2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Graph-Based Deep Reinforcement Learning Approach for Alliance Formation Game based Robot Swarm Task Assignment

Yang Lv, Jinlong Lei, Peng Yi

Abstract—**This paper explores the robot swarm task allocation problem based on alliance formation game theory, which treats each robot as an autonomous agent capable of forming strategic alliances for task completion, with a focus on optimizing overall system revenue and mutual benefits. To resolve the problem, we introduce a unique graph-based Deep Reinforcement Learning (DRL) framework named AFGNet_DDQN. Firstly, we construct an Allocation Feature Graph (AFG) that intricately maps the complex interactive relationships and allocation features among robots and tasks, and develop the AFGNet architecture to efficiently extract features from the graph nodes. Then thorugh reconstructing a Markov Decision Process (MDP) within this graph, we implement an advanced version of Double Deep Q-Networks (DDQN) algorithm, adapted for our graph-based framework. This setup allows for the effective learning and optimization of task allocation strategies through localized interactions among the robots. Finally, our empirical results demonstrate the superiority of our framework over traditional methods, especially in terms of scalability and robustness.**

Index Terms—**robot swarm, multi-robot systems, networked robots, dynamic task allocation.**

I. Introduction

With the advances of robot technology, robot swarms comprising a multitude of cooperative robots have become indispensable for addressing intricate tasks across tremendous fields such as mobile sensor networks [1], military air operations [2], and environmental monitoring [3]. The efficiency of robot swarms in these intricate environments is contingent upon the development of refined task allocation methodologies. In particularly, they require not only proficient task allocation strategies to optimize overall system performance but also necessitate rapid response capabilities, so as to enhance the overall efficiency and responsiveness of the system [4]. Consequently, the design of efficient and intelligent task allocation algorithms is a pivotal step in the progression of intelligent system development.

This work focuses on the allocation of tasks in scenarios where robots outnumber tasks, focusing on effectively grouping robots for task allocation. This is typical of Single-Task Robots and Multi-Robot Tasks (ST-MR) systems [5], where each robot is dedicated to a single task, while a task may require multiple robots. The existing methods for the considered problem can be divided into two main branches: centralized and distributed methods [6]. Centralized methods, including optimization and heuristic approaches, are noted for their theoretical ability to provide optimal solutions. Though optimization methods such as exhaustive search and integer programming (see e.g., [7], [8]) are highly effective for small-scale problems, they face significant challenges due to rapidly increasing computational complexity in complex environments as the number of robots increases. In contrast, heuristic methods [9] like genetic algorithms [10] and particle swarm optimization [11] can provide near-optimal solutions for NP-hard problems with lower computational complexity and better adaptability, but these methods may not guarantee global optimality.

With the advancement of drone technology and communication networking, distributed algorithms, particularly top-down and bottom-up strategies, offer effective methodologies to address large-scale robot task allocation problems. Top-down methods, such as market-based auction algorithms [12], [13], optimize task allocation through hierarchical task decomposition and distributed decision-making. On the other hand, bottom-up approaches [14], [15] rely on swarm intelligence and individual agents' local decisions and responses to achieve task allocation. Although distributed methods excel in scalability and robustness, these algorithms are inherently polynomial-time and may not meet the rapid response requirements in large-scale robot swarms. For instance, in search and rescue missions involving robot swarms [16], a swift task allocation response is crucial. Therefore, the primary challenge in designing task allocation algorithms for large-scale robot swarms lies in achieving rapid and adaptive response scheduling, while simultaneous maintaining flexibility and scalability.

The paper was sponsored by the National Key Research and Development Program of China under No 2022YFA1004701, the National Natural Science Foundation of China under Grant No. 72271187 and No. 62373283, and partially by Shanghai Municipal Science and Technology Major Project No. 2021SHZDZX0100.

Yang Lv is with Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University Shanghai 200092, China, (email: 726564418@qq.com).

Jinlong Lei and Peng Yi are with the Department of Control Science and Engineering, Tongii University, Shanghai, 201804, China; The Shanghai Research Institute for Intelligent Autonomous Systems,Shanghai, 201804, China;Shanghai Institute of Intelligent Science and Technology, Tongji University 200092, China, (email: leijinlong@tongji.edu.cn; pengyi@tongji.edu.cn).

To tackle these challenges, our research models the ST-MR task allocation problem as an alliance formation game [17] and introduces an innovative end-to-end Deep Reinforcement Learning (DRL) method for learning efficient allocation schemes. In this model, each robot is an autonomous agent, forming alliances based on individual preferences, aiming for mutual benefits and optimized total revenue. We then construct an allocation feature graph (AFG) for robot swarms, which represents complex features of robots and tasks, and the interactions among them. Drawing from literature [18], we frame a Markov Decision Process (MDP) on this graph and introduce a graph-based DRL method named AFGNet_DDQN. Our approach begins with the AFGNet, a novel network architecture we developed to extract feature embeddings from the graph nodes. It is worth noting that our proposed Graph Neural Network (GNN), AFGNet, is specifically tailored to manage allocation feature graphs in task allocation, capturing both the state of robots and the status of other communicable nodes. Subsequently, we employ an enhanced version of Double Deep Q-Networks (DDQN) that is specifically adapted for our constructed allocation feature graph. Extensive experiments with synthetic benchmarks clearly show that our method outperforms traditional distributed iterative methods in efficiency and decision quality. Notably, our approach exhibits strong generalizability, with models trained on smaller scales adeptly handling larger-scale problem decision-making. This adaptability underscores our method's practicality in swiftly and effectively solving large-scale ST-MR challenges, surpassing conventional methods in rational scheduling capabilities.

