2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Learning for Dynamic Subteaming and Voluntary Waiting in Heterogeneous Multi-Robot Collaborative Scheduling

Williard Joshua Jose and Hao Zhang

***Abstract*— Coordinating heterogeneous robots is essential for autonomous multi-robot teaming. To execute a set of dependent tasks as quickly as possible, and to complete tasks that cannot be addressed by individual robots, it is necessary to form subteams that can collaboratively finish the tasks. It is also advantageous for robots to wait for teammates and tasks to become available in order to form better subteams or reduce the overall completion time. To enable both abilities, we introduce a new graph learning approach that formulates heterogeneous collaborative scheduling as a bipartite matching problem that maximizes a reward matrix learned via imitation learning. We design a novel graph attention transformer network (GATN) that represents the problem of collaborative scheduling as a bipartite graph, and integrates both local and global graph information to estimate the reward matrix using graph attention networks and transformers. By relaxing the constraint of one-to-one correspondence in bipartite matching, our approach allows multiple robots to address the same task as a subteam. Our approach also enables voluntary waiting by introducing an idle task that the robots can select to wait. Experimental results have shown that our approach well addresses heterogeneous collaborative scheduling with dynamic subteam formation and voluntary waiting, and outperforms the previous and baseline methods.**

I. INTRODUCTION

Collaborative multi-robot teaming has been widely studied over the past decades to address a broad range of real-world applications [1], [2], [3], [4], such as space exploration [5], connected driving [6], [7], search and rescue [8], [9], and hospital and warehouse assistance [10], [11]. To scalably and efficiently deploy multi-robot teams, collaborative scheduling is essential for the robots to both autonomously allocate and collaboratively complete complex tasks, often with the goal of minimizing the overall task completion time. Collaborative scheduling must address the inherent heterogeneity in both robots and tasks. Collaborative robots may be heterogeneous, equipped with different sensors and actuators with varying payloads, e.g., in the scenario of multi-robot assisted manufacturing in Fig. 1. Each task to be scheduled may also have different requirements on the robot's capabilities (e.g., mobility and manipulation) and capacities (e.g., payload).

Given its importance, various techniques were implemented to address heterogeneous collaborative scheduling. Classical methods are designed based on heuristics [12], [13]. However, these techniques typically provide greedy solutions and cannot always lead to optimal results. Several other techniques use integer linear programming in collaborative scheduling to optimize the overall time to complete all tasks [14], [15], [16]. However, these methods cannot run in real-time when solving problems with complex constraints caused by robot/task heterogeneity and task dependency. Recently, learning-based methods have shown impressive promise to balance both scheduling performance and algorithm runtime [17], [18], [19]. However, there still exist several limitations to the state-of-the-art learning methods for heterogeneous collaborative scheduling. First, most existing learning methods cannot build subteams to combine the strengths of individual robots to complete the tasks that cannot be tackled by each individual robot. Second, previous approaches generally do not consider situations where a robot voluntarily determines to wait in order to shorten the overall task completion time. In particular, voluntary waiting and subteam formation are interrelated, i.e., forming subteams may require robots to voluntarily wait for other teammates to become available. The interplay between subteam formation and voluntary waiting has not been studied yet by existing learning-based methods.

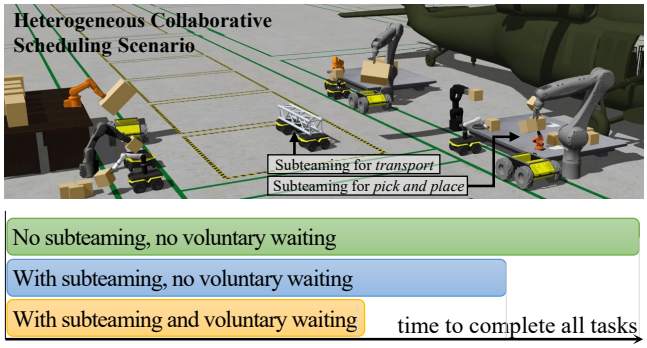

Fig. 1. A motivating scenario for heterogeneous multi-robot collaborative scheduling with dynamic subteaming and voluntary waiting in an assembly application. Subteaming allows robots to dynamically build subteams to address tasks that cannot be completed by individual robots, while voluntary waiting enables robots to wait for additional robots and tasks to be available in order to form better subteams or reduce the overall task completion time.

In this paper, we introduce a learning-based approach called *Learning for Voluntary Waiting and Subteaming* (LVWS), which enables robots to voluntarily wait and dynamically form subteams in order to minimize the overall task completion time in real-time heterogeneous collaborative scheduling. Our approach uses an undirected graph to encode a team of robots, whose nodes represent the heterogeneous robots with different capabilities and payload capacities, and whose edges represent the relationships of the robots (e.g., communication connections). Similarly, our method represents tasks and their

*This work was partially supported by NSF CAREER Award IIS-2308492 and DARPA Young Faculty Award (YFA) D21AP10114-00.

Williard Joshua Jose and Hao Zhang are with the Human-Centered Robotics Laboratory in the Manning College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01002, USA. Email: {wjose,hao.zhang}@umass.edu.

dependencies using a directed graph. Then, we design a new graph attention transformer network (GATN) to compute feature embeddings of the robots and tasks, which uses an attention mechanism for each node to fuse information from its connected neighbors, and employs a transformer for each node to encode contextual information (e.g., node position in the graph). Heterogeneous collaborative scheduling is then formulated as a deep bipartite graph matching problem, which maximizes a reward matrix computed using the robot and task embeddings. Our approach learns to group robots into a subteam that receives increased rewards when the subteam is assigned an available task. Our approach also implements voluntary waiting by designing an idle task that each robot can choose to wait for forming subteams and receiving better rewards at a later time.

