2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Collaborative Scheduling with Adaptation to Failure for Heterogeneous Robot Teams

Peng Gao[1], Sriram Siva[2], Anthony Micciche[3], and Hao Zhang[3]

***Abstract*— Collaborative scheduling is an essential ability for a team of heterogeneous robots to collaboratively complete complex tasks, e.g., in a multi-robot assembly application. To enable collaborative scheduling, two key problems should be addressed, including allocating tasks to heterogeneous robots and adapting to robot failures in order to guarantee the completion of all tasks. In this paper, we introduce a novel approach that integrates deep bipartite graph matching and imitation learning for heterogeneous robots to complete complex tasks as a team. Specifically, we use a graph attention network to represent attributes and relationships of the tasks. Then, we formulate collaborative scheduling with failure adaptation as a new deep learning-based bipartite graph matching problem, which learns a policy by imitation to determine task scheduling based on the reward of potential task schedules. During normal execution, our approach generates robot-task pairs as potential allocations. When a robot fails, our approach identifies not only individual robots but also subteams to replace the failed robot. We conduct extensive experiments to evaluate our approach in the scenarios of collaborative scheduling with robot failures. Experimental results show that our approach achieves promising, generalizable and scalable results on collaborative scheduling with robot failure adaptation.**

I. Introduction

Collaborative multi-robot systems have been widely studied over the past few decades due to their scalability, robustness, and effectiveness [1], [2]. For a multi-robot system, collaborative scheduling is a fundamental ability for multiple robots to collaboratively complete complex tasks and adapt to dynamic changes such as robot failures. It is widely studied in a wide range of robotics applications, such as multi-robot search and rescue [3], [4], [5], human-robot collaboration [6], [7], [8], and intelligent manufacturing [9], [10], [11].

To realize collaborative scheduling, two essential problems must be addressed. First, during normal execution, a sequence of tasks should be optimally scheduled to a team of robots that may have heterogeneous capabilities. Second, robots may fail, and the team of robots should autonomously adapt to such failures during collaborative scheduling. For example, as shown in Figure 1, when a team of heterogeneous robots collaboratively assembles a helicopter, a policy needs to schedule assembling tasks to individual robots with the best resource use and shortest completion time until all tasks are completed.

*This work was partially supported by NSF CAREER Award IIS-1942056 and DARPA Young Faculty Award (YFA) D21AP10114-00.

[1]Peng Gao is with the Department of Computer Science, University of Maryland, College Park, MD 20742, USA. Email: gaopeng@umd.edu.

[2]Sriram Siva is with the Department of Computer Science, Colorado School of Mines, Golden, CO 80401, USA. Email: sivasriram@mines.edu.

[3]Anthony Micciche and Hao Zhang are with the Manning College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01002, USA. Email: {amicciche, hao.zhang}@umass.edu.

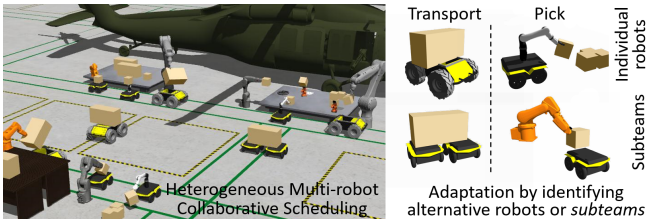

Fig. 1. A motivating scenario for heterogeneous multi-robot collaborative scheduling with failure adaptation in an assembly application. When a robot fails, the scheduling policy should identify alternative individual robots and subteams to replace the failed robot in order to complete the tasks (e.g., pick or transport).

Additionally, when a robot fails, the policy must identify alternative individual robots or *subteams* to replace the failed robot to complete the tasks.

Due to the importance of collaborative scheduling, a variety of methods were proposed. Traditional methods typically use heuristics [12], [13] or integer linear programming [14], [15], [16] to address this problem. However, they cannot run in real time to address real-world problems with complex constraints (e.g., task dependency or complex resource limitation). Very recently, learning-based methods demonstrate promising performance in terms of effectiveness and efficiency on collaborative scheduling [17], [18]. However, they still face several difficulties. First, previous learning-based approaches cannot identify subteams as alternatives to adapt as a team to failures, but generally consider individual robots only. Second, during task scheduling, most methods do not encode complex states of both tasks and robots, such as task dependencies and robot resource constraints. Third, the policy learned by the previous methods often do not generalize well to new robots, tasks and application scenarios.

In this paper, we introduce a novel approach that integrates deep bipartite graph matching and imitation learning (IL) in a unified framework for heterogeneous multi-robot collaborative scheduling with failure adaptation. Specifically, we use a graph to represent all tasks: each node represents a task and is associated with a vector to encode its required robot capability and resource capacity; an edge between two nodes denotes dependency of the tasks. We use a graph attention network [19] to encode task relationships. Then, we formulate collaborative scheduling as a novel learning-based bipartite graph matching problem that learns a policy to determine task scheduling based on the reward of potential task schedules. When a robot fails during task execution, our approach generates a pool of subteams and uses a reward function learned by IL to evaluate the subteams to replace the failed robot.

Our key contribution is the introduction of the novel unified learning-based approach that addresses heterogeneous multi-robot collaborative scheduling with robot failure adaptation. Specific novelties of the paper include:

- We design a new graph attention network (GAT) with a graph-based representation to encode complex relationships among tasks. As the embedding vector computed by GAT has a fixed length, our approach is agnostic to the changing number of tasks, thus improving the approach's generalizability.
- We propose a novel deep learning-based bipartite graph matching method for heterogeneous multi-robot collaborative scheduling with failure adaptation. Our approach learns a policy via bipartite graph matching that is trained using IL and is able to identify not only individual robots but also subteams to replace failed robots.

