Autonomous Robots
https://doi.org/10.1007/s10514-021-09997-2

Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints

Zheyuan Wang[1] · **Chen Liu**[1] · **Matthew Gombolay**[1]

Received: 31 January 2021 / Accepted: 26 June 2021

Abstract
Robot teams are increasingly being deployed in environments, such as manufacturing facilities and warehouses, to save cost and improve productivity. To efficiently coordinate multi-robot teams, fast, high-quality scheduling algorithms are essential to satisfy the temporal and spatial constraints imposed by dynamic task specification and part and robot availability. Traditional solutions include exact methods, which are intractable for large-scale problems, or application-specific heuristics, which require expert domain knowledge to develop. In this paper, we propose a novel heterogeneous graph attention network model, called ScheduleNet, to learn scheduling policies that overcome the limitations of conventional approaches. By introducing robot- and proximity-specific nodes into the simple temporal network encoding temporal constraints, we obtain a heterogeneous graph structure that is nonparametric in the number of tasks, robots and task resources or locations. We show that our model is end-to-end trainable via imitation learning on small-scale problems, and generalizes to large, unseen problems. Empirically, our method outperforms the existing state-of-the-art methods in a variety of testing scenarios involving both homogeneous and heterogeneous robot teams.

Keywords Multi-robot coordination · Planning and scheduling · Graph neural networks · Imitation learning

1 Introduction

Given the recent developments in robotic technologies and the increasing availability of collaborative robots (cobots), multi-robot systems are increasingly being adopted in various manufacturing and industrial environments (Yan et al. 2013). Research in related areas (e.g., multi-robot communication, team formation and control, path planning, task scheduling and routing) has also received significant attention (Flushing et al. 2017). In this paper, we focus on the problem of multi-robot task allocation and scheduling (Nunes et al. 2017) with both temporal and spatial constraints, which captures the key challenges of final assembly manufacturing with robot teams. To achieve an optimal schedule for a user-specified objective, the robots must be allocated with the proper number of tasks and process these tasks with optimal order, while satisfying temporal constraints such as task deadlines and wait constraints. The addition of spatial constraints (i.e., a specific work area can only be occupied by one robot at a time and robots must maintain a minimum distance from other agents while performing a task) makes scheduling difficult because one must reason through inter-coupled, disjunctive sequencing constraints that impact shared resource utilization.

Traditionally, the scheduling problem is formulated as a mixed-integer linear program (MILP), which can be approached with either exact methods or hand-crafted heuristics. However multi-robot scheduling with both temporal and spatial constraints is generally NP-hard (Solomon 1986). Exact methods fail to scale to large-scale problems, which is exacerbated by the need for near real-time solutions to prevent factory slow-downs. Moreover, efficient approximation algorithms are hard to design. They not only require

This is one of the several papers published in Autonomous Robots comprising the Special Issue on Robotics: Science and Systems 2020.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-021-09997-2.

✉ Zheyuan Wang
pjohnwang@gatech.edu

Chen Liu
chen.liu@gatech.edu

Matthew Gombolay
matthew.gombolay@cc.gatech.edu

[1] Georgia Institute of Technology, Atlanta, GA, USA

Published online: 02 August 2021

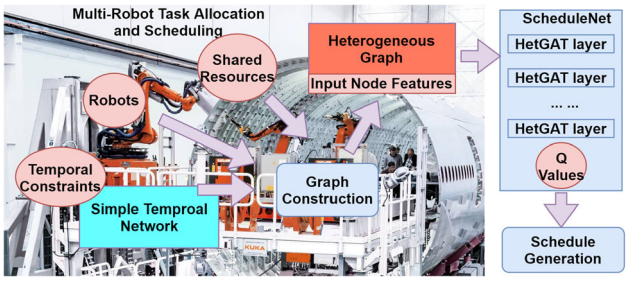


Fig. 1 Overview of the proposed ScheduleNet, which operates on the heterogeneous graph constructed by augmenting the STN of the problem, and predicts Q-values for scheduling. Courtesy: KUKA Robotics

domain specific knowledge with respect to each problem configuration that usually takes years to gain, but they also require accurate feature engineering to encode such knowledge, which leaves much to be desired (Raghavan et al. 2006).

In recent years, deep neural networks have brought about breakthroughs in many domains, including image classification, nature language understanding and drug discovery, as they can discover intricate structures in high-dimensional data without hand-crafted feature engineering (LeCun et al. 2015). Promising progress has also been made towards learning heuristics for combinatorial optimization problems by utilizing graph neural networks to learn meaningful representations of the problem to guide the solution construction process (Zhou et al. 2018). Yet this research focuses on significantly easier problems with a simpler graphical structure, e.g. the traveling salesman problem (TSP).

We propose a novel heterogeneous graph attention network model, called ScheduleNet, to learn heuristics for solving the multi-robot task allocation and scheduling problems with upper- and lowerbound temporal and spatial constraints. Figure 1 shows the overall framework of our proposed method. We extend the simple temporal network (STN) (Dechter et al. 1991) that encodes the temporal constraints into a heterogeneous graph by adding nodes denoting various components, such as workers (human or robot) and physical locations or other shared resources. By doing so, ScheduleNet directly operates on the heterogeneous graph in a fully-convolutional manner and can estimate the Q-function of state-action pairs to be used for schedule generation. We show that ScheduleNet is end-to-end trainable via imitation learning on small-scale problems and generalizes to large, unseen problems with an affordable increase in computation cost. This flexibility allows us to set a new state of the art for multi-robot coordination and in autonomously learning domain-specific heuristics for robotic applications.

This paper extends our prior conference paper (Wang and Gombolay 202) which first introduced ScheduleNet. This prior work was limited in modeling only teams of robots with homogeneous task performance (i.e., each robot was equally proficient in completing a given task) and could only consider a more restricted set of shared resource constraints. We build upon this prior work in three key ways. First, we extend ScheduleNet to efficiently reason about coordinating teams of heterogeneous robots (i.e., robots have varying proficiencies in completing each task). Second, we expand the types of spatial constraints ScheduleNet can consider from 1-Dimensional (1D) locations to 2D areas with minimum-distance constraints. Third, to improve ScheduleNet's ability to coordinate heterogeneous teams with such spatial constraints, we further augment our approach by proposing novel schedule synthesis strategies. These extensions and the accompanying empirical validation and robot demonstration serve to provide a more holistic view of ScheduleNet's capabilities with emphases on its flexibility, scalability and generalizability. Our results show that ScheduleNet outperforms benchmark approaches when under both the homogeneous and heterogeneous cases. Our extension even solves random problem instances with up to 10 heterogeneous robots and 200 tasks when no other baseline can solve even a single such instance.

The paper is organized as follows. Section 2 summarizes prior work in related areas. Section 3 presents an overview of the problem of interest and its Markov Decision Process (MDP) formulation for schedule construction. In Sect. 4 we describe the ScheduleNet graph structure and its building block layer. Section 5 presents the imitation learning algorithm to train the ScheduleNet model. Experimental results are split into two sections: Sect. 6 focuses on problems involving homogeneous robots; Sect. 7 covers results evaluated with heterogeneous task completion. Section 8 demonstrates the use of ScheduleNet in a simulated environment for airplane fuselage construction. Section 9 discusses major takeaways of our algorithm and its limitations, followed by concluding remarks in Sect. 10.

2 Related work

Task assignment and scheduling for multi-robot teams has been studied with various real-world applications, such as manufacturing, warehouse automation and delivery systems Nunes et al. (2017). Korsah et al. (2013) devised a widely accepted taxonomy, iTax, to categorizethe Multi-Robot Task Allocation (MRTA) problem. In a more recent survey, Nunes et al. (2017) further categorized the extensive research in multi-robot task allocation and identified possible solution approaches there within.

2.1 Traditional techniques

Task allocation is essentially an optimization problem, and the most common formalism to capture its constraints

is Mixed Integer Linear Programming (MILP). As exact methods for solving MILP yield exponential complexity, researchers have combined MILP and constraint programming (CP) methods into a hybrid algorithm using decomposition (Benders 1962; Gombolay et al. 2013; Ren and Tang 2009) to accelerate computation. Other hybrid approaches exploited heuristic schedulers to gain better scalability (Castro and Petrovic 2012; Chen and Askin 2009).

Nikou et al. (2017) presented a fully-automated procedure for controller synthesis for multi-robot systems under temporal constraints over high-level tasks. A decentralized abstraction that provides a time and space discretization of the multi-agent system is designed to enable efficiently computing the individual runs which provably satisfy the high-level task specifications. Gombolay et al. (2018) proposed a MILP-based task assignment and scheduling algorithm that takes advantage of an analytical sequencing test to satisfy temporospatial constraints, allowing the algorithm to scale better to large-scale multi-agent problems. Bogner et al. (2018) proposed an integer linear programming (ILP) model to coordinate the assignment of tasks between humans and robots in the assembly of printed circuit boards. The ILP model utilizes heuristics to find optimized solutions within a reasonable time while also accounting for the different capabilities of humans and robots. Solovey et al. (2020) presented the FLOWDEC algorithm for efficient task allocation with heterogeneous agents. By decomposing the heterogeneous task allocation problem into a set of homogeneous subproblems, FLOWDEC achieves an approximation factor of at least 1/2 of the optimal reward, achieving several orders of magnitude improvement in computation time over an exact technique.

Search algorithms have also been applied to solve task scheduling problems. Kartal et al. (2016) developed an efficient, satisficing and centralized Monte Carlo Tree Search-based algorithm combined with branch-and-bound to solve the multi-robot task allocation problem with temporal and spatial constraints. Their approach offers theoretical guarantees and finds optimal solutions for some non-trivial data sets. Zhang et al. (2020) employed a genetic algorithm for heterogeneous human-robot collaboration problems by encoding problem solutions as chromosomes and repeatedly crossing over and mutating the solutions to find the optimal schedule. To decouple the key computational challenges of multi-robot allocation under task uncertainty and temporal constraints, Choudhury et al. (2020) presented a hierarchical algorithm, SCoBA, which uses multi-agent conflict resolution with single-agent policy tree search and prove that SCoBA is both optimal in expectation and complete under mild assumptions. Wang et al. (2020) modeled the heterogeneous task scheduling problem with an edge-weighted and vertex-weighted block sequence graph and found solution with genetic algorithm. More recently, Hari et al. (2020) developed an approximation algorithm for the task allocation and scheduling problem involving humans and robots by combining ideas from vehicle routing and scheduling theory, with a guaranteed upper bound on the approximation ratio.

Unfortunately, these methods typically requires domain-specific expert knowledge to design efficient heuristics to guide the schedule search process. Furthermore, these algorithms are tied to a single objective function. In contrast, our learning-based approach can automatically learn such domain-specific knowledge via Q-learning based imitation learning and work under different objectives.

2.2 Learning-based techniques

Another class of techniques utilizes machine learning to learn the optimal strategy for solving scheduling optimization problems. Wang and Usher (2005) developed a Q-learning based method for the single-machine dispatching rule selection problem. Wu et al. (2011) proposed a multi-agent reinforcement learning method, called the ordinal sharing learning (OSL) method, for job-scheduling problems such as load balancing in grids.

More recently, Shiue et al. (2018) incorporated an offline learning module and a Q-learning-based RL module for the real-time scheduling problem. Casalino et al. (2021) proposed a probabilistic framework using time Petri nets to model the uncertainty in the time to complete tasks by humans and robots. The distribution of these time durations can be learned during the process of human-robot interaction. However, these methods typically depend on customized, hand-crafted features to achieve satisfying results. In our method, we exploit the power of deep learning models to automatically learn useful features.

2.3 Graph neural networks

Graph neural networks (GNN) are a class of deep neural networks that learn from unstructured data by representing objects as nodes and relations as edges and aggregating information from nearby nodes. GNNs have been widely applied in graph-based problems such as node classification, link prediction and clustering, and show convincing performance (Xu et al. 2019). Fout et al. (2017) introduced a GNN model that learns latent representation of proteins by performing convolutions over local neighborhoods of nodes and predicts if a pair of proteins belong to an interface. Hamaguchi et al. (2017) used GNNs to compute embeddings for entities not already stored in a knowledge base and answer queries concerning such entities. With the introduction of attention mechanisms to the graph convolutional layer, Veličković et al. (2017) developed the graph attention network (GAT) model that learns to attend to different neighbors with different weights, and were able to

further improve the performance of GNNs on a number of graph benchmark datasets. Hamilton et al. (2018) proposed a general framework that learns to generate node feature embeddings by sampling and aggregating features of the node's neighbors, and demonstrated strong performance in node classification tasks, such as predicting paper subject categories in citation graphs with each paper represented as a node in the graph. Yan et al. (2018) modeled human skeletons across time as graphs and proposed a temporal graph network for human activity recognition. Wang et al. (2019c) applied graph convolutional networks to point cloud classification and segmentation by exploiting GNN's capability to aggregate information from local neighborhoods. A more comprehensive review of GNN approaches and applications can be found in Wu et al. (2021).

Compared to the pervasive use of GNNs in classification problems, efforts of applying GNNs in solving combinatorial optimization problems are less investigated. Khalil et al. (2017) input the node embeddings learned by a GNN into a Q-learning module and achieved better performance than previous heuristics on solving minimum vertex cover, maximum cut and TSPs. Kool et al. (2019) combined GNNs and policy gradient methods to learn a policy for TSP and two variants of the Vehicle Routing Problem (VRP), which obtained solutions close to the optimal results for up to 100 nodes. Wang and Gombolay (2019) developed a GNN-based model operating on the STN to generate schedules for coordinating multi-robot teams. However, their method uses homogeneous graphs and hard-codes robot and location information as node features, making it not scalable when the number of robots changes. More discussions of applying GNN to combinatorial optimization problems can be found in Bengio et al. (2021).

Heterogeneous GNNs, which directly operate on heterogeneous graphs containing different types of nodes and links, have shown good interpretability and model expressiveness compared to traditional GNNs (Wang et al. 2019b; Yang et al. 2020), but such a model has never been applied to combinatorial optimization problems. We are the first to utilize heterogeneous GNNs for scheduling multi-robot teams.

3 Problem overview

3.1 Problem statement

We consider the problem of coordinating a multi-robot team in the same space, with temporal and resource / spatial constraints. The problem falls into the single-task robots (ST), single-robot tasks (SR), time-extended assignment (TA) category with cross-schedule dependencies [XD] under the iTax taxonomy proposed by Korsah et al. (2013). We describe its components using a six-tuple $< \boldsymbol{r}, \boldsymbol{\tau}, \boldsymbol{d}, \boldsymbol{w}, \boldsymbol{Loc}, z >$. $\boldsymbol{r}$ consists of all the robot agents available. $\boldsymbol{\tau}$ are the tasks to be performed. Each task τ_i is associated with a start time s_i and a finish time f_i and takes a certain amount of time $dur_{i,r}$ for robot r to complete. We introduce s_0 as the time origin and f_0 as the time point when all tasks are completed, so that the schedule has a common start and end point. $\boldsymbol{d}$ contains the deadline constraints. $d_i \in \boldsymbol{d}$ specifies before which task τ_i has to be completed, i.e., $f_i \leq d_i$. $\boldsymbol{w}$ is the set of wait constraints. $w_{i,j} \in \boldsymbol{w}$ specifies the wait time between task τ_i and task τ_j (e.g., "task τ_i should wait at least 10 min after task τ_j finishes" translates into $s_i - f_j \geq w_{i,j} = 10$). $\boldsymbol{Loc}$ is the list of all task locations. Finally, z is an objective function to minimize that can take different forms depending on end-user applications.

A solution to the problem consists of an assignment of tasks to agents and a schedule for each agent's tasks, such that all constraints are satisfied and the objective function is minimized.

Equations 1–9 give the mathematical program formation of our problem under this setting, where two types of binary decision variables are used: 1) $A_{r,i} = 1$ for the assignment of robot r to task τ_i and 2) $X_{i,j} = 1$ denoting task τ_i finishes before task τ_j starts. We also have continuous decision variables, s_i, $f_i \in [0, \infty)$, corresponding to the start and finish times of task τ_i, respectively. Additionally, $\boldsymbol{L_{proximity}}$ is the set of task pairs, $< \tau_i, \tau_j >$, that should be separated along the time axis due to the presence of pairwise proximity constraints on agents performing tasks at the corresponding locations in $\boldsymbol{Loc}$. These equations can be passed to a MILP-based solver to search for optimal solutions.

$$\min(z) \tag{1}$$

$$\sum_{r \in \boldsymbol{r}} A_{r,i} = 1, \forall \tau_i \in \boldsymbol{\tau} \tag{2}$$

$$f_i - s_i = \sum_{r \in \boldsymbol{r}} dur_{i,r} A_{r,i}, \forall \tau_i \in \boldsymbol{\tau} \tag{3}$$

$$f_i - s_0 \leq d_i, \forall d_i \in \boldsymbol{d} \tag{4}$$

$$s_i - f_j \geq w_{i,j}, \forall w_{i,j} \in \boldsymbol{w} \tag{5}$$

$$(s_j - f_i) A_{r,i} A_{r,j} X_{i,j} \geq 0, \forall \tau_i, \tau_j \in \boldsymbol{\tau}, \forall r \in \boldsymbol{r} \tag{6}$$

$$(s_i - f_j) A_{r,i} A_{r,j} (1 - X_{i,j}) \geq 0, \forall \tau_i, \tau_j \in \boldsymbol{\tau}, \forall r \in \boldsymbol{r} \tag{7}$$

$$(s_j - f_i) X_{i,j} \geq 0, \forall (\tau_i, \tau_j) \in \boldsymbol{L_{proximity}} \tag{8}$$

$$(s_i - f_j)(1 - X_{i,j}) \geq 0, \forall (\tau_i, \tau_j) \in \boldsymbol{L_{proximity}} \tag{9}$$

We begin our experiments in Sect. 6 by considering the problem of coordinating a set of homogeneous robots to complete a set of tasks given temporal constraints and 1D task location constraints (i.e., no two robots can be in the same place at the same time). Here, homogeneity in robot teams refers to teams comprised of robots that are equally proficient in completing a task (i.e., $dur_{i,r} = dur_i, \forall r \in \boldsymbol{r}, \forall \tau_i \in \boldsymbol{\tau}$). This modeling setup is motivated by common manufactur-

ing scenarios in which work locations are along a line with robots moving across a rail to perform assembly tasks of a large workpiece, e.g., the Boeing 777 Fuselage Automated Upright Build process (Tang and Webb 2019). Under this setting, $\boldsymbol{L_{proximity}}$ consists of all task pairs that require the same location to be completed.

Second, we examine scheduling problems involving heterogeneous robots moving with 2D proximity constraints (Sect. 7). The relaxation to heterogeneous robot teams allows for the full expressivity where $dur_{i,r} \neq dur_i$ in general. Furthermore, expanding from 1D location constraints to 2D proximity constraints allows us to model an open factory floor concept where a certain distance must be maintained between robots while executing tasks. In these experiments, we extend $\boldsymbol{L_{proximity}}$ to include task pairs whose locations fall within the minimum allowed safety distance.

As z varies depending on application-specific goals, we mainly report the results of minimizing the makespan (i.e., overall process duration, $z = \max_i f_i$) as a generic objective function. To show the generalization of our method, in Sect. 6.4, we also consider an application-specific case where we try to minimize the weighted sum of the completion time of all tasks ($z = \sum_i c_i f_i$). We use this objective function as an analogy to the minimization of weighted tardiness in job-shop scheduling (Essafi et al. 2008).

3.2 MDP formulation

Given the problem statement, we learn greedy heuristics that construct solutions by appending tasks to an individual robot's partial schedule based on maximizing a score Q-function approximated with a graph neural network parameterized by θ. We formalize the problem of constructing the schedule as a Markov Decision Process (MDP) using a five-tuple $< x_t, u, T, R, \gamma >$ that includes:

- State x_t at a decision-step t includes the temporal constraints of the problem, represented by a STN, the location information, and all robots' partial schedules constructed so far.
- Action $u = < \tau_i, r_j >$ corresponds to appending an unscheduled task τ_i at the end of the partial schedule of robot r_j.
- Transition T corresponds to deterministically adding the edges associated with the action $u = < \tau_i, r_j >$ into the STN and updating the partial schedule of the selected robot. In accordance with problem constraints, we impose the following two rules when updating the STN: 1) $s_i \leq s_k, \forall \tau_k \in \{unscheduled\}$; 2) $f_i \leq s_m, \forall \tau_m \in \{unscheduled | (\tau_i, \tau_m) \in \boldsymbol{L_{proximity}}\}$.
- Reward R of a state-action pair is defined as the change in objective values after taking the action, calculated as $R = -1 \times (Z_{t+1} - Z_t)$. Z_t denotes the partial objective function at state x_t and is calculated only using scheduled tasks. For example, while minimizing makespan, $Z_t = max_i f_i, \tau_i \in \{partial\ schedules\}$. The reward is multiplied by -1.0 as the objective is minimization. We further divide Z_t by a factor $D > 1$ if x_t is not a termination state. We use D to balance between finding the highest immediate reward (local optimal) and finding the global optimal schedules. If the action results in an infeasible schedule in the next state, a large negative reward M_{inf} is assigned to Z_{t+1}.
- Discount factor, γ.

Algorithm 1: Solve a problem instance using ScheduleNet via scheduling-through-simulation

Input: Problem components $< \boldsymbol{r}, \boldsymbol{\tau}, \boldsymbol{d}, \boldsymbol{w}, \boldsymbol{Loc} >$, maximum allowed time t_{max}
Output: A schedule generated by ScheduleNet

Initialize all robots' pratial schedules as $\varnothing$;
Initialize problem state x with $< \boldsymbol{r}, \boldsymbol{\tau}, \boldsymbol{d}, \boldsymbol{w}, \boldsymbol{Loc} >$;
$t \leftarrow 0$;
while $t \leq t_{max}$ **do**
  Collects all available robots at t into $\boldsymbol{r_{avail}}$;
  $r \leftarrow$ `pickRobot`($\boldsymbol{r_{avail}}$);
  while $r \neq NULL$ **do**
    Collect all unscheduled tasks at t into $\boldsymbol{\tau_{avail}}$;
    if $\boldsymbol{\tau_{avail}} \neq \varnothing$ **then**
      Build the heterogeneous graph g from x;
      Generate input features for nodes in g;
      Run ScheduleNet on g to predict $Q_\theta(x, u)$;
      $\tau \leftarrow \text{argmax}_{\tau \in \boldsymbol{\tau_{avail}}} Q_\theta(x, u)|_{u=<\tau,r>}$;
      Append τ into r's partial schedule and update x;
      if *x becomes infeasible* **then**
        Exit with an infeasible schedule;
      else if *all tasks have been assigned* **then**
        Exit with a feasible schedule;
      end
      Remove r from $\boldsymbol{r_{avail}}$;
      $r \leftarrow$ `pickRobot`($\boldsymbol{r_{avail}}$);
    else
      break;
    end
  end
  $t \leftarrow t + 1$;
end
Exit with an infeasible schedule;

3.3 Schedule generation

Our learned heuristic relies on the evaluation function $Q(x, u)$, which will be learned using a collection of problem instances, to estimate the total discounted future reward of state-action pairs and select accordingly. We use scheduling-through-simulation to generate schedules as it has been shown in Wang and Gombolay (2019) that this process achieves better performance than using decision-step-based generation. Algorithm 1 illustrates the process of generat-

ing schedules for a problem instance using ScheduleNet via scheduling-through-simulation. In scheduling-through-simulation, starting from $t = 0$ (here t refers to time points instead of decision steps), at each time step the policy first collects all the available robots not working on a task into a set $\boldsymbol{r_{avail}} = \{r_j | r_j \text{ is available}\}$. Then, the policy picks a robot (denoted as the $pickRobot$ function) from $\boldsymbol{r_{avail}}$ and tries to assign τ_i using $\tau := \text{argmax}_{\tau \in \boldsymbol{\tau_{avail}}} Q_\theta(x, u)$, where $\boldsymbol{\tau_{avail}}$ is the set of unscheduled tasks and only Q-values associated with r_j are considered. This task allocation step repeats until no robot is available; then, the simulation moves to the next time step, $t + 1$. When considering a team of homogeneous robots, $pickRobot$ can be implemented as simply picking robots from $\boldsymbol{r_{avail}}$ in the same order as their initial index or another static priority queue. However, we empirically found that such a static strategy for ScheduleNet was not efficient for the more difficult problem of task allocation with heterogeneous robot teams. As such, we developed a set of additional task allocation strategies for dynamically picking heterogeneous robots from $\boldsymbol{r_{avail}}$:

- *First available*—Pick the first robot in $\boldsymbol{r_{avail}}$ according to their original index.
- *Minimum average time on unscheduled tasks*—Compute the average time it takes for each robot in $\boldsymbol{r_{avail}}$ to complete unscheduled tasks, and pick the robot with the smallest such time.
- *Minimum time on any one unscheduled task*—Find the minimum time it takes for each robot in $\boldsymbol{r_{avail}}$ to complete any one unscheduled task, and pick the robot with the smallest such time.
- *Minimum average time on all tasks*—Compute the average time it takes for each robot in $\boldsymbol{r_{avail}}$ to complete all tasks, both scheduled and unscheduled, and pick the robot with the smallest such time.

When solving a given problem instance, we run ScheduleNet in parallel with each task allocation strategy variants for the $pickRobot$ function. Among the feasible solutions produced by the same model with each of the four strategies, we keep the one that yields the best objective function score. This result ensemble of different robot-picking variants proves to find not only more feasible schedules, but also schedules with better makespans than any single strategy alone, as each strategy may work better than another in certain problems but not the others.

4 Heterogeneous graph attention network

Traditional graph neural networks (GNNs) operate on homogeneous graphs to learn a universal feature update scheme for all nodes. We instead cast the task scheduling problem into a heterogeneous graph structure, and propose a novel heterogeneous graph attention network, ScheduleNet, that learns per-edge-type message passing and per-node-type feature reduction mechanisms on this graph. One advantage of ScheduleNet is that it directly estimates the Q-values of state-action pairs as its output node features. In this section, we first describe how to construct the heterogeneous graph given a problem state, x_t, starting with homogeneous robot teams and then introducing necessary extensions for heterogeneous robots. Next, we present the building block layer used to assemble a ScheduleNet of arbitrary depth (through stacking this layer), which we call the heterogeneous graph attention (HetGAT) layer.

4.1 Heterogeneous graph representation

4.1.1 Homogeneous robot case

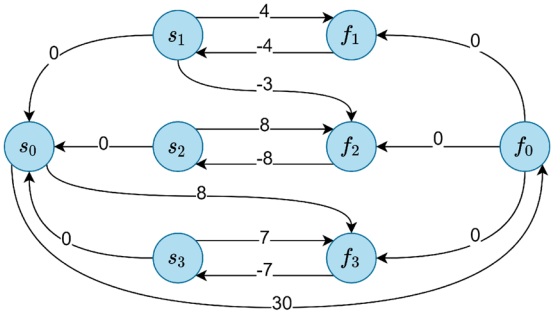
The temporal constraints in multi-robot task allocation and scheduling problems have been commonly modeled as STNs because the consistency of the upper and lower bound constraints can be efficiently verified in polynomial time (Tsamardinos 2000). STNs also allow for encoding set-bounded uncertainty. However, as we develop multiple homogeneous agents, physical constraints, etc., we also have latent disjunctive variables that augment the graph to account for each agent being able to perform only one task at a time and for only one robot occupying a work location at a time, which is known as the Disjunctive Temporal Problem (Tsamardinos and Pollack 2003). To learn a more expressive and scalable representation of the problem, we extend the STN formulation into a heterogeneous graph using the construction process illustrated in Algorithm 2. In a heterogeneous graph, we use a three-tuple, in the form of $< srcName, edgeName, dstName >$, to specify the edge type/relation that connects the two node types (from source node to destination node), which can also be denoted as $(srcName \xrightarrow{edgeName} dstName)$.

In traditional STN formulations, each task, τ_i, is represented by two event nodes: its start time node, s_i, and finish time node, f_i. The directed, weighted edges encode the temporal constraints associating corresponding nodes. Under the homogeneous robot team setting, task duration is deterministic without knowing the actual assignment. Exploiting this fact, we develop a **novel simplification trick** to reduce the model complexity. That is, after running Johnson's algorithm (Johnson 1977) on the original STN to find its minimum distance graph, we remove all finish time nodes (except f_0) from the distance graph to obtain a new STN. The simplified STN, using only half the nodes, still reserves all the necessary temporal constraints. In this way, each task can be represented by its start time node with task duration now serving as its

Algorithm 2: Construct the heterogeneous graph for modeling homogeneous robot teams

Input: STN, locations $\boldsymbol{Loc}$, robots $\boldsymbol{r}$ and their partial schedules, available actions $\boldsymbol{u_{avail}}$
Output: Heterogeneous graph representation

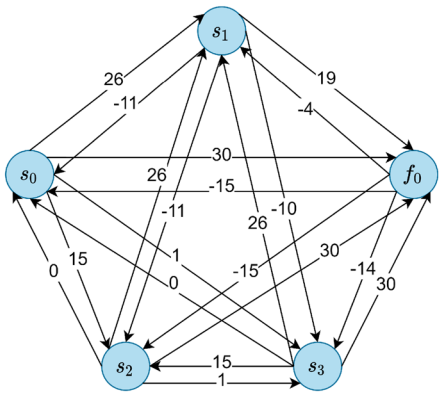
Run Johnson's algorihtm on STN to find its minimum distance graph, g_d;
Remove all f_i's from g_d, except f_0;
Use g_d as the new STN and s_i as the task node of τ_i;
foreach *robot* r_j **do**
  Add a robot node, r_j;
  foreach τ_m *assigned to* r_j **do**
    Add an edge $\tau_m \rightarrow r_j$;
  end
end
Connect robot nodes with each other;
foreach *location* L_k **do**
  Add a location node, L_k;
  foreach τ_m *located in* L_k **do**
    Add an edge $\tau_m \rightarrow L_k$;
  end
end
Connect location nodes with each other;
Add a state node st, connect all other nodes to it;
foreach $u_n =< \tau_n, r_n > \in \boldsymbol{u_{avail}}$ **do**
  Add a value nodes v_n;
  Add an edge $\tau_n \rightarrow v_n$;
  Add an edge $r_n \rightarrow v_n$;
  Add an edge $st \rightarrow v_n$;
end
Add self-loops;
return g_d.



(a) Original STN

Src \ Dst	s_0	f_0	s_1	f_1	s_2	f_2	s_3	f_3
s_0	0	30	26	30	15	23	1	8
f_0	-15	0	-4	0	-15	-7	-14	-7
s_1	-11	19	0	4	-11	-3	-10	-3
f_1	-15	15	-4	0	-15	-7	-14	-7
s_2	0	30	26	30	0	8	1	8
f_2	-8	22	18	22	-8	0	-7	0
s_3	0	30	26	30	15	23	0	7
f_3	-7	23	19	23	8	16	-7	0

(b) Table of shortest distances from each source (src) node to each destination (dst) node



(c) Simplified Minimum Distance Graph

Fig. 2 An example STN consisting of 3 tasks: **a** the original STN with placeholder start and finish nodes, s_0 and f_0; **b** The shortest distances between all pairs of source (src) and destination (dst) nodes found by an all pairs shortest path (APSP) algorithm, with blue denoting the nodes/edges that are maintained in the simplified graph (**c**) and orange denoting nodes/edges that are pruned; **c** the simplified minimum distance graph with f_i removed for each task, with the duration of each task encoded in the input node features

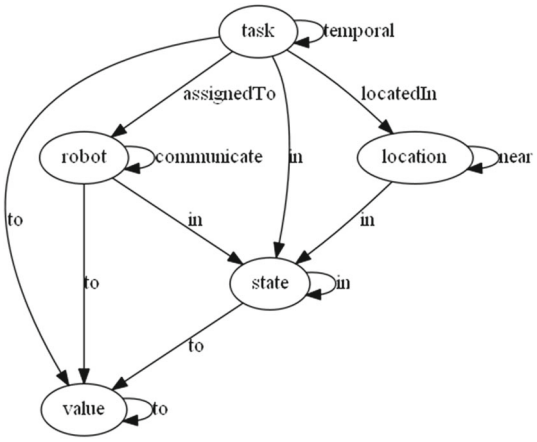
node feature. Figure 2 illustrates this process with an example problem consisting of 3 tasks. Tasks 1, 2, and 3 are shown with durations 4, 8, and 7, respectively. Task 3 has a deadline constraint: $f_3 \leq 8$. There is a wait constraint between task 1 and task 2: $s_1 \geq f_2 + 3$. Distances in the blue cells of Fig. 2b are used to construct the graph in Fig. 2c. The representation shown in Fig. 2c effectively describes the temporal constraint representation in Fig. 2c for the purposes of performing an all-pairs shortest path computation as input to our GNN model. The task durations represented by edges in Fig. 2a are preserved implicitly in the graph edges shown in Fig. 2c and are captured by our GNN as node features for the corresponding tasks. Given the partial schedule at the current state, we generate the initial input features of each task node as follows: the first two dimensions are the one-hot encoding of whether a task has been scheduled [1 0] or not [0 1]; the next dimension is the task duration. We denote the edge type from STNs using $(task \xrightarrow{temporal} task)$ as they encode the temporal constraints.

To extend the simplified STN, we add robot and location nodes equaling the number of robots and locations in the problem, respectively. A robot node is connected to the task nodes that have been assigned to it, with edge relation $(task \xrightarrow{assignedTo} robot)$. All robots are connected with each other to enable message flow between them, with edge relation $(robot \xrightarrow{communicate} robot)$. The initial feature of a robot node is the number of tasks assigned so far. In a similar manner, a location node is connected to the task nodes in that location, with edge relation $(task \xrightarrow{locatedIn} location)$. All location nodes are connected with each other, with the

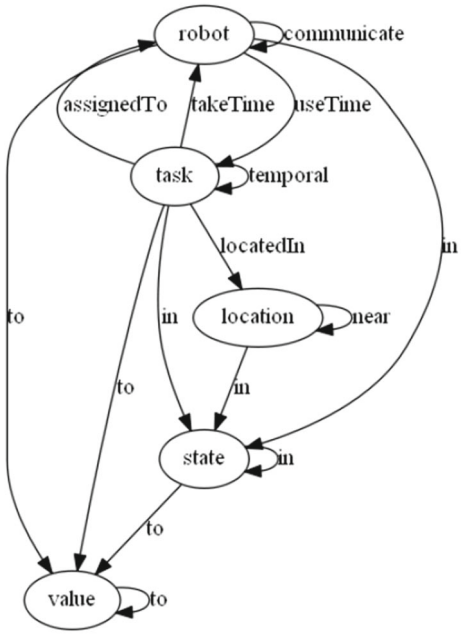
relation ($location \xrightarrow{near} location$). The initial feature of a location node is the number of tasks in that location.

As the Q-function is based on state-action pairs, we also expect the network to learn a state embedding of the problem from all the task, robot, and location node embeddings. To achieve this, we add a state summary node into the graph structure. The state summary node is connected to all the task, robot and location nodes, with edge types ($task \xrightarrow{in} state$), ($robot \xrightarrow{in} state$), ($location \xrightarrow{in} state$), respectively. The initial features of the graph summary node include the number of total tasks, the number of currently scheduled tasks, the number of robots and the number of locations.

Once the node embeddings are computed using the heterogeneous graph, it is possible to learn a separate Q network consisting of several fully-connected (FC) layers to predict the Q-value of a state-action pair, taking as input the concatenation of embeddings from corresponding state, task, and robot nodes. However, designing a separate Q network on top of GNNs is computationally expensive and not memory efficient, especially when evaluating a large number of state-action pairs at once for parallel computing. Instead, we propose to add value nodes in the graph to directly estimate the Q-values. A value node is connected to corresponding nodes with edge types denoted as ($task \xrightarrow{to} value$), ($robot \xrightarrow{to} value$), ($state \xrightarrow{to} value$). The initial feature of a value node is set to 0. During evaluation, the heterogeneous graph is constructed with the needed Q-value nodes covering task nodes in $\boldsymbol{\tau_{avail}}$ and robot node of r_j, as mentioned in Sect. 3.3. As we are calculating the minimum distance graph of a STN while constructing the heterogeneous graph, we can further filter out the tasks in $\boldsymbol{\tau_{avail}}$ of which the lower bound of task start time is greater than the current time. For all nodes, self-loops are added so that their own features from previous layers are considered for the next layer's computation. The metagraph (or network schema) of the graph constructed with Algorithm 2 is shown in Fig. 3a, which summarizes all the node types and edge types.



(a) Homogeneous Robot Team Metagraph



(b) Heterogeneous Robot Team Metagraph

Fig. 3 Metagraph of the heterogeneous graph built from the STN by adding robot, location, state, and value nodes: **a** team of homogeneous robots; **b** team of heterogeneous robots

4.1.2 Extension for heterogeneous task completion

In a setting of heterogeneous robot teams, the duration of a task depends on the robot which is assigned to the task. We recall that our problem state consists of a partial schedule in which some tasks have already been assigned an agent and then sequenced. For those tasks, we know their duration, which is given by the assigned robot (i.e., $dur_{i,r}$ for task τ_i assigned robot r). However, the duration of unscheduled tasks is yet to be determined, as no robot has been assigned. As such, Eq. 3 can only be described with a relaxed set bound as shown in Eq. 10. Here $dur_{i,min}, dur_{i,max}$ are the minimum and maximum amounts of time task τ_i can be finished. Specifically, $dur_{i,min} = \min_r dur_{i,r}$ and $dur_{i,max} = \max_r dur_{i,r}$.

$$dur_{i,min} \leq f_i - s_i \leq dur_{i,max}, \forall \tau_i \in \{unscheduled\} \quad (10)$$

Unfortunately, this set bound nullifies our novel simplification trick in Sect. 4.1.1 for reducing the graph's–and in turn our algorithm's–complexity. To recover our simplification in this more expressive setting, we extend the set of task node features to include multiple descriptive statistics describing the task's possible completion times among all robots where the size is non-parametric in the number of robots. Specifi-

cally, the input features of task τ_i consists of the minimum, maximum, mean and standard deviation of $\{dur_{i,r}, \forall r \in \boldsymbol{r}\}$. This modeling approach achieves a potent balance between model complexity and expressivity.

Additionally, we encode information about completion times for unscheduled tasks by augmenting the heterogeneous graph obtained from Algorithm 2. We add two new edge types between nodes of unscheduled tasks and robot nodes: $(task \xrightarrow{takeTime} robot)$ and $(robot \xrightarrow{useTime} task)$. The edge attribute encodes the time used for a robot to complete the task connected via this edge.

A further change we make to Algorithm 2 concerns the handling of edges between location nodes. Because locations are expanded to 2D spatial areas in the heterogeneous robot case, we only connect locations that fall within the minimum allowed safety distance to represent the proximity constraints, instead of connecting location nodes with each other.

Figure 3b shows the augmented metagraph with heterogeneous robot teams, including the newly-added edge types between robot nodes and task nodes. The edge type $(location \xrightarrow{near} location)$, although shares the same name as in Figure 3a, now encodes the 2D proximity constraints.

4.2 Heterogeneous graph attention layer

4.2.1 Homogeneous robot case

The feature update process in a HetGAT layer is conducted in two steps: per-edge-type message passing followed by per-node-type feature reduction. During message passing, each edge type uses a distinct weight matrix, $W_{edgeName} \in \mathbb{R}^{D \times S}$, to process the input feature from the source node, N_{src}, and sends the computation result to the destination node, N_{dst}'s mailbox. S is the input feature dimension of N_{src}, and D is the output feature dimension of N_{dst}. In the case that several edge types share the same name, we use $W_{srcName,edgeName}$ to distinguish between them. For example, we distinguish edge types coming into the state nodes by $W_{task,in}$, $W_{robot,in}$, $W_{loc.,in}$ and $W_{state,in}$. As for edge type $(task \xrightarrow{temporal} task)$ which is the only weighted edge in our heterogeneous graph formulation of homogeneous robots, the edge attribute, $edge$, is also sent after transformed by $W_{tempEdge} \in \mathbb{R}^{D \times 1}$.

Feature reduction happens inside each node's mailbox. For each edge type that a node has, the HetGAT layer computes per-edge-type aggregation result by weighing received messages, stored in its mailbox, along the same edge type with normalized attention coefficients that are feature-dependent and structure-free. Those results are then merged to compute a node's output feature. We note that, in the case of coordinating teams of homogeneous robots, task type nodes only ever serve as destination nodes for other task nodes. Task nodes can serve as source nodes for non-task type nodes (e.g., robot nodes). This flow of information from task nodes to robot nodes enables us to extract embeddings for each robot. Embeddings for tasks are extracted from the underlying STN, which already captures information regarding the robots' homogeneous task completion times. When we augment our model in Sect. 4.1.2 to account for heterogeneous task completion times, we additionally consider edges from non-task type nodes (e.g., robot nodes) to task nodes to extract task embeddings that account for robot heterogeneity. The feature update formulas of different node types are listed in Eqs. 11–15.

$$\text{Task } h_i' = \sigma\left(\sum_{j \in N_{temporal}(i)} \alpha_{ij}^{temporal}\left(W_{temporal} h_j + W_{tempEdge} edge_{ji}\right)\right) \tag{11}$$

$$\text{Robot } h_i' = \sigma\left(\sum_{j \in N_{assginedTo}(i)} \alpha_{ij}^{assignedTo} W_{assignedTo} h_j + \sum_{k \in N_{comm.}(i)} \alpha_{ik}^{comm.} W_{comm.} h_k\right) \tag{12}$$

$$\text{Location } h_i' = \sigma\left(\sum_{j \in N_{locateIn}(i)} \alpha_{ij}^{locatedIn} W_{locatedIn} h_j + \sum_{k \in N_{near}(i)} \alpha_{ik}^{near} W_{near} h_k\right) \tag{13}$$

$$\text{State } h_i' = \sigma\left(\sum_{j \in N_{task,in}(i)} \alpha_{ij}^{task,in} W_{task,in} h_j + \sum_{k \in N_{robot,in}(i)} \alpha_{ik}^{robot,in} W_{robot,in} h_k + \sum_{m \in N_{loc.,in}(i)} \alpha_{im}^{loc.,in} W_{loc.,in} h_m + W_{state,in} h_i\right) \tag{14}$$

$$\text{Value } h_q' = \sigma\left(W_{task,to} h_t + W_{robot,to} h_r + W_{state,to} h_s + W_{value,to} h_q\right) \tag{15}$$

In Eqs. 11–15, $N_{edgeName}(i)$ is the set of incoming neighbors of node i along a certain edge type, and $\sigma()$ represents

the ReLU nonlinearity. Prior work has shown that attention mechanisms are beneficial for representation learning on homogeneous graphs (Veličković et al. 2017; Kool et al. 2019). Thus, we extend the attention models from prior work to reason about task scheduling with heterogeneous graph networks. Specifically, the per-edge-type attention coefficient, $\alpha_{ij}^{edgeName}$, is calculated based on source node features and destination node features (plus edge attributes if applicable). More specifically, the attention coefficient for edge type ($task \xrightarrow{temporal} task$) is calculated by Equation 16, where $\mathbf{a}_{temporal}^{T}$ is the learnable weights, $||$ is the concatenation operation, and $\sigma'()$ is the LeakyReLU nonlinearity (with a negative input slope of 0.2). Softmax function is used to normalize the coefficients across all choices of j.

$$\alpha_{ij}^{temp.} = \text{softmax}_j\Big(\sigma'\Big(\mathbf{a}_{temp.}^{T} \big[W_{temp.}\mathbf{h}_i || W_{temp.}\mathbf{h}_j || W_{tempEdge} edge_{ji}\big]\Big)\Big) \tag{16}$$

For edge types connecting the same type of nodes, the attention coefficients can be computed by Equation 17.

$$\alpha_{ij}^{edgeName} = \text{softmax}_j\Big(\sigma'\Big(\mathbf{a}_{edgeName}^{T} \big[W_{edgeName}\mathbf{h}_i || W_{edgeName}\mathbf{h}_j\big]\Big)\Big) \tag{17}$$

However, Equation 17 does not hold for edges where the source node, h_j, and destination node, h_i, are of different node types. Take the edge type ($task \xrightarrow{assignedTo} robot$) as an example, the message passing weights, $W_{assignedTo}$, are only defined and trained for processing the source node type features (of task nodes) and are thus not adequate for processing the destination node type features (of robot nodes) for attention computation. Therefore, we change Equation 17 into Equation 18 by using both $W_{edgeName}$ and $W_{dstType}$ to account for differing types of source and edge nodes. While these additional parameters improve model expressivity, there is a cost in terms of computational memory and speed. In practice, we find that we can achieve a helpful tradeoff between expressivity and computational costs by employing weight sharing. Specifically, we set $W_{dstType}$ to be equal to $W_{comm.}$, W_{near}, and $W_{state,in}$ when the destination node type is robot, location and state, respectively.

$$\alpha_{ij}^{edgeName} = \text{softmax}_j\Big(\sigma'\Big(\mathbf{a}_{edgeName}^{T} \big[W_{dstType}\mathbf{h}_i || W_{edgeName}\mathbf{h}_j\big]\Big)\Big) \tag{18}$$

To stabilize the learning process, we utilize the multi-head attention proposed in Veličković et al. (2017), adapting it to fit the heterogeneous case. We use K independent HetGAT layers to compute node features in parallel and then merge the results as the multi-headed output via the concatenation operation for each multi-head layer in ScheduleNet, except for the last layer which employs averaging. Considering that ScheduleNet utilizes a fully convolutional structure where the last graph layer directly predicts Q-values as the 1-dimensional output feature of value nodes, merging multi-head results with concatenation is no longer viable for the last layer as it would give a K-dimensional output.

4.2.2 Extension for heterogeneous task completion

Because the newly-added edge type ($robot \xrightarrow{useTime} task$) only accounts for unscheduled tasks, the feature update formula for scheduled tasks remains the same as Eq. 11. For unscheduled tasks, we change Eq. 11 by including terms accounting for the message coming through the new edge type, as shown in Eq. 19.

$$\begin{aligned} h_i' = \sigma\Bigg(& \sum_{j \in N_{temporal}(i)} \alpha_{ij}^{temporal} (W_{temporal} h_j \\ & + W_{tempEdge} edge_{ji}) \\ & + \sum_{k \in N_{useTime}(i)} \alpha_{ik}^{useTime} (W_{useTime} h_k \\ & + W_{useTimeEdge} edge_{ki}^{'}\Bigg) \end{aligned} \tag{19}$$

In Eq. 19, $edge_{ki}^{'}$ is the attribute of the new edge, and the corresponding attention coefficient, $\alpha_{ik}^{useTime}$, is computed by Eq. 20.

$$\alpha_{ik}^{useTime} = \text{softmax}_k\Big(\sigma'\Big(\mathbf{a}_{useTime}^{T} \big[W_{temp.}\mathbf{h}_i || W_{useTime}\mathbf{h}_k || W_{useTimeEdge} edge_{ki}^{'}\big]\Big)\Big) \tag{20}$$

Similarly, the addition of new edge type ($task \xrightarrow{takeTime} robot$) changes the feature update equation of robot nodes from Eq. 12 in the homogeneous case to Eq. 21 in the heterogeneous case.

$$\begin{aligned} h_i' = \sigma\Bigg(& \sum_{j \in N_{assginedTo}(i)} \alpha_{ij}^{assignedTo} W_{assignedTo} h_j \\ & + \sum_{k \in N_{comm.}(i)} \alpha_{ik}^{comm.} W_{comm.} h_k \end{aligned}$$

$$+ \sum_{m \in N_{takeTime}(i)} \alpha_{im}^{takeTime} (W_{takeTime} h_m + W_{takeEdge} edge''_{mi}) \quad (21)$$

In Eq. 21, $edge''_{mi}$ is the attribute of the corresponding edge, and the corresponding attention coefficient, $\alpha_{im}^{takeTime}$, is computed according to Eq. 22.

$$\alpha_{im}^{takeTime} = \text{softmax}_m \Big(\sigma' \Big(\mathbf{a}_{takeTime}^T \Big[W_{comm.} \mathbf{h}_i || W_{takeTime} \mathbf{h}_m || W_{takeTimeEdge} edge''_{mi} \Big] \Big) \Big) \quad (22)$$

Even though locations are extended from 1D to 2D areas, Eq. 13 still applies to location nodes. Because now $N_{near}(i)$ only considers neighbor locations falling within the allowed safety distance instead of all locations, and W_{near}, α_{ik}^{near} will learn to encode the corresponding proximity constraints. Finally, we note that the update equations for state and value nodes (Eqs. 14 and 15) remain the same as in the homogeneous robot case.

5 Imitation learning

Under the MDP formulation, our goal is to learn a greedy policy for sequential decision making. Thus, it is natural to consider reinforcement learning algorithms (e.g., Q-learning) for training ScheduleNet. However, reinforcement learning relies on finding feasible schedules from scratch to learn useful knowledge. In our problem, most permutations of the schedule are infeasible. As a result, reinforcement learning policies spend much more time than allowed before learning anything of value from exploring infeasible solutions.

Instead, we leverage imitation learning methods that learn from high-quality schedules to accelerate the learning process for quick deployment. In real-world scheduling environments, we often have access to high-quality, manually-generated schedules from human experts who currently manage the logistics in manufacturing environments. Moreover, it is practical to optimally solve small-scale problems with exact methods. Given the scalability of the heterogeneous graph, we expect that exploiting such expert data on smaller problems to train the ScheduleNet can generalize well towards solving unseen problems, even in a larger scale.

Let D_{ex} denote the expert dataset that contains all the state-action pairs of schedules either from exact solution methods or the domain experts. For each transition in D_{ex}, we calculate the total reward from current step t until termination step n using $R_t^{(n)} = \sum_{k=0}^{n-t} \gamma^k R_{t+k}$ and regress the corresponding Q-value predicted by ScheduleNet towards this value as shown in Equation 23, where the supervised learning loss, L_{ex}, is computed as the mean squared error between $R_t^{(n)}$ and our current estimate on the expert action u_{ex}.

$$L_{ex} = \left\| Q(x_t, u_{ex}) - R_t^{(n)} \right\|^2 \quad (23)$$

To fully exploit the expert data, we ground the predicted Q-values of alternative actions u_{alt} (not selected by the expert) to a value below $R_t^{(n)}$ using the loss shown in Equation 24, where q_o is a positive constant empirically picked as an offset, and N_{alt} is the number of alternative actions at step t. In accordance with the schedule generation scheme, N_{alt} only considers actions involving the same robot selected by the expert.

$$L_{alt} = \frac{\sum \left\| Q(x, u_{alt}) - \min \left(\begin{bmatrix} Q(x, u_{alt}) \\ R_t^{(n)} - q_o \end{bmatrix} \right) \right\|^2}{N_{alt}} \quad (24)$$

The min term in Equation 24 ensures that the gradient propagates through all the unselected actions that have Q-values predicted by ScheduleNet to be higher than $R_t^{(n)} - q_o$. The difference from Piot et al. (2014) lies in that they only train on the unselected action with the max Q-value. The total supervised loss is shown in Equation 25, where L_2 is the L2 regularization term on the network weights, and λ_1, λ_2 are weighting parameters assigned to different loss terms empirically.

$$L_{total} = L_{ex} + \lambda_1 L_{alt} + \lambda_2 L_2 \quad (25)$$

6 Experimental results on homogeneous robots

In this section, we evaluate the performance of ScheduleNet on problems involving homogeneous robot teams. We show the results of optimizing a generic objective function, which is the minimization of total makespan. In Sect. 6.4, we also consider the application-specific objective function mentioned in Sect. 3.1, in which we minimize the weighted sum of task completion time, to investigate how ScheduleNet generalizes under different use cases.

6.1 Dataset

To evaluate the performance of ScheduleNet, we generate random problems based on Gombolay et al. (2013). We simulate multi-agent construction of a large workpiece, e.g. an airplane fuselage, with three different configurations: a two-robot team, a five-robot team, and a ten-robot team. Task duration is generated from a uniform distribution in the interval [1, 10]. Approximately 25% of the tasks have absolute deadlines drawn from a uniform distribution in the interval $[1, N \times T]$, where N is the number of total tasks. We use $T = 5$ for two-robot teams, $T = 2$ for five-robot teams, and $T = 1$ for ten-robot teams. Approximately 25% of the tasks have wait constraints, and the duration of non-zero wait constraints is drawn from a uniform distribution in the interval [1, 10]. We set the number of locations in a problem to be the same as the number of robots, and each task's location is picked randomly.

For each team configuration, problems are generated in three scales: small (16–20 tasks), medium (40–50 tasks) and large (80–100). For each problem scale, we generate 1000 problems for testing. To train the ScheduleNet model, we generate another 1000 small problems with two-robot teams. We run Gurobi with a cutoff time of 15 min on generated problems to serve as exact baselines for test set and expert demonstrations for training set. This resulted in a total of 17,513 transitions for training. To further examine the scalability of ScheduleNet, we also generate 100 ten-robot team problems in extra-large scale (160–200 tasks), and set the Gurobi cutoff time to be 1 hour, as the MILP formulation involves 300,000+ general constraints and 160,000+ binary variables.

6.2 Benchmark

We benchmark ScheduleNet against these methods:

- *EDF*—A ubiquitous heuristic algorithm, earliest deadline first (EDF), that selects from a list of available tasks the one with the earliest deadline, assigning it to the first available worker.
- *Tercio*—A state-of-the-art scheduling algorithm for this problem domain, Tercio (Gombolay et al. 2018). Tercio is a hybrid algorithm that combines mathematical optimization for task allocation and an analytical sequencing test to ensure temporal and spatial feasibility. Hyperparameters are chosen from Gombolay et al. (2018).
- *HomGNN*—A neural-network-based method proposed in Wang and Gombolay (2019). Their method uses a homogeneous GNN to exact problem embedding from the STN, and a separate Q-network consisting of two FC layers to predict the Q-value. We denote this model as HomGNN and use the same hyper-parameters in Wang and Gombolay (2019).
- *Exact*—Gurobi, a commercial optimization solver widely used for mixed integer linear programming. Its results represent the exact baseline.

6.3 Evaluation results

Metrics—For minimizing the makespan, we use the following metric for evaluation purpose. **M1**: Percentage of problems solved within optimality ratio. A problem is considered solved by an algorithm if the ratio, r, between the objective value it finds and the optimal value is within a certain range (e.g., $r = \frac{z_{algorithm}}{z_{optimal}} \leq 1.1$). Gurobi solutions are used as the optimal value. If the algorithm finds a solution of the problem which Gurobi fails to solve within cutoff time, we set $r = 1$ on this problem during evaluation. By calculating this metric with different optimal ratios, we can obtain a comprehensive view of how the solution quality an algorithm finds is distributed.

Model details—We implement ScheduleNet using PyTorch (Paszke et al. 2019) and Deep Graph Library (Wang et al. 2019a). The ScheduleNet used in training/testing is constructed by stacking four multi-head HetGAT layers (the first three use concatenation, and the last one uses averaging). The feature dimension of hidden layers = 64, and the number of heads = 8. We set $\gamma = 0.99$, $D = 3.0$ and used Adam optimizer (Kingma and Ba 2014) through training. The training procedure used a learning rate of 10^{-4}, $\lambda_1 = 0.9$, $\lambda_2 = 0.1$, $q_o = 3.0$ and batch size = 8. Both training and evaluation were conducted on a Quadro RTX 8000 Graphics Processing Unit (GPU).

The ScheduleNet was trained on small problems of two-robot teams and the same model was evaluated on all the different problem scales and team configurations. As HomGNN is not scalable in number of robots, for each team configuration, we trained a new model on 1000 small problems and used it for evaluation on the rest. Figures 4, 5 and 6 compared the evaluation results of different methods using **M1**, where optimality ratio ranges from 1 to 2 with intervals of 0.05 by default.

For small problems, as far as small optimal ratio ($r \leq 1.2$) is concerned, ScheduleNet outperformed three other heuristics (EDF, Tercio, and HomGNN) by a large margin, and achieved significantly closer results to the exact method. This result shows the effectiveness of ScheduleNet in finding high-quality feasible schedules. The only case where HomGNN performed similarly was when examined under a large optimal ratio ($r \geq 1.8$), indicating HomGNN was able to find more low-quality solutions, which is often not preferred.

For medium problems, both EDF and Tercio tended to find high-quality schedules, but with a low percentage, while

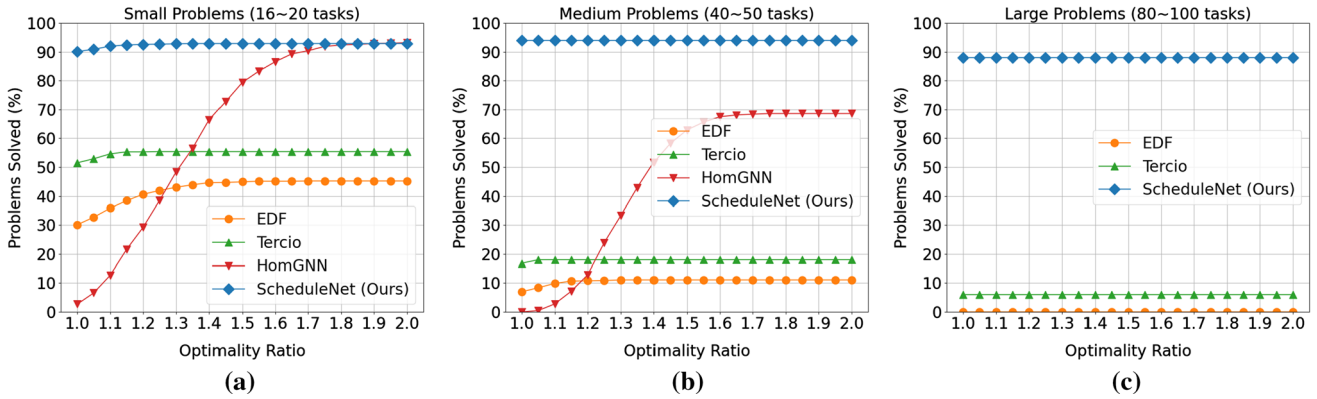


Fig. 4 Evaluation results on problems of two-robot teams of homogeneous robots: **a** small problems; **b** medium problems; **c** large problems

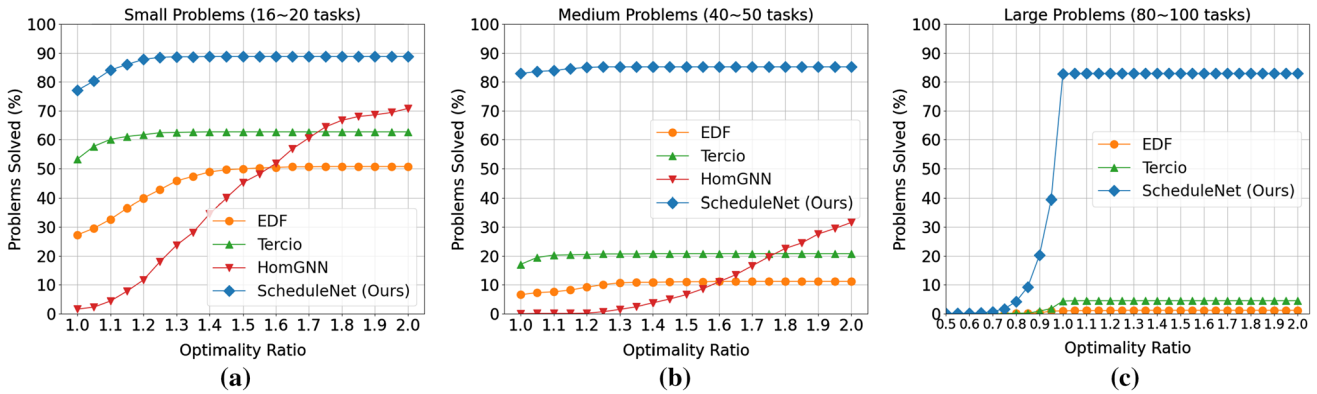


Fig. 5 Evaluation results on problems of five-robot teams of homogeneous robots: **a** small problems; **b** medium problems; **c** large problems. For 40% of the large problems, ScheduleNet's solutions outperform Gurobi within cutoff time as denoted by data points left of the 1.0 optimality ratio

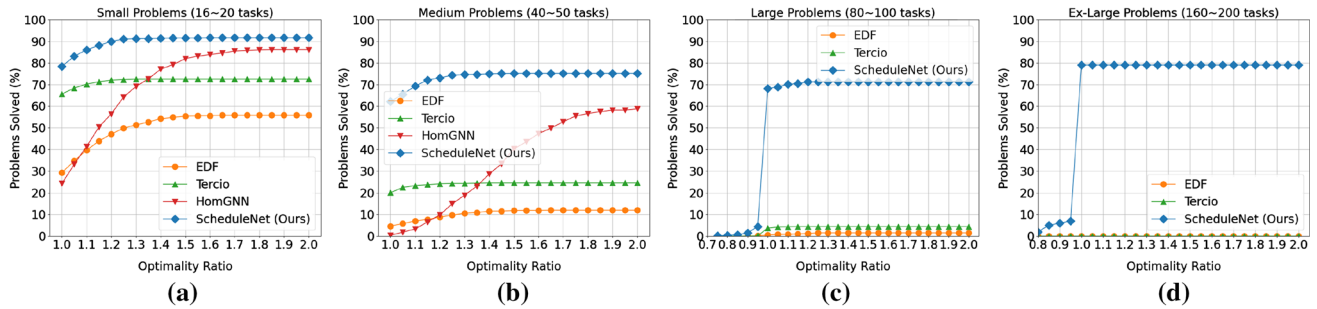


Fig. 6 Evaluation results on problems of ten-robot teams of homogeneous robots: **a** small problems; **b** medium problems; **c** large problems; **d** ex-large problems. In the large and ex-large problems cases, ScheduleNet is able to find solutions that outperform Gurobi as denoted by data points left of a 1.0 optimality ratio

HomGNN found more feasible low-quality solutions. Again, ScheduleNet model significantly outperformed the other three methods. Even though only trained with small problems, the performance of ScheduleNet remained consistent in solving medium and large problems, where a notable performance drop was observed for other methods. HomGNN failed to find solutions on large problems within Gurobi cutoff time (at least 40 min vs. 15 min), thus was not reported. During evaluation on large and ex-large problems, we found that for some problems the solutions found by SchedulerNet had better makespans than those found by Gurobi under its cutoff time. Therefore, we extended the opimality ratio to the smallest value under which ScheduleNet still solved at least one problem in Figs. 5c and 6c, d. For ex-large problems, Gurobi failed to find most of the feasible solutions within the one hour cuttoff time (8 solved out of 100), while ScheduleNet managed to find substantially more feasible schedules (79 solved). These results demonstrated that our model can

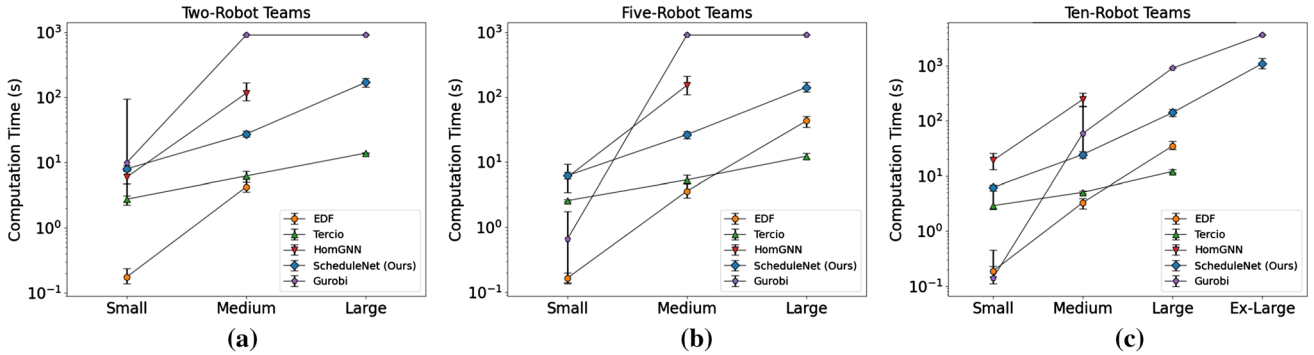


Fig. 7 Running time statistics on different problems of homogeneous robots: **a** two-robot teams; **b** five-robot teams; **c** ten-robot teams. Error bars denote the 25th and 75th percentile. Results for EDF, Tercio, and HomGNN are not shown in cases when no solutions are found within the allowed cutoff time

transfer knowledge learned on small problem to help solve larger problems, by exploiting the scalabililty within heterogeneous graph formulation.

We reported computation time of different methods in Fig. 7, where only feasible solutions were counted for each method. Due to differences in implementation details, CPU/GPU utilization, besides directly comparing the raw numbers, we also focused on the time changes of each method with respect to increasing problem sizes. When problem size increased, the performance of ScheduleNet stayed consistent with an affordable increase in computation time, which was less than Gurobi. This was largely due to the fully convolutional structure as well as the STN simplification trick that greatly reduced its model complexity and computation cost. As ten-robot team imposes a larger number of robot-related constraints than other team sizes, it took Gurobi less time to find solutions for ten-robot problems than two- and five-robot problems. In contrast, HomGNN failed to scale up to 100 tasks within Gurobi cuttoff time. This was mainly due to its structure, where FC layers are stacked on top of a GNN for Q-value prediction, making the model complexity proportional to $2 \times N_{task} \times N_{action}$ during parallel evaluation. As a comparison, the structure complexity of ScheduleNet is only proportional to $N_{task} + N_{action}$, considering $N_{robot}, N_{location} \ll N_{task}$.

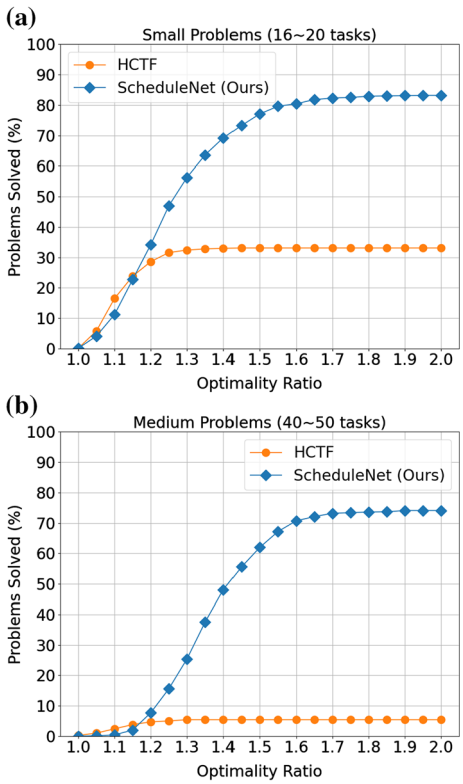


Fig. 8 Evaluation results of minimizing the weighted sum of completion times on five-robot teams of homogeneous robots: **a** small problems; **b** medium problems

6.4 Application-specific objective function

To evaluate the performance of our proposed method under a different objective function, $z = \sum_i c_i f_i$, we generated problems involving five-robot teams with two scales: small and medium, following the same parameters as used in Sect. 6.1. Additionally, each task was associated with a real number cost, c, drawn from a uniform distribution in the interval [1, 10]. This cost is added to the input features of task nodes. For each problem scale, 1000 problems were generated for testing. We generated 1000 small problems for training the ScheduleNet. We ran Gurobi on all problems with a cutoff time of 15 min to serve as exact baselines. We used the same set of parameters during training as used in the total makespan case, except $q_o = 30$, considering the reward was generally larger. We compare ScheduleNet against a Highest Cost Tardiness First (HCTF) priority heuristic which assigns the task with the highest cost to the first available worker in every scheduling decision. Figure 8 shows the evaluation results. For $r \leq 1.2$ both methods solved similar number of

problems. However, under larger optimaliy ratios, ScheduleNet started to outperform HCTF, resulting in a better overall performance.

7 Experimental results on heterogeneous task completion

In this section, we evaluate the performance of ScheduleNet for coordinating robots that are heterogeneous in task completion time with the objective of minimizing the team's makespan.

7.1 Dataset and benchmark

We generate random problems following the same setting and hyper-parameters as described in Sect. 6.1, with the following two differences:

1. For each task, τ_i, we sample a *mean* value from [1, 10] and a *gap* value from [1, 3], both with uniform distributions. Then we sample $dur_{i,r}$ for each robot from a uniform distribution in the interval $[mean-gap, mean+gap]$ (clamped at [1, 10] if applicable).
2. The task locations are randomly sampled from the 2D map. We use 2x2 for two-robot teams, 3x3 for five-robot teams, and 5x5 for ten-robot teams. The safety distance is set to 1. If $|Loc_i - Loc_j| \leq 1$, then $(\tau_i, \tau_j) \in \boldsymbol{L_{proximity}}$.

Considering that HomGNN (Wang and Gombolay 2019) is not designed for handling heterogeneous task completion among robots, we benchmark ScheduleNet against the remaining set of methods described in Sect. 6.2: EDF, Tercio and Exact (i.e., a MILP solved by Gurobi).

7.2 Evaluation results

Model Details—For performance evaluation, we use the same **M1** metric as used in Sect. 6.3. The ScheduleNet model also consisted of four multi-head HetGAT layers (the first three use concatenation, and the last one uses averaging). The feature dimension of hidden layers = 64, and the number of heads = 8. We set $\gamma = 0.95$, $D = 3.0$ and used Adam optimizer. The training procedure used a learning rate of 3×10^{-4}, $\lambda_1 = 0.9$, $\lambda_2 = 0.3$, $q_o = 3.0$ and batch size = 8. Both training and evaluation were conducted on a Quadro RTX 8000 GPU.

Same as in the homogeneous robot case, we trained the ScheduleNet for heterogeneous robot teams on small problems of two-robot teams and evaluate the same model on varying problem scales and team configurations. Evaluation results using **M1** are shown in Figs. 9, 10 and 11 where optimality ratios range from 1 to 2 with intervals of 0.05 by default.

For small and medium problems, ScheduleNet outperformed EDF and Tercio for medium-to-large optimality ratios ($r \geq 1.2$) other than small problems of ten-robot teams, and obtained results close to Tercio for $r \geq 1.5$ while consistently beating EDF. The improvement in performance of ScheduleNet over the baseline models was particularly significant in medium problems of two-robot and five-robot teams in which ScheduleNet found feasible solutions for more than half of the problems whereas Tercio and EDF found less than 15% for medium-to-high optimality ratios ($r \geq 1.2$). Compared to evaluation results for homogeneous robots, high-quality schedules for heterogeneous robots were much harder to find for all three methods with a much lower success rate overall in finding a feasible schedule.

For large and extra-large problems, we extended the optimality ratios (measured relative to the solution returned by Gurobi) to the smallest value under which ScheduleNet solved at least one problem. In addition to finding schedules that were more optimal than Gurobi for $\sim$ 10% of problems, ScheduleNet models significantly outperformed EDF and Tercio by finding feasible solutions for $>$ 30% of large problems compared to less than 5% by the latter baselines. Notably, for extra-large problems, EDF, Tercio and Gurobi all failed to find any feasible solutions, whereas ScheduleNet was able to find schedules for up to 24% of the problems. These results further demonstrate ScheduleNet's capability of generalizing learned knowledge to solving larger unseen problems.

Running time statistics of different methods were shown in Fig. 12, where we counted only feasible solutions found by each method. Because these problems of coordinating heterogeneous robot teams were much harder than the homogeneous robot case (Fig. 7), computation times in heterogeneous case (Fig. 12) increased for all three methods. Furthermore, Gurobi timed out significantly more frequently. For ScheduleNet, similar time change patterns with respect to increasing problem scales can be observed as the homogeneous robot case. Nonetheless, computation times of ScheduleNet are shorter than those of Gurobi and show a much better balance between solution quality and solving speed than EDF and Tercio, making ScheduleNet much more viable in practice.

8 Robot demonstration

We demonstrate our trained ScheduleNet model to coordinate the work of a five-robot team in a simulated environment for airplane fuselage construction, covering both homogeneous robot case in 1D space and heterogeneous robot case in 2D space. Our demo leverages the Robotarium, a remotely acces-

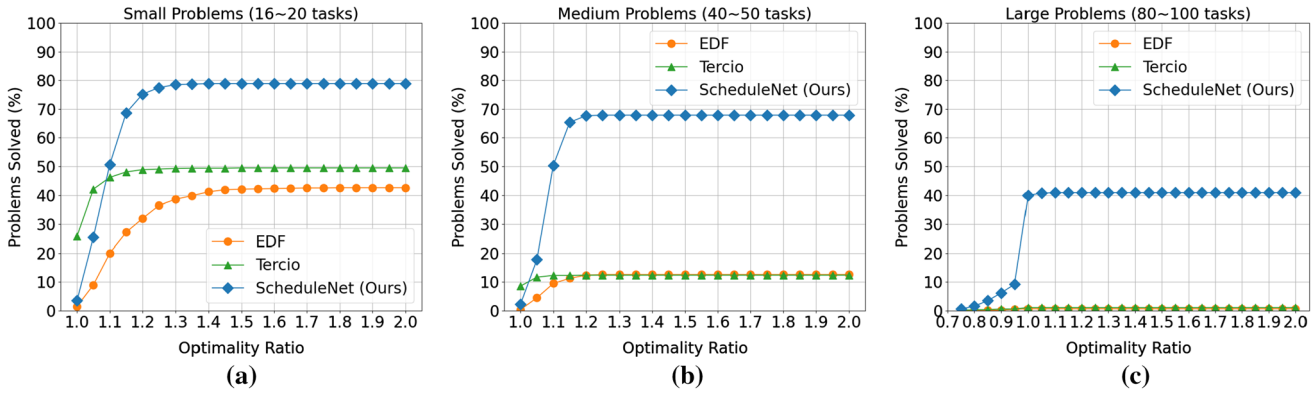


Fig. 9 Evaluation results on problems of two-robot teams of heterogeneous robots: **a** small problems; **b** medium problems; **c** large problems

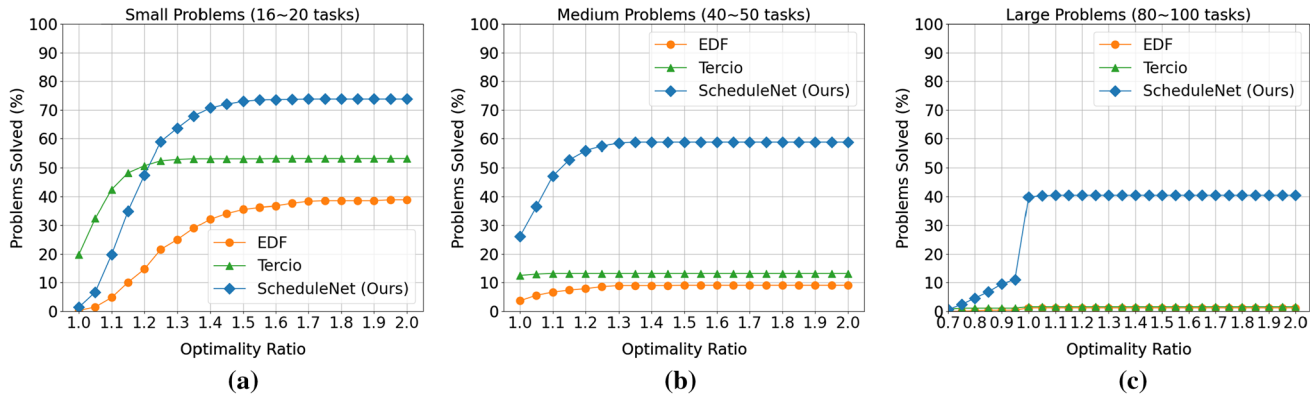


Fig. 10 Evaluation results on problems of five-robot teams of heterogeneous robots: **a** small problems; **b** medium problems; **c** large problems

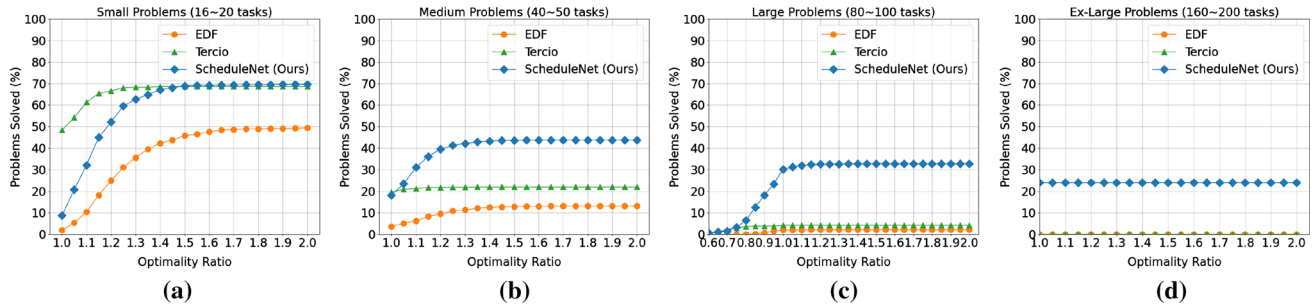


Fig. 11 Evaluation results on problems of ten-robot teams of heterogeneous robots: **a** small problems; **b** medium problems; **c** large problems; **d** ex-large problems

sible swarm robotics research testbed with GRITSBot X robots (Wilson et al. 2020). Examples of scheduling homogeneous robot teams and heterogeneous robot teams are shown in Fig. 13a, b, respectively. The ScheduleNet outputs for each step are depicted in bar plots at the bottom of each figure, with the selected task assignment highlighted in red. A detailed breakdown of the scheduling process with those examples can be found in the supplementary video.

9 ScheduleNet discussion

Our empirical results and analysis demonstrates that ScheduleNet establishes a state-of-the-art in autonomously learning heuristics for coordinating teams of robots in a computationally efficient framework. In particular, we demonstrate that our approach:

1. Outperforms prior work in multi-robot scheduling both in terms of schedule optimality and the total number of feasible schedules found (Figs. 4, 5, 6, 9, 10, 11).

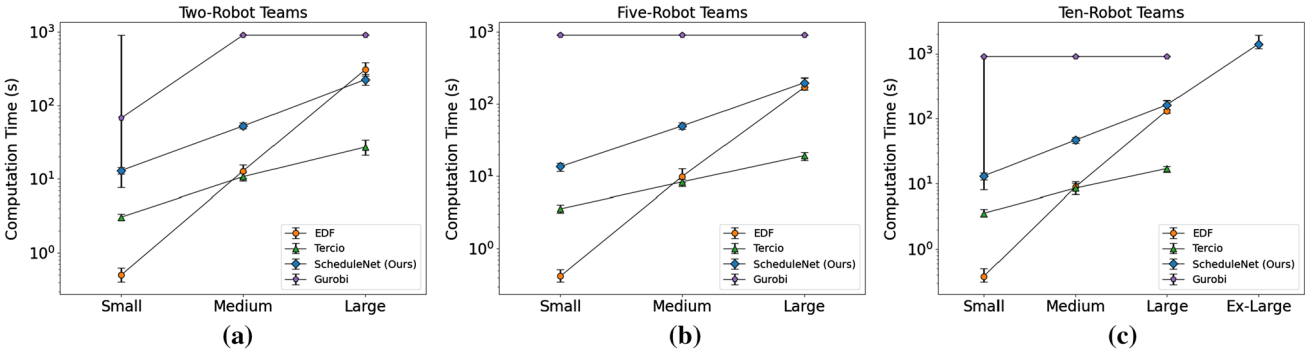


Fig. 12 Running time statistics on different problems of heterogeneous robots: **a** two-robot teams; **b** five-robot teams; **c** ten-robot teams. Error bars denote the 25th and 75th percentile

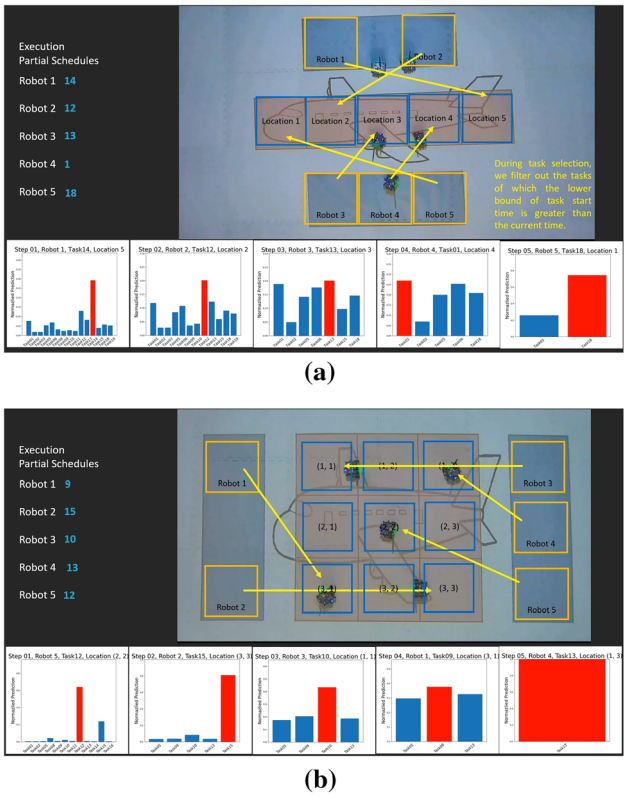


Fig. 13 Our demonstration of a 5-robot team completing tasks for airplane fuselage assembly. The ScheduleNet outputs for each step are plotted at the bottom, with the selected task assignment highlighted in red. **a** team of homogeneous robots with 1D task locations; **b** team of heterogeneous robots with 2D task locations (Color figure online)

2. Achieves this superior performance in a scalable framework that allows us to train via imitation-based Q-learning on smaller problems to provide high-quality schedules on much larger problems (Figs. 6d, 11d).
3. Autonomously learns scheduling policies on multiple application domains (Fig. 5 vs. Fig. 8, attaining an order of magnitude speedup vs. an exact method (Figs. 7, 12).
4. Leverages a highly flexible framework that models homogeneous robots and heterogeneous robots via graph augmentation. Given the high expressiveness of heterogeneous graphs, our research opens up future opportunities in designing graph-based learning algorithms in multi-robot research.

