A Learning Approach to Multi-robot Task Allocation with Priority Constraints and Uncertainty

Fuqin Deng[1,2,3], Huanzhao Huang[1], Lanhui Fu[1], Hongwei Yue[1,4],
Jianmin Zhang[1,*], Zexiao Wu[3], Tin Lun Lam[2,5,*]
[1]School of Intelligent Manufacturing, Wuyi University, Jiangmen 529020, Guangdong, China
[2]The Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen 518000, Guangdong, China
[3]The 3irobotix Co.,Ltd, Shenzhen 518000, Guangdong, China
[4]School of physics and information engineering, Guangdong University of Education,
Guangzhou 510303, Guangdong, China
[5]School of Science and Engineering, the Chinese University of Hong Kong, Shenzhen, 518000, Guangdong, China.
*Corresponding authors' e-mail: zjm99_2001@126.com and tllam@cuhk.edu.cn

***Abstract*—Multi-robot task allocation has an important impact on the efficiency of multi-robot collaboration. For single-shot allocation without complicated constraints, some exact algorithms and heuristic algorithms can find the optimal solution efficiently. However, considering the priority constraints and uncertain execution time of robots for multiple times of allocation in an approximate dynamic programming environment, traditional methods such as heuristic algorithms have limited performance. To obtain better performance, we propose a method based on deep reinforcement learning. Specifically, we first use the directed acyclic graph to describe the priority relationship between tasks. Then we propose a graph neural network with a hierarchical attention mechanism to extract the characteristics of the task groups. Finally, we design the policy network to solve the approximate dynamic programming problem of multi-robot task allocation. Through training on the dataset of a given environment, the policy network can gradually refine the decision-making process by reinforcement learning. Experiment results show that the proposed modeling and solving method can find better solutions than existing heuristic algorithms. Furthermore, the learned strategy can be directly applied in other untrained environments with superior performance.**

***Index Terms*—Multi-robot task allocation, Deep reinforcement learning, Graph neural network.**

I. Introduction

MULTI-ROBOT systems can accomplish complex tasks by enabling multiple robots to work together and provide reliable and efficient solutions for practical applications such as search and rescue [1], patrol [2], and manufacturing [3]. In multi-robot systems, the multi-robot task allocation (MRTA) problem has a significant impact on the efficiency of robot cooperation. MRTA problem can be described as follow. Given multiple tasks and robots, under the condition of satisfying constraints, robots need to form a coalition to perform the tasks cooperatively. This work aims to arrange robots to form different coalitions in different periods to complete various tasks.

MRTA needs to face two problems in the real environment. In the first problem, there are priorities between tasks, and the robots need to complete tasks in a certain order. The priority between tasks is derived from the requirements of the real environment. We regard a set of tasks with the priority relationship as the task group. For example, in a smart factory, two orders require the same product, and the product needs to be manufactured according to the production process. Firstly, multiple robots are required to carry parts to the working area. Secondly, the robots need to cooperate to assemble multiple complex parts into products. Then the products are divided into two batches before packing and shipping. The whole process can be regarded as a task group. In this task group, the highest priority task is to carry the parts, the second priority task is to assemble the parts, and the last two tasks are to pack and ship the product. In the second problem, the time required for each robot to perform the task is uncertain. In many applications, other uncertain events may occur while robots are performing a task [4], which affects the execution time of the robot. In addition, the execution time of robots is related to the number of robots in the coalition and the characteristics of the task itself. In the ideal situation, the more robots in the coalition, the shorter the robot execution time. However, the characteristics of the task itself may make this situation untenable. For example,

This work was supported in part by the National Key R&D Program of China (2020YFB1313300), Shenzhen Peacock Plan of Shenzhen Science and Technology Program (Grant No.KQTD2016113010470345), the funding (AC01202101103) from the Shenzhen Institute of Artificial Intelligence and Robotics for Society, the special projects in key fields of Guangdong Provincial Department of Education (2019KZDZX1025), Innovative Program for Graduate Education (503170060259) and School-enterprise Cooperation Projects HX19029, HX20199, HX20247, HX21008 from Wuyi University. We would also like to thank Dr. Li NanNan from Macau University of Science and Technology for his valuable discussion on this paper.

when shipping parts, assigning too many robots will lead to robot congestion, and reduce the efficiency of the robot. It is difficult for us to have an accurate knowledge of the characteristics of each task, which also brings uncertainty to the task execution time of the robot. If the uncertainty in the MRTA problem is not addressed, the actual performance of the generated scheme will be much worse than expected. Therefore, this work aims to solve the MRTA problem with priority constraints and uncertain execution time.