In summary, this work introduces a novel end-to-end Deep Reinforcement Learning (DRL) approach to solve the Single-Task Multi-Robot (ST-MR) task allocation problem, modeled as an alliance formation game. The main contributions include: (i) Development of an Allocation Feature Graph (AFG): We construct an AFG for robot swarms to represent the features of robots and tasks and their interactions. This graph serves as a foundational element for our DRL model. (ii) Innovative Network Architecture and Enhanced DDQN Adaptation (AFGNet_DDQN): We design AFGNet, a unique network architecture for extracting feature embeddings from the AFG's nodes, combined with an advanced adaptation of the Double Deep Q-Networks (DDQN) algorithm. This integrated approach is tailored to our AFG framework, to optimize the efficiency and accuracy of task allocation decisions. (iii) Demonstration of Superior Performance and Generalizability: Extensive experiments showcase that our method significantly surpasses traditional methods in efficiency and decision quality, demonstrating remarkable scalability and adaptability from smaller-scale training to larger-scale problem decision-making.

The paper is organized as follows: Section II introduces the robot swarm task allocation problem and models it as an alliance formation game. Section III details our proposed AFGNet_DDQN method. Section IV presents simulation results and discussions. Section V concludes the study.

II. Problem Descriptions

In this section, we will introduce the robot swarm task allocation problem and formulate its model based on Coalition Formation Game (CFG).

A. Introduction to Robot Swarm Task Allocation Problem

The robot swarm task allocation problem involves assigning a group of homogeneous robots to various tasks with the objective to maximize overall revenue. It comprises three main elements: the set of target tasks, the set of robots, and the utility function that evaluates the reward for robots collaboratively completing a task.

Define a set of tasks $\mathcal{M} = \{1, 2, ..., M\}$ and a set of robot $\mathcal{N} = \{1, 2, ..., N\}$ situated within a two-dimensional space. Each task $j \in \mathcal{M}$ is characterized as a multi-robot required task, necessitating collaborative effort from multiple robots for its completion. We use Π_j to represent the subset or coalition of robots allocated to task $j \in \mathcal{M}$, and $|\Pi_j|$ to indicate the cardinality of this coalition. Each task j has a maximum capacity, $\overline{h}_j$, which is the largest number of robots it can accommodate, ensuring $|\Pi_j| \leq \overline{h}_j$. Upon successful allocation, each task $j \in \mathcal{M}$ dispenses a reward to the coalition of robots Π_j, with the reward amount depending on the task's characteristics and the number of participating robots. We define the reward function r_j^{task} for the task $j \in \mathcal{M}$ as:

$$r_j^{task} = \mathcal{F}(j, |\Pi_j|). \tag{1}$$

The utility function of robot $i \in \mathcal{N}$ for selecting the task $j \in \mathcal{M}$, denoted by $u_i(j, |\Pi_j|)$, consists of two parts. The first part u_i^1 is the reward fed back to the robot upon task completion, which is equally distributed among participating robots as $u_i^1 = r_j^{task}/|\Pi_j|$. The second part u_i^2 describes the cost of the robot $i \in \mathcal{N}$ for selecting the task $j \in \mathcal{M}$, which increases with the the Euclidean distance $d_{i,j}$ between robot $i \in \mathcal{N}$ and task $j \in \mathcal{M}$. Particularly, we set $u_i^2 = \beta d_{i,j}$, where $\beta > 0$ is a penalty coefficient. Therefore, the utility function u_i for robot $i \in \mathcal{N}$ choose task $j \in \mathcal{M}$ is defined as:

$$u_i(j, |\Pi_j|) = u_i^1 - u_i^2 = \frac{\mathcal{F}(j, |\Pi_j|)}{|\Pi_j|} - \beta d_{i,j}. \tag{2}$$

B. Modeling Based on Coalition Formation Game

Since the class of Coalition Formation Game (CFG) provides an effective framework for analyzing the complex coalition formation process, we model the task allocation problem of a robot swarm as a CFG. In this model, each robot is considered as an independent participant, choosing to join specific coalitions based on their preferences.

Coalition Formation of Robot Swarm: The coalition formation of a robot swarm can be defined as a series of subsets separated from the set of robots $\mathcal{N}$: $\Pi = \{\Pi_1, \Pi_2, ..., \Pi_M\}$. Here, $\Pi_j \in \mathcal{N}$ represents the robot coalition for executing task $j \in \mathcal{M}$. Since each robot can only choose one task, the union $\cup_{j=1}^{M} \Pi_j = \mathcal{N}$, and the intersection $\Pi_j \cap \Pi_k = \emptyset, j \neq k \in \mathcal{M}$. If robot $i \in \mathcal{N}$ chooses the target task g_i, then the coalition it belongs to is Π_{g_i}, i.e., $\Pi_{g_i} = \{\Pi_j \in \Pi | i \in \Pi_j\}$.

Definition 1 (Robot Swarm Coalition Formation Game): It is a triplet $(\mathcal{N}, \mathcal{M}, \mathcal{Q})$, where (1) $\mathcal{N} = \{1, 2, ..., N\}$, a set of robots; (2) $\mathcal{M} = \{1, 2, ..., M\}$, a set of tasks; (3) $\mathcal{Q} := \{\mathcal{Q}_1, \mathcal{Q}_2, ...\mathcal{Q}_N\}$, a set of preference relations of the robots. For robot $i \in \mathcal{N}$, $\mathcal{Q}_i$ describes its preference relation over the robot coalition set Π. Specifically, for any two coalitions $\Pi_j, \Pi_k \in \Pi$ including robot i, $\Pi_j \succ_i \Pi_k$ implies that robot i strongly prefers Π_j to Π_k, and $\Pi_j \sim_i \Pi_k$ means that the preference regarding Π_j and Π_k is indifferent for robot i.

In CFG, coalition preferences are based on individual utility or the coalition's total utility. To balance individual and collective interests, we employ a balanced preference ordering method [19] for robot preferences.

Definition 2 (Balanced Preference Ordering): For any robot $i \in \mathcal{N}$, and for any two alliances Π_j and Π_k:

$$\begin{aligned}\Pi_j \succ_i \Pi_k \Leftrightarrow u_i(j, |\Pi_j|) + \sum_{i' \in \Pi_j/\{i\}} u_{i'}(j, |\Pi_j|) \\ + \sum_{i' \in \Pi_k/\{i\}} u_{i'}(k, |\Pi_k| - 1) > u_i(k, |\Pi_k|) + \\ \sum_{i' \in \Pi_j/\{i\}} u_{i'}(j, |\Pi_j| - 1) + \sum_{i' \in \Pi_k/\{i\}} u_{i'}(k, |\Pi_k|).\end{aligned} \tag{3}$$

where $u_i(j, |\Pi_j|)$ is the utility that robot i receives from being in alliance $|\Pi_j|$, $\sum_{i' \in \Pi_j/\{i\}} u_{i'}(j, |\Pi_j|)$ is the combined utility of all other robots i' in alliance $|\Pi_j|$ except robot i, $\sum_{i' \in \Pi_k/\{i\}} u_{i'}(k, |\Pi_k| - 1)$ is the utility that other robots i' would receive if they joined alliance $|\Pi_k|$ with the current size of $|\Pi_k|$ minus one (assuming robot i leaves $|\Pi_k|$). The term $\sum_{i' \in \Pi_k/\{i\}} u_{i'}(k, |\Pi_k| - 1)$ represents the utility that other robots i' would receive if they joined alliance Π_k assuming robot i leaves Π_k.

Definition 3 (Equilibrium Alliance Division): For any robot $i \in \mathcal{N}$, if the alliance division Π satisfies $\Pi_{g_i} \succ_i \Pi_k, \forall \Pi_k \in \Pi$, then Π is considered an equilibrium alliance division.

In this division, each robot tends to maintain its current alliance rather than join others. In other words, no robot can unilaterally change its alliance to gain greater benefits, either individually or collectively. The following lemma orignated from Theorem 2 in [19] shows some nice properties of the coalition formation game with balanced preference ordering.

Lemma 1: In a coalition formation game with balanced preference ordering, there is at least one stable alliance division. Furthermore, when solving the problem of maximizing total revenue, the optimal solution to this problem constitutes a stable alliance division.

According to [20], under the condition of decreasing individual utility functions, a Nash equilibrium alliance division can be obtained after $N(N+1)/2$ iterations using a distributed iterative method. However, as the number of robots increases, the time spend of obtaining an equilibrium solution grows exponentially. To address the computational cost issues in traditional distributed iterative methods, we consider using Deep Reinforcement Learning (DRL) to learn an end-to-end strategy for alliance division. In the following section, we will introduce the method in details.

III. AFGNet_DDQN for Robot Swarm Task Assignment

In this section, we provide a comprehensive overview of our algorithm, which combines graph neural networks and reinforcement learning to solve the task allocation problem in robot swarms. Our approach involves several key components: the construction of an Allocation Feature Graph (AFG), the embedding of this graph using a specialized network (AFGNet), and the application of an enhanced Double Deep Q-Network (DDQN) for learning optimal task allocation strategies.

A. Construction of the Allocation Feature Graphs

To precisely represent the allocation status features, alongside the distinct characteristics of task nodes and robot nodes in robot swarm task assignment problem, we thus develop the following allocation feature graph.

a) ***Allocation Feature Graphs (AFG):*** We let the AFG $\mathcal{G} = \{\mathcal{N}, \mathcal{E}, \mathcal{M}, \mathcal{H}\}$ illustrate the intricate interactive relationship and allocation feature among robots and tasks. Within this graph, $\mathcal{N} = \{1, 2, ..., N\}$ and $\mathcal{M} = \{1, 2, ..., M\}$ represents the set of robots and tasks in a two-dimensional space, respectively. Interaction among robots is depicted by edges $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$, with $\varepsilon_{i,i'} \in \mathcal{E}$ indicating information exchange between robots i and i'. The set $\mathcal{H} \subset \mathcal{N} \times \mathcal{M}$ illustrates robot-task connections, where $\xi_{i,k} \in \mathcal{H}$ signifies robot i's access to task k's information, including utility and attributes. Each robot i's neighboring set of robots is defined as $\mathcal{N}_i = \{i' \in \mathcal{N} : (i, i') \in \mathcal{E}\}$. Furthermore, we introduce $\mathcal{W} \in \mathbb{R}^{N \times N}$ as the weight matrix for the robot nodes. For any robot node $i \in \mathcal{N}$ and another node $k \in \mathcal{N}$, if $k \in \mathcal{N}_i$, $w_{i,k} \in \mathcal{W}$ denotes the Euclidean distance between these nodes, i.e., $w_{i,k} = d_{i,k}$; otherwise, $w_{i,k} = 0$.

