) is utilized to fuse temporal and spatial information.

Based on the above observations, in this work we innovatively propose a novel hierarchical GNN model (i.e., GCN-based interagent layer and GAT-based intergroup layer) together with a GAT-based inter-Robot communication model for the MNRP. The hierarchical GNN model can help achieve local graph relation representation involving potential relations between agents, leading to more efficient navigation policies, while the GAT-based communication model can further enhance the navigation efficiency by effective communications.

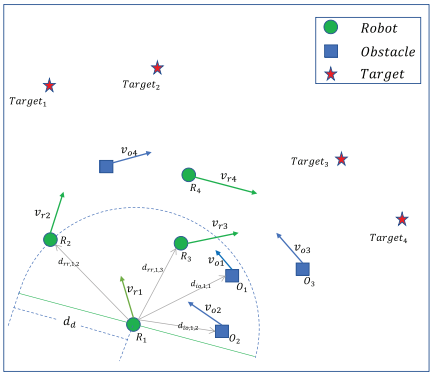

Fig. 1. Illustration of MRNP.

III. Preliminaries and Problem Formulation

A. Graph Attention Network

GAT [38] is a novel convolution-style neural network that can efficiently process graph-structured data. GAT provides an efficient way to compute the node embeddings. In GAT, each node of the graph can assign different weights to its neighboring nodes according to their characteristics. The node embedding h_i can be computed from neighboring nodes $\{\mathcal{N}_i\}$ that are connected to the node i, as $h_i = \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}h'_j)$, where $\mathbf{W} \in \mathbb{R}^{F'\times F}$ is a weight matrix, and h'_i represents state feature of node i. The attention weight $\alpha_{ij} = \text{softmax}_j(e_{ij})$ measures the importance of node j to node i in calculating node-embedding vector h_i, where $e_{ij} = \boldsymbol{a}(\mathbf{W}h'_i, \mathbf{W}h'_j)$, and $\boldsymbol{a}$ is a multilayer feed-forward neural network.

B. Problem Formulation

In this section, we present the problem formulation of MRNP in dynamic and complex environments with moving obstacles. This problem can be regarded as a partially observable Markov decision process (POMDP), where N robots aim to arrive at N targets without collisions within a minimum expected time. The state vector s_t^r of the robot r at time t is composed of the robot's position $[p_t^{rx}, p_t^{ry}]$, speed $[v_t^{rx}, v_t^{ry}]$, and orientation θ_t^r, and is denoted as $s_t^r = [p_t^{rx}, p_t^{ry}, v_t^{rx}, v_t^{ry}, \theta_t^r]$. The observable states $s_t^o = [p_t^{ox}, p_t^{oy}, v_t^{ox}, v_t^{oy}, r_o]$ of moving obstacles at time t include position $[p_t^{ox}, p_t^{oy}]$, speed $[v_t^{ox}, v_t^{oy}]$, and radius of the circumcircle r_o, while the unobservable states $s_t^h = [p_{ogx}, p_{ogy}, v_{\text{pref}}, \theta_t^o]$ consist of goal position $[p_{ogx}, p_{ogy}]$, preferred speed v_{pref}, and orientation θ_t^o. As shown in Fig. 1, the detection range of the robot is d_d, and v_{oi} and v_{rj} denote the speeds of obstacle i and robot j, respectively. The radius of all robots is the same, and the radius of moving obstacles is not consistent. The robot's target position is denoted as $s_g = [p_{gx}, p_{gy}]$. The navigation policy of robots generates the actions a_t at each time step, $a_t \sim \pi_\theta(a_t|s_t^r, \tilde{s_t^r}, \tilde{s_t^o}, s_g)$, where, $\tilde{s_t^r}$ and $\tilde{s_t^o}$ are the states of robots and obstacles observed locally by the robot, respectively, and θ is the parameter of policy. Eventually, the MRNP in dynamic environment can be formulated as

$$\min_{\pi_\theta} \ \mathbb{E}\left[t_{g\max}|\pi_\theta, s_{1:N}^r, s_{1:M}^o, s_{g,1:N}\right] \tag{1}$$

$$\text{s.t.} \ \ \forall i, j \in [1, N], k \in [1, M] \tag{2}$$

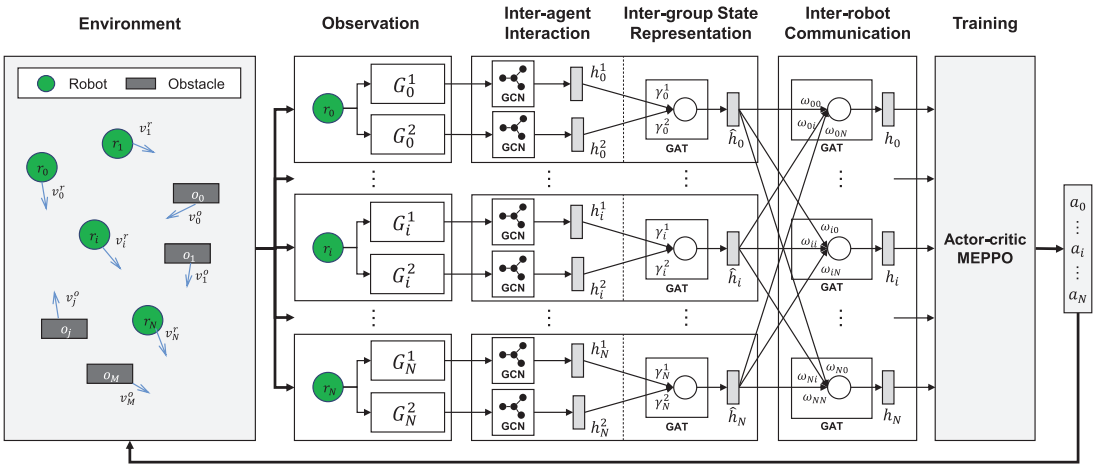

Fig. 2. Overview of our HRMR-Navi framework.

$$d_{rr,i,j} > 2r_r \tag{3}$$

$$d_{ro,i,k} > r_r + r_o \tag{4}$$

$$d_{g,i} < d_{\min} \tag{5}$$

where N and M are the number of robots and obstacles, respectively. $d_{rr,i,j}$ denotes the distance between robot i and robot j, and $d_{ro,i,k}$ represents the distance between robot i and obstacle k. The constraint (3) gives the requirement of avoiding collisions between robots. r_r and r_o represent the radius of the robots and the circumcircle of the obstacles, respectively. The constraint (4) should be met to avoid collisions between robots and obstacles. The distance $d_{g,i}$ in constraint (5) is utilized to determine whether the robot i reaches the target position according to whether it is less than the minimum distance $d_{\min}$. $t_{g\max} = \max_{i\in[1,N]} t_{gi}$, where t_{gi} is the navigation time of robot i. This problem has been proved to be NP complete [16], and we design an efficient DRL-based solution to solve this problem.