For the priority constraints in the task group, one way is to use mathematical formulas to describe the priority relationship between tasks, and the whole problem can be formulated as a mathematical model [5]. However, with the increase in the scale of the problem, the efficiency of solving the problem will become very low. Another way is to divide tasks into executable tasks and non-executable tasks [6]. This method is simple, but it does not consider the impact of the structure of the task group on the MRTA problem. To make full use of the structural information of the task group, two problems need to be considered. Firstly, we should consider how to describe the priority relationship between tasks in the task group, and then how to extract information from them. For the first problem, the directed acyclic graph (DAG) is commonly used to describe the priority relationship between tasks [7]. For the second problem, we notice the excellent performance of graph neural network (GNN) in processing graph structure data recently [8] and consider using GNN to process DAG.

For the uncertainty of execution time, we cannot get all knowledge about execution time in most cases, so it is difficult to establish a mathematical model. In order to get more information about execution time, a natural idea is to learn from historical data and design efficient heuristic algorithms, but this requires expert knowledge and a lot of work. We would like to be able to automate the entire process of learning from data and designing strategies. Deep reinforcement learning (DRL) is an appropriate choice for this purpose as a kind of machine learning method. In DRL, the agent interacts with the environment through a Markov decision process (MDP). At each time step, the agent is in a given environment state and chooses an action according to its policy, then the agent gets a reward from the environment and enters a new state. The goal of DRL is to train the agent by maximizing the expectation of future rewards [9]. Once the problem is established as an MDP, the agent can be learned in the data from the actual working conditions of a multi-robot system, independent of inaccurate assumptions [10].

The main contribution of this paper is to propose a DRL based method to solve the MRTA problem of multiple task groups and uncertain robot execution time. Firstly, DAG is used to model the priority constraints in the task group. On this basis, the MDP for the MRTA problem is proposed. Then, we propose a GNN using the hierarchical attention mechanism. This GNN can encode the nodes in the graph, extract the key information features and output various types of embedding vectors. Finally, the policy network is designed based on the GNN. Experimental data show that our proposed method is superior to the existing heuristic algorithms designed for MRTA problems, and the learned strategy can be applied to MRTA problems that are much larger than those used in training.

This paper is organized as follows. Section II gives a discussion of the related work. Section III is the statement of the problem. The proposed method is introduced in Section IV. In Section V, we show the setups of our experiments and analysis of our results. In Section VI, we sum up our study.

II. Related works

According to [11], our MRTA problem belongs to the category of the single-task robots (ST), multi-robot tasks (MR), time-extended allocation (TA) with cross-schedule dependencies [XD] under the iTax taxonomy. Because the problem requires multiple robots to form a coalition to complete one task at a time, the efficiency of robots will be affected by other robots in the coalition. Here, we mainly investigated two aspects: i) Methods for solving MRTA problems belonging to ST-MR-TA [XD] and the applicability of these methods to our problem is discussed, ii) Current progress of deep reinforcement learning and its application in MRTA.

A. Previous Methods

At present, there have been many studies on MRTA problems fit within ST-MR-TA [XD]. (Unless otherwise specified, the MRTA problems mentioned later belong to ST-MR-TA [XD].) For priority constraints, Korsah [5] describes the constraints as mathematical formulas and models them as mixed-integer linear programming (MILP) problems, and uses an accurate algorithm to solve the problems. The algorithm produces the optimal scheme, which is executed by a group of indoor robots. However, its solution time increases exponentially with the increase of problem scale [12], which is unacceptable for most applications. To improve the solution efficiency, Jones [13] uses the genetic algorithm to search for the optimal solution in all feasible scheme and get good performance in the MRTA problem of the disaster response. These two algorithms allocate all tasks at one time before all robots begin to execute, and accurately plan the schedule of each robot because they are based on the assumption that the execution time of the robot is determined. However, in many practical applications, the environment is dynamic, which makes the execution time of the robot uncertain. In this case, we cannot generate a complete schedule for the robot before the task starts. Therefore, it needs to be allocated multiple times. At present, few methods can deal with the MRTA problem with uncertainty. There are decentralized methods and heuristic methods.