We also note our algorithm's limitations. First, high-quality expert data are required to train ScheduleNet as the loss function assumes the experts choose the optimal scheduling action at each time step. Therefore, sub-optimal expert data would likely lead to degrade model performance. In this paper, we assume it is practical to obtain high-quality solutions for small-scale problems where the problem complexity allows for exact algorithms to be used. Nonetheless, we propose in future work to investigate computational formulations of ScheduleNet that can explicitly reason about sup-optimality in expert scheduling demonstrations. Second, although ScheduleNet scales to different problem and team sizes, we observe a performance drop when the problem scale increases. Considering that our approach only trains on small scale problems, applying the learned representation to large scale problems, we propose further exploration into fine-tuning and transfer learning methods for improving the performance of ScheduleNet as problem sizes increases. Another limitation is that ScheduleNet assumes task durations to be deterministic and known a priori. Task performance in the real world is subject to many sources of uncertainty or randomness (Pinedo 2012), such as machine breakdowns, unexpected releases of high priority jobs, or uncertainty in the processing times. Although the speedup of ScheduleNet vs. exact methods allows us to dynamically re-schedule in a timely manner in response to unexpected disturbance during execution, we propose to extend our approach to reason about stochasticity in task completion times to ensure robust schedule execution.

10 Conclusion

We presented a novel heterogeneous graph attention network model, called ScheduleNet, to learn a scalable policy for multi-robot task allocation and scheduling problems. By introducing robot- and proximity-specific nodes into the simple temporal network that encodes the temporal constraints, we obtained a heterogeneous graph structure that is nonparametric in the number of tasks, robots and task resources. We showed that the model is end-to-end trainable via imitation learning with expert demonstrations, and generalizes well to large, unseen problems. Empirically, we showed that our method outperformed existing state-of-the-art methods in a variety of testing scenarios involving both homogeneous robot teams and heterogeneous robot teams. Future research includes incorporating reinforcement learning algorithms for fine-tuning ScheduleNet on large problems for potential performance gain, extending the model to allow scheduling mixed human-robot teams, as well as deploying the trained network in a real-world scenario.

Declarations

Funding This work was supported in part by the Office of Naval Research under grant GR10006659 and Lockheed Martin Corporation under grant GR00000509.

References

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research, 290*(2), 405–421. https://doi.org/10.1016/j.ejor.2020.07.063.

Bogner, K., Pferschy, U., Unterberger, R., & Zeiner, H. (2018). Optimised scheduling in human–robot collaboration—A use case in the assembly of printed circuit boards. *International Journal of Production Research*, *56*(16), 5522–5540.

Casalino, A., Zanchettin, A. M., Piroddi, L., & Rocco, P. (2021). Optimal scheduling of human–robot collaborative assembly operations with time petri nets. *IEEE Transactions on Automation Science and Engineering*, *18*(1), 70–84. https://doi.org/10.1109/TASE.2019.2932150.

Castro, E., & Petrovic, S. (2012). Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem. *Journal of Scheduling*, *15*(3), 333–346.

Chen, J., & Askin, R. G. (2009). Project selection, scheduling and resource allocation with time dependent returns. *European Journal of Operational Research*, *193*(1), 23–34.

Choudhury, S., Gupta, J., Kochenderfer, M.J., Sadigh, D.,& Bohg, J. (2020). Dynamic multi-robot task allocation under uncertainty and temporal constraints. In *Proceedings of Robotics: Science and Systems (RSS)*. https://doi.org/10.15607/rss.2020.xvi.068.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, *49*(1–3), 61–95.

Essafi, I., Mati, Y., & Dauzère-Pérès, S. (2008). A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. *Computers & Operations Research*, *35*(8), 2599–2616.

Flushing, E. F., Gambardella, L. M., & Di Caro, G. A. (2017). Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1861–1868). IEEE.

Fout, A., Byrd, J., Shariat, B., & Ben-Hur, A. (2017) Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, pp. 6530–6539.

Gombolay, M., Wilcox, R., & Shah, J. (2013) Fast scheduling of multi-robot teams with temporospatial constraints. In *Robotics: Science and system*, pp. 49–56.

Gombolay, M. C., Wilcox, R. J., & Shah, J. A. (2018). Fast scheduling of robot teams performing tasks with temporospatial constraints. *IEEE Transactions on Robotics*, *34*(1), 220–239.

Hamaguchi, T., Oiwa, H., Shimbo, M., & Matsumoto, Y. (2017). *Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach*. arXiv preprint arXiv:1706.05674.

Hamilton, W. L., Ying, R., & Leskovec, J. (2018). *Inductive representation learning on large graphs*. arXiv:1706.02216.

Hari, S. K. K., Nayak, A., & Rathinam, S. (2020). An approximation algorithm for a task allocation, sequencing and scheduling problem involving a human–robot team. *IEEE Robotics and Automation Letters*, *5*(2), 2146–2153.

Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, *24*(1), 1–13.

Kartal, B., Nunes, E., Godoy, J., & Gini, M. (2016). Monte Carlo tree search with branch and bound for multi-robot task allocation. In *The IJCAI-16 workshop on autonomous mobile service robots*, Vol. 33.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in neural information processing systems*, pp. 6348–6358.

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.

Kool, W., van Hoof, H., & Welling, M. (2019). Attention, learn to solve routing problems! In International conference on learning representations.

Korsah, G. A., Stentz, A., & Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, *32*(12), 1495–1512.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Nikou, A., Boskos, D., Tumova, J., & Dimarogonas, D. V. (2017) Cooperative planning for coupled multi-agent systems under timed temporal specifications. In *2017 American Control Conference (ACC)* (pp. 1847–1852). IEEE.

Nunes, E., Manner, M., Mitiche, H., & Gini, M. (2017). A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, *90*, 55–70.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8024–8035.

Pinedo, M. (2012). *Scheduling*, Vol. 29. Springer.

Piot, B., Geist, M., & Pietquin, O. (2014). Boosted bellman residual minimization handling expert demonstrations. In *Joint European Conference on machine learning and knowledge discovery in databases* (pp. 549–564). Springer.

Raghavan, H., Madani, O., & Jones, R. (2006). Active learning with feedback on features and instances. *Journal of Machine Learning Research*, *7*, 1655–1686.

Ren, H., & Tang, L. (2009). An improved hybrid milp/cp algorithm framework for the job-shop scheduling. In *2009 IEEE international conference on automation and logistics* (pp. 890–894). IEEE.

Shiue, Y. R., Lee, K. C., & Su, C. T. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering*, *125*, 604–614.

Solomon, M. M. (1986). On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks*, *16*(2), 161–174.

Solovey, K., Bandyopadhyay, S., Rossi, F., Wolf, M. T., & Pavone, M. (2020). Fast near-optimal heterogeneous task allocation via flow decomposition. arXiv preprint arXiv:201103603.

Tang, G., & Webb, P. (2019). Human–robot shared workspace in aerospace factories. In *Human–robot interaction: Safety, standardization, and benchmarking*, pp. 71–80.

Tsamardinos, I. (2000). *Reformulating temporal plans for efficient execution*. Master's thesis, University of Pittsburgh.

Tsamardinos, I., & Pollack, M. E. (2003). Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, *151*(1–2), 43–89.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). *Graph attention networks*. arXiv preprint arXiv:1710.10903.

Wang, H., Chen, W., & Wang, J. (2020). Coupled task scheduling for heterogeneous multi-robot system of two robot types performing complex-schedule order fulfillment tasks. *Robotics and Autonomous Systems*, *131*, 103560.

Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Ma, C., Huang, Z., Guo, Q., Zhang, H., Lin, H., Zhao, J., Li, J., Smola, A. J., Zhang, Z. (2019a). Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR workshop on representation learning on graphs and manifolds*.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P. S. (2019b). Heterogeneous graph attention network. In *The World Wide Web Conference* (pp. 2022–2032). ACM.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019c). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, *38*(5), 1–12.

Wang, Y. C., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, *18*(1), 73–82.

Wang, Z., & Gombolay, M. (2019). *Learning to dynamically coordinate multi-robot teams in graph attention networks*. arXiv preprint arXiv:1912.02059.

Wang, Z., & Gombolay, M. (2020). Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints. In *Robotics: Science and System XVI*.

Wilson, S., Glotfelter, P., Wang, L., Mayya, S., Notomista, G., Mote, M., et al. (2020). The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Systems Magazine*, *40*(1), 26–44.

Wu, J., Xu, X., Zhang, P., & Liu, C. (2011). A novel multi-agent reinforcement learning approach for job scheduling in grid computing. *Future Generation Computer Systems*, *27*(5), 430–439.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems,* 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386..

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *International conference on learning representations*.

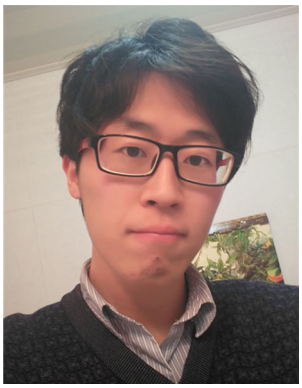
Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32, pp. 7444–7452.

Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, *10*(12), 399.

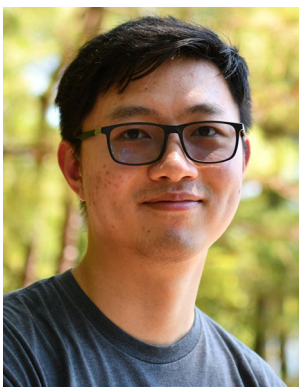
Yang, X., Deng, C., Liu, T., & Tao, D. (2020). Heterogeneous graph attention network for unsupervised multiple-target domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2020.3026079.

Zhang, S., Chen, Y., Zhang, J., & Jia, Y. (2020). Real-time adaptive assembly scheduling in human-multi-robot collaboration according to human capability. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3860–3866. IEEE.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2018). *Graph neural networks: A review of methods and applications*. arXiv preprint arXiv:1812.08434.



Zheyuan Wang is a Ph.D. student in the School of Electrical and Computer Engineering (ECE) at the Georgia Institute of Technology. He received the B.S. degree and the M. E. degree, both in Electrical Engineering, from Shanghai Jiao Tong University. He also received the M.S. degree from ECE, Georgia Tech. He is currently a Graduate Research Assistant in the Cognitive Optimization and Relational (CORE) Robotics lab, led by Prof. Matthew Gombolay. His current research interests include deep learning for optimization and its application in multi-robot coordination.



Chen Liu received his M.S. (2020) in computer science from Georgia Institute of Technology in Atlanta, GA and B.S. (2016) in computer science from Carnegie Mellon University in Pittsburgh, PA. His research interests include robotics and computer vision, specifically task scheduling, object detection and tracking. He worked as a software engineer at Snap Inc. in Los Angeles, CA from 2016 to 2019, and is joining Matroid in Palo Alto, CA as a deep learning engineer.

Dr. Matthew Gombolay is an Assistant Professor of Interactive Computing at the Georgia Institute of Technology. He received a B.S. in Mechanical Engineering from the Johns Hopkins University in 2011, an S.M. in Aeronautics and Astronautics from MIT in 2013, and a Ph.D. in Autonomous Systems from MIT in 2017. Gombolay's research interests span robotics, AI/ML, human–robot interaction, and operations research. Between defending his dissertation and joining the faculty at Georgia Tech, Dr. Gombolay served as a technical staff member at MIT Lincoln Laboratory, transitioning his research to the U.S. Navy, earning him an R&D 100 Award. His publication record includes a best paper award from American Institute for Aeronautics and Astronautics, a finalist for best student paper at the American Controls Conference, and a finalist for best paper at the Conference on Robot Learning. Dr Gombolay was selected as a DARPA Riser in 2018, received 1st place for the Early Career Award from the National Fire Control Symposium, and was awarded a NASA Early Career Fellowship for increasing science autonomy in space.