IV. HRMR-Navi Approach Design

This section presents the design of our DRL-based HRMR-Navi framework, elaborating the scheme of state representation, the reward function design, and the DRL framework.

A. Overview of HRMR-Navi Framework

HRMR-Navi is a hierarchical graph network representation learning framework, which models all the entities as a graph to reason about the latent relations among all the agents. As illustrated in Fig. 2, the framework represents multirobot environment as a graph, and calculates a node embedding vector for each robot node, which concisely summarizes the state of each robot relative to its local observed environment. When the distance between two robots is less than a predefined threshold, the two robots communicate with each other so as to exchange their node embedding vector information. After multihop communications, each robot will have an updated encoding. The computed encoding is subsequently utilized to generate the Q-values and actions. The model is trained in an end-to-end manner using our newly proposed MEPPO algorithm, which is a proximal policy optimization (PPO) algorithm based on the maximum entropy reinforcement learning.

B. State Representation

The hierarchical graph network is employed to transform each robot's local observation into a high-dimensional node state encoding based on potential feature of agents to represent the interaction among agents (robots and obstacles). After the robots obtain the local state encoding, a communication model is then employed to help robots obtain effective global information to enhance the cooperation performance.

1) GCN-Based Interagent Interaction: First, the agents observed by robot i are clustered into two distinct groups G_i^k ($k = 1, 2$), namely, robot group and obstacle group. The interactions between observed obstacles or robots play an important role in robot navigation. In particular, the robots should actually consider the potential interaction between moving obstacles on the planning path, however, moving obstacles are noncooperative. Therefore, we try to employ GCN to obtain a high-level robot state representation considering the relation between moving obstacles through multilayer convolution. We model the environment as a graph to infer the relations among all agents and use a GCN to reason about the interaction of agents. We model robots and obstacles as a directed graph, where a graph edge represents the amount of attention between two agents. Note that this paired relation is an unknown priori, so a paired similarity function can be used to infer such relation. After inferring the relation among agents, the GNN spreads the information among the observed agent nodes to calculate the state representation of agents.

We use two multilayer perceptrons (MLPs) $f_r()$ and $f_o()$ to embed the states of robots and moving obstacles into an embedding vector. Then, an adjacency feature matrix A is defined, where the first row is the embedding vector of the robot and the remaining rows are the embedding vectors of observed obstacles or robots. For matrix A, an embedded Gaussian as the similarity function [39] is leveraged to compute the matrix relation of agents. The pairwise relation of agents is represented by

$$f(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)} \tag{6}$$

where $\theta(x_i) = W_\theta x_i$, $\phi(x_j) = W_\phi x_j$, $x_i = A[i:]$ is the state embedding vector of agent i, and W_θ and W_ϕ are learnable parameters. The relation matrix of agents is given as

$$R = \text{softmax}\left(AW_\theta W_\phi^T A^T\right). \tag{7}$$

Subsequently, GCN is utilized to compute interaction features among agents. The formula of feature interaction between agents is given as

$$H^{l+1} = \sigma\left(RH^l W^l + H^l\right) \tag{8}$$

where W^l is the learned parameter matrix, σ is an activation function, and H^l is the feature matrix of layer l. The features of the $l+1$ level node i are weighted to aggregate the features of the l level adjacent nodes according to the relation stored in the matrix R. Given $H^0 = A$, after L layer propagates, we obtain the state encoder matrix H^L. The first row of H^L is the group-level node embedding vector h_i^k for robot i of group k encoding its local interactions.

2) GAT-Based Intergroup State Representation: After obtaining node-embedding vectors h_i^1 for robot group and h_i^2 for obstacle group of robot i, the information-condensed state representation of robot i is represented as

$$\hat{h}_i = \sum_{k=1}^{2} \gamma_i^k h_i^k \tag{9}$$

where $\hat{h}_i$ contains the relations between robot i and its locally observed agent groups. The intergroup attention weight γ_i^k instructs robot i to pay more attention to the obstacle agent or the robot agent to achieve cooperative navigation. The intergroup attention weight is computed with the softmax function

$$\left(\gamma_i^1, \gamma_i^2\right) \propto \exp\left(\left[\left(\mathsf{h}_\mathsf{i}^1\right)^\mathsf{T} \mathsf{W}_\mathsf{k}{}^\mathsf{T} W_q W_\theta x_i, \left(\mathsf{h}_\mathsf{i}^2\right)^\mathsf{T} \mathsf{W}_\mathsf{k}{}^\mathsf{T} W_q W_\theta x_i\right]\right) \tag{10}$$

where x_i is the state vector of robot i, and W_q, W_k, and W_θ are the *query*, *key*, and *value* parameters, respectively.

The hierarchical state representation, combining interagent and intergroup relations, greatly improves the efficiency of multirobot cooperative navigation in dynamic environments, especially, when robots need to consider cooperation and obstacle avoidance at the same time. This will be empirically proven by the experiments in Section V.

3) Inter-Robot Communication: Due to the local observability of the environment, the effect of one robot's decision making will be negatively affected by its unobservable robots, resulting in the relatively low performance of the robot's decision-making system. Therefore, the way of information sharing between robots becomes essential to the cooperation performance of the whole system. Each robot needs to be able to accumulate information from its neighboring robots. Thus, a reasonable communication strategy rationally becomes the key to information sharing. The local encoding $\hat{h}_i$ computed by the hierarchical relational graph network represents the robot's understanding of its own state and the observed environment. The attention-based message propagation mechanism of our HAMR-Navi works as follows. First, each robot i computes a query vector $Q_i = W_Q \hat{h}_i$, a key vector $K_i = W_k \hat{h}_i$, and a value vector $V_i = W_v \hat{h}_i$, where W_Q, W_k, and W_v are learnable parameters. After receiving query-value pair (Q_j, V_j) from all of its neighbors $j \in \mathcal{N}(i)$, robot i assigns weight $w_{ij} = \text{softmax}([Q_j(K_i)^T]/d_k)$ to the exchanged information from its neighbor robots, where d_k is the dimensionality of the key vector. Afterwards, all the received messages are aggregated by computing a weighted sum of its neighbors' values, and then executing a linear transformation $V_i = W_{\text{out}} \Sigma w_{ij} V_j$, where W_{out} is also a learnable parameter. Eventually, the robot updates its node embedding h_i by using a neural network to conduct a nonlinear transformation of its current encoding $\hat{h}_i$ concatenated with V_i and targets state encoder, which is obtained by encoding the target state with two MLPs.