II. Related Work

In this section, we provide a review of related works in both problem and solution domains.

In the problem domain, heterogeneous multi-robot collaborative scheduling is mainly related to two research problems [20], including single-task robots, single-robot tasks (ST-SR) and the single-task robots, multi-robot tasks (ST-MR). ST-SR approaches require that each robot only executes one task and one task can only be executed by one robot at a time, e.g., to address the Traveling Salesman Problem (TSP) [17], [21] and the Vehicle Routing Problem (VRP) [22], [23]. ST-MR allows each single task to be executed by a team of robots, which is also known as coalition formation [24], [25], e.g., to address the Multi-mode Multi-Processor Machine Scheduling Problem [26]. The number of robots is assumed to be known to execute each task in ST-MR. Additional constraints considered in these problems also include heterogeneity [27], [28], capacity [23], [29], task dependency [30], [31], and spatial [32], [33] and temporal constraints [34], [35].

In the solution domain, existing methods for multi-robot collaborative scheduling can be categorized into three groups, including integer linear programming (ILP) [14], [15], [16], heuristic-based scheduling [12], [13], and deep learning [17], [18]. ILP-based methods often have an exponential complexity and therefore it is not usually practical to deploy them on a real-time system solving large problems. Heuristic scheduling techniques accelerate task scheduling by integrating heuristics, including using colony [36], genetic [37] and simulated annealing algorithms [38]. However, these methods often cannot deal with complex constraints [39]. Recently, deep learning-based methods have been implemented to some success, e.g., addressing TSP by reinforcement learning (RL) [17], [40], location coverage by behavior cloning [41], and multi-robot coordination via inverse RL [39], [42].

Although deep learning-based approaches achieve promising efficiency (compared to ILP techniques) and effectiveness (compared to heuristic methods), previous learning methods cannot encode the complex relationship of tasks and robots, or identify subteams for failure adaptation.

III. Approach

Notation. Matrices are represented as boldface capital letters, e.g., $\mathbf{M} = \{\mathbf{M}_{i,j}\} \in \mathbb{R}^{n \times m}$. $\mathbf{M}_{i,j}$ denotes the element in the i-th row and j-th column of $\mathbf{M}$, and $\mathbf{M}_{:,i}$ denoting the i-th column of $\mathbf{M}$. Vectors are denoted as boldface lowercase letters $\mathbf{v} \in \mathbb{R}^n$ and scalars are denoted as lowercase letters.

A. Problem Definition

Given a team of N heterogeneous robots and M tasks, we represent the problem of collaborating scheduling as a tuple $\mathcal{M} = \{\mathcal{R}, \mathcal{T}, \mathbf{P}\}$, where $\mathcal{R} = \{\mathbf{r}_i\}^N$ is a set robot states, $\mathcal{T} = \{\mathbf{t}_i\}^M$ is a set of task states, and $\mathbf{P}$ is a task dependency matrix. Each robot state $\mathbf{r}_i = [\mathbf{c}_i^r, w_i^r]$ has two attributes. The binary vector $\mathbf{c}_i^r \in \{0,1\}^K$ denotes the capabilities of the i-th robot and K is the number of possible capabilities. Each variable in $\mathbf{c}_i^r$ indicates whether a robot has the corresponding capability. For example, given a set of capabilities [Ground Navigating, Grasping, Flying], $\mathbf{c}_i^r = [1, 0, 0]$ indicates that the i-th robot can navigate on the ground, but cannot grasp or fly. The scalar $w_i^r \in \mathbb{R}$ represents resource capacities (e.g., robot payload).

A task state $\mathbf{t}_i = [\mathbf{s}_i^t, \mathbf{c}_i^t, w_i^t]$ includes three attributes. $\mathbf{s}_i^t \in \mathbb{R}^3$ represents the state of the i-th task, which consists of three variables to indicate whether the task is schedulable, assigned, and incomplete, respectively. For example, if the i-th task is schedulable and incomplete, then $\mathbf{s}_i^t = [1, 0, 1]$. $\mathbf{c}_i^t \in \mathbb{R}^K$ is a one-hot vector that encodes the required robot capabilities to execute the i-th task. The scalar $w_i^t \in \mathbb{R}$ denotes the required resource to execute the task. Task dependency is defined as matrix $\mathbf{P} \in \{0,1\}^{M \times N}$, with $\mathbf{P}_{i,j} = 1$ encoding that the j-th task must be executed after the i-th task.

Given the above representations, our objective is to address the following problems by a unified approach:

- During normal execution with no failure, we aim to learn a policy that optimally assign tasks to robots. The policy can be denoted as a scheduling matrix $\mathbf{A} = \{\mathbf{A}_{i,j}\}^{N \times M}$, $\mathbf{A}_{i,j} \in \{0,1\}$, where $\mathbf{A}_{i,j} = 1$ if the j-th task is assigned to the i-th robot.
- When a robot fails, we aim to adapt robots to the failure as a team by identifying alternative individual robots or subteams to replace the failed robot in order to ensure all tasks to be completed.

The overview of our approach is illustrated in Figure 2.

B. Bipartite Graph Matching for Collaborative Scheduling

We formulate collaborative scheduling as a deep bipartite graph matching (DBGM) problem. We represent the problem as a bipartite graph $\mathcal{G} = \{\mathcal{R}, \mathcal{T}, \mathbf{A}\}$ [43], and design DBGM methods to identify schedules between the robot set $\mathcal{R}$ and the task set $\mathcal{T}$ by choosing edges $\mathbf{A}$ of the bipartite graph $\mathcal{G}$.