The main contribution of this work is the introduction of our novel learning-based LVWS method to address heterogeneous collaborative scheduling. Two specific novelties include:

- We propose a novel graph attention transformer network to learn representations for collaborative scheduling from graphs of heterogeneous robots and complex tasks. This GATN can learn on graphs with arbitrary structures and is agnostic to the number of robots and tasks.
- We enable two interrelated abilities for robot teammates to dynamically form subteams and voluntarily wait under a unified learning-based method. These abilities not only allow a robot team to improve the overall task completion time, but also enable subteams to tackle tasks that cannot be completed by individual robots.

II. RELATED WORK

Collaborative multi-robot task scheduling is a family of problems characterized by different robot and task constraints [20], [21], [22]. Robot constraints can include whether the robots can perform one or more tasks simultaneously (single-task vs. multiple-task robots) and support one or more physical capabilities with varying payloads (heterogeneous robots). Task constraints can include task durations, task capability requirements, task order and precedence requirements, and whether tasks are fixed or can change over time.

A. Single-Task Single-Robot (ST-SR) Scheduling

Several previous approaches addressed ST-SR scenarios for single-task robots and single-robot tasks, which are usually solved with heuristics [23], greedy methods [12], [24], search algorithms [25], [26], and integer linear programming [27], [28]. However, heuristic-based and greedy approaches are not guaranteed to obtain an optimal solution. Also, integer linear programming-based approaches have exponential time complexity and cannot be used to address real-world large collaborative scheduling problems in practice [29]. Recently, there have been learning-based methods proposed to solve multi-robot scheduling. These include reinforcement learning (RL) [17], [30] and inverse RL [31], [32]. These approaches perform better than the heuristic and greedy methods and run faster than integer linear programming techniques. However, these learning-based approaches generally cannot well address the scenarios with complex heterogeneous robot capability constraints and temporal task constraints.

B. Single-Task Multiple-Robot (ST-MR) Scheduling

Several methods were implemented to address the scenarios with single-task robots and multiple-robot tasks where two or more robots can complete tasks jointly. Most existing works utilized homogeneous robots to complete tasks with larger payloads cooperatively [21], [33]. In addition, several methods were implemented for heterogeneous robot scheduling to perform perception and action tasks [34], [35], and mitigate unexpected failures [19], [36], [28]. However, they limit robot subteam assignment to scenarios with short time horizons or no temporal task dependencies. Recently, learning-based methods were developed to encode tasks as directed graphs [32], [37], which permits reasoning over longer time horizons while respecting task prerequisites in the solution. Scheduling delays were also implemented for multiple agents to improve load management for reliability [38] and resource utilization in ride-sharing scenarios [39]. However, the previous learning-based methods cannot learn to form subteams [38], [39], or use heuristics [19], [28] to select agents for subteam formation. In heterogeneous collaborative scheduling, subteaming and waiting are interrelated. However, the research problem of unified learning to jointly enable subteaming and waiting has not been well studied yet.

III. APPROACH

Notation. We denote matrices as boldface uppercase letters (e.g., $\mathbf{M} = \{M_{i,j}\} \in \mathbb{R}^{n \times m}$, an $n \times m$ matrix whose element in the i-th row and j-th column is $M_{i,j}$), vectors as boldface lowercase letters (e.g., $\mathbf{v} \in \mathbb{R}^d$, a d-dimensional vector whose i-th element is v_i), and scalars as lowercase letters (e.g., s).

A. Problem Definition

Given a team of N heterogeneous robots, we represent the team as an undirected graph $\mathcal{G}^r = (\mathcal{R}, \mathbf{Q})$, where $\mathcal{R} = \{\mathbf{r}_i\}^N$ represents the set of vertices, and $\mathbf{Q} \in \{0,1\}^{N \times N}$ is a matrix denoting the edges to represent the network connection among the robots. The vertex $\mathbf{r}_i = [a_i^r, \mathbf{c}_i^r, w_i^r]$ represents the states of the i-th robot, which includes three variables to indicate the i-th robot's availability, heterogeneous capabilities (e.g., manipulation and mobility), and payload capacity, respectively. For example, $[a_i^r = 1, \mathbf{c}_i^r = [1,1], w_i^r = 20]$ indicates that the i-th robot is available, can both manipulate and move, and has a payload capacity of 20 units. In this work, we assume that all robots in the team are connected, i.e., $\mathbf{Q}$ is a matrix of ones.

Similarly, in order to represent a set of M dependent tasks, we implement a directed acyclic graph $\mathcal{G}^t = (\mathcal{T}, \mathbf{P})$. The set of the graph's vertices $\mathcal{T} = \{\mathbf{t}_j\}^M$ denotes the tasks, and each $\mathbf{t}_j = [\mathbf{s}_j^t, \mathbf{c}_j^t, w_j^t]$ includes three elements, where $\mathbf{s}_j^t \in \{0,1\}^3$ encodes if the j-th task is ready, assigned, and incomplete, $\mathbf{s}_j^t$ is an indicator vector encoding the required robot capabilities, and w_j^t indicates the minimum payload required to complete the task. For example, $\mathbf{s}_j^t = [1,0,1]$ indicates that the j-th task is queued, but not assigned or completed; $\mathbf{c}_j^t = [1,1]$ indicates

that the task requires a robot or subteam to have both the manipulation and mobility capabilities; and $w_j^t = 25$ indicates that the task requires the robot or subteam to have a minimum payload of 25 units. Task dependencies are represented by the directed edges $\mathbf{P}^{M\times M}$, where $P_{i,j} = 1$ represents that the i-th task is a dependency of the j-th task. The j-th task is considered ready, i.e., $(s_j^t)_1 = 1$, if all its dependent tasks have been completed.

Then, we formally represent the problem of heterogeneous multi-robot collaborative scheduling using a bipartite graph $\mathcal{G} = \{\mathcal{G}^r, \mathcal{G}^t, \mathbf{A}\}$, and mathematically formulate it as a deep bipartite graph matching problem that sequentially allocates tasks $\mathbf{s}_i^t$ to individual and subteams of robots in the team $\mathcal{R}$, where the allocation is encoded by the scheduling matrix $\mathbf{A} = \{A_{i,j}\}^{N\times M}$ with $A_{i,j} = 1$ indicating that the j-th task $\mathbf{t}_j \in \mathcal{T}$ is allocated to robot $\mathbf{r}_i \in \mathcal{R}$ at a timestep. Given this problem formulation, our proposed LVWS approach aims to achieve the following objectives under a unified deep learning framework:

- Learn to estimate the scheduling matrix $A_{i,j}$ as a policy to schedule tasks to not only individual robots but also subteams of robots in order to minimize the overall task completion time. For example, in the case of allocating the j-th task to a subteam that includes robots i and k, we have $A_{i,j} = A_{k,j} = 1$.
- Learn for each robot to voluntarily wait for other robots and tasks to become available, in order to form subteams to reduce the overall task completion time and/or address tasks that cannot be completed by individual robots.

B. LVWS for Heterogeneous Collaborative Scheduling

We develop our novel LVWS approach, as demonstrated in Fig. 2, to enable voluntary waiting and dynamic subteaming for heterogeneous collaborative scheduling. Specifically, we design a graph attention transformer network that integrates graph networks (to encode local graph structure) and transformers (to encode contextual information). Given a graph, a graph attention network (GAT) [40], denoted as $\Psi(\cdot)$, is used to compute the embedding of a node by the l-th layer of the GAT, as follows:

$$\mathbf{h}_i^{(l+1)} = \alpha_{i,i}\mathbf{W}\mathbf{h}_i^{(l)} + \text{LeakyReLU}\left(\sum_{P_{i,j}\neq 0} \alpha_{i,j}\mathbf{W}\mathbf{h}_j^{(l)}\right) \tag{1}$$

where $\mathbf{W}$ denote the parameters of $\Psi(\cdot)$, and $\alpha_{i,j}$ is the GAT attention function that can be computed by:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{W}\mathbf{h}_i^{(l)} \parallel \mathbf{W}\mathbf{h}_j^{(l)})\right)}{\sum_{P_{i,j}\neq 0}\exp\left(\text{LeakyReLU}(\mathbf{W}\mathbf{h}_i^{(l)} \parallel \mathbf{W}\mathbf{h}_j^{(l)})\right)} \tag{2}$$

where $\parallel$ denotes a concatenation operator, and LeakyReLU$(\cdot)$ is a leaky rectified linear unit. Given the robot team graph $\mathcal{G}^r = (\mathcal{R}, \mathbf{Q})$, the GAT $\Psi^r(\cdot)$ calculates node embeddings for the robots, where $\mathcal{H}^r = \Psi^r(\mathcal{G}^r) = \{\mathbf{h}_i^r\}^N$. Moreover, given the task graph $\mathcal{G}^t = (\mathcal{T}, \mathbf{P})$, the GAT $\Psi^t(\cdot)$ computes node embeddings for the tasks, where $\mathcal{H}^t = \Psi^t(\mathcal{G}^t) = \{\mathbf{h}_j^t\}^M$.

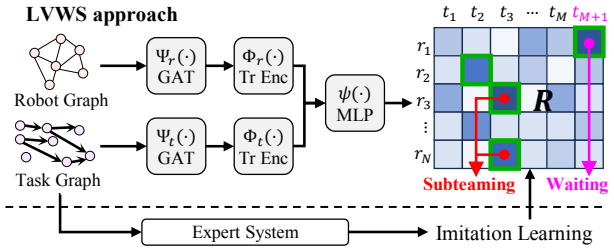

Fig. 2. Overview of our LVWS approach. By maximizing a scheduling reward matrix $\mathbf{R}$ that is estimated using imitation learning, LVWS integrates GATs and transformers to learn a policy to determine robot-task assignments. LVWS enables subteaming by grouping multiple robots that collectively meet a task's requirements on capabilities and payload capacities. LVWS enables voluntary waiting by allowing robots to select the idle task t_{M+1}.