The attention mechanism described above enables the agent to selectively pay attention to the messages received from its neighbors. Multihop communications are utilized to achieve the spread of information between robots that might not be directly connected with each other, owing to that the agent (robot and moving obstacle) network may be sparsely connected. After multihop communications, each robot has an updated node embedding h_i. Afterwards, this node embedding is fed into two neural networks to predict the state value of robot and the probability distribution of all possible behaviors, respectively. Ultimately, each robot extracts an action from the distribution and takes this action. Algorithm 1 describes the whole working procedure of the state representation scheme.

Algorithm 1: State Representation

```
for robot i = 1, 2, ... do
    Cluster the agents observed locally into robot group G_i^1 and obstacle group C_i^2;
    Get high level local state h_i^1 and h_i^2 using GCN;
    Calculate state representations h_i^1 and h_i^2 considering the relation between observed obstacles and robots using GCN, respectively;
    Get high level local state representation ĥ_i using GAT;
end
for robot i = 1, 2, ... do
    Compute global state representation h_i of robot i through attention-based inter-Robot communication scheme, where the weights are obtained via GAT;
end
Return h = {h_1, h_2, ...};
```

C. Reward Function

Maximizing the reward of a series of actions is the ultimate goal of the DRL algorithm. An appropriate reward function should be designed in line with the specific task so as to ensure the task to be well accomplished. The multirobot group not only needs to meet these constraints to arrive at the targets as soon as possible but also needs to be collision free. Meanwhile, it is also necessary to promote the cooperation of multirobot groups. To this end, for robot i at time t, we

design an augmented reward function $R_i^t = R_{e,i}^t + R_{c,i}^t$, where $R_{e,i}^t$ is the reward from environments, and $R_{c,i}^t$ is the punishment in case of collisions between robots or robot obstacle. $R_{e,i}^t$ is given as

$$R_{e,i}^t = \begin{cases} 1.0 - \alpha \frac{t}{t_{\text{limit}}}, & \text{if} \quad p_i = p_g \\ -d_{\text{all}}^t + \beta\left(\text{goal}_{\text{dist}}^{t-1} - \text{goal}_{\text{dist}}^t\right), & \text{otherwise} \end{cases} \tag{11}$$

where t_{limit} is a time limit for arriving at destination. In order to encourage robots to reach the targets as soon as possible, this reward will decay monotonically over time. d_{all}^t is the sum of the distances from all the robots to their targets at time t. $\text{goal}_{\text{dist}}^t$ is the average of the total distance of all robots from their targets at time t. β is a hyperparameter. $R_{e,i}^t$ endows robots a positive or negative reward according to the upcoming $\text{goal}_{\text{dist}}^t$ closer or further to their goals, respectively.

On the other hand, $R_{c,i}^t$ is defined as follows:

$$R_{c,i}^t = \begin{cases} -0.25, & \text{if} \quad d_{rr} < 2 * r_r \\ -0.25, & \text{if} \quad d_{ro} < r_r + r_o \\ \eta\left(d^t - d_{\text{disc}}\right), & \text{if} \quad d^t < d_{\text{disc}} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where η is a hyperparameter, d_{rr}, and d_{ro} are used to determine whether robot i collides with other robots or obstacles, d^t represents the distance from robot i to its target at time t, and d_{disc} represents the minimum discomfort distance between robots and between robots and obstacles, which is utilized to encourage earlier collision avoidance.

D. Deep Reinforcement Learning Framework

The local encoding represents the robot's understanding of its own state and the observed environment, which is computed by a hierarchical relational graph network. Robots communicate with each other using GAT [38] only when their distance is smaller than a predefined threshold. The GAT enables the robots to selectively process messages coming from its neighbors. The model is trained in an end-to-end manner using the actor–critic policy gradient PPO algorithm. Notably, our proposed model has a specific distributed feature, where it can be trained and executed in an absolutely decentralized manner. Compared with the latest PPO2 algorithm [40] which uses the entropy as a regularizer to augment exploration, in our approach, a better way is drawn on the maximum entropy framework to provide a substantial improvement in exploration and robustness.

1) MEPPO Algorithm Design: The maximum entropy reinforcement is different from the standard maximum expected reward utilized in legacy reinforcement learning to generalize the standard objective with an entropy term, such that the optimal policy additionally aims to maximize its entropy at each visited state [36]

$$\pi^\star = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma\left(r(s_t, a_t, s_{t+1}) + \alpha H(\pi(.|s_t))\right)\right] \tag{13}$$

where γ is a discount factor, α is the temperature parameter, and $H(\pi(.|s_t))$ is the entropy of policy distribution in s_t state.

To enhance the exploration ability of the robots, we propose an improved PPO algorithm based on the Maximum Entropy Reinforcement Learning, named MEPPO. First, we define the policy gradient loss and value loss. In the Maximum Entropy Reinforcement Learning framework, the reward $r = r^{\text{ex}} + r^{\text{in}}$ is composed of r^{ex} and r^{in}, where the external reward r^{ex} is empowered by the environment and the internal r^{in} is defined in line with the policy entropy. The value estimation needs to minimize a square-error loss

$$L_t^v(\psi) = \left(V_\psi(s_t) - V_t^{\text{targ}}\right)^2 = \left(V_\psi(s_t) - \hat{R}_t\right)^2. \tag{14}$$

The advantage estimator is given as

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t-1}\delta_{T-1} \tag{15}$$

where

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{16}$$

$$r_t = r_t^{\text{ex}} + r_t^{\text{in}} = r_t^{\text{ex}} + \alpha H(\pi(.|s_t)). \tag{17}$$

Then, the policy gradient loss of the MEPPO algorithm can be computed as

$$L_t^\pi(\theta) = \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\hat{A}^{\pi_{\theta'}}(s_t, a_t), f\left(\rho, \hat{A}^{\pi_{\theta'}}(s_t, a_t)\right)\right) + \alpha H(\pi(.|s_t)) \tag{18}$$

where

$$f(\rho, A) = \begin{cases} (1+\rho)A, & \text{if} \quad A > 0 \\ (1-\rho)A, & \text{otherwise.} \end{cases} \tag{19}$$

The descriptions of all notations mentioned above are summarized in Table I.