We further represent the task set as a directed graph $\mathcal{G}^t = \{\mathcal{T}^t, \mathbf{P}^t\}$, where the edge set $\mathbf{P}^t$ encodes task dependencies. Then, we use a graph attention network (GAT) [19] to encode attributes of the tasks. Node embeddings of the graph $\mathcal{G}^t$ are computed as $\mathcal{H}^t = \{\mathbf{h}_i^t\} = \Psi(\mathcal{G}^t)$, where $\mathbf{h}_i^t$ is the embedding vector of the i-th task and Ψ is the GAT. $\mathbf{h}_i^t$ encodes not only

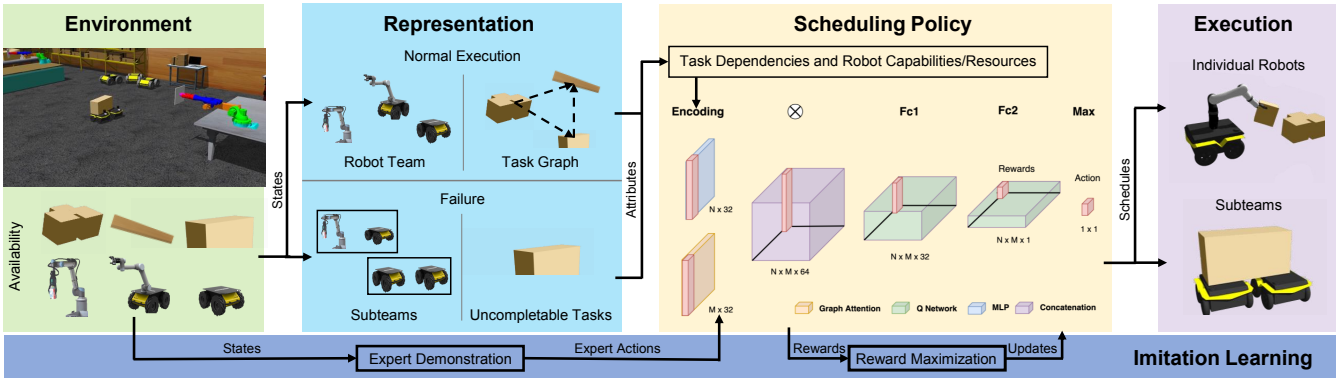

Fig. 2. Overview of our approach that integrates deep bipartite graph matching and imitation learning in a unified framework for heterogeneous multi-robot collaborative scheduling with failure adaptation. In normal execution, our policy is able to determine task scheduling based on the reward of potential robot-task pairs. In failure cases, our policy is also able to evaluate the subteams to replace the failed robot.

the i-th task's attributes, but also the attributes of its depending tasks. $\mathbf{h}_i^t$ is computed using Ψ by:

$$\mathbf{h}_i^{t,l+1} = \alpha_{i,i}\mathbf{W}^e\mathbf{h}_i^{t,l} + \text{LeakyReLU}\left(\sum_{\mathbf{P}_{i,j}\neq 0} \alpha_{i,j}\mathbf{W}^e\mathbf{h}_j^{t,l}\right) \quad (1)$$

where $\mathbf{W}^e$ denotes the trainable parameter of Ψ, LeakyReLU is a non-linear activation function, $l \in \{1, 2, \ldots, L\}$ is the number of layers in the forward process of Ψ. Each task embedding vector does not just encode its own attributes but also consider its dependent tasks attributes. $\alpha_{i,j}$ is the attention of the j-th task to the i-th task, which is defined as:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{W}^e\mathbf{h}_i^{t,l} \| \mathbf{W}^e\mathbf{h}_j^{t,l})\right)}{\sum_{\mathbf{P}_{i,j}\neq 0} \exp\left(\text{LeakyReLU}(\mathbf{W}^e\mathbf{h}_i^{t,l} \| \mathbf{W}^e\mathbf{h}_j^{t,l})\right)} \quad (2)$$

where $\exp(\cdot)$ is the exponential function, $\|$ is the concatenation operation. Eq. (2) uses SoftMax to normalize the impacts of dependent tasks on the i-th task. The embedding vectors are agnostic to the number of tasks, thus improving our method's generalizability.

We also compute robot embedding vectors by $\mathbf{h}^r = \Phi(\mathbf{r}^r)$, where Φ denotes a multi-layer perceptron that consists of one linear layer. Then, we define a reward matrix $\mathbf{R} \in \mathbb{R}^{N\times M}$ with $\mathbf{R}_{i,j}$ denoting the reward of assigning the j-th task to the i-th robot. $\mathbf{R}_{i,j}$ can be computed by $\mathbf{R}_{i,j} = \psi(\mathbf{h}_i^r \| \mathbf{h}_j^t)$ using a Q-network ψ that consists of two linear layers and a Relu activation function.