b) ***Robot Node Feature on Graph:*** For the robot nodes of this graph, we primarily focus on the tasks selected by each robot. To effectively represent this information, we adopt a one-hot encoding scheme. In this encoding, for each robot node $i \in \mathcal{N}$, we create a vector ν_i of length M, where M is the total number of tasks. If a robot selects a specific task, the corresponding position in ν_i is marked as 1, while all other positions set to 0. Therefore, the robot note features on the graph can be expressed as $\mathcal{V} = \{\nu_1, \nu_2, ..., \nu_N\} \in \mathbb{R}^{N \times M}$.

c) ***Task Node Feature on Graph:*** Task nodes are characterized by the rewards they offer to each robot, reflecting the payoff for successful task completion. We establish $\mathcal{H} = \{h_1, h_2, ..., h_M\} \in \mathbb{R}^{M \times N}$ as the payoff matrix. The jth row of this matrix specifies the rewards available to each robot from task node j, as determined by formula (2). To maintain uniformity in features between robot and task nodes, we present task node features in a columnar layout, specifically defining $\mathcal{U} = \mathcal{H}^T$. Thus, the task node features on the graph are represented as $\mathcal{U} = \{\mu_1, \mu_2, ..., \mu_N\}$.

The graph-based model excels in providing a comprehensive state representation, and hence can effectively map out the interconnectedness and dependencies prevalent in robot swarm environments.

B. AFG based Graph Embedding Network

To capture the structural features of allocation feature graphs, we propose a graph-based deep learning architecture named AFGNet for embedding representation.

*a) **Node Embedding:*** In this subsection, we examine the handling of various feature types in graph $\mathcal{G}$. This includes task node features $\mathcal{U}$, robot node features $\mathcal{V}$, and the edge features $w_{i,k}$, for edges $\varepsilon_{i,k} \in \mathcal{E}$ connecting robot nodes $i, k \in \mathcal{N}$. We utilize multi-layer perceptron (MLP) with parameter θ and nonlinear activation functions σ to effectively synthesize and refine these features for enhanced feature embedding. Furthermore, we use the notation $[x||y]$ to denote the concatenation of two vectors x and y, which will be used below.

To enhance the representational power of the robot nodes' original features, we apply a dimensionality expansion using MLP with parameters $\theta_1 \in \mathbb{R}^{M \times d}$. This MLP operation increases the dimension of each robot node's feature vector ν_i from $\mathbb{R}^M$ to a higher-dimensional space $\mathbb{R}^d$, facilitating more nuanced feature representation.

$$\nu_i' = \sigma(\text{MLP}_{\theta_1} \nu_i). \tag{4}$$

In parallel with the robot nodes, the task nodes' original features undergo a dimensionality increase for better feature representation. Using another MLP with parameters $\theta_2 \in \mathbb{R}^{M \times d}$, each task node feature vector μ_i is expanded from $\mathbb{R}^M$ to $\mathbb{R}^d$.

$$\mu_i' = \sigma(\text{MLP}_{\theta_2} \mu_i). \tag{5}$$

To create a comprehensive embedding for each robot node, we combine the enhanced robot node features ν_i' and task node features μ_i'. This unified embedding δ_i encapsulates both the robot's attributes and its relationship with tasks, essential for effective task allocation.

$$\delta_i = \nu_i' + \mu_i'. \tag{6}$$

This part focuses on learning the embeddings that describe each robot node's connections within its neighborhood $\mathcal{N}_i$. For each robot node i, we first concatenate the features of its neighboring nodes ν_k and their corresponding edge weights $w_{i,k}$. This concatenated data is then processed through a MLP with parameters $\theta_4 \in \mathbb{R}^{d \times d}$, followed by a nonlinear activation function σ. After averaging these features across all neighbors in $\mathcal{N}_i$, we feed them into another MLP with parameters $\theta_3 \in \mathbb{R}^{(M+1) \times (d-1)}$. The final output, η_i, is the embedding that encapsulates the spatial and relational information of robot node i with respect to its neighboring nodes.

$$\eta_i = \sigma\left(\text{MLP}_{\theta_3}\left[\frac{1}{|\mathcal{N}_i|}\sum_{k \in \mathcal{N}_i} \sigma(\text{MLP}_{\theta_4}[w_{i,k}||\nu_k])||(|\mathcal{N}_i|)\right]\right). \tag{7}$$

To refine each robot node's embedding, we integrate its current combined embedding δ_i with the relational embedding η_i and the weighted embeddings of its neighbors. We start by concatenating($||$) the current combined embedding of robot node i, δ_i, with its relational embedding, η_i. Additionally, we incorporate the weighted embeddings of its neighbors, obtained by averaging the weighted product of each neighbor's embedding δ_k and the corresponding edge weight $w_{i,k}$. This concatenated data is then processed through additional MLP layers with parameters θ_5 and θ_6, and a nonlinear activation function σ. The result is the final updated embedding δ_i', which now includes both the individual characteristics of the robot node and the contextual information from its neighborhood.

$$\delta_i' = \sigma\left[\text{MLP}_{\theta_5}\left[\delta_i||\sigma\left(\text{MLP}_{\theta_6}\left[\frac{1}{|\mathcal{N}_i|}\sum_{k \in \mathcal{N}_i} w_{i,k}\delta_k||\eta_i\right]\right)\right]\right]. \tag{8}$$

*b) **Stacking and Pooling:*** The embedding process within the AFGNet layer is key to transforming the original graph $\mathcal{G}$ into node embeddings δ_i'. In this study, we stack L identical AFGNet layers, each capable of independent training, to achieve a deeper and more nuanced representation of the graph's structure. This stacking of multiple layers, culminating in the final embedding $\delta_i'^{(L)}$, is crucial because it allows the model to learn complex hierarchical patterns within the data [21].

After applying L layers of AFGNet, the embeddings of the robot nodes undergo a mean pooling process to condense the individual and collective information into a unified representation. Specifically, linear transformation MLP_{θ_7} with parameters $\theta_7 \in \mathbb{R}^{d \times d}$ and nonlinear activation functions σ are performed on the sum of all robot node embeddings, $\sum_{i \in \mathcal{N}} \delta_i'^{(L)}$. Each individual robot node's embedding $\delta_i'^{(L)}$ is then combined through concatenation($||$), yielding two distinct d-dimensional vectors. These vectors are then concatenated to form the state embedding s of graph $\mathcal{G}$:

$$s = \left[\sigma(\frac{\text{MLP}_{\theta_7}}{N}\sum_{i \in \mathcal{N}} \delta_i'^{(L)})||\delta_i'^{(L)}\right]. \tag{9}$$

This process allows for a comprehensive representation that encompasses both individual node details and the aggregated information of the swarm. For simplicity, this entire graph embedding process is summarized as:

$$s = \text{AFGNet}^L(\mathcal{V}, \mathcal{U}, \mathcal{G}|\theta). \tag{10}$$

C. Reformulation as Markov Decision Process

In this section, we reformulate the robot swarm task allocation as a Markov Decision Process (MDP) on graphs. Initially, each robot's node feature ν_i starts as a zero vector, reflecting no task assignment. Robots make sequential actions a_t based on the current graph state s_t. The MDP is defined by $H = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where γ represents the discount factor.

*a) **State and Action Spaces:*** The state space $\mathcal{S}$ consists of graph $\mathcal{G}$'s embedded states, output by AFGNet. The action space $\mathcal{A}$, representing the set of all tasks $\mathcal{M}$, reflects each robot's task choice.

*b) **State Transition Function:*** The function $\mathcal{P}(\mathcal{S}, \mathcal{A})\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}'$ describes how the graph state evolves post-action. It divides into P^r for robot nodes, updating ν_i based on action a_i, and P^m for task nodes, adjusting tasks' participant counts and subsequent rewards using formula (2) based on action a_t.

*c) **Reward Function:*** The reward function is defined as the difference between the maximum utilities of states s_t and s_{t+1}. It is important to note that when s_t is known, the utility of each robot can be calculated using formula (2), since the variables needed to compute the utility are included in s_t. Thus, the utility of state s_t is given by $u(s_t) = \sum_{i=1}^{t} u_i(j, |\Pi_j|)$. Based on this, the reward function can be expressed as:

$$r(s_t, a_t, s_{t+1}) = u(s_{t+1}) - u(s_t). \tag{11}$$

Since each robot node is equally important, we set the discount factor γ to 1, ensuring that future rewards are valued as much as immediate ones. Consequently, the cumulative reward R of this allocation strategy can be represented as:

$$R = \sum_{t=1}^{N} r(s_t, a_t, s_{t+1}) = u(s_N) - u(s_0) = u(s_N). \tag{12}$$

Here, s_N is the final state, representing system's total utility post task allocation, and s_0 is the initial state. The objective of the MDP is to maximize R, leading to an equilibrium coalition partition, as stated in Lemma 1.

D. DDQN based Training Process

DQN is a reinforcement learning algorithm tailored for solving Markov Decision Process (MDP) problems. To effectively manage the robot swarm task allocation on graphs, we implement the Double Deep Q-Network (DDQN) algorithm to train a policy network.

*a) **Prediction Network:*** The prediction network, denoted as $\mathrm{MLP}_{\omega_\pi}$, is a multi-layer perceptron with parameters $\omega_\pi \in \mathbb{R}^{2d}$. At each decision step, $\mathrm{MLP}_{\omega_\pi}$ takes the graph's embedding state s_t as input and outputs Q-values for all potential actions from that state, represented as $Q(s_t, a|\omega_\pi)$:

$$Q(s_t, a|\omega_\pi) = \mathrm{MLP}_{\omega_\pi}(s_t). \tag{13}$$

*b) **Sampling Strategy:*** Action selection at each step hinges on the Q-values forecasted by the Prediction Network. The probability of picking each action is determined by applying the softmax function to all the $Q(s_t, a|\omega_\pi)$ values:

$$\pi(a_t|s_t) = \frac{\exp(Q(a_t, s_t|\omega_\pi))}{\sum_{a'_t \in Q} \exp\left(Q(a'_t, s_t|\omega_\pi)\right)}, \forall a_t \in \mathcal{A}. \tag{14}$$

In the training phase, we sample actions based on the policy π to facilitate exploration. Conversely, during testing, the choice of actions is made greedily, selecting those with the highest probability.

*c) **Target Network:*** The target network MLP_{ω_t} has the same network structure with the prediction network, but with diffirence parameters $\omega_t \in \mathbb{R}^{2d}$. It is used to compute the target Q-values for updating the parameters of the prediction network. For each decision step , $\mathrm{MLP}_{\omega_\pi}$ inputs the next state of resulting from action a', denoted as s_{t+1}, and inputs the Q-values for all possible actions from state s_{t+1}, denoted as $\widehat{Q}(s_{t+1}, a'|\omega_t)$.

$$\widehat{Q}(s_{t+1}, a'|\omega_t) = \mathrm{MLP}_{\omega_\pi}(s_{t+1}). \tag{15}$$

*d) **Network Updating:*** In AFGNet_DDQN, the AFGNet 's parameters θ and the Prediction Network's parameters ω_π are updated regularly with each training step based on the loss function in formula (16), while the Target Network's parameters ω_t are updated every fixed number of steps c, using the parameters ω_π of the prediction network.

$$\begin{aligned} L(\theta, \omega_\pi) &= \mathbb{E}[\left(r_t - Q(s_t, a_t; \omega_\pi) + \gamma y_{\text{target}}\right)^2]; \\ s_t &= \text{AFGNet}^L(\mathcal{V}_t, \mathcal{U}_t, \mathcal{G}|\theta); \\ y_{\text{target}} &= \widehat{Q}(s_{t+1}, \underset{a_{t+1} \in \mathcal{A}}{\operatorname{argmax}} Q(s_{t+1}, a_{t+1}|\omega_\pi)|\omega_t); \\ s_{t+1} &= \text{AFGNet}^L(\mathcal{V}_{t+1}, \mathcal{U}_{t+1}, \mathcal{G}|\theta). \end{aligned} \tag{16}$$

Algorithm 1 presents a comprehensive training pseudocode that integrates three key stages: initialization, experiential data generation and network training.

Algorithm 1: Training Process of AFGNet_DDQN

```
   Input: AFGNet, Prediction Network and Target
          Network with trainable parameters $\theta$, $\omega_\pi$, and
          $\omega_t$; $\mathcal{G} = \{\mathcal{N}, \mathcal{E}, \mathcal{M}, \mathcal{H}\}$
1  Initialize: Experience replay buffer $\mathcal{D}$;
2  while Training is not terminated do
3      Receive initial feature $\mathcal{V}_0$ and $\mathcal{U}_0$;
4      Extract embeddings using AFGNet:
         $s_0 = \text{AFGNet}^L(\mathcal{V}_0, \mathcal{U}_0, \mathcal{G}|\theta)$;
5      for t = 1 to N do
6          Sample $a_t \sim \mathrm{MLP}_{\omega_\pi}(\cdot|s_t)$ Execute actions $a_t$
            ;
7          Receive reward $r_{t+1}$ and next feature $\mathcal{V}_{t+1}$
            and $\mathcal{U}_{t+1}$;
8          Extract embeddings using AFGNet:
            $s_{t+1} = \text{AFGNet}^L(\mathcal{V}_{t+1}, \mathcal{U}_{t+1}, \mathcal{G}|\theta)$;
9          Store ($s_t$, $a_t$, $s_{t+1}$, $r_{t+1}$ ) in replay buffer $\mathcal{D}$;
10     end
11     Sample a random minibatch of $\mathcal{S}$ samples from
        $\mathcal{D}$;
12     Compute the AFGNet_DDQN's loss $L$, and
        optimize the AFGNet 's parameters $\theta$ and the
        Prediction Network's parameters $\omega_\pi$ ;
13     Update Target Network parameters $\omega_t$ by $\omega_\pi$
        every $c$ times;
14 end
```

IV. Experiment

This section will evaluate the effectiveness of our proposed task allocation method for robot swarms through simulation experiments.

A. Experimental Setup

*a) **Evaluation Instances**:* Similarly to the related research, this study generated necessary allocation feature graphs instance followed the process described in [20] for training and testing. Specifically, in each instance, M tasks and N robots were uniformly and randomly distributed in a $1m$ x $1m$ area. In this article, we only consider the case where $M = 5$.

*b) **Utility Functions**:* We detail the task rewards and individual utility functions in our experiment. When multiple robots form a coalition to execute a task, they collectively receive a reward. While the total reward increases with the number of participating robots, the marginal contribution of additional robots diminishes. The reward approaches its maximum when the number of robots equals $\overline{h}_j$. Specifically, the task reward function is defined as:

$$r_j^{task} = r_j^{ini} \cdot log_{\varepsilon_j}(|\Pi_j| + \varepsilon_j - 1).$$

where r_j^{ini} is the initial reward when a single robot forms the coalition, uniformly drawn from $[N/M, 2 * N/M]$. The penalty coefficient β is set to 1, leading to the robot utility function:

$$u_i(j, |\Pi_j|) = u_i^1 - u_i^2 = \frac{r_j^{ini} \cdot log_{\varepsilon_j}(|\Pi_j| + \varepsilon_j - 1)}{|\Pi_j|} - d_{i,j}.$$

*c) **Configuration**:* For AFGNet, the number of iterations L was set to 3, and the dimension d of vector embedding was set to 64. The training process included 2.5 million iterations, with a batch size of 256, and the target network was updated every 1000 iterations.The Adam optimizer was used with a learning rate of 10^{-4}. During the first 10% of training, the exploration rate was linearly reduced from 1 to 0.05.

B. Training Results and Discussion

In our study, AFGNet_DDQN was pre-trained across varying robot scales (N=20, 60, 100) and subsequently benchmarked against two established algorithms. Firstly, to ascertain the efficacy of our bespoke AFGNet embedding network, we juxtaposed it with a "structure-to-vector" (S2V) based graph network embedding architecture [22], thus establishing our first benchmark, S2V_DDQN. Secondly, to underscore the superiority of the Double Deep Q-Network (DDQN) over traditional Deep Q-Network (DQN) in addressing the challenges presented in this study, we introduced AFGNet_DQN as our second benchmark. It is crucial to note that the network architectures and parameter configurations of these benchmarks were aligned with our proposed method for consistency.

The convergence trajectories of all methodologies are illustrated in Fig.1. It can be observed that our AFGNet_DDQN consistently outperformed in terms of reward acquisition, irrespective of the robot count. More significantly, compared to AFGNet_DQN, the AFGNet_DDQN achieved markedly higher rewards, substantiating the enhanced efficiency of the DDQN training architecture over the conventional DQN in addressing the robot swarm allocation problem. When we switch from AFGNet to S2V, although S2V_DDQN converges effectively, it often gets stuck at local optima because it doesn't include task-specific features, which limits its ability to gain higher rewards.

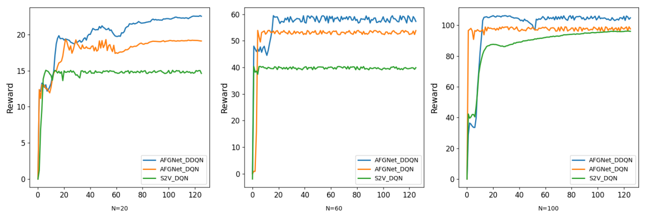

Fig. 1: Average reward by S2V_DDQN, AFGNet_DQN and AFGNet_DDQN in different robot scales.

C. Comparative Experiments

To further evaluate the efficacy of our AFGNet_DDQN, we execute a series of performance comparisons using models trained through the three distinct methods. We generated 100 varied initial test cases, each characterized by randomly positioned tasks and robot starting points, to assess these trained policies. Our evaluation metrics included the average total utility and the rate at which robots achieved an equilibrium coalition partition (NS_rate). The outcomes of these evaluations are detailed in Table I.

The empirical result reveal that our method consistently outperformed the others, achieving the highest average total utility across all tested scenarios. In scenarios with small (N=20) and medium (N=60) robot groups, AFGNet_DDQN demonstrated a marked advantage over S2V_DDQN. However, in large-scale scenarios (N=100), the performance gap between AFGNet_DDQN and S2V_DDQN narrowed, echoing our observations from the training phase. From the standpoint of NS_rate, our method achieved an 80% NS_rate, demonstrating its effectiveness across various scales. On the other hand, while AFGNet_DQN showed promising results in medium and small-scale scenarios, its performance notably deteriorated in larger-scale tests. This decline in NS_rate could be attributed to potential overfitting issues inherent in the traditional DQN when applied to large-scale problems, leading to diminished effectiveness.

D. Scalability Performance in Large-Scale Instances

In this section, we explore the generalizability of our AFGNet_DDQN method for large-scale robot systems. We pretrained our model with 20, 60, and 100 robots and tested it on scenarios with 500 and 1000 robots. For benchmarking, we compared it with a well-established Decentralized Algorithm (DA) [20]. We created 100 diverse test scenarios,

TABLE I: Comparison of test results between S2V_DDQN, AFGNet_DQN and AFGNet_DDQN

Scale	Average Total Utility			NS_rate		
	S2V_ DDQN	AFGNet _DQN	AFGNet _DDQN	S2V_ DDQN	AFGNet _DQN	AFGNet _DDQN
20	14.85	*19.31*	**22.61**	*77.90%*	77.30%	**81.80%**
60	42.37	*48.00*	**59.51**	48.70%	*76.60%*	**80.80%**
100	96.18	*96.99*	**106.21**	*66.64%*	53.88%	**78.00%**

TABLE II: Comparison of test results between S2V_DDQN, AFGNet_DQN and AFGNet_DDQN

Metrics	Scale	DA	Pretrain_20	Pretrain_60	Pretrain_100
Average Total Utility	500	*619.11*	564.4	**620.23**	*503.95*
	1000	*1247.53*	1106.39	**1250.75**	1000.21
NS_rate	500	**80.50%**	73.60%	*79.50%*	76.47%
	1000	*80.00%*	71.81%	**81.10%**	68.77%
Time Cost	500	181.83	**2.66**	**2.66**	*2.67*
	1000	714.88	**10.42**	*10.5*	11.1

and the results are presented in Table II. As can be seen, our model trained with 60 nodes, performs similarly to the DA algorithm in terms of Average Total Utility and NS_rate but requires only 2% of the DA algorithm's processing time, demonstrating significant computational efficiency.

However, our model's performance varies at different scales. Smaller-scale training leads to underfitting, limiting the model's ability to handle complex dynamics in larger environments. Conversely, training on overly large systems may cause overfitting, reducing adaptability to new scenarios. Thus, selecting an appropriate training scale is crucial for consistent model performance across various system sizes.

V. Conclusion and Discussion

In this study, we proposed a novel graph-based deep reinforcement learning method, AFGNet_DDQN, to tackle the robot swarm task allocation problem modeled as a coalition formation game. We have designed a network architecture, based on allocation feature graph embedding, to illuminate the interplay between robot node characteristics, task node features, and the state of task allocation. Then by reconstructing the problem within a Markov Decision Process (MDP) on graphs, we effectively depicted a sequential task allocation process. In our training process, we used an enhanced Double Deep Q Network (DDQN) algorithm to achieve an end-to-end training strategy. Our methodology demonstrated notable efficacy and superiority in both performance and decision-making speed in task allocation, as corroborated by thorough experimental validation.

References

[1] C. H. Caicedo-Nunez and M. Zefran, "Distributed task assignment in mobile sensor networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2485–2489, 2011.

[2] Y. Liu, M. A. Simaan, and J. B. Cruz Jr, "An application of dynamic nash task assignment strategies to multi-team military air operations," *Automatica*, vol. 39, no. 8, pp. 1469–1478, 2003.

[3] M. Dorigo, G. Theraulaz, and V. Trianni, "Swarm robotics: Past, present, and future [point of view]," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, 2021.

[4] S. Poudel and S. Moh, "Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey," *Vehicular Communications*, vol. 35, p. 100469, 2022.

[5] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[6] H. Chakraa, F. Guérin, E. Leclercq, and D. Lefebvre, "Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art," *Robotics and Autonomous Systems*, p. 104492, 2023.

[7] P. Ghassemi, D. DePauw, and S. Chowdhury, "Decentralized dynamic task allocation in swarm robotic systems for disaster response," in *2019 International Symposium on Multi-robot and Multi-agent Systems (MRS)*, pp. 83–85, IEEE, 2019.

[8] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann, "Multi-robot task allocation and scheduling considering cooperative tasks and precedence constraints," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3949–3956, IEEE, 2020.

[9] J. Chen, J. Wang, Q. Xiao, and C. Chen, "A multi-robot task allocation method based on multi-objective optimization," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1868–1873, 2018.

[10] W. Zhu, L. Li, L. Teng, and W. Yonglu, "Multi-uav reconnaissance task allocation for heterogeneous targets using an opposition-based genetic algorithm with double-chromosome encoding," *Chinese Journal of Aeronautics*, vol. 31, no. 2, pp. 339–350, 2018.

[11] W. H. Lim and N. A. M. Isa, "Particle swarm optimization with dual-level task allocation," *Engineering Applications of Artificial Intelligence*, vol. 38, pp. 88–110, 2015.

[12] K. Clinch, T. A. Wood, and C. Manzie, "Auction algorithm sensitivity for multi-robot task allocation," *Automatica*, vol. 158, p. 111239, 2023.

[13] G. Oh, Y. Kim, J. Ahn, and H.-L. Choi, "Market-based distributed task assignment of multiple unmanned aerial vehicles for cooperative timing mission," *Journal of Aircraft*, vol. 54, no. 6, pp. 2298–2310, 2017.

[14] K. Sakurama and H.-S. Ahn, "Multi-agent coordination over local indexes via clique-based distributed assignment," *Automatica*, vol. 112, p. 108670, 2020.

[15] M. Khoo, T. A. Wood, C. Manzie, and I. Shames, "A distributed augmenting path approach for the bottleneck assignment problem," *IEEE Transactions on Automatic Control*, vol. 69, no. 2, pp. 1210–1217, 2024.

[16] S. H. Alsamhi, A. V. Shvetsov, S. Kumar, S. V. Shvetsova, M. A. Alhartomi, A. Hawbani, N. S. Rajput, S. Srivastava, A. Saif, and V. O. Nyangaresi, "Uav computing-assisted search and rescue mission framework for disaster and harsh environment mitigation," *Drones*, vol. 6, no. 7, p. 154, 2022.

[17] T. Li and J. Chen, "Alliance formation in assembly systems with quality-improvement incentives," *European Journal of Operational Research*, vol. 285, no. 3, pp. 931–940, 2020.

[18] T. Barrett, W. Clements, J. Foerster, and A. Lvovsky, "Exploratory combinatorial optimization with reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3243–3250, 2020.

[19] J. Chen, Q. Wu, Y. Xu, N. Qi, X. Guan, Y. Zhang, and Z. Xue, "Joint task assignment and spectrum allocation in heterogeneous uav communication networks: A coalition formation game-theoretic approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 440–452, 2020.

[20] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous hedonic game for task allocation in a large-scale multiple agent system," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1534–1548, 2018.

[21] J. Sun, L. Zhang, G. Chen, P. Xu, K. Zhang, and Y. Yang, "Feature expansion for graph neural networks," in *International Conference on Machine Learning*, pp. 33156–33176, PMLR, 2023.

[22] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.