TABLE I
SUMMARY OF NOTATIONS

Notations	Description
$V_\psi(s_t)$	Value function estimated by neural network
$\hat{R}_t$	Returns obtained from the environment
V_t^{targ}	Equivalent to $\hat{R}_t$
$\hat{A}_t$	Advantage function value
$\hat{A}^{\pi_{\theta'}}$	Advantage function value of policy $\pi_{\theta'}$
$H(\pi(.\|s_t))$	Entropy of policy distribution in s_t state
γ	Discount factor
α	Temperature parameter utilized to tradeoff the importance of entropy and reward.
ρ	Ensure that the gap between $\pi_\theta(a_t\|s_t)$ and $\pi_{\theta'}(a_t\|s_t)$ is small

2) Training Algorithm Design: Since each robot processes incoming messages from other robots in a different way and receives different observations, they may take different actions even if they share parameters. Furthermore, robots in this MRNP are homogeneous, thus, a common policy can be shared. Thereby, we develop a common navigation policy for all robots. The navigation policy is modeled as a combination of a collision-free policy and a dynamic target selection policy. As these two policies are coupled, thus, they cannot be trained separately. Therefore, we propose a DRL method to learn them at the same time. We apply the MEPPO, an actor–critic method to learn the navigation policy, in consideration of the continuous action space and dynamic complex environment. The training algorithm is described in Algorithm 2.

Algorithm 2: MEPPO Algorithm for N Robots

Initialize policy parameter θ;
Initialize value-function parameters ψ;
Initialize Temperature α;
for each episode $\in [1, ...n]$ **do**
  Reset the environment with the initial state, s_{init};
  for each *step* **do**
    Compute high level state representation h using Algorithm 1;
    for *robot* $i = 1, 2, ...$ **do**
      Receive state s_i^t;
      Sample action $a_i^t \sim \pi_\theta(a_i^t|s_i^t)$;
      Execute action a_i^t to robot i;
      Collect state s_i^{t+1}, reward r_i^t and action a_i^t;
      Compute rewards-to-go $\hat{R}_i^t$;
      Compute advantage estimates $\hat{A}_i^t$;
    end
  end
  Optimize policy loss $L^\pi(\theta)$ via Equation 18, with Adam optimizer and learning rate l_a;
  $\theta \leftarrow \theta - l_a \nabla_\theta L^\pi(\theta)$;
  Optimize value loss $L^v(\psi)$ via Equation 14, with Adam Optimizer and learning rate l_v;
  $\psi \leftarrow \psi - l_v \nabla_\psi L^v(\psi)$;
end

V. Performance Evaluation

To validate the effectiveness of our proposed approach, in this section, we implemented a prototype of the HRMR-Navi framework on top of the OpenAI Gym [41] under dynamic environments. Besides, we conducted extensive experiments to compare HRMR-Navi with state-of-the-art methods, and also conducted a series of ablation experiments to verify the necessity of key components of HRMR-Navi.

A. Experimental Settings

The experiments were conducted in a simulated dynamic environment with different numbers of moving obstacles and robots within a square area. Fig. 3 shows an example of a square crossing scenario with three robots and three moving obstacles, which are randomly positioned on a 3 m × 3 m square with a random perturbation added to their (x, y) coordinates. The obstacles are controlled by ORCA [8]. To introduce diversity, the parameters of ORCA are sampled from the Gaussian distribution. The maximum speed v_{pref} of robots is set to 0.2 m/s. The rotation angles of robots are between $-60°$ and $60°$. Each robot is controlled by the policy trained via HRMR-Navi. In addition, to fully evaluate the performance and generalization of our approach, the robots are assumed to be invisible to obstacles. Therefore, the simulated obstacles react only to obstacles but not to the robots. The navigation policy is trained using Algorithm 2 on a virtual machine with a 2.30 GHz Xeon CPU and a Nvidia Tesla P100 graphics card on the Google Colab platform. Both the robot encoder f_r and the obstacle encoder f_o take the 4-D states as input, and the

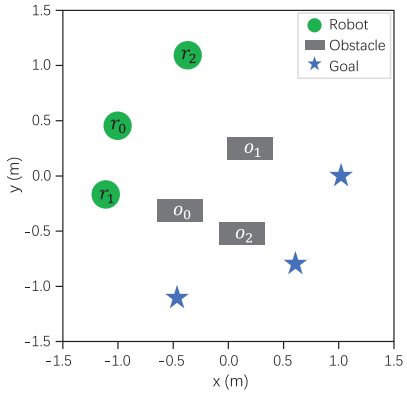

Fig. 3. Example of multiagent dynamic environment.

output dimension is 128. Each encoder is a single ReLu fully connected layer. The output dimension of W^l is 128. When communicating with neighbor robots, the attention module uses 128-D queries, keys, and values to calculate the attention. Adam optimizer is used to train all parameters. The learning rate l_a and l_v are set to 0.001. The discount factor γ is set to 0.9. Each episode lasts up to 320 timesteps. Each MEPPO update is executed after accumulating experience for 640 timesteps on 24 parallel processes.

B. Baseline Algorithms

To verify the superiority of our approach, we implemented three state-of-the-art approaches, i.e., *ORCA* [8], *GA3C-CADRL* [12], and *TA-Policy* [17] for multidimensional comparisons. In addition, we implemented two ablation methods: 1) a trimmed HRMR-Navi algorithm without hierarchical GNN and 2) HRMR-Navi using PPO2 as the training algorithm to conduct ablation experiments for comparisons.

1) *ORCA*, which is a popular VO-based policy used as a baseline algorithm for performance comparison.
2) *GA3C-CADRL*, which is a reinforcement learning-based algorithm and utilizes LSTM to encode observations of an arbitrary number of other agents breaking the limitation of fixed observation size.
3) *TA-Policy*, which is also a learning-for-communication approach like our approach and is designed to process only local observation during the execution phase. Another reason we choose TA-Policy as the baseline algorithm is that, similar to our approach, TA-Policy also considers both the mobility of obstacles and the cooperation of robots at the same time.
4) *HRMR-Navi (non-hierar)*, which is the ablation method without a hierarchical GNN. Specifically, compared with the method HRMR-Navi, HRMR-Navi (non-hierar) does not utilize GAT to calculate the agent intergroup relation.
5) *HRMR-Navi (ppo2)*, which is the ablation method using PPO2 as the training algorithm.

C. Experimental Results

To fully evaluate the performance of the proposed approach, we conduct a task, in which n robots cooperatively reach n different targets without collisions. The initial positions of

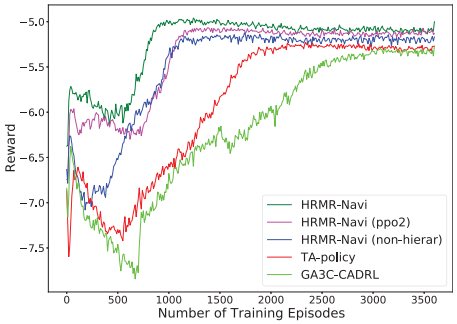

Fig. 4. Comparison of mean episode rewards of HRMR-Navi and its variants with the other two RL methods.

robots, targets, and obstacles are all randomly generated. The evaluation results will be demonstrated with respect to various performance metrics, including convergence speed, success rate, mean episode reward, and extra time.