In order to encode task dependencies and robot capabilities for a heterogeneous team, we mask the rewards as:

$$\mathbf{O} = \mathbf{R} \circ \mathbf{M}^d \circ \mathbf{M}^c \quad (3)$$

where $\circ$ denotes the element-wise product operator and $\mathbf{O}$ denotes the masked rewards given task dependencies and robot capabilities. $\mathbf{O}$ can also be treated as the weights of edges $\mathbf{A}$ in the bipartite graph $\mathcal{G}$, as $\mathbf{O}_{i,j}$ indicates the reward of the schedule $\mathbf{A}_{i,j}$. The task dependency mask $\mathbf{M}^d = \{\mathbf{M}_{i,j}^d\}^{N\times M}$ is defined as:

$$\mathbf{M}_{:,j}^d = \begin{cases} 1 & \sum_{k=1}^{M} \mathbf{P}_{k,j} = 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

If the j-th task's all dependent tasks are completed (encoded by $\mathbf{P}$), then $\mathbf{M}_{:,j}^d = 1$. $\mathbf{M}^c = \{\mathbf{M}_{i,j}^c\}^{N\times M}$ denotes the robot capability mask, defined as:

$$\mathbf{M}_{i,j}^c = \begin{cases} 1 & (\mathbf{c}_i^r)^\top \mathbf{c}_j^t = \|\mathbf{c}_j^t\|_1 \text{ and } w_i^r > w_j^t \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

If the i-th robot is able to address the j-th task, i.e., the robot's capabilities and resources satisfy the task's requirements, then $\mathbf{M}_{i,j}^c = 1$. The final collaborative scheduling policy $\mathbf{A}^*$ is computed by bipartite graph matching to maximize the edge weights $\mathbf{O}$ of the bipartite graph $\mathcal{G}$ as follows:

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \sum_{i,j} \mathbf{A}_{i,j}\mathbf{O}_{i,j} \quad \text{s.t.} \sum_i \mathbf{A}_{i,j} \leq 1, \sum_j \mathbf{A}_{i,j} \leq 1 \quad (6)$$

The constraints enforce that each robot / subteam can at most execute one task and each task can be assigned to one robot / subteam at most.

C. Adaptation to Robot Failure

Robots may fail in the real world, e.g., because of actuator, sensor, and computer malfunctions. Robot failure may cause tasks or the whole mission to fail. For any task that cannot be completed by individual robots, we name it an *uncompletable* task. To address uncompletable tasks due to robot failure, we introduce an adaptation method that teams up the remaining available robots as a *subteam* to replace the failed robot, which is presented in Algorithm 1.

Algorithm 1 recursively forms a pool of potential subteams $S = \{\mathbf{s}_i\}$ that are able to execute each uncompletable task $\mathbf{t}_u$. When a subteam is identified that can satisfy the requirement of an uncompletable task, it is added to the subteam set (Line 7). In addition, if the remaining available robots cannot form a subteam with K robots (Line 11), then Algorithm 1 ends the current iteration and move to the next one. The main function (Line 1) identifies possible subteams with various numbers of teammates. The complexity of the algorithm is $O(KC)$ where C is the combination $C = \binom{N}{K}$. Given the subteam set S, we use the same Q-network ψ to compute the reward of assigning an individually uncompletable task to a subteam.

The DBGM method for collaborative scheduling with adaptation to failure is presented in Algorithm 2. Line 2 collects

Algorithm 1: Generate Subteams

Input : Individually uncompletable task $\mathbf{t}_u = \{\mathbf{c}_u^t, w_u^t\}$
Available robot set $\mathcal{R} = \{\mathbf{r}_j\}^N$ where $\mathbf{r}_j = \{\mathbf{c}_j^r, w_j^r\}$
Maximum subteam size K
Output : Subteam set S

1: **for** $k < K$ **do**
2: Initialize subteam indicator $\mathbf{s} = [0, \ldots, 0] \in \mathbb{R}^N$;
3: Initialize position pointer $p = 1$;
4: Update subteam set $S \leftarrow$ Generate Subteams(p, $\mathbf{t}_u$, N, k, $\mathbf{s}$);
5: **end**
6: **Function** *Generate Subteams (p, $\mathbf{t}_u$, N, k, $\mathbf{s}$)*:
7: **if** $|\mathbf{s}| = k$ *and* $< \mathbf{c}^s, \mathbf{c}^t >= \|\mathbf{c}^t\|_1$ *and* $w^s \geq w^t$ **then**
8: Add subteam $\mathbf{s}$ to subteam set S;
9: **return**;
10: **end**
11: **if** $p = N + 1$ *or* $|\mathbf{s}| + N - p + 1 < K$ **then**
12: **return**;
13: **end**
14: $\mathbf{s}_p = 1$, Generate Subteams ($p + 1$, $\mathbf{t}_u$, N, k, $\mathbf{s}$);
15: $\mathbf{s}_p = 0$, Generate Subteams ($p + 1$, $\mathbf{t}_u$, N, k, $\mathbf{s}$);
16: **return** Subteam set S

available robots and tasks, respectively. Lines 5 checks if an individually uncompletable task exists. If yes, Lines 6-8 form subteams and compute the embedding vectors for the task and the subteams. If there is no individually uncompletable task, Lines 11-12 compute embedding vectors of the tasks and robots for normal execution. Lines 14-16 greedily traverse all robot/subteam-task pairs and choose the pair with the highest reward as the task scheduling. This scheduling process is repeated until no robots/subteams or tasks are available.

D. Imitation Learning for DBGM Training

Learning the parameters of our DBGM network needs large amount of training data that is not feasible to manually label. Inspired by the recent work [39], we design an expert system to generate synthetic expert demonstrations on collaborative scheduling for training. The expert system generates scheduling demonstrations by solving the optimization problem:

$$\min_{\mathbf{A},\mathbf{X},s_i,f_i} \left(\sum f_j + \frac{\sum_{i,j}(w_i^r - w_j^t)\mathbf{A}_{i,j}}{\sum_{i,j}(w_i^r + w_j^t)\mathbf{A}_{i,j}} \right), \forall \mathbf{t}_j \in \mathcal{T}, \forall \mathbf{r}_i \in \mathcal{R} \tag{7}$$

$$\text{s.t.} \sum_r \mathbf{A}_{i,j} = 1, \forall \mathbf{t}_j \in \mathcal{T}, \forall \mathbf{r}_i \in \mathcal{R} \tag{8}$$

$$f_i - s_i = d_i, \ \forall \mathbf{t}_i \in \mathcal{T} \tag{9}$$

$$s_i - \mathbf{P}_{ij} f_j \geq 0, \ \forall \mathbf{t}_i, \mathbf{t}_j \in \mathcal{T} \tag{10}$$

$$\left(\frac{\mathbf{c}_i^{r\top} \mathbf{c}_j^t}{\|\mathbf{c}_j^t\|_1} - \mathbf{A}_{i,j} \right) \geq 0 \ , \forall \mathbf{t}_j \in \mathcal{T}, \forall \mathbf{r}_i \in \mathcal{R} \tag{11}$$

$$(w_i^r - w_j^t)\mathbf{A}_{i,j} \geq 0, \ \forall \mathbf{t}_j \in \mathcal{T}, \forall \mathbf{r}_i \in \mathcal{R} \tag{12}$$

$$(s_j - f_i)\mathbf{A}_{k,i}\mathbf{A}_{k,j}\mathbf{X}_{i,j} \geq 0, \forall \mathbf{t}_i, \mathbf{t}_j \in \mathcal{T}, \forall \mathbf{r}_k \in \mathcal{R} \tag{13}$$

$$(s_i - f_j)\mathbf{A}_{k,i}\mathbf{A}_{k,j}(1 - \mathbf{X}_{i,j}) \geq 0, \forall \mathbf{t}_i, \mathbf{t}_j \in \mathcal{T}, \forall \mathbf{r}_k \in \mathcal{R} \tag{14}$$

where s_i and f_i are the start and finish times of the i-the task, w_r^r and w_t^t are the resources of the r-th robot and the t-th task, and d_i is the expected duration of the i-th task.

Algorithm 2: DBGM for Collaborative Scheduling with Failure Adaptation

Input : Problem tuple $\mathcal{M} = \{\mathcal{R}, \mathcal{T}, \mathbf{P}\}$
Output : Scheduling matrix $\mathbf{A}^*$

1: **while** *Not all tasks in $\mathcal{T}$ are assigned* **do**
2: Collect available robots $\mathcal{A}_{robot}$ and tasks $\mathcal{A}_{task}$;
3: **while** $\mathcal{A}_{robot} \neq Null$ *and* $\mathcal{A}_{task} \neq Null$ **do**
4: Check uncompletable tasks $\mathbf{t}_u$;
5: **if** $\mathbf{t}_u \neq Null$ **then**
6: Generate subteam pool S using Algorithm 1;
7: Compute $\mathbf{h}_1^t = \Psi(\mathcal{G}^u)$;
8: Compute $\{\mathbf{h}_j^r\} = \Phi^r(\mathbf{r}_j), j \in \mathcal{S}$;
9: **end**
10: **else**
11: Compute $\{\mathbf{h}_i^t\} = \Psi(\mathcal{G}^t), t \in \mathcal{A}_{task}$;
12: Compute $\{\mathbf{h}_j^r\} = \Phi(\mathbf{r}_j), j \in \mathcal{A}_{robot}$;
13: **end**
14: Compute masked reward $\mathbf{O}$ by Eq. (3);
15: Compute $\mathbf{A}_{i,j}^*$ by Eq. (6) for collaborative scheduling;
16: Update $\mathcal{A}_{task}$, $\mathcal{A}_{robot}$, and $\mathcal{T}$;
17: **end**
18: Update dependency matrix $\mathbf{P}$;
19: **if** *Task is completed* **then**
20: Update $\mathbf{P}$, $\mathcal{A}_{task}$, and $\mathcal{T}$;
21: **end**
22: **end**
23: **return** $\mathbf{A}^*$

Solving this optimization problem in Eqs. (7-14) provides the optimal $\mathbf{A}$, $\mathbf{X}$, s_i, and f_i, where $\mathbf{X}_{i,j} = 1$ encodes that the i-th task is completed before the j-th task starts. The objective function in Eq. (7) is a sum of two terms. The first term is used to minimize the overall time to complete all tasks. The second term is used to minimize the resource use for scheduled tasks, and the resource use is normalized between $[0, 1]$. We design eight constrains to generate high-quality demonstrations. Eq. (8) enforces that only one robot can be assigned to a task. Eq. (9) enforces positive task execution time. Eq. (10) enforces that task execution satisfies the task dependencies encoded by $\mathbf{P}$. Eq. (11) enforces that, if the j-th task is scheduled to the i-th robot , the robot must have all the capabilities required by the task. Eq. (12) enforces that each task must be executable by the corresponding scheduled robot. Eqs. (13-14) enforces that robots can only perform one task at a time. We use Gurobi [44] to solve this constrained optimization problem. Although the optimization in Eqs. (7-14) cannot be solved in real time (thus cannot be directly used for task scheduling in practice), it offers synthetic demonstration data for training DBGM.

We use imitation learning to learn the Q-network ψ in our DBGM method to imitate the expert scheduling policy. Given an expert demonstration, we decompose it into a sequence of state-action pairs $\mathbf{A}^{\mathrm{e}} \rightarrow \{(\mathbf{r}^{\mathrm{e}})_i, (\mathbf{t}^{\mathrm{e}})_j\}$, which encodes the expert scheduling of all tasks at different start time. To imitate demonstrations, we use the imitation loss function [39]:

$$L_{exp} = \|\mathbf{R}_{i,j} - \mathbf{Q}_{i,j}\|_1 \tag{15}$$

where $\mathbf{R}_{i,j} = \psi((\mathbf{h}^{\mathrm{e}})_i^r \| (\mathbf{h}^{\mathrm{e}})_j^t)$ denotes the reward computed by the DBGM's Q-network ψ that uses the embedding vectors

$(\mathbf{h}^{\mathrm{e}})_i^r = \Phi((\mathbf{r}^{\mathrm{e}})_i)$ and $(\mathbf{h}^{\mathrm{e}})_j^t = \Psi((\mathbf{t}^{\mathrm{e}})_j)$ as the input, and $\mathbf{Q}_{i,j}$ denotes the accumulated reward associated with the expert demonstration of assigning the j-th task to the i-th robot. We can compute it by $\mathbf{Q}_{i,j} = \sum_k \beta^k q_k$ during the period of time when the i-th robot executes the j-th task, where q denotes the immediate reward that can be computed by $q = \exp(\frac{w^r - w^t}{w^r})$, and $\beta \in (0, 1)$ is a discount factor. The loss in Eq. (15) trains ψ to generate a similar reward to the expert's reward.

In addition, we train our DBGM approach not to choose the scheduling that is not used in the expert demonstrations. Thus, we design another loss to enforce the reward of executing the schedules that are not selected by the expert is smaller than the expert's reward $\mathbf{Q}_{i,j}$, which can be defined as:

$$L_{non} = \|\mathbf{R}_{i,j} - \min\{\mathbf{R}_{i,j}, \mathbf{Q}_{i,j} - z\}\|_1 \tag{16}$$

where $\mathbf{R}_{i,j} = \psi((\mathbf{h}^{\mathrm{n}})_i^r \| (\mathbf{h}^{\mathrm{n}})_j^t)$ denotes the reward computed by the DBGM's Q-network ψ, whose input is the embedding vectors of the robots and tasks that are not used in the expert demonstrations. z is a constant to ensure that the reward of the scheduling not used by the expert is always smaller than the reward of the expert scheduling.

The final objective function designed to train the DBGM's Q-network ψ is as $L = L_{exp} + \lambda L_{non}$, where λ is a hyperparameter to balance the two loss functions. We use ADMM [45] as the optimization solver to train our Q-network ψ using expert demonstrations.

IV. Experiments

In this section, we present the setup of our extensive experiments, as well as discuss the experimental results on collaborative scheduling and team forming for failure adaptation.

A. Experimental Setups

We employ both synthetic data and Gazebo simulations in the Robot Operating System (ROS) to evaluate our approach. For training, we randomly generate 100 problems, which are solved by our expert system to generate demonstrations. The parameters used to generate the problems are as follows: (1) The number of robot capabilities is set to 2, including mobility and manipulation. (2) We consider payload to be the resource, and assume the payload required by robots and tasks to follow a uniform distribution $U(2, 20)$. (3) The task dependency is generated randomly and 20% of the tasks have dependencies. In our implementation, the GAT has two layers, and we assign the dimension of $\mathbf{W}^e$ to 32. We set the hyperparameter value $\lambda = 0.1$, and the discount factor $\beta = 0.01$.

We compare our approach with two methods, including (1) Randomized policy (RAND) that randomly schedules tasks to available robots and subteams that satisfy the task's capability and resource requirement; (2) Greedy resource policy (GRP) that schedules a task to the robot/subteam that satisfies the task's requirement and has the minimum resource. We employ three metrics for method evaluation and comparison, including (1) Idle Rate (IR) that is computed as the number of idle robots during the execution of all tasks, which indicates the efficiency of a schedule policy; (2) MakeSpan (MS) that is defined as the time of completing all tasks; and (3) Resource Cost (RC) that is computed as the cost of robot resources to complete all tasks. Smaller values indicate better performance for all of the three metrics.

TABLE I

Quantitative results of heterogeneous multi-robot collaborative scheduling using synthetic data.

Method	6r-18t			6r-100t			10r-100t		
	MS	IR	RC	MS	IR	RC	MS	IR	RC
RAND	6.06	18.36	75.06	24.95	49.72	311.00	14.95	49.49	277.20
GRP	5.83	16.98	59.15	22.10	32.60	274.40	14.00	40.00	236.35
Ours	5.73	16.38	57.57	21.05	26.30	266.50	13.25	32.50	236.70

B. Results on Multi-Robot Collaborative Scheduling

We first evaluate our approach in three synthetic scenarios, including (1) *6r-18t* that contains 200 problems with a team of 6 heterogenous robots and 18 tasks, (2) *6r-100t* that contains 20 problems with 6 robots and 100 tasks, and (3) *10r-100t* that contains 20 problems with 10 robots and 100 tasks.

The quantitative results are shown in Table I. In the *6r-18t* scenario, we observe that RAND performs poorly, and GRP performs much better as it greedily minimizes resource cost. However, since GRP does not consider task dependency and robot heterogeneity, it does not perform well on MS and IR. In this scenario, our method obtains the best performance over all metrics. For large problems, since the expert system cannot provide a solution within reasonable time, we use the DBGM model trained in the scenario of *6r-18t*, and evaluate it in the scenarios of *6r-100t* and *10r-100t*. In the scenario of *6r-100t*, our approach outperforms the baseline methods on all metrics. In the scenario of *10r-100t*, our method performs similarly to GRP on the RC metric, and significantly outperforms GRP on MS and IR. These results indicate that our approach generalized well to unseen scenarios to address large problems.

We further evaluate our approach using Gazebo simulations in ROS. In these multi-robot simulations, a team of heterogeneous robots are simulated to pick and transport a collection of boxes. The robot team includes 3 Jackal robots and 2 Husky robots (from Clearpath Robotics), with a payload capacity of 5 and 12 units, respectively. The simulation includes 18 tasks, and all have dependencies. The boxes involved in these tasks require the payload capability between 2 and 8 units. As some boxes require >5 units of payload resources, they cannot be transported by individual Jackal robots, but a subteam of two Jackal robots provide sufficient payload to transport them. In the Gazebo simulations, scheduling policies are computed in a centralized way, which are sent to each robot or subteam to execute their scheduled tasks.

Figure 3 depicts an example of the qualitative results using Gazebo simulations in ROS to demonstrate our approach for multi-robot heterogeneous collaborative scheduling. We can observe that the boxes are assigned to appropriate robots, and the robots go back to the standby area only after they complete all tasks. In addition, we quantitatively evaluate our approach in these Gazebo simulations, and compare it with GRP. The GRP approach obtains an average of MS $= 192.9$ and RC $=$ 67. Our approach achieves an average of MS $= 170.2$ and

Fig. 3. An example of the qualitative results obtained using Gazebo simulations in ROS to evaluate and demonstrate our method of collaborative scheduling for a team of heterogeneous robots to collaboratively pick and transport a collection of boxes.

TABLE II

RESULTS OF FORMING SUBTEAMS OF TWO ROBOTS TO EXECUTE INDIVIDUALLY UNCOMPLETABLE TASKS.

Required	16	18	20	22	24	26	28
Subteam	[3, 4]	[1, 4]	[3, 6]	[3, 6]	[1, 6]	[1, 3]	[1, 2]
Payload	[12, 5]	[14, 5]	[12, 10]	[12, 10]	[14, 10]	[14, 12]	[14, 14]

RC $= 60$, which greatly outperforms GRP over both metrics. This is because our method learns from expert demonstrations that explicitly consider both execution time and resource use (e.g., encoded by the two terms in Eq. (7)). Also, our approach achieves an average of IR $= 7$, which significantly outperform GRP that obtains an average of IR $= 12$.

C. Results on Forming Subteams for Failure Adaptation

We define 10 robot failure cases to evaluate our approach's performance on adaptation to robot failure that causes individually uncompletable tasks during the scheduling process. For example, when the robot with the largest payload capacity fails, the payload required by a task is greater than the payload capacity of all remaining individual robots. In this situation, robots may have to form subteams to collaboratively address the individually uncompletable tasks.

In this set of experiments, we simulate a team of 6 heterogeneous robots with payload capacities of $[14, 14, 12, 5, 5, 10]$. The experimental results are shown in Table II, in which the first row shows the payload required by the task, the second row shows indices of the robots that form a subteam, and the third row presents the respective payload capacity of the two robots in the subteam. We observe that our approach is able to form a subteam to execute an individually uncompletable task. For example, for a task that requires 18 units of payload, no robots can individually execute the task; our approach selects robots 1 and 4 (with the payload of 14 and 5 units respectively) to execute the task with minimum payload use. When robots with a large payload capacity fail, possible combinations of robots with a small payload capability are explored by our method to form subteams in order to replace the failed robots.

We further investigate our method's execution speed given various number of tasks, number of robots, and subteam size. The results of execution speed are obtained by averaging the execution time of 20 problems on a Linux machine with an i7

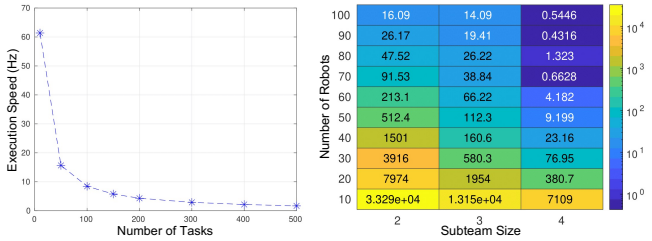

Fig. 4. Results on execution speed (Hz) of our approach given various (a) number of tasks and (b) number of robots and subteam size.

16-core CPU and 16G memory. As illustrated in Figure 4(a), the execution speed monotonically decreases as the number of tasks increases. When the number of tasks is 18, our approach runs at 50 Hz. When the number of tasks is 100, the execution speed is 10 Hz, and the speed drops to 2 Hz when the number of tasks is 500. From Figure 4(b), we observe that when the subteam size is smaller than 3, our approach obtains promising execution speeds, with a minimum speed of 14.09 Hz when the team includes 100 robots. When the number of robots is smaller than 50 and the subteam size is smaller than 3, our approach obtains the execution speed of more than 100 Hz. For a larger problem that includes a number of 100 robots with subteams that include 4 robots, our approach obtains an execution speed of around 0.5 Hz.

V. CONCLUSION

In this paper, we have proposed a novel approach that integrates deep bipartite graph matching and imitation learning to perform heterogeneous multi-robot collaborative scheduling with adaptation to robot failure. We formulate collaborative scheduling as a bipartite graph matching problem. During normal execution, our approach encodes complex robot and task attributes based on deep graph learning. In robot failure cases, our approach generates a pool of potential subteams to replace the failed robots based upon its capability and capacity. The policy network is trained through imitation learning with expert schedules provided by our designed expert solver. Extensive experimental results show that our approach achieves generalizable and scalable results on heterogeneous multi-robot collaborative scheduling with robot failure adaptation.

REFERENCES

[1] S. Vorotnikov, K. Ermishin, A. Nazarova, and A. Yuschenko, "Multi-agent robotic systems in collaborative robotics," in *International Conference on Interactive Collaborative Robotics*, 2018.

[2] F. Vicentini, "Collaborative robotics: a survey," *Journal of Mechanical Design*, vol. 143, no. 4, 2021.

[3] B. Reily, C. Reardon, and H. Zhang, "Leading multi-agent teams to multiple goals while maintaining communication," in *Robotics: Science and Systems*, 2020.

[4] N. Fung, J. Rogers, C. Nieto, H. I. Christensen, S. Kemna, and G. Sukhatme, "Coordinating multi-robot systems through environment partitioning for adaptive informative sampling," in *International Conference on Robotics and Automation*, 2019.

[5] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191 617–191 643, 2020.

[6] S. Zhang, Y. Chen, J. Zhang, and Y. Jia, "Real-time adaptive assembly scheduling in human-multi-robot collaboration according to human capability," in *IEEE International Conference on Robotics and Automation*, 2020.

[7] P. Gao and H. Zhang, "Bayesian deep graph matching for correspondence identification in collaborative perception," in *Robotics: Science and Systems*, 2021.

[8] E. Matheson, R. Minto, E. G. Zampieri, M. Faccio, and G. Rosati, "Human–robot collaboration in manufacturing applications: a review," *Robotics*, vol. 8, no. 4, p. 100, 2019.

[9] Z. Li, A. V. Barenji, J. Jiang, R. Y. Zhong, and G. Xu, "A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand," *Journal of Intelligent Manufacturing*, vol. 31, no. 2, pp. 469–480, 2020.

[10] P. Gao, B. Reily, S. Paul, and H. Zhang, "Visual reference of ambiguous objects for augmented reality-powered human-robot communication in a shared workspace," in *International Conference on Human-Computer Interaction*, 2020.

[11] Y. Huang, Y. Zhang, and H. Xiao, "Multi-robot system task allocation mechanism for smart factory," in *IEEE 8th Joint International Information Technology and Artificial Intelligence Conference*, 2019.

[12] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *AAAI Conference on Artificial Intelligence*, 2015.

[13] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[14] E. Suslova and P. Fazli, "Multi-robot task allocation with time window and ordering constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[15] P. Ghassemi, D. DePauw, and S. Chowdhury, "Decentralized dynamic task allocation in swarm robotic systems for disaster response," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*, 2019.

[16] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *IEEE International Conference on Robot & Human Interactive Communication*, 2021.

[17] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *International Conference on Learning Representations*, 2018.

[18] Z. Wang, C. Liu, and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," *Autonomous Robots*, pp. 1–20, 2021.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *International Conference on Representation Learning*, 2018.

[20] 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.

[21] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2530–2537, 2020.

[22] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in Neural Information Processing Systems*, 2018.

[23] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *IEEE International Conference on Robotics and Automation*, 2018.

[24] A. Dutta and A. Asaithambi, "One-to-many bipartite matching based coalition formation for multi-robot task allocation," in *IEEE International Conference on Robotics and Automation*, 2019.

[25] P. Mazdin and B. Rinner, "Distributed and communication-aware coalition formation and task assignment in multi-robot systems," *IEEE Access*, vol. 9, pp. 35 088–35 100, 2021.

[26] H. Baek, K. G. Shin, and J. Lee, "Response-time analysis for multi-mode tasks in real-time multiprocessor systems," *IEEE Access*, vol. 8, pp. 86 111–86 129, 2020.

[27] H. Wang, W. Chen, and J. Wang, "Coupled task scheduling for heterogeneous multi-robot system of two robot types performing complex-schedule order fulfillment tasks," *Robotics and Autonomous Systems*, vol. 131, p. 103560, 2020.

[28] S. Mayya, D. S. D'antonio, D. Saldaña, and V. Kumar, "Resilient task allocation in heterogeneous multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1327–1334, 2021.

[29] P. Ghassemi and S. Chowdhury, "Multi-robot task allocation in disaster response: Addressing dynamic tasks with deadlines and robots with range and payload constraints," *Robotics and Autonomous Systems*, vol. 147, p. 103905, 2022.

[30] J. Motes, R. Sandström, H. Lee, S. Thomas, and N. M. Amato, "Multi-robot task and motion planning with subtask dependencies," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3338–3345, 2020.

[31] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann, "Multi-robot task allocation and scheduling considering cooperative tasks and precedence constraints," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2020.

[32] B. Reily and H. Zhang, "Team assignment for heterogeneous multi-robot sensor coverage through graph representation learning," in *IEEE International Conference on Robotics and Automation*, 2021.

[33] M. Gombolay, R. Wilcox, and J. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints," *Robotics: Science and Systems*, 2013.

[34] X. Chen, P. Zhang, G. Du, and F. Li, "A distributed method for dynamic multi-robot task allocation problems with critical time constraints," *Robotics and Autonomous Systems*, vol. 118, pp. 31–46, 2019.

[35] 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*, pp. 1–17, 2021.

[36] X. Chen, P. Zhang, G. Du, and F. Li, "Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems," *IEEE Access*, vol. 6, pp. 21 745–21 757, 2018.

[37] S. M. Mousavi and R. Tavakkoli-Moghaddam, "A hybrid simulated annealing algorithm for location and routing scheduling problems with cross-docking in the supply chain," *Journal of Manufacturing Systems*, vol. 32, no. 2, pp. 335–347, 2013.

[38] L. Zhang and T. Wong, "An object-coding genetic algorithm for integrated process planning and scheduling," *European Journal of Operational Research*, vol. 244, no. 2, pp. 434–444, 2015.

[39] 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.

[40] M. Strens and N. Windelinckx, "Combining planning with reinforcement learning for multi-robot task allocation," in *Adaptive Agents and Multi-Agent Systems II*, 2004, pp. 260–274.

[41] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," *Advances in Neural Information Processing Systems*, 2019.

[42] F. Duvallet and A. Stentz, "Imitation learning for task allocation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3568–3573.

[43] M. D. Plummer and L. Lovász, *Matching Theory*, 1986.

[44] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[45] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.