1) Decentralized Method: The decentralized method is coordinated by robots in various ways, and the robots determine the tasks they perform respectively. Among the decentralized methods to solve the MRTA problem, the market and negotiation-based algorithm and the distributed constrained-based algorithm have received more attention [4]. At present, both algorithms have been studied for priority constraints and uncertainty [7], [14]–[16]. This kind of method is robust when dealing with dynamic problems, but it is at the cost of reducing the solution quality [17].

2) Heuristic method: Heuristic method determines the robot coalition for the task according to rules. It is an easy-to-implement method and can deal with uncertainty. Ramchurn et al. [12] proposed a heuristic algorithm for MRTA in search and rescue. Zhang and Parker also introduced several general heuristic algorithms [18] to solve the MRTA problem, which can quickly generate acceptable solutions in a dynamic environment. However, the previous heuristic algorithms do not take into account the priority constraints between tasks. Therefore, Bischoff et al. [6] divides the current set of executable tasks and non-executable tasks according to the priority of tasks, so that the algorithm can directly consider the priority constraints of tasks. Compared with the decentralized method, the heuristic method has significantly improved the solution quality. However, designing efficient heuristic algorithms requires rich expert experience and repeated experiments. It may be difficult to design efficient heuristic algorithms in the face of more complex problems.

In order to overcome the limitations of heuristic methods, we consider enabling the scheduler to automatically learn the design of scheduling strategy through a data-driven way. DRL is an ideal technique for this purpose [19].

B. Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a technology that combines the optimal control ability of reinforcement learning (RL) with the data mining ability of deep learning (DL) [20]. Deep reinforcement learning uses the MDP to model the problem, which is suitable for dealing with the uncertainty in the problem. Recently, the idea of deep reinforcement learning as a solution to combinatorial optimization problems has been widely explored. One strategy is to use graphs to represent the problem and solve the problem by graph neural network combined with reinforcement learning. This method has been widely used in routing problems [21]–[23]. However, the routing problem abstracts from an undirected graph and does not take into account the priority relationship between task nodes. In the problem of machine scheduling, Zhang modeled the job shop scheduling problem as a directed acyclic graph (DAG) and modified the graph isomorphism network to deal with directed graphs [19]. But the problem it solves is different from the MRTA problem we studied. In [10], a DRL method is proposed for the data cluster scheduling problem in cloud computing. It also uses DAGs to describe the time dependence between its tasks. A GNN dedicated to the scheduling problem is designed to extract different types of embedding. We get inspiration from it, improve its GNN and apply it to the MRTA problem.

At present, there is not much work to use deep reinforcement learning in MRTA problems. In [3], they introduce an algorithm combining graph attention network and imitation learning to solve the MRTA in manufacturing. However, this work focuses on the problem of the single-task robots (ST), single-robot tasks (SR), and time-extended allocation (TA) with cross-schedule dependencies [XD], so its method can not be directly applied to our problem.

III. Problem Statement

In the problem, we consider a set of task groups $g = \{g_1, ..., g_n\}$. There are also a group of robots $r = \{r_1, ..., r_m\}$. Each task group is composed of a group of tasks. Task group g_i has a group of tasks $g_i = \{t_1^i, ..., t_l^i\}$. Each task t_j^i has an estimated total workload $d_{t_j^i}$, which represents the time required for one robot to complete the task under normal conditions. There is a priority between tasks in the task group, which represents the process to complete the action. The priority between tasks makes it possible for each task to have one or more predecessor and successor tasks [7]. A task cannot be executed until all its predecessor tasks are completed. Each robot can only perform one task at a time, and each task requires multiple robots to form a robot coalition to perform. Robot coalition C_j^i used to execute task t_j^i. Robots in the coalition contribute equally to the task. But robots do not have to execute the task at the same time, they only need to complete the assigned part and then move on to the next assigned task.

When robots form a coalition to execute tasks, the execution time of robots in the coalition is affected by various factors, so when the task is completed is uncertain. In this MRTA problem, the execution time of the robot is affected by the following factors:

- The execution time of each robot is uncertain because each robot will encounter various uncertain situations when performing tasks so the time required for them to complete the same workload is not necessarily the same.
- Although the workload of robots in the coalition is the same, the execution time of robots is affected by the number of robots in the coalition. In reality, the more robots in the coalition, the higher the efficiency is not necessarily. When there are too many robots in the coalition, the time for robots to perform tasks may increase due to congestion.

The goal is to find an allocation strategy to minimize the average completion time of task groups. The reason for choosing this goal is to reduce the running time of the whole multi-robot system and to reduce the resources consumed by the robot. Due to the uncertainty of the execution time of the robots and the priority between the tasks, the start time

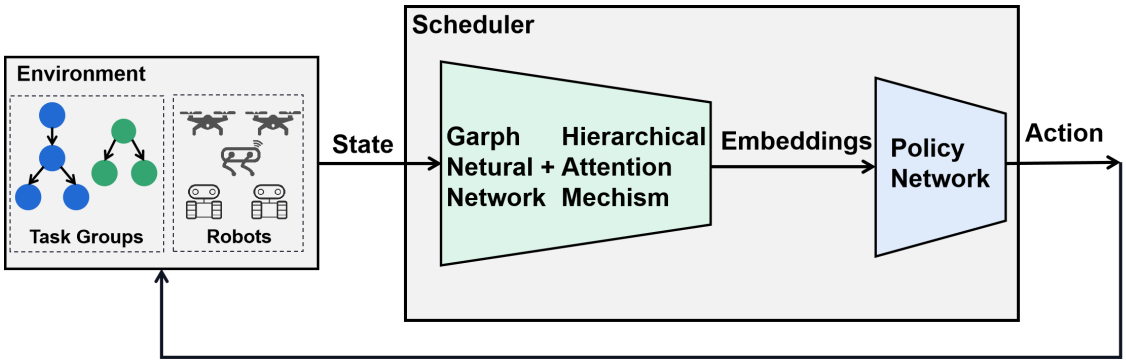

Figure 1. Overall framework of proposed method.

of the future task depends on the actual completion time of the previous predecessor task. So a single-shot allocation is not enough [24].

IV. Proposed Method

In this section, we will introduce our method. Firstly, we model the task group as DAG, and then based on this, we model the MRTA problem as the MDP. Finally, we introduce the design of the scheduler, which includes GNN and policy network. Figure 1 shows the overall framework of the proposed method. The state information is first input to GNN, the embedded vector output by GNN is input to the policy network, and the policy network outputs the action to the environment. The reason why GNN is used to process state information is that the policy network cannot directly process the arbitrary number of task and task groups, while GNN can process any number of DAGs and encode the information in DAGs as embedded vectors.

A. Model Task Group as DAG

For a task group, we use a directed acyclic graph $G_i = (t, E)$ to represent the relationship between tasks in the task group g_i. Node t corresponds to tasks, and edge E represents priority constraints between tasks. The directed edge $e_{jk} \in E$ indicates that task t_k^i should complete before executing task t_j^i. Figure 2 shows the directed acyclic graph of the example in the introduction. In the graph G_1, t_1^1 is a parts transporting task, t_2^1 represents the part assembly, t_3^1 and t_4^1corresponds to the tasks of packing and shipping products.

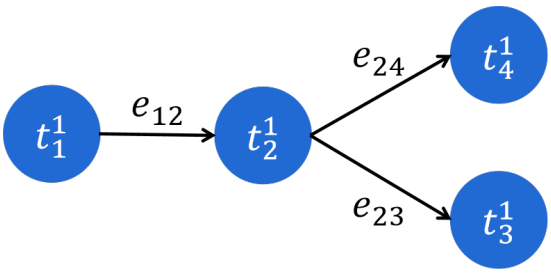

Figure 2. A directed acyclic graph G_1 of task group g_1.

B. Formulate Problem as Markov Decision Process

Solving an MRTA problem instance can be regarded as determining the robot coalition for each task at different times. Based on this, we formulate the problem as a Markov decision process. It includes four elements: state, action, state transition and reward.

- State: the state s_k at the decision step k includes DAGs representing the unfinished task groups, the current executable task list, and the list of available robots. The state of task t_l^i is represented by feature vectors $x_v^i = (w_1, w_2, q_{t_v^i})$. Where w_1 represents the unfinished workload of the task, w_2 represents the total workload, and $q_{t_v^i}$ represents the number of robots currently working on the task. The executable task list records which tasks can be executed at present. The list of available robots gives the number of available robots.
- Action: action a_k taken in the state s_k represents a part of the task allocation scheme. The scheduling decision is decomposed into a set of two-dimensional action sequences (t, p), t and p represent the output of the next task to be executed and the number of parallel robots of the task group in which the task is located respectively. According to the number of parallel robots in the task group, the number of robots assigned to the task can be determined. This design can reduce the space of action that must be explored and optimized during training [10].
- State transition: update the node state and list information in DAGs according to the action. If all tasks in the task group have been completed, remove the DAG corresponding to the task group from the current state.
- Reward: the goal of reward R is to minimize the average completion time of all task groups. We set the reward function obtained after the a_k as:

$$R = -(d_k - d_{k-1}) J_k \tag{1}$$

d_k represents the time corresponding to the decision step k, and J_k represents the number of task groups that the robot has not completed in time $[d_{k-1}, d_k)$. According

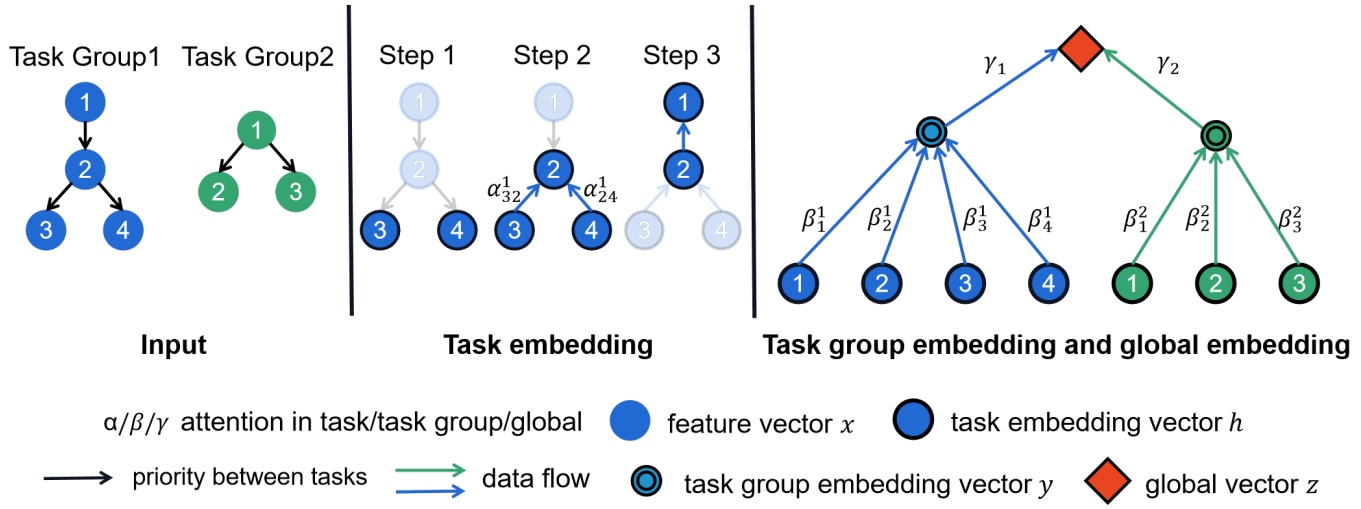

Figure 3. An example of using GNN with hierarchical attention mechanism to generate embedding vectors at different levels.

to little's law [25], it effectively minimizes the average completion time of all task groups.

C. GNN with hierarchical attention mechanism

In order to extract the features in DAGs, we added a hierarchical attention mechanism to the GNN dedicated to scheduling [10]. The design of the GNN is based on two intuitions inspired by Yang's work [26]: firstly, MRTA instances have a hierarchical structure (tasks form task groups, task groups form an MRTA instance). Therefore, when processing input state information, we build the representation of tasks, then aggregate the representation of tasks into the representation of task groups, and finally aggregate the representation of task groups into a global representation. Second, even the same task or task group may have different importance in different MRTA instance. So we add a hierarchical attention mechanism to GNN. GNN outputs three different levels of embedding vectors: task embedding vector, task group embedding vector, and global embedding vector. Figure 3 shows a simple example of generating embedding vectors using GNN.

Task embedding vector can capture information about tasks and their successors. To calculate all task embedding vectors, it is necessary to carry out multiple times messages passing steps starting from the task node with the lowest priority of DAG. In each message passing step, the task node aggregates the information from all its successor task nodes, and its embedded vector is expressed as follows:

$$h_v^i = MLP_{\theta_1}\left(\sum_{u \in S(t_v^i)} \alpha_{uv} MLP_{\theta_2}\left(h_u^i\right)\right) + MLP_{\theta_3}\left(x_v^i\right) \quad (2)$$

Where h_v^i is the task embedding vector of task node t_v^i in the DAG G_i, $MLP_{\theta_1}(\cdot)$, $MLP_{\theta_2}(\cdot)$ and $MLP_{\theta_3}(\cdot)$ are multi-layer perceptron (MLP) ; $S(t_v^i)$ represents the set of successor task nodes of task node t_v^i. α_{uv} is the attention coefficient. Used to indicate the importance of successor tasks. The attention coefficient is calculated as follows:

$$\alpha_{uv} = \frac{\exp\left(\text{LeakyReLU}\left(a^T\left[h_u^i || MLP_{\theta_3}\left(x_v^i\right)\right]\right)\right)}{\sum_{k \in S(v)} \exp\left(\text{LeakyReLU}\left(a^T\left[h_k^i || MLP_{\theta_3}\left(x_v^i\right)\right]\right)\right)} \quad (3)$$

Where a is a learnable weight vector, $||$ means that the two vectors are concatenated. LeakyReLU Nonlinear with negative input slope α= 0.2 .

Task group embedding vector aggregates information for all tasks in a task group. The task group embedding vector regards all task nodes in a task group as its successor task nodes. Similar to (2), the embedding vector of task group g_i is calculated as follows:

$$y^i = MLP_{\theta_4}\left(\sum_{u \in G(i)} \beta_u^i MLP_{\theta_5}\left(h_u^i\right)\right) \quad (4)$$

Where β_u^i is the attention factor. Unlike the attention mechanism in the task, this attention is used to determine the importance of tasks to the entire task group. It is similar to using the attention mechanism to extract words that are important to a sentence [26]. So the calculation of the attention coefficient is different.

$$\beta_u^i = \frac{\exp\left(\text{LeakyReLU}\left(a_{DAG}^T h_u^i\right)\right)}{\sum_{k \in G_i} \exp\left(\text{LeakyReLU}\left(a_{DAG}^T h_k^i\right)\right)} \quad (5)$$

Where a_{DAG} is the weight vector, which is randomly initialized and learned randomly during the training process.

The global embedding vector z summarizes the information of all task group embedding vectors. z is calculated the same way as those of task group embedding vectors.

D. Policy Network

To select the appropriate action, the embedding vectors output from the GNN also needs to be processed. Since actions are divided into tasks and the number of robots, they need to be calculated separately.

- Task node selection: for the task node t_v^i in DAG graph G_i, first calculate the score of t_v^i:

$$\lambda_v^i = MLP_{\theta_6}\left(h_v^i, y^i, z\right) \tag{6}$$

The function of MLP is to map the input embedding vector to the scalar value, and λ_v^i represents the score of choosing task node t_v^i. Then input the score into the softmax function to calculate the probability of selecting task node t_v^i:

$$prob\left(t_v^i\right) = \frac{\exp\left(\lambda_v^i\right)}{\sum_{k\in\Phi}\exp\left(\lambda_k^{G(k)}\right)} \tag{7}$$

Where $G(k)$ is the task group to which task t_k belongs. Φ is the set of all executable tasks.

- Number of robots: For each DAG G_i, the policy network also calculates its score firstly:

$$\eta_p^i = MLP_{\theta_7}\left(y^i, z, p\right) \tag{8}$$

Where p is the number of robots allocated to G_i. Similar to task node selection, the policy network applies the softmax function to these scores to calculate the probability of selecting different numbers of the robot.

E. Training Algorithm

We use the REINFORCE algorithm [27] to train the policy network. REINFORCE is a well-known policy gradient method and has been widely used in various fields. It updates network parameters using the stochastic gradient ascent algorithm based on the observed returns during training, which increases the probability of actions that increase the reward.

V. Experiment

A. Experimental setup

The experiment is completed on a computer equipped with Intel i9-9900k CPU and NVIDIA 3090 graphics card with 24GB memory.

In order to generate the random MRTA problem instance, we use the method of Suslova et al. [7] to generate the directed acyclic graph of the task group. The number of task nodes in the directed acyclic graph is randomly selected from 2 to 15. The estimated total workload of each task is sampled uniformly from the full interval of $[1000, 10000]$. It does not take time for the robot to transfer between tasks in the task group. And the time for robots to transfer between tasks belonging to different task groups is 1000.

To simulate the uncertainty of robot execution time, firstly, $\delta(d_{t_j^i}, |C_j^i|)$ is used to estimate the execution time of robots in the coalition, which is deterministic. $|C_j^i|$ is the number of robots in coalition C_j^i. Then, Gaussian distribution is used to model the uncertainty of robot execution time. In coalition C_j^i, the execution time of each robot follows the following distribution:

$$d_{t_j^i}^{r_m} \sim N\left(\delta\left(d_{t_j^i}, |C_j^i|\right), \mu\right) \tag{9}$$

μ is a constant, and $d_{t_j^i}^{r_m}$ is the actual execution time for robot r_m to perform task t_j^i. We use homogeneous robots to perform tasks to simplify experiments.

The code implementation of the scheduler uses TensorFlow [28]. For each problem scale, we train the scheduler for 12000 iterations. All multi-layer perceptrons contain three hidden layers. We used Adam optimizer [29] for gradient descent. In the training process, we use the Adam optimizer to set the constant learning rate lr = 1×10^{-3}.

To verify the performance of the proposed method in solving the MRTA problem, we compare the proposed method with three methods. They are random selection algorithm (random), greedy constructive heuristic (GCH) [6] and MinInterfere [18]. The random selection algorithm randomly selects the task assignment robots from the executable tasks. The main idea of the greedy constructive heuristic is to calculate the impact of all executable tasks on the objective function and assign all available robots to the tasks with the smallest objective function increment. Mininterfere allows as many tasks as possible to start as soon as possible, so the robot will be relatively evenly allocated to each task. Since Mininterfere does not take priority constraints into account in its design, we refer to the method of [6] to enable Mininterfere to handle priority constraints.

TABLE I
AVERAGE TASK GROUP COMPLETE TIME OF VARIOUS METHODS IN DIFFERENT SCALES
(SCALE:NUMBER OF ROBOTS ×NUMBER OF TASK GROUP)

Scale	Method			
	Random	***GCH***	***MinInterfere***	***Ours***
5×5	59083.2	51706.7	55607.6	**49079.7**
10×10	82585.3	72879.3	51951.2	**49807.2**
15×15	108738.0	91227.8	51648.9	**49337.5**
20×20	133422.5	106195.0	50971.3	**48895.8**

B. Results

In the first group of experiments, we train and test four groups of problem examples with different scales. We randomly generate 50 examples for each scale to test and compare the average completion time of task groups of solutions generated by different methods. The final results are shown in table I. The first column indicates the number of robots and task groups under the corresponding problem scale. It can be observed from table I that the quality of solutions generated by our proposed method is better than that of the baseline method under different scales. When the

TABLE II
AVERAGE TASK SOLUTION TIME

Scale	Method			
	Random	***GCH***	***MinInterfere***	***Ours***
5×5	1.3ms	1.4ms	1.3ms	5.0ms
10×10	1.4ms	1.4ms	1.4ms	5.4ms
15×15	1.4ms	1.6ms	1.7ms	5.8ms
20×20	1.4ms	1.9ms	1.8ms	5.9ms

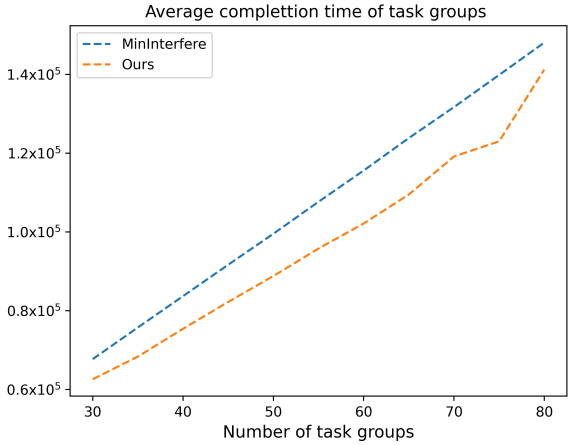

Figure 4. Generalization result of our method in larger scales.

number of robots and task groups is small, the performance gap of all methods is small. However, with the increase of the number of robots and task groups, the performance of the baseline method begins to decline rapidly, and our method can still maintain good performance. In particular, the random selection algorithm and GCH assign all robots to one task, which will reduce the execution efficiency of robots when the number of robots increases. These results show that our method can generate better solutions for MRTA problems of different scales, especially in large-scale problems.