1) *Convergence Speed:* The rate of convergence to the minimum return during training.
2) *Mean Episode Reward:* Average rewards on 24 parallel processes per episode.
3) *Success Rate:* The ratio of successful cases without any collision or being stuck somewhere within the maximum time step during the navigation.
4) *Extra Time:* The difference between the average travel time over all robots and the lower limit of the travel time (i.e., the average time cost of going straight toward goal at the preferred speed) [11].

1) Convergence Speed and Mean Episode Reward: Intuitively, a good reward function results in an efficient navigation policy with fewer collisions. If a robot is closer to the target and has fewer collisions, there will be a higher average reward.

Fig. 4 shows the comparison results of the mean episode rewards (averaged per ten episodes) of HRMR-Navi, HRMR-Navi (non-hierar), HRMR-Navi (ppo2), TA-Policy, and GA3C-CADRL. In the training phase, as revealed in Fig. 4, our algorithm HRMR-Navi converges faster to the optimal reward. This is mainly due to the fact that the state of each robot is effectively represented by relational graph learning through the hierarchical graph network. HRMR-Navi converges to better rewards faster than HRMR-Navi (non-hierar), which also indicates that hierarchical graph network architecture can effectively facilitate the cooperation among robots while avoiding obstacle. From another perspective, except for GA3C-CADRL, all the other four methods are active-communication schemes. However, with the benefit of attention network our method HRMR-Navi can converge faster, which indicates that the attention-based communication module using GAT can encourage more effective communications among robots leading to a more efficient navigation policy.

2) Success Rate and Extra Time: In order to effectively evaluate the quality of navigation paths planned by various methods in a dynamic environment, we choose two typical metrics: 1) success rate and 2) extra time. The experimental results of four algorithms are summarized in Table II.

TABLE II
SUCCESS RATE AND EXTRA TIME

Metrics	Methods	Number of Robots				
		1	2	3	4	5
Success rate	HRMR-Navi	1	1	1	0.97	0.93
	HRMR-Navi (ppo2)	1	1	1	0.96	0.91
	HRMR-Navi (non-hierar)	1	1	0.98	0.95	0.90
	TA-policy	1	1	0.96	0.94	0.88
	GA3C-CADRL	1	1	0.97	0.92	0.85
	ORCA	1	0.96	0.94	0.89	0.83
Extra time (s) / std	HRMR-Navi	1.15/0.72	1.64/1.45	2.88/1.67	3.76/2.32	5.46/2.13
	HRMR-Navi (ppo2)	1.12/0.80	1.84/1.55	3.12/1.81	3.95/2.63	5.79/2.51
	HRMR-Navi (non-hierar)	1.53/0.79	2.07/1.75	3.52/2.46	4.41/2.72	6.34/3.06
	TA-policy	1.28/0.87	2.63/1.81	4.16/3.15	5.47/2.91	7.52/3.22
	GA3C-CADRL	1.34/0.74	2.52/1.78	4.43/2.82	5.61/3.14	8.05/3.61
	ORCA	1.52/0.96	2.77/1.97	4.55/3.08	6.24/3.52	8.49/3.90

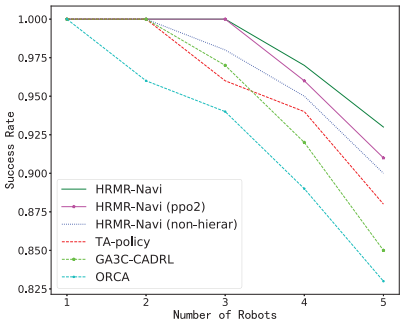

Fig. 5. Comparison of success rate of HRMR-Navi and its variants with the other three methods.

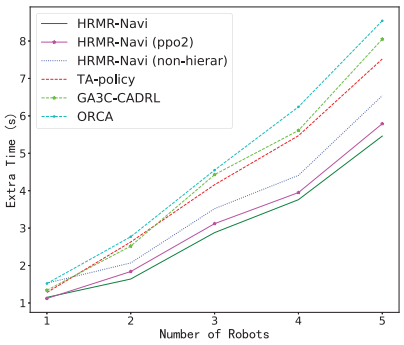

Fig. 6. Comparison of the extra time of HRMR-Navi and its variants with the other three methods.

Figs. 5 and 6 compare our HRMR-Navi with HRMR-Navi (non-hierar), HRMR-Navi (ppo2), TA-Policy, GA3C-CADRL, and ORCA over various policies in terms of success rate and extra time. Fig. 5 exhibits the percentage of successfully completed tasks under different densities of robots. A higher success rate implies that the robots are more likely to avoid obstacles to reach different targets within a specified maximum time range. As revealed in the experimental results, all four algorithms work well for low-density cases, but their performance decreases as the density of agents increases. Nevertheless, as expected, HRMR-Navi achieves the lowest performance degradation rate as the number of robots and obstacles increases. This achievement primarily benefits from the fact that a certain action of the robot is induced at a certain state by analyzing and interpreting the interagent and intergroup attention weights in our algorithm, so that robots can adaptively choose to pay more attention to obstacle avoidance or to cooperation at a certain time.

The robot's full understanding of the relation between agents greatly helps predict the behaviors of other agents and their possible impacts on itself, thus, it can more efficiently generate obstacle avoidance behavior in the appropriate position and reach the globally optimal targets as soon as possible. Moreover, HRMR-Navi achieves a better performance than HRMR-Navi (ppo2), which is due to that the augmented ppo algorithm (MEPPO) with maximizing entropy significantly enhances the robot's exploration ability leading to a better stability. This is connected with a more fundamental merit, where the model trained by HRMR-Navi enables robots to better explore and understand the inherent and high-level connotation of the environment. In addition, GA3C-CADRL only utilizes an LSTM to encode the observations of robots, without an effective reason about the relationship beween agents, so that the behavior selection policy is vulnerable to encounter a freezing point problem.