To represent the global contextual information of the entire team and all the tasks, we introduce a transformer encoder [41], [42], denoted as $\Phi(\cdot)$. For a graph $\mathcal{G}$, given its vertex embeddings $\mathcal{H} = \{\mathbf{h}_j\}$ computed from the GAT $\Psi(\mathcal{G})$, the transformer $\Phi(\cdot)$ first integrates all the embeddings to update the global embedding $\mathbf{g}$ at the l-th layer by:

$$\bar{\mathbf{g}}^{(l)} = \text{LN}\left(\sum_j \left(\beta_j \mathbf{U}_1 \mathbf{h}_j^{(l)}\right) + \mathbf{g}^{(l)}\right) \tag{3}$$

where $\mathbf{U}_1$ are the parameters, LN$(\cdot)$ is the layer normalization operation, and β_j is the scaled self-attention function [41], [43] that can be computed by:

$$\beta_j = \text{softmax}\left(\frac{\mathbf{U}_2\mathbf{g}\cdot\mathbf{U}_3\mathbf{h}_j}{\sqrt{\dim(\mathbf{g})}}\right) \tag{4}$$

where softmax$(\cdot)$ is the softmax function, $\mathbf{U}_2$ and $\mathbf{U}_3$ are its parameters, and $\dim(\mathbf{g})$ is the dimensionality of $\mathbf{g}$. Then, the global embedding $\mathbf{g}$ is propagated across layers by:

$$\mathbf{g}^{(l+1)} = \text{LN}\left(\text{MLP}(\bar{\mathbf{g}}^{(l)}) + \bar{\mathbf{g}}^{(l)}\right) \tag{5}$$

where MLP$(\cdot)$ is a multi-layer perceptron network. Given the robot graph $\mathcal{G}^r$, we compute its embedding $\mathbf{g}^r = \Psi(\mathcal{G}^r)$ to encode the global team information. Similarly, given the task graph $\mathcal{G}^t$, we compute its embedding $\mathbf{g}^t = \Psi(\mathcal{G}^t)$ to encode the status and dependencies of all tasks. Besides the global graph embeddings, the same transformer layers in Eqs. (3-5) are also applied to update the local node embeddings $\{\mathbf{h}_i^r\}^N$ and $\{\mathbf{h}_j^t\}^M$ for each robot and task, respectively.

Given the bipartite graph $\mathcal{G} = \{\mathcal{G}^r, \mathcal{G}^t, \mathbf{A}\}$ and the embeddings computed from our GATN, heterogeneous collaborative scheduling can be formally formulated as a bipartite graph matching problem, which determines the scheduling matrix $\mathbf{A}$ by maximizing the rewards associated with the edges between $\mathcal{R}$ and $\mathcal{T}$ in the bipartite graph $\mathcal{G}$. In order to enable ***voluntary waiting***, we introduce a new idle task that can be selected by each robot to wait without taking other tasks. This idle task is modeled as a vertex $\mathbf{t}_{M+1} \in \mathcal{T}$ that is not connected with other vertices in $\mathcal{T}$. Then, taking into account voluntary waiting modeled by $\mathbf{t}_{M+1}$, we introduce the reward matrix $\mathbf{R} = \{R_{i,j}\}^{N\times(M+1)}$ that is associated with the scheduling

matrix $\mathbf{A} = \{A_{i,j}\}^{N\times(M+1)}$ in $\mathcal{G}$. $R_{i,j}$ indicates the reward when the i-th robot selects the j-th task, which is defined by:

$$R_{i,j} = \psi\left(\mathbf{h}_i^r \parallel \mathbf{h}_j^t \parallel \mathbf{g}^r \parallel \mathbf{g}^t\right) \tag{6}$$

where $\psi(\cdot)$ is a MLP to calculate $R_{i,j}$ from the embeddings. $R_{i,j}$ integrates not only the local information of the i-th robot and j-th task (through $\mathbf{h}_i^r \parallel \mathbf{h}_j^t$), but also the global context of the graphs $\mathcal{R}$ and $\mathcal{T}$ (through $\mathbf{g}^r \parallel \mathbf{g}^t$).

In addition to voluntary waiting, we enable the new capability of dynamic ***subteaming***. First, we relax the requirement of one-to-one correspondence in classic bipartite graph matching to allow that same task to be scheduled to multiple robots. Second, we require that the multiple robots assigned to a task must satisfy the task's requirements on the capabilities, which can be modeled by the constraint $\bigvee_i A_{i,j}\mathbf{c}_i^r \cdot (\mathbf{c}_j^t)^\top = \|\mathbf{c}_j^t\|_1$, where $\vee$ denotes the disjunction operator that is a product of ORs. Third, we require that the multiple robots assigned to a task must satisfy its requirement on the payload capacity, so that the subteam has sufficient resources to complete the task, which can be mathematically modeled by the constraint $\sum_i A_{i,j} w_i^r \geq w_j^t$.

To enable the new capabilities of subteaming and voluntary waiting, we formulate bipartite matching for heterogeneous collaborative scheduling as a constrained optimization problem defined as:

$$\mathbf{A}^* = \underset{\mathbf{A}}{\operatorname{argmax}} \sum_{i,j} A_{i,j} R_{i,j} \tag{7}$$

$$\forall \mathbf{r}_i \in \mathcal{R} | a_i^r = 1, \forall \mathbf{t}_j \in \mathcal{T} | (s_j^t)_1 = 1 \tag{8}$$

$$\text{s.t.} \sum_j A_{i,j} \leq 1 \tag{9}$$

$$\left(\bigvee_i A_{i,j}\mathbf{c}_i^r\right) \cdot (\mathbf{c}_j^t)^\top = \|\mathbf{c}_j^t\|_1 \tag{10}$$

$$\sum_i A_{i,j} w_i^r \geq w_j^t \tag{11}$$

The objective of this optimization problem in Eq. (7) is to compute the scheduling matrix $\mathbf{A}^*$ as a policy by maximizing the overall reward in order to assign ready tasks (represented by $\forall \mathbf{t}_j \in \mathcal{T} | (s_j^t)_1 = 1$) to individual or a subteam of robots that are available (represented by $\forall \mathbf{r}_i \in \mathcal{R} | a_i^r = 1$). Eq. (9) is the constraint requiring that each robot can take at most one task. Eqs. (10) and (11) are constraints on the capability and payload capacity requirements in order to enable subteaming.

C. Training LVWS through Imitation Learning

Since LVWS is a deep graph learning approach, we need a large amount of training data to learn all model parameters of the network components, including GATs, transformers, and MLPs. Since it is difficult to manually label the optimal task-robot assignment for all possible scenarios, we adopt imitation learning that uses an expert system to provide demonstrations as our training data. Following previous methods on imitation learning to generate scheduling demonstrations [44], [45], the expert system is implemented using dynamic programming, which divides the scheduling problem into multiple smaller scheduling sub-problems by removing feasible robot/subteam-task assignments. This expert system has an exponential time complexity [19] and cannot be executed in real-time (thus, it is not practical for real-time task scheduling).

With the scheduling demonstrations from the expert system as the training data, the objective function of using imitation learning to train LVWS can be defined as:

$$\mathcal{L} = \|(\mathbf{R} - \mathbf{E}) \circ \mathbf{X}\|_1 + \lambda \|\mathbf{R} - \mathbf{E}\|_1 \tag{12}$$

where the hyperparameter λ balances the two terms. The first term is a loss that is designed to train LVWS to estimate a reward $\mathbf{R}$ similar to the expert reward $\mathbf{E} = \{E_{i,j}\}^{N\times(M+1)}$. Each $E_{i,j}$ is an accumulated expert reward for scheduling the j-th task to the i-th robot in the demonstrations, which is computed as $S_{i,j} = \sum_k \gamma^k e_k$, where e_k is the immediate expert reward for finishing the tasks at timestep k and γ is a reward discount factor. $\mathbf{X} = \{X_{i,j}\}^{N\times(M+1)}$ is a mask to enforce learning positive rewards from feasible assignment demonstrations, where $X_{i,j} = 1$ if it is feasible to schedule the j-th task to the i-th robot, and $X_{i,j} = 0$ otherwise. The second term in Eq. (12) is developed for our approach not to learn from the assignments that are not shown in the expert demonstrations, by making our model receive zero rewards for invalid robot-task assignments. In order to train our LVWS approach using the objective function in Eq. (12), we adopt gradient descent based on the ADAM optimizer [46], [47].

D. Time Complexity Analysis

We analyze the time complexity of LVWS with respect to the number of robots N and the number of tasks M. Constructing the robot and task graphs requires $O(N^2)$ and $O(M^2)$ time complexity, respectively. If the GAT has L_g layers and K_g attention heads, computing the embeddings using the GATs takes $O(N^2 L_g K_g)$ and $O(M^2 L_g K_g)$ for the robot and task graphs, respectively. Similarly, applying the transformer encoders requires $O(N^2 L_{tr} K_{tr})$ and $O(M^2 L_{tr} K_{tr})$ time complexity, respectively, where L_t is the number of layers and K_t is the number of attention heads in the transformer encoder. Computing the reward matrix has a complexity of $O(NML_m)$, where L_m is the number of layers of the MLP. Solving the optimization problem in Eqs. (7-11) requires to iterate through all elements of $\mathbf{R}$, and sequentially select the robot-task assignment with maximum reward, which has a time complexity of $O(NM^2)$.

Combining these together, the dominating term in the time complexity per timestep is $O(N^2 + NM^2)$. At each timestep where a decision on robot-task assignments must be made, $\mathbf{R}$ must be dynamically computed using the graphs $\mathcal{G}^r$ and $\mathcal{G}^t$ at the timestep. The number of timesteps T depends on how many tasks can be completed simultaneously. In the worst case, only one task can be completed at a time, which leads to $\lceil M/N \rceil \leq T \leq M$. Therefore, the overall worst-case time complexity of our approach is $O\left((N^2 + NM^2)T\right)$.

IV. EXPERIMENTS

In our experments, we evaluate the LVWS approach using synthetic data and Gazebo simulations in the Robot Operating

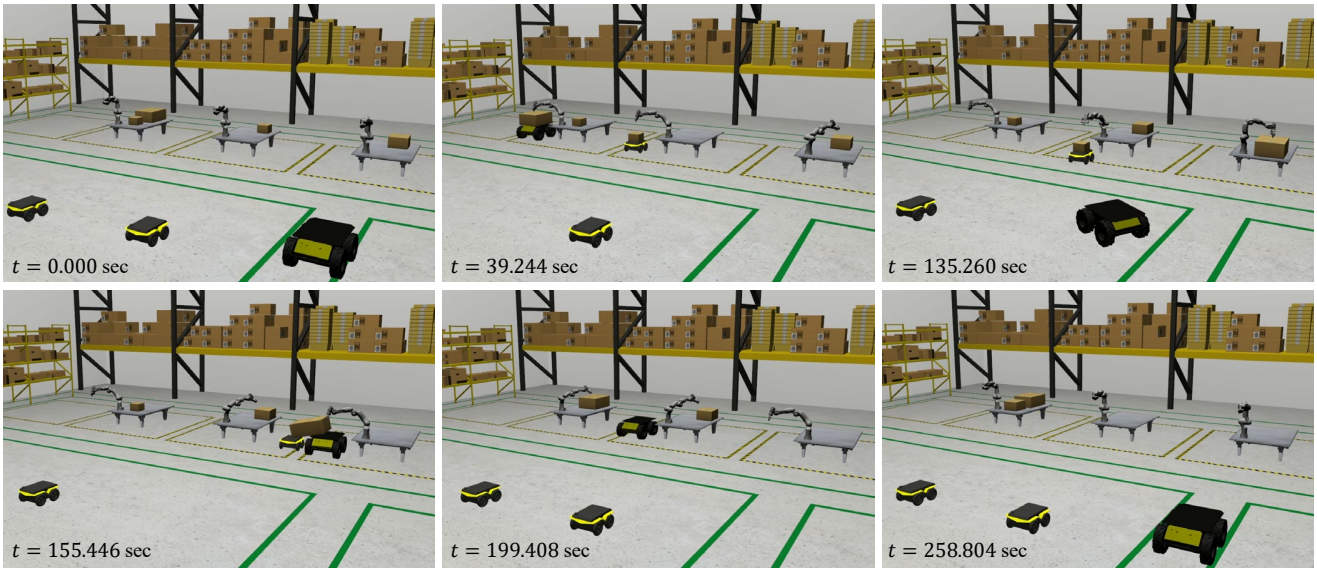

Fig. 3. Qualitative results from the robot-assisted manufacturing case study using Gazebo simulations to evaluate and demonstrate LVWS.

System (ROS). Model training and execution are performed centralized on a 14-core Intel Core i7 machine with 96 GB RAM and Titan RTX GPU. We compare our LVWS approach with four baseline methods: (1) a randomized policy (**RAND**) that randomly assigns each task to available individual robots that meet the task's requirements, (2) a randomized policy with subteam formation (**RAND w/ subteam**) that randomly assigns tasks to both individual robots and subteams that meet the requirements, (3) a greedy heuristic-based policy (**GRP**) that assigns each task to the robot or subteam that meets the task's requirements and has the minimum payload capacity, and (4) the dynamic bipartite graph matching (**DBGM**) [19] that uses graph attention networks to perform collaborative scheduling. To quantitatively evaluate LVWS and compare it with other methods, two metrics are used: (1) **makespan** (MS) [32], [19], which is defined as the time (in seconds) that is used to complete all the tasks and (2) **suboptimality** [48], which is defined as the percent increase in the makespan of a solution compared to the exact solution's makespan. For both metrics, smaller values indicate better performance.

TABLE I

QUANTITATIVE RESULTS OF LVWS AND COMPARISONS WITH OTHER METHODS USING THE SUBOPTIMALITY METRIC ON DATASETS I AND II.

Method	6 Robots, 18 Tasks (*6r-18t*)	3-8 Robots, 8-20 Tasks (*8r-20t-var*)
RAND	23.00%	–
RAND w/ subteams	11.80%	18.04%
GRP	15.80%	19.93%
DBGM	13.80%	–
LVWS [Ours]	**0.80%**	**13.81%**

A. Quantitative Results on Collaborative Scheduling

We generate three synthetic datasets to train and evaluate the approaches for heterogeneous collaborative scheduling. Dataset I includes 1,000 problem instances with 6 robots and 18 tasks each (*6r-18t*). Dataset II contains 10,000 instances with a varying number of 3-8 robots and 8-20 tasks (*8r-20t-var*). Dataset III contains 20 instances with 6 robots and 100 tasks (*6r-100t*), which is collected to evaluate the approaches' scalability to a large number of tasks. We set the number of robot capabilities to be between 2-3, the robot payload capacity to be between 1-30, and the task payload requirement to be between 1-30 that is sampled from a Poisson distribution. Moreover, we generate a randomized directed acyclic graph for each instance to model task dependencies. 800 instances in Dataset I and 8,000 instances in Dataset II are used for training, and the remaining in Datasets I-III for testing.

TABLE II

QUANTITATIVE RESULTS USING THE METRIC OF MAKESPAN WHEN SCALING THE METHODS WITH A LARGER NUMBER OF TASKS.

Method	6 Robots, 100 Tasks (*6r-100t*)
RAND	24.85
RAND w/ subteams	24.70
GRP	25.85
DBGM	23.05
LVWS [Ours]	**22.00**

The quantitative results over Datasets I and II are shown in Table I. In the *6r-18t* scenario, we can observe that RAND performs the worst as expected, while GRP performs better due to its objective of always assigning a task to the robot with the minimum payload that can complete the task. DBGM performs better than RAND and GRP but it performs worse than RAND with subteam formation. This is because DBGM only forms subteams as necessary for failure situations, while RAND w/ subteams attempts to choose among all possible subteams when assigning tasks. Our method LVWS optimizes subteam formation together with voluntary waiting for specific robot-task assignments when needed and achieves the lowest suboptimality compared with the exact solution. In the *8r-20t-var* scenario, both RAND and DBGM cannot run because these policies only work with a fixed number of robots and

tasks for the entire dataset. We observe similar results showing LVWS outperforming the other methods.

We further evaluate the methods on Dataset III to study the scalability of these methods to a larger number of tasks. The quantitative results are shown in Table II. Because the exact solution is intractable and cannot be computed reasonably, we use the makespan metric to evaluate our approach and compare it with other methods. It can be observed that LVWS outperforms all the compared methods, demonstrating that our LVWS approach can scale with larger number of tasks even when trained with smaller datasets.

B. Qualitative Case Study in Gazebo Simulations

To intuitively demonstrate our LVWS approach, we perform a case study of heterogeneous collaborative scheduling in an assembly cell manufacturing environment using ROS Gazebo simulations. We simulate a heterogeneous team of six robots with different capabilities and payload capacities, including three Panda robotic arms (from Franka Emika), two Jackal mobile robots (from Clearpath), and one Husky mobile robot (from Clearpath). In the case study illustrated in Fig. 4, the environment includes three assembly cells. Each cell includes one or more fixed robot arms with the manipulation capability but have different payload capacities. The mobile robots have the mobility capability to transport objects among the cells. The assembly problem in the case study includes two types of tasks. The first type of tasks pertains to inter-assembly cell object transportation, where each distinct source-destination pair identifies a unique task. The transportation tasks can be defined by transitions between different steps during assembly, where intermediate products must be moved to other locations for further processing. The second type of tasks pertains to the assembly cell tasks, which are performed on the objects during assembly. Our case study contains four objects processed by 14 tasks across the three assembly cells.

The qualitative results obtained by our LVWS approach in Gazebo on this case study are demonstrated in Fig. 3. We can observe that LVWS successfully schedules all tasks to individual robots and subteams to complete all the tasks. At time 39.244s, the Jackal robot is waiting for the additional task to become available while the other robots perform their tasks. At time 155.446s, the Jackal and Husky mobile robots form a subteam to jointly carry the large object Obj1 that has a payload requirement of 7 units, which cannot be carried by either a Jackal robot (with a payload capacity of 2-3 units) or the Husky robot (with a capacity payload of 6 units). The makespan (i.e., overall task completion time) obtained by our LVWS approach is 258.804s for the heterogeneous team of six robots to complete all 14 tasks in the Gazebo simulation.

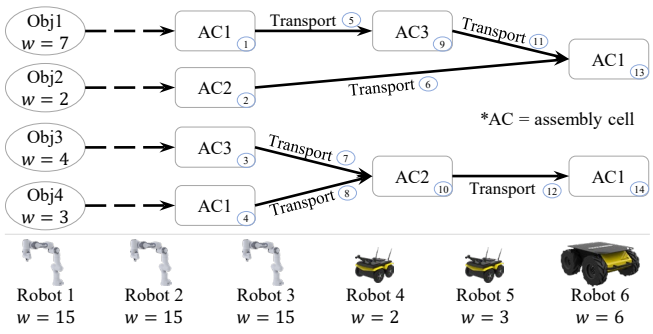

Fig. 4. Overview of the case study setup that includes 6 heterogeneous robots to transport and manipulate 4 objects in 3 assembly cells. The assembly process is divided into 14 tasks that must be completed by the robots.

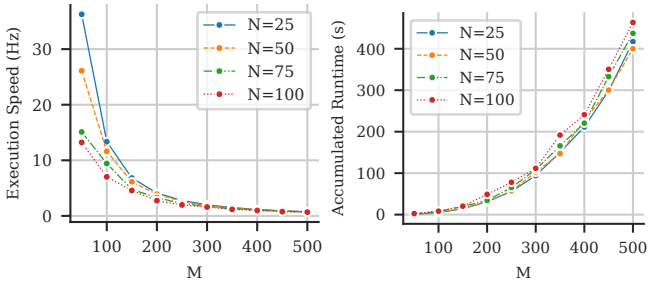

Fig. 5. Execution speed (left) and accumulated runtime (right) of our LVWS approach given varying N and M.

C. Execution Speed and Accumulated Runtime Analysis

We experimentally analyze the execution speed and the accumulated runtime of LVWS, as illustrated in Fig. 5. The number of robots varies from 25 to 100, while the number of tasks varies from 50 to 500. We observe that the execution speed of LVWS at $M = 50$ is 36.26 Hz and decreases with increasing M. At $M = 500$, the execution speed is 0.68 Hz. This allows LVWS to be used in online scheduling scenarios where the execution time of LVWS is faster than the rate at which tasks are finished. We also investigated the accumulated runtime, defined as the total time (in seconds) used for an approach to compute the scheduling solution. Fig. 5 shows that the accumulated runtime of LVWS quadratically increases with respect to the number of tasks M. With 25 robots and 50 tasks, LVWS can compute an optimal scheduling in 0.94 s. For a larger number of robots ($N = 100$) and tasks ($M = 500$), LVWS computes an optimal schedule in 463.43 s. This growth rate is polynomial, approximately between quadratic and cubic, which agrees with the theoretical time complexity discussed in Section III-D.

V. CONCLUSION

In this paper, we have proposed a new deep graph learning method for collaborative scheduling formulated as maximum bipartite matching between robots and tasks. We introduced a graph attention transformer network to generate a reward matrix used for allocating tasks to robots. By introducing dynamic subteaming and voluntary waiting, LVWS generates schedules that can reduce the overall task completion time and enable multiple robots to collaboratively address tasks that cannot be handled by individual robots. We evaluated our LVWS approach using three synthetic datasets and high-fidelity simulations in ROS Gazebo. Experimental results have shown that our approach well addresses heterogeneous collaborative scheduling and enables subteaming and voluntary waiting, which outperforms the baseline and previous methods across diverse numbers of robots and tasks.

REFERENCES

[1] F. Vicentini, "Collaborative robotics: A survey," *Journal of Mechanical Design*, vol. 143, no. 4, p. 040802, 2021.

[2] J. K. Verma and V. Ranga, "Multi-robot coordination analysis, taxonomy, challenges and future scope," *Journal of Intelligent & Robotic Systems*, vol. 102, pp. 1–36, 2021.

[3] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2017.

[4] P. Gao, R. Guo, H. Lu, and H. Zhang, "Multi-view sensor fusion by integrating model-based estimation and graph learning for collaborative object localization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[5] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, R. Giubilato, *et al.*, "The arches space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5315–5322, 2020.

[6] P. Gao, R. Guo, H. Lu, and H. Z. Zhang, "Regularized graph matching for correspondence identification under uncertainty in collaborative perception," in *Robotics Science and Systems*, 2021.

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

[8] 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.

[9] Z. Kashino, G. Nejat, and B. Benhabib, "Aerial wilderness search and rescue with ground support," *Journal of Intelligent & Robotic Systems*, vol. 99, pp. 147–163, 2020.

[10] S. Botelho and R. Alami, "Multi-robot cooperation through the common use of mechanisms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[11] M. Smyrnakis and S. M. Veres, "Coordination of control in robot teams using game-theoretic learning," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1194–1202, 2014.

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

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

[14] C. A. Valle and J. E. Beasley, "Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment," *Computers & Operations Research*, vol. 125, p. 105090, 2021.

[15] F. Xue, H. Tang, Q. Su, and T. Li, "Task allocation of intelligent warehouse picking system based on multi-robot coalition." *KSII Transactions on Internet & Information Systems*, vol. 13, no. 7, 2019.

[16] B. Fu, W. Smith, D. Rizzo, M. Castanier, M. Ghaffari, and K. Barton, "Learning task requirements and agent capabilities for multi-agent task allocation," *arXiv preprint arXiv:2211.03286*, 2022.

[17] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *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, 2022.

[19] P. Gao, S. Siva, A. Micciche, and H. Zhang, "Collaborative scheduling with adaptation to failure for heterogeneous robot teams," in *IEEE International Conference on Robotics and Automation*, 2023.

[20] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[21] 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.

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

[23] A. Messing, G. Neville, S. Chernova, S. Hutchinson, and H. Ravichandar, "Grstaps: Graphically recursive simultaneous task allocation, planning, and scheduling," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 232–256, 2022.

[24] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," in *European Control Conference*, 2019.

[25] 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.

[26] 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.

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

[28] G. Neville, S. Chernova, and H. Ravichandar, "D-itags: A dynamic interleaved approach to resilient task allocation, scheduling, and motion planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1037–1044, 2023.

[29] H. Aziz, H. Chan, Á. Cseh, B. Li, F. Ramezani, and C. Wang, "Multi-robot task allocation—complexity and approximation," in *International Conference on Autonomous Agents and Multiagent Systems*, 2021.

[30] A. Elfakharany, R. Yusof, and Z. Ismail, "Towards multi-robot task allocation and navigation using deep reinforcement learning," in *Journal of Physics: Conference Series*, 2020.

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

[32] 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.

[33] N. Seenu, K. C. RM, M. Ramya, and M. N. Janardhanan, "Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems," *Industrial Robot: the International Journal of Robotics Research and Application*, vol. 47, no. 6, pp. 929–942, 2020.

[34] E. Seraj, L. Chen, and M. C. Gombolay, "A hierarchical coordination framework for joint perception-action tasks in composite robot teams," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 139–158, 2021.

[35] E. Seraj, A. Silva, and M. Gombolay, "Multi-uav planning for cooperative wildfire coverage and tracking with quality-of-service guarantees," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 2, p. 39, 2022.

[36] 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.

[37] W. Gosrich, S. Mayya, S. Narayan, M. Malencia, S. Agarwal, and V. Kumar, "Multi-robot coordination and cooperation with task precedence relationships," in *IEEE International Conference on Robotics and Automation*, 2023.

[38] H. Wu, A. Ghadami, A. E. Bayrak, J. M. Smereka, and B. I. Epureanu, "Task allocation with load management in multi-agent teams," in *International Conference on Robotics and Automation*, 2022.

[39] J. Ke, F. Xiao, H. Yang, and J. Ye, "Learning to delay in ride-sourcing systems: A multi-agent deep reinforcement learning framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2280–2292, 2020.

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[43] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2015.

[44] J. A. Gromicho, J. J. Van Hoorn, F. Saldanha-da Gama, and G. T. Timmer, "Solving the job-shop scheduling problem optimally by

dynamic programming," *Computers & Operations Research*, vol. 39, no. 12, pp. 2968–2977, 2012.
[45] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196–210, 1962.
[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.
[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.
[48] I. Spasojevic, X. Liu, A. Ribeiro, G. J. Pappas, and V. Kumar, "Active collaborative localization in heterogeneous robot teams," in *Robotics Science and Systems*, 2023.