Then we tested the computational efficiency of our method. We calculated the average solution time required for each task. The final results are in table II. It can be seen that the average task solution time does not change significantly with the expansion of the scale, indicating that our method is less affected by the problem scale. Although the solution time of our method is relatively long compared with other baseline methods, it is acceptable considering that the solution time is still within a reasonable range and the performance is improved.

Finally, we evaluated the generalization performance of the proposed method. In this experiment, we use the strategy of training in the scale of 20 robots and 20 task groups to solve the MRTA instances with more task groups and observe the average task completion time. We compare the proposed method with the best performing baseline method. The number of robots is fixed at 20, and the number of task groups ranges from 25 to 80, and the number of task groups increases at intervals of 5. The final results are shown in Figure 4. It can be observed that even solving the problems larger than the training scale, the performance of the proposed method is still better than that of the best baseline method. This shows that the proposed method can gain experience in the process of training, to deal with the state not seen in training, indicating the proposed method has a certain generalization ability. The generalization ability enhances the generality of the method and reduces the time cost of training on the corresponding scale, which is conducive to the application in practical MRTA problems.

VI. CONCLUSION

We present a method based on DRL to solve the MRTA problem with priority constraints and uncertain execution time of robots. Firstly, we use the DAG to describe the priority constraints and establish a Markov decision process for the MRTA problem. Then a graph neural network with the hierarchical attention mechanism is proposed, and a policy network is designed based on it. Trained policy networks can generate efficient solutions for MRTA problems of different scales. The experimental results show that the proposed method is superior to the existing heuristic methods. Future research will extend the method to MRTA with more constraints.

REFERENCES

[1] S. Ramchurn, A. Farinelli, K. Macarthur, M. Polukarov, and N. Jennings, "Decentralised coordination in robocup rescue," *The Computer Journal*, 2009.

[2] P. Fazli, A. Davoodi, and A. K. Mackworth, "Multi-robot repeated area coverage," *Autonomous robots*, vol. 34, no. 4, pp. 251–276, 2013.

[3] Z. Wang and M. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.

[4] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[5] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, "xbots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 115–122.

[6] 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)*. IEEE, 2020, pp. 3949–3956.

[7] E. Suslova and P. Fazli, "Multi-robot task allocation with time window and ordering constraints," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6909–6916.

[8] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[9] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[10] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 270–288.

[11] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[12] S. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, and N. R. Jennings, "Coalition formation with spatial and temporal constraints," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 3-Volume 3*, 2010, pp. 1181–1188.

[13] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," *Autonomous robots*, vol. 30, no. 1, pp. 41–56, 2011.

[14] E. Nunes, M. McIntire, and M. Gini, "Decentralized allocation of tasks with temporal and precedence constraints to a team of robots," in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2016, pp. 197–202.

[15] N. Hooshangi and A. A. Alesheikh, "Agent-based task allocation under uncertainties in disaster environments: An approach to interval uncertainty," *International journal of disaster risk reduction*, vol. 24, pp. 160–171, 2017.

[16] R. Stranders, F. M. Delle Fave, A. Rogers, and N. Jennings, "U-gdl: A decentralised algorithm for dcops with uncertainty," 2011.

[17] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "A mathematical programming approach to collaborative missions with heterogeneous teams," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 396–403.

[18] Y. Zhang and L. E. Parker, "Multi-robot task scheduling," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2992–2998.

[19] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, "Learning to dispatch for job shop scheduling via deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1621–1632, 2020.

[20] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[21] 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.

[22] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, "Learning heuristics over large graphs via deep reinforcement learning," *arXiv preprint arXiv:1903.03332*, 2019.

[23] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large tsp instances," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 7474–7482.

[24] S. Choudhury, J. K. Gupta, M. J. Kochenderfer, D. Sadigh, and J. Bohg, "Dynamic multi-robot task allocation under uncertainty and temporal constraints," *Autonomous Robots*, vol. 46, no. 1, pp. 231–247, 2022.

[25] D. Chhajed and T. J. Lowe, *Building intuition: insights from basic operations management models and principles*. Springer Science & Business Media, 2008, vol. 115.

[26] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[27] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: A system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.