Fig. 6 presents the comparison results from the perspective of the average extra time taken by the robots to reach targets under different robot densities. A lower extra time reveals that the robots are less likely to get stuck and reach different targets as soon as possible. As shown in Fig. 6, our algorithm HRMR-Navi achieves the best performance with the least extra time. The key reason behind this achievement is owing to that the hierarchical GNN architecture can fully consider the interactions between obstacles and the robots. When there are no obstacles in the direction toward the selected target of the robot, it can go straight to the target. Since our method HRMR-Navi can reason both obstacle–robot interactions and obstacle–obstacle interactions, the trained policies are capable of avoiding obstacles and avoiding unnecessary steering behaviors with as little cost as possible. What is more, the customized MEPPO algorithm can fully explore the environment and enable robots to always find a relatively optimal path, so as to avoid the robots getting stuck. Thereby, the robots will be encouraged to reach their targets as soon as possible. As shown in Table II, compared with baseline algorithms, our HRMR-Navi achieves a lower extra time standard deviation, which also proves the stability of our approach.

3) Knowledge Transferability: To evaluate the knowledge transferability of our approach, we first obtain policies trained for three robots and five robots, and then the trained policies are directly applied to evaluate a new scenario that has a different number of robots, without model retraining. When testing the performance of the policy trained for five robots in some scenarios with more than five robots, the size of the robots and obstacles are reduced to half of the original size. The performance evaluation results are summarized in Tables III and IV, from which we can see that HRMR-Navi significantly outperforms HRMR-Navi (non-hierar). This proves that our hierarchical GNN model greatly enhances the knowledge transferability of the algorithm.

As revealed in Fig. 7(a) and (b), the trained policy shows impressive success rates in robot clusters with different sizes. The results demonstrate that HRMR-Navi is able to utilize its experience obtained from related but different tasks to solve new tasks. Moreover, HRMR-Navi exhibits better transferability compared with the ablation algorithm HRMR-Navi

TABLE III
SUCCESS RATES OF TRANSFERABILITY FOR POLICY TRAINED OF THREE AGENTS (*S%*)

Methods	Number of Robots					
	N-1	N=3	N+1	N+2	N+3	N+4
HRMR-Navi	99	96	92	87	83	48
HRMR-Navi (non-hierar)	97	95	79	61	39	22

TABLE IV
SUCCESS RATES OF TRANSFERABILITY FOR POLICY TRAINED OF FIVE AGENTS (*S%*)

Methods	Number of Robots						
	N-4	N-2	N-1	N=5	N+1	N+2	N+3
HRMR-Navi	89	93	94	93	90	85	82
HRMR-Navi (non-hierar)	76	80	83	85	78	62	49

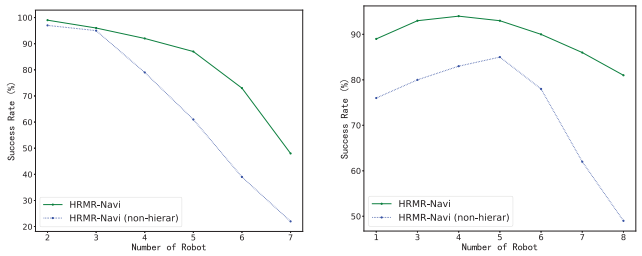

Fig. 7. Evaluation of knowledge transferability: the policies trained for three and five robots are utilized to evaluate a new scenario that has a different number or size of robots. (a) Transferability of the policy trained for three agents. (b) Transferability of the policy trained for five agents.

(non-hierar) as the number of robots increases. This evidently proves that the hierarchical GNN architecture is beneficial to achieving a policy with better transferability, since this hierarchy can be leveraged to obtain the state representation that takes the interactions between agents into account. The transferability exhibited by the policies trained by our method plays a significant role in exploring general policies.

4) Trajectory Analysis: In our HRMR-Navi framework, the robots are expected to achieve two objectives: the first one is to automatically select the targets that minimize the sum of the distances of all robots to their respective targets; and the second one is to select the appropriate linear velocity and angular velocity to reach the targets as soon as possible while avoiding collisions with moving obstacles. Notably, robots are invisible to dynamic obstacles, so the obstacles only respond to obstacles but not to robots. This setting is helpful for a clean test to verify the ability of our method to infer obstacle–robot interaction and obstacle–obstacle interaction without affecting obstacles' behaviors. Note that HRMR-Navi requires no prior knowledge about the dynamic environment, which makes learning an efficient navigation policy a challenging task.

To more intuitively demonstrate the effectiveness of the trained policy, Fig. 8 shows the robots' motion trajectories in environments with different numbers of agents. The simple and direct way to evaluate the efficiency of these path plannings is to compare the indicated time consumed by the robots to reach their goals. Fig. 8 reveals that the robots have

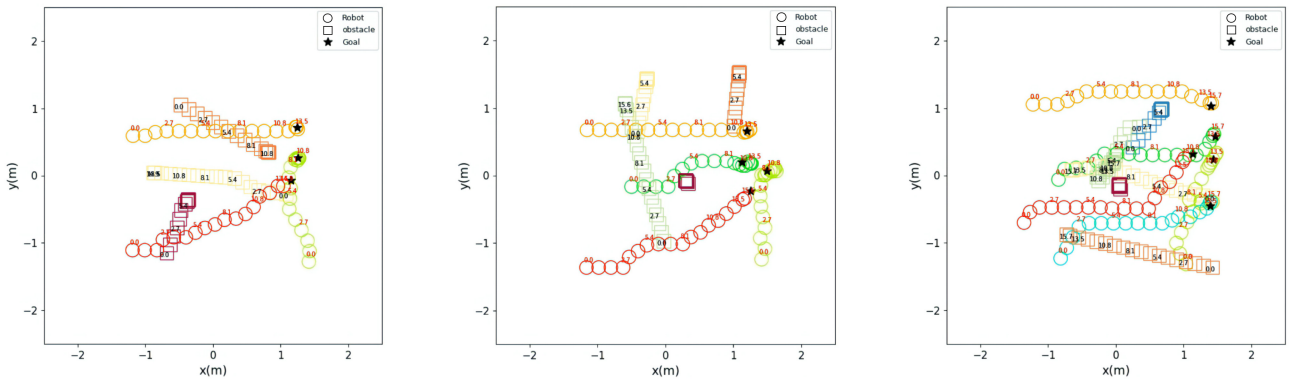

Fig. 8. HRMR-Navi trajectories with $n \in \{3, 4, 5\}$ robots. The circles represent robots, and the red numbers denote the time at robot's position. The squares represent obstacles, and black numbers denote the time at obstacle position. The stars indicate the targets of robots. (a) Trajectories with three robots. (b) Trajectories with four robots. (c) Trajectories with five robots.

successfully learned to cooperate in selecting targets and finding the shortest paths, and also have learned to avoid dynamic obstacles at the same time. As shown in Fig. 8(a), the robots have not selfishly selected the targets closest to themselves, but the targets that can minimize the overall distance of all robots to their targets. For example, the red and yellow robots are farther from all targets than the cyan robot, so they chose the targets closest to themselves. Comparatively, the cyan robot is much closer to all targets than the other two robots, and even the distance between the cyan robot and its farthest target is closer than that between the red robot and its nearest target. Nevertheless, the cyan robot does not selfishly choose its nearest target so as to minimize the overall path length of all robots.

From Fig. 8, we can also observe that all robots have successfully reached the dynamically selected targets within the time limit. Meanwhile, as the trajectory depicts, neither the robots nor the obstacles appear at the same position at the same time, which means that the robots successfully avoid collision with all dynamic moving obstacles. Besides, it also reveals that, as the number of robots and obstacles increases, it is more difficult for robots to avoid other agents (robots and obstacles), resulting in more turns for robots, which will take more time to reach their targets. However, the moving paths of robots are not always absolutely optimal, for instance, the red robot in Fig. 8(b) travels a longer path with some redundant turn actions. This is mainly due to the dynamic selection of targets and the interference of other agents' behaviors, where the target selected by the same robot at different time points may change due to the change of positions of other agents, resulting in redundant turn actions.

Beyond the above, the GCN layer of HRMR-Navi also plays a vital role in guaranteeing the navigation efficiency, where the GCN layer can effectively extract the relation between agents, so that the robots can obtain an information-condensed state representation. To evaluate the effect of the GCN layer, we conduct a group of experiments in a scene with three robots and three obstacles, and depict the navigation trajectories generated by policies trained with and without GCN layers, as shown in Fig. 9. Specifically, Fig. 9(a) and (b) demonstrate

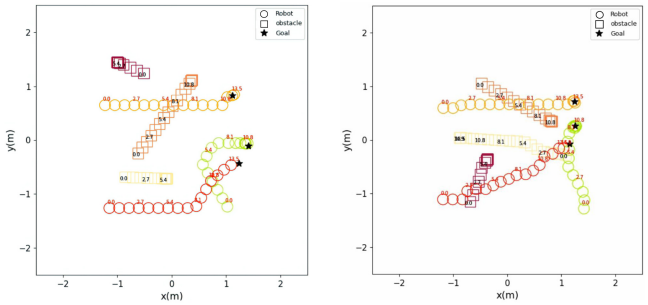

Fig. 9. Trajectories with $L \in \{0, 2\}$ GCN layers. (a) Trajectories of three robots without GCN. (b) Trajectories of three robots with GCN.

the robots' trajectories without GCN layers and with two GCN layers, respectively. It can seen from Fig. 9(a) and (b), policies trained with GCN layers can generate smoother trajectories than policies trained without GCN layers, with fewer turns and trajectory offsets.

Specifically, as shown in Fig. 9(a), the cyan robot deviates to the left abnormally without the threat of collision with other agents, resulting in a longer moving path. Comparatively, in Fig. 9(b), all robots reach their targets more smoothly at a smaller cost without collisions. The key reason behind this achievement is mainly owing to that the GCN layer helps the robots fully understand their relations with other agents so that they can correctly predict the influence of other agents on them and make better behavioral decisions.

5) Performance in Complex Hybrid Environment With Both Dynamic and Static Obstacles: To further verify the performance of our method in more complex environment, we simulate a complex hybrid environment with both dynamic and static obstacles. In this simulation, the number of dynamic obstacles is equal to that of robots. Meanwhile, for each scenario, we set up two additional static obstacles, whose radius is twice that of the dynamic obstacles.

Table V presents the evaluation results of four algorithms in such hybrid environment in term of success rate and extra time. As revealed, all four methods achieve a high success

TABLE V
PERFORMANCE IN HYBRID ENVIRONMENT

Metrics	Methods	Number of Robots				
		1	2	3	4	5
Success rate	HRMR-Navi	1	1	0.99	0.97	0.95
	HRMR-Navi (ppo2)	1	1	0.98	0.95	0.93
	HRMR-Navi (non-hierar)	1	1	0.97	0.94	0.91
	TA-policy	1	1	0.97	0.91	0.87
	GA3C-CADRL	1	1	0.96	0.89	0.85
	ORCA	1	0.98	0.95	0.88	0.83
Extra time (s) / std	HRMR-Navi	2.24/1.15	2.83/1.88	3.25/2.32	4.48/2.67	6.13/2.96
	HRMR-Navi (ppo2)	2.46/1.25	2.97/1.70	3.36/2.36	4.98/2.89	6.42/3.14
	HRMR-Navi (non-hierar)	2.85/1.58	3.37/2.08	4.45/2.75	5.42/3.22	7.18/3.29
	TA-policy	3.15/1.41	3.92/2.61	4.96/2.96	6.13/3.09	8.46/3.47
	GA3C-CADRL	3.08/1.52	3.98/2.75	5.13/3.16	6.75/3.44	8.81/3.68
	ORCA	3.49/1.73	4.28/2.61	5.46/3.52	7.46/3.78	9.56/4.14

TABLE VI
PERFORMANCE OF SCALABILITY

Metrics	Methods	Number of Robots			
		12	15	18	20
Success rate	HRMR-Navi	0.97	0.96	0.93	0.91
	HRMR-Navi (ppo2)	0.96	0.93	0.91	0.88
	HRMR-Navi (non-hierar)	0.92	0.89	0.85	0.82
	TA-policy	-	-	-	-
	GA3C-CADRL	0.82	0.65	0.56	0.50
	ORCA	0.71	0.63	0.53	0.48
Extra time (s) / std	HRMR-Navi	5.86/2.35	7.26/3.10	9.23/3.26	11.09/3.45
	HRMR-Navi (ppo2)	6.04/3.53	7.86/3.42	9.95/3.37	12.18/3.53
	HRMR-Navi (non-hierar)	6.75/2.88/	8.33/3.57	11.79/3.74	14.16/3.82
	TA-policy	-	-	-	-
	GA3C-CADRL	9.57/2.84	11.68/3.93	13.42/3.89	15.18/3.97
	ORCA	11.72/3.42	14.18/4.36	16.58/4.91	18.24/4.28

rate, among which HRMR-Navi achieves the highest, which is up to 9.2%, 11.7%, and 14.5% higher than TA-policy, GA3C-CADRL, and ORCA, respectively. However, as for the performance of extra time, compared with the scenario with only dynamic obstacles (as shown in Table II), the scenario with additional static obstacles takes a longer extra time to complete the navigation task. Intuitively, this is largely due to the interference of static obstacles, where the robot has to detour if the static obstacles are located on its path. Comparatively, dynamic obstacles and robots may arrive at the same position at different times, and if the robot predicts a possible collision, it can slightly slow down to avoid the dynamic obstacle without detouring with less cost. Therefore, in the scenario with only dynamic obstacles, it may take less extra time to complete the navigation task. Nevertheless, our method still significantly outperforms other baseline algorithms.

6) Scalability Analysis: As an important indicator of practicability, the scalability of an algorithm determines its deployability in real-world production environment. To verify the scalalibity performance of HARM-Navi, we conduct a series of experiments in several larger-scale robot network environments with n robots ($n \in \{12, 15, 18, 20\}$) and ten obstacles, where the radius of both robots, each episode lasts up to 400 timesteps and obstacles is halved compared with the previous experiments and the size of the experimental site is expanded to 3.5 m $\times$ 3.5 m.

Table VI presents the evaluation results of four algorithms in terms of success rate and extra time. It can be seen that the policies trained by HRMR-Navi achieve the highest success rate and lower extra time, which are 36.7%–89.6% higher than ORCA, and the more robots the greater the advantage of HRMR-Navi. HRMR-Navi (non-hierar) also exhibits an impressive performance. Comparatively, the method TA-policy fails to converge for all four scenarios, demonstrating a poor scalability. This is mainly due to the fact that TA-policy lacks an effective communication mechanism among robots and cannot effectively capture the interactions among agents. In conclusion, the results prove the convincing performance of our method HRMR-Navi in terms of scalability. This achievement is primarily owning to that hierarchical graph network architecture can effectively help interpret the interactions between agents and the effective attention-based communication model can facilitate the robots obtain more useful information for decision making from other robots.

VI. CONCLUSION

To address the MRNP, this article presented a novel DRL-based multirobot navigation method named HRMR-Navi that can simultaneously learn a dynamic target selection policy together with a collision avoidance policy. By adopting a hierarchical graph network model together with an attention-based communication model for each robot, HRMR-Navi greatly facilitates the communication and cooperation between robots under DRL. Meanwhile, a novel training algorithm MEPPO is proposed to further enhance the exploration ability of robots, aiming to achieve more efficient and stable movement in a complex dynamic environment. Since our approach enables the modeling of both interagent relations between robots and intergroup relations between robots and obstacles, the latent interaction among agents can be fully investigated to benefit the robot navigation. Comprehensive experimental results demonstrated that, compared with state-of-the-art DRL methods, our approach can not only significantly improve both the convergence rate and overall navigation time for various complex environments but also dramatically reduce the collision rate as the number of agents grows. The results also proved that the policies learned by our approach can be well transferred to accommodate different environments for robot clusters with different scales.

REFERENCES

[1] H. Hong, H. Jung, K. Park, and S. Ha, "SeMo: Service-oriented and model-based software framework for cooperating robots," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2952–2963, Nov. 2018.

[2] H. Hamann and A. Reina, "Scalability in computing and robotics," *IEEE Trans. Comput.*, vol. 71, no. 6, pp. 1453–1465, Jun. 2022.

[3] Q. Zhu, A. Sangiovanni-Vincentelli, S. Hu, and X. Li, "Design automation for cyber-physical systems," *Proc. IEEE*, vol. 106, no. 9, pp. 1479–1483, Sep. 2018.

[4] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1421–1434, Sep. 2017.

[5] M. Lukic, A. Barnawi, and I. Stojmenovic, "Robot coordination for energy-balanced matching and sequence dispatch of robots to events," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1416–1428, May 2015.

[6] R. N. Darmanin and M. K. Bugeja, "A review on multi-robot systems categorised by application domain," in *Proc. 25th Mediterr. Conf. Control Autom.*, 2017, pp. 701–706.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[8] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. Int. Symp. Robot. Res. (ISRR)*, 2009, pp. 3–19.

[9] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 6252–6259.
[10] J. Yu et al., "INCAME: Interruptible CNN accelerator for multirobot exploration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 4, pp. 964–978, Apr. 2022.
[11] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 285–292.
[12] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 3052–3059.
[13] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.
[14] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 656–663, Apr. 2017.
[15] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 6382–6393.
[16] Y. Jin, Y. Zhang, J. Yuan, and X. Zhang, "Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning," in *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2019, pp. 2897–2901.
[17] R. Han, S. Chen, and Q. Hao, "Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 448–454.
[18] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, no. 5, pp. 4282–4286, 1995.
[19] G. Ferrer, A. G. Zulueta, F. H. Cotarelo, and A. Sanfeliu, "Robot social-aware navigation framework to accompany people walking side-by-side," *Auton. Robots*, vol. 41, no. 4, pp. 775–793, 2017.
[20] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2008, pp. 1928–1935.
[21] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.
[22] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5567–5576.
[23] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 7265–7275.
[24] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4292–4301.
[25] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2961–2970.
[26] G. Sartoretti et al., "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.
[27] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[28] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2020, pp. 7236–7243.
[29] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2754–2761, Apr. 2020.
[30] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, "Graph policy gradients for large scale robot control," in *Proc. Annu. Conf. Robot Learn. (CoRL)*, 2020, pp. 823–834.
[31] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Proc. Annu. Conf. Robot Learn. (CoRL)*, 2019, pp. 671–682.
[32] L. Zhou et al., "Graph neural networks for decentralized multi-robot submodular action selection," 2021, *arXiv:2105.08601*.
[33] Y. Zhou, J. Xiao, Y. Zhou, and G. Loianno, "Multi-robot collaborative perception with graph neural networks," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2289–2296, Apr. 2022.
[34] J. Ding, L. Zhang, and J. Cheng, "Multi-robot path planning based on spatio-temporal information in large-scale unknown environment," in *Proc. Int. Conf. Mechatronics Mach. Vis. Pract. (M2VIP)*, 2021, pp. 795–800.
[35] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Dept. Comput. Sci., Univ. Washington, Seattle, WA, USA, 2010.
[36] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1352–1361.
[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1856–1865.
[38] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[39] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7794–7803.
[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[41] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

Ting Wang (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2015.

He is currently an Associate Professor with the Software Engineering Institute, East China Normal University, Shanghai, China. His research interests include cloud/edge computing, data center networks, machine learning, and AI-aided intelligent networking.

Xiao Du (Student Member, IEEE) received the B.S. degree in electronic information engineering from Leshan Normal University, Leshan, China, in 2015. He is currently pursuing the Ph.D. degree with the Software Engineering Institute, East China Normal University, Shanghai, China.

His research interests include multiagent reinforcement learning, distributed computing, and machine learning.

Mingsong Chen (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2010.

He is currently a Professor with the Software Engineering Institute, East China Normal University, Shanghai, China. His research interests include cloud computing, design automation of cyber–physical systems, parallel and distributed systems, and formal verification techniques.

Keqin Li (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York at New Paltz, New Paltz, NY, USA. He is also a Distinguished Professor with Hunan University, Changsha, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing.