Learning Coordination Policies over Heterogeneous Graphs for Human-Robot Teams via Recurrent Neural Schedule Propagation

Batuhan Altundas[1], Zheyuan Wang[2], Joshua Bishop[2] and Matthew Gombolay[2]

Abstract— As human-robot collaboration increases in the workforce, it becomes essential for human-robot teams to coordinate efficiently and intuitively. Traditional approaches for human-robot scheduling either utilize exact methods that are intractable for large-scale problems and struggle to account for stochastic, time varying human task performance, or application-specific heuristics that require expert domain knowledge to develop. We propose a deep learning-based framework, called HybridNet, combining a heterogeneous graph-based encoder with a recurrent schedule propagator for scheduling stochastic human-robot teams under upper- and lower-bound temporal constraints. The HybridNet's encoder leverages Heterogeneous Graph Attention Networks to model the initial environment and team dynamics while accounting for the constraints. By formulating task scheduling as a sequential decision-making process, the HybridNet's recurrent neural schedule propagator leverages Long Short-Term Memory (LSTM) models to propagate forward consequences of actions to carry out fast schedule generation, removing the need to interact with the environment between every task-agent pair selection. The resulting scheduling policy network provides a computationally lightweight yet highly expressive model that is end-to-end trainable via Reinforcement Learning algorithms. We develop a virtual task scheduling environment for mixed human-robot teams in a multi-round setting, capable of modeling the stochastic learning behaviors of human workers. Experimental results showed that HybridNet outperformed other human-robot scheduling solutions across problem sizes for both deterministic and stochastic human performance, with faster runtime compared to pure-GNN-based schedulers.

I. INTRODUCTION

With collaborative robots (cobots) becoming more available in the industrial and manufacturing environments, robots and humans increasingly share the same work space to collaborate on tasks [1]. By removing the cage around traditional robot platforms and integrating robots into dynamic, final assembly operations, manufacturers can see improvements in reducing a factory's footprint and environmental costs as well as increased productivity [2]. In this paper, we focus on the problem of multi-agent task allocation and scheduling [3] with mixed human-robot teams over multiple iterations of the same task allocation problem. Our work accounts for and leverages stochastic, time-varying human task performance to quickly solve task allocation problems among team members to achieve a high-quality schedule with respect to the application-specific objective function while satisfying the temporal constraints (i.e., upper and lower bound deadline, wait, and task duration constraints).

Compared to task scheduling within multi-robot systems, the inclusion of human workers makes scheduling even more challenging because, while robots can be programmed to carry out certain tasks at a fixed rate, human workers typically have latent, dynamic, and task-specific proficiencies. Effective collaboration in human-robot teams requires utilizing the distinct abilities of each team member to achieve safe, effective, and fluent execution. For these problems, we must consider the ability of humans to learn and improve in task performance over time. To exploit this property, a scheduling algorithm must reason about a human's latent performance characteristics in order to decide whether to assign the best worker to a task now versus giving more task experience to a person who is slower but has a greater potential for fluency at that particular task. However, it is non-trivial to infer human strengths and weaknesses while ensuring that the team satisfies requisite scheduling constraints, due to the uncertainty introduced by variability in task execution behavior across different individuals, as well as uncertainty on future task performance affected by human's learning effects with practice [4]. Moreover, a lack of consideration for human preferences and perceived equality may, in the long run, put efficient behavior and fluent coordination at a contradiction [5].

Recent advances in scheduling methods for human-robot teams have shown a significant improvement in the ability to dynamically coordinate large-scale teams in final assembly manufacturing [6], [7]. Prior approaches typically rely on an assumption of deterministic or static worker-task proficiencies to formulate the scheduling problem as a mixed-integer linear program (MILP), which is generally NP-hard [8]. Exact methods are hard to scale and often fail to consider the time-varying stochastic task proficiencies of human workers over multi-round schedule execution that could result in significant productivity gains. The heuristic approaches may be able to determine task assignments; however, such approaches required domain specific knowledge that takes years to gain. We desire a scalable algorithmic approach that can automatically learn to factor in the human behavior for fast and fluent human-robot teaming.

Advancements in artificial intelligence have fostered the idea of leveraging deep neural networks (DNNs) to solve a plethora of problems in operations research [9]. DNNs can be trained to automatically explore the problem struc-
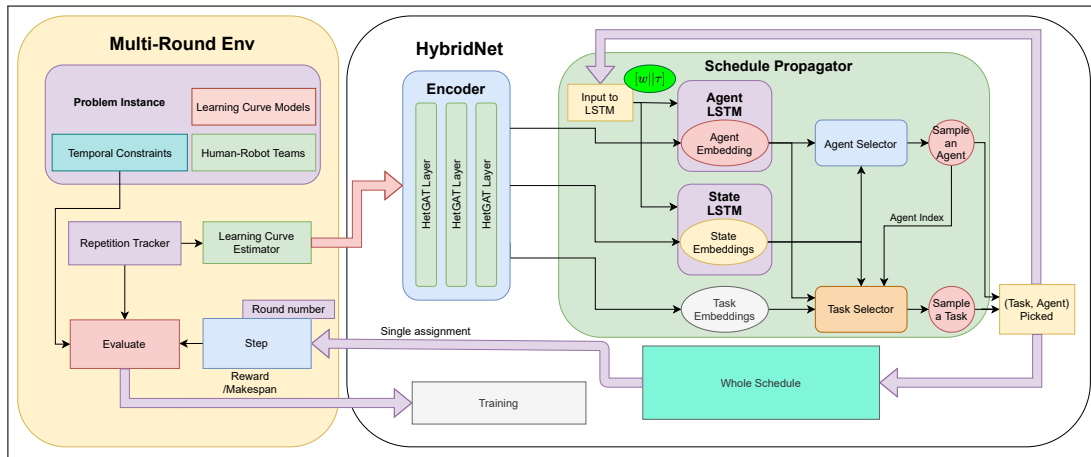
Fig. 1. Overview of Multi-Round Environment with HybridNet Scheduler. Left: The Multi-Round Scheduling Environment is developed to simulate a human-robot scheduling problem over multiple iterative rounds of execution, accounting for changes in human task performance. Right: HybridNet consists of a heterogeneous graph-based encoder to extract high-level embeddings of the problem and a recurrent schedule propagator for fast schedule generation.

ture and discover useful representations in high-dimensional data towards constructing high-quality solutions, without hand-crafted feature engineering [10]. Particularly, promising progress has been made in learning scalable solvers with graph neural networks via imitation learning (IL) or reinforcement learning (RL), outperforming state-of-the-art, approximate methods [11], [12], [13].

To overcome the limitations of prior work, we propose a deep learning-based framework, called HybridNet, for scheduling stochastic human-robot teams under temporal constraints. Figure 1 shows the overall framework of our proposed method operating in a multi-round environment. HybridNet utilizes a heterogeneous graph-based encoder and a recurrent schedule propagator. The encoder extracts high level embeddings of the scheduling problem using a heterogeneous graph representation of the problem extended from the simple temporal network (STN) [14]. By formulating task scheduling as a sequential decision-making process, the recurrent propagator uses Long Short Term Memory (LSTM) cells to carry out fast schedule generation. The resulted policy network provides a computationally lightweight yet highly expressive model that is end-to-end trainable via reinforcement learning algorithms.

The primary contributions of our work are:

- We propose a deep learning-based framework, HybridNet, for human-robot coordination under temporal constraints. HybridNet consist of a Heterogeneous Graph-based encoder and a Recurrent Schedule Propagator. The encoder extracts relevant information about the initial environment, while the Propagator generates the consequential models of each task-agent assignments based on the initial model. Inspired by the sensory encoding and recurrent processing of the brain, this approach allows for fast schedule generation, removing the need to interact with the environment between every task-agent pair selection.
- We develop a virtual task scheduling environment for mixed human-robot teams in a multi-round setting, capable of modeling the stochastic learning behavior

of human workers. We make our environment OpenAI gym-compatible and expect it to serve as a testbed to facilitate the development of human-robot scheduling algorithms. The implementation is publicly available.[1]

- We present a novel policy model that jointly learns how to pick agents and tasks without interacting with the environment between intermediate scheduling decisions and only needs a single reward at the end of schedule. By factoring in the action space into an agent selector and a task selector, we enable conditional policy learning with HybridNet. We account for the state and agent models when selecting the agents, and combine the information regarding the tasks, selected agent and the state for task assignment. As a result, HybridNet is end-to-end trainable via Policy Gradients algorithms.
- We conducted extensive experiments to validate HybridNet across a set of problem sizes. Results showed HybridNet consistently outperformed prior human-robot scheduling solutions under both deterministic and stochastic settings.

II. RELATED WORK

A. Multi-Agent Scheduling Problem

Task assignment and scheduling of multi-agent systems is an optimization problem that has been studied for real world applications, both for Multi-Robot Task Allocation(MRTA) problem using traditional techniques [15] and deep learning based techniques [16] as well as for human-robot collaboration [7]. Task Allocation can be formalised by Mixed Integer Linear Programming (MILP) to capture it's constraints. The exponential complexity of solving the MILP can be accelerated through constraint programming methods [7], [17], [18] or heuristic schedulers to leverage better scalability [19], [20]. Zhang et al. encoded task schedules as chromosomes for a genetic algorithm that optimized schedules for heterogeneous human-robot collaboration by repeatedly crossing over and mutating the solutions to find the optimal schedule. [21]

---

[1]https://github.com/altundasbatu/HybridNet_IROS2022

Gombolay et al. present an algorithm to capture domain knowledge through scheduling policy requiring domain-expert demonstrations [22]. Wang et al. propose ScheduleNet, a Heterogenous Graph Neural Networks-based model for task allocation under temporospatial constraints, trained through Imitation Learning using optimal schedule [23]. ScheduleNet relies on interactive scheduling scheme, with constant update of an environment before reaching a complete schedule. These approaches require optimal schedules generated by other expert systemsto train and have high computational complexity that make their implementation costly.

B. Modeling Human-Robot Teams

As advancements in robot capability progress, they become safer and effective to use in conjunction with humans to complete specialized works. Liu et al. presents a model of human task completions, showing an increase in the task efficiency as a result of learning. This paper shows that prediction of human performance enhances the ability of the scheduling systems to explicitly reason about the agents' capabilities [4]. Prior work on behavioral teaming and the natural computational intractability of large-scale schedule optimization suggests that robots can offer a valuable service by designing and adapting schedules for human teammates.

In our system, we leverage the findings of Liu et al. to account for humans learning over time, both in problem generation as part of the environment and a learning curve predictor as part of the scheduling policy. The human learning curve follows an exponential function of generic form over the course of multiple iterations as shown in Equation 1 [4]:

$$y = c + ke^{-\beta i} \tag{1}$$

where i is the number of iteration the human has previously executed a task and c, k, β parameters. We further account for the stochastic-nature of human learning in our environment.

C. Graph Neural Networks

Graph Neural Networks (GNNs) 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 [24]. GNNs have been widely applied in graph-based problems such as node classification, link prediction and clustering, and they have shown to have an impressive performance [25]. The Heterogeneous Graph Attention Network presented in Wang et al. utilizes Deep Learning Algorithms to address the Scheduling Problem, showing improved performance compared to non-Deep Learning Schedulers such as Earliest-Deadline First (EDF) [26] and Tercio [7] at the cost of increased computational complexity [23].

D. LSTM Based Sequence Prediction

The impact of the LSTM network has been notable in language modeling [27], speech-to-text transcription[28], machine translation [29], and other applications that involve

predictive modeling [30], [31], [32]. The advantage of this lengthier path generated through the recurrent nature of the neural network is that it affords an opportunity to build a certain degree of intuition that can prove beneficial during all phases of the process [30], [33].

III. HUMAN-ROBOT SCHEDULING PROBLEM

A. Problem Overview

In this paper, we focus on the problem of human-robot task allocation and scheduling with temporal constraints [15]. We describe the problem components using a 4-tuple $\langle a, \tau, d, w \rangle$ form. a represents all agents that belong to the human-robot team, τ represents all the tasks to be performed. Each task, τ_i, and agent, a_j, have a task completion duration $dur(\tau_i, a_j)$ and agents are capable of completing a sequence of tasks in order. d contains the set of deadline constraints, where $d_i \in d$ specifies the tasks depending on τ_i [23]. w is the set of wait constraints where $w_{ij} \in w$ denotes the wait time between tasks τ_i and τ_j. A Schedule, S, is a sequence of task-agent pairs $\langle \tau_i, a_j \rangle$ such that S contains all tasks in τ.

B. Multi-Round Scheduling Environment

The Multi-Round Scheduling Environment is developed to simulate a human-robot scheduling problem over multiple iterative rounds of execution, accounting for changes in the task performance of human workers based on previous round. Each round is a step in the OpenAI Gym-compatible environment, taking as input the complete set of task-agent pairs for the scheduling problem, simulating the sequential assignment of tasks to agents.

Each round's execution is considered finished when all the tasks are assigned to one of the agents or if the provided schedule is determined to be infeasible under the problem constraints. The environment checks the feasibility of the provided schedule given the constraints of the problem, and computes the total duration of task completion of the schedule if the schedule is feasible. If the schedule does not satisfy the constraints, it is determined to be infeasible and the list of tasks that could not been scheduled are returned.

We formulate the Multi-Round Scheduling Environment as a Partially Observable Markov Decision Process (POMDP) using a six-tuple $\langle S, A, T, R, \Omega, O, \gamma \rangle$ below:

- **States**: The problem state S is a state of the Multi-Round Environment consistent of the state of the Agents.
- **Actions**: Actions at round t within the Multi-Round Environment refers to a complete set of Task Allocations made up of a list of task-agent pairs, denoted as $A_t = [\langle \tau_{i_1}, a_{j_1} \rangle, \langle \tau_{i_2}, a_{j_2} \rangle, ...]$ to be executed in order.
- **Transitions**: T corresponds to executing the action in Multi-Round Scheduling Environment and proceed to next time step.
- **Rewards**: R_t is based on the scheduling objective a user wants to optimize. In Section III-E we show how to compute R_t when optimizing makespan.
- **Observations**: Ω is the estimated performance of all the task-agent pairs, plus the observable constraints.

- Observation Function: O is handled by the *Learning Curve Estimator* explained in the Section III-D.
- Discount factor, γ

C. Agent Models

The Multi-Round Environment stores the Agent information, allowing the environment to keep track of each agent and which tasks it has previously completed. The update of the Environment happens at the end of each round, allowing agents to modify themselves based on their internal models. to update the model based on the selected (task-agent) pairs for each round.

1) Determinitic Robot Model: We generate the robot task completion times randomly through uniform distribution.

2) Stochastic Human Model: We generate the human task completion times randomly based on Equation 1, such that the Environment can be setup to provide Deterministic and Stochastic performance for human learning. The task duration parameters of the human learning model, c, k, β, in Equation 1 are built from the randomly selected initial task completion time for round 0. For Stochastic performance, the standard deviations are used to sample from a Normal Distribution as presented in Liu et al. [4].

D. Learning Curve Estimator

The scheduler is given an estimate of the performance of the human agents for each task based on the information about the task duration of the previous executions of the task-agent pair through the *Learning Curve Estimator* as part of our OpenAI Gym-like Environment In our paper, we have implemented a black box model based on the insights presented in Liu et al.[4] to simulate a Stochastic Human Learning Estimator. As an Agent completes a task in multiple rounds, the Agent Model records the task completion duration, allowing *Learning Curve Estimator* to predict the next task-agent duration more accurately. To represent the increase in accuracy from increase in information, we implemented a Learning Curve Estimator that generates an estimate of the human agent performance using the actual task performance as the mean of a Gaussian Distribution with noise that exponentially decreases with the number of repetitions of the same task for that agent in previous rounds.

E. Reward Design

The total reward, R_t, for the schedule generated by the multi-round scheduling environment is calculated based on feasible, A', and infeasible, $\tilde{A}'$, subsets of task allocations, such that $A_t = A'_t \cup \tilde{A}'_t$. Specifically, the reward, R_t, is based on the expected reward for the feasible subset of task-agent assignments, $R_t(A'_t)$, and the reward from the assignment of the infeasible subset of task-agent assignments, $R_t\tilde{A}'_t...$, based on the point estimate of the reward from assigning the incomplete task to the agent that will complete it in the longest possible duration, multiplied by an infeasible coefficient C_i as shown in equation 2:

$$R_t = \sum_{i \in A'_t} R(\tau_i, a_i) + C_i \max_{a_j} \left(\sum_{i \in \tilde{A}'} R(\tau_i, a_j) \right) \quad (2)$$

The Total Schedule Reward, R_S, favors schedules with more feasible task allocations and enables learning from infeasible explorations during training.

IV. HYBRIDNET SCHEDULING POLICY

As shown in Figure 1, our HybridNet framework consists of a heterogeneous graph-based encoder to learn high level embeddings of the scheduling problem, and a recurrent schedule propagator to generate the team schedule sequentially. This hybrid network architecture enables directly learning useful features from the problem structure, owing to the expressiveness of heterogeneous graph neural networks, and at the same time efficiently constructing the schedule with our LSTM-based propagator. As a result, HybridNet does not require interacting with the environment between every task-agent pair selection, which is necessary but computationally expensive in prior work [16], [23].

We denote the policy learned by HybridNet as $\pi_\theta(A|S)$, with θ representing the parameters of the neural network. At round t, an action takes the form of an ordered sequence of scheduling decisions, $A_t = \{d_1, d_2, ..., d_n\}, d_i = \langle \tau_i, a_j \rangle$, where a latter decision, d_i, is conditioned on its former ones, $d_{1:i-1}$. Then, the policy can be factorized as

$$p_\theta(A_t|S_t) = \prod_{i=1}^{n} p_\theta(d_i|S_t, d_{1:i-1}) \quad (3)$$

Using the Recurrent Schedule Propagator, HybridNet recursively computes the conditional probability, $p_\theta(d_i|S_t, d_{1:i-1})$, for sampling a task-agent pair. At the end, the network collects all the decisions and sends to the environment for execution.

A. Heterogeneous Graph Encoder

We build our Encoder using the heterogeneous graph attention (HetGAT) layer proposed in [23] that has been shown effective in representation learning of multi-agent scheduling problems. At the start of each round for a given human-robot scheduling problem, the heterogeneous graph representation is built by extending from the simple temporal network (STN) that encodes the temporal constraints to include agent nodes and a state summary node. The metagraph of the resulted graph is shown in Figure 2, which summarizes all the node types and edge types. Then, a HetGAT layer computes the output node features by performing per-edge-type message passing followed by per-node-type feature reduction, while utilizing a feature-dependent and structure-free attention mechanism. We refer interested readers to [23] for full details of implementing a HetGAT layer.

By stacking several HetGAT layers sequentially, we construct the Encoder that utilizes multi-layer structure to extract high-level embeddings of each node that will be send to the propagator for schedule generation. We follow the same
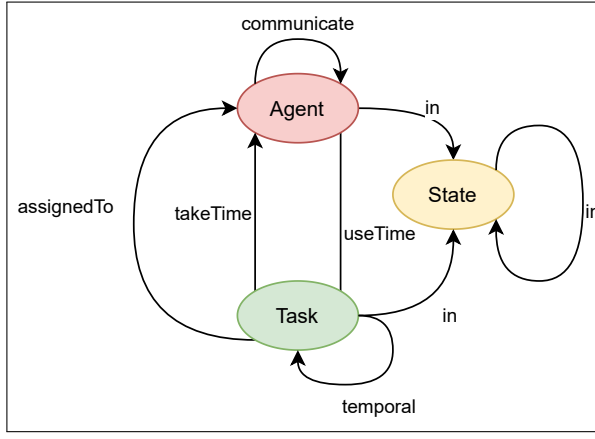
Fig. 2. Metagraph of the heterogeneous graph built from the STN by adding agent and state summary nodes.

hyper-parameters for HetGAT layers as provided in Wang et al. [23]

B. Recurrent Schedule Propagator

The HetGAT layers are computationally complex and require interactive scheduling to generate the initial model. By utilizing an LSTM based Recurrent Predictor, we propagate forward consequences of each task-agent assignment, recreating the encoded information about the environment without relying on the initial HetGAT Layer, significantly reducing the computational complexity of our scheduler.

The Recurrent Schedule Propagator takes as input the Task, State and Agent embeddings generated by the Heterogeneous Graph Encoder and sequentially generates task-agent pairs based on the encoded information. To predict the consecutive encoding of state and agents, we use an LSTM Model to recursively generate the Agent and State after each assignment of a task to an agent, without interacting with the Environment, outputting the sequential task-agent assignment for the complete set of tasks. The pseudo-code for scheduling generation with HybridNet is presented in Algorithm 1.

As $d_i = \langle \tau_i, a_j \rangle$, we further factor $p_\theta(d_i|S_t, d_{1:i-1})$ into an agent selector and a task selector. That is, $\pi_{factor}(d|\cdot) = \pi_{agent}(a_j|\cdot) \cdot \pi_{task}(\tau_i|a_j,\cdot)$. This factorization allows the policy to capture the underlying composite and conditional nature of the scheduling decisions, where the task to schedule is strongly dependent on the picked agent.

The Agent Selector selects the new agent for the next decision d based on the state and agent information. Specifically, the concatenated state-agent embeddings are processed by a feed-forward neural network, f_a, to compute the likelihood of selecting each agent for the next task-agent pair, using Equation 4. A softmax operation is performed to convert the raw predictions into a probability distribution. After the selection of the agent, the agent embedding of the chosen agent is updated based on the selected task and state embeddings, as state change only happens for the assigned agent. This approach allows the agent selector to consider how busy each agent is, based on the inherent information

---

Algorithm 1 Psuedocode for Schedule Generation
Input: graph g, features f, unscheduled-Tasks u
Output: *schedule*

1: $schedule = [\ ], i = 1$
2: $(h_{a_1}, c_{a_1}, h_{t_1}, c_{t_1}, h_{s_1}, c_{s_1}) \leftarrow Encoder(g, f)$
3: **while** $|u| \neq 0$ **do**
4:     $p_{a_i} \leftarrow AgentSelector(h_{s_i}, h_{a_i})$
5:     $a_i \leftarrow Sampling(p_{a_i})$
6:     $p_{t_i} \leftarrow TaskSelector(h_{t_i}, h_{s_i}, a_i)$
7:     $t_i \leftarrow Sampling(p_{t_{i-1}})$
8:     $schedule.append(\langle t_i, a_i \rangle)$
9:     $unscheduledTasks.remove(t_i)$
10:     **if** $|unscheduledTasks| == 0$ **then**
11:       **return** *schedule*
12:     **end if**
13:     $i \leftarrow i + 1$
14:     $h_{s_i}, c_{s_i} \leftarrow LSTM_s((h_{t_{i-1}}[t_i], h_{a_{i-1}}[a_i]),$
                            $h_{a_{i-1}}, c_{a_{i-1}})$
15:     $h_{a_i}, c_{a_i} \leftarrow LSTM_a((h_{t_{i-1}}[t_i], h_{a_{i-1}}[a_i]),$
                            $h_{a_{i-1}}, c_{a_{i-1}})$
16: **end while**

presented in the embeddings.

$$\pi_{agent}(a_j|s) = softmax_i(f_a([h_{a_j}||h_s])) \qquad (4)$$

Next, the Schedule Propagator uses the Task Selector to assign the task for the selected agent based on the state, agent and unscheduled task embeddings. As shown in Equation 5, the Task Selector concatenates the state, selected agent and the unscheduled task embeddings and passes the combined information to a feedforward neural network, f_τ, to calculate the likelihood of the task being assigned to the selected agent. After assigning to an agent for execution, the tasks are removed from the list of unscheduled tasks. Since the calculation of likelihood of each task is independent of each other up to the last softmax operation, the model is scalable and can be used for differentproblem sizes.

$$\pi_{task}(\tau_i|a_j, s) = softmax_i(f_\tau([h_{\tau_i}||h_{a_j}||h_s])) \qquad (5)$$

The key component of the Schedule Propagator is the use of LSTM. As shown in line 12 of Algorithm 1, after each task-agent pair selection, the state and agent embeddings are updated using the state LSTM and agent LSTM, respectively. The LSTM Cell stores the hidden and cell data from the previous step of the task allocation and predicts the next step based on the input using the Equation 6 [33].

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_t, x_t] + b_i) \\
\tilde{c}_t &= tanh(W_c[h_{t-1}, x_t] + b_c) \\
c_t &= f_t c_{t-1} + i_t \tilde{c}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned} \qquad (6)
$$

Where the Encoder produces initial hidden state, h_1 and initial cell state c_1 as an output in the form of $[h_1, c_1]$.

During testing, we utilize a batched sampling strategy for further performance gains. Specifically, we generate multiple schedules for the same task allocation problem every round.

We select the best performing schedule by computing the estimated makespan utilizing the Learning Curve Estimator and provide it to the Multi-Round Environment. More sampling improves solution quality at increased computation.

C. Stochastic Policy Learning

We train HybridNet in multi-round scheduling environments using Policy Gradient methods that seek to directly optimize the model parameters based on rewards received from the environment [34]. Specifically, we compute the gradient of the model using the sum of the log likelihood of Agent and Task Selectors, as shown in Equation 7:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi(\sum_t^T A_t^{\pi_\theta}(s_t, \langle \tau_i, a_i \rangle) \\ \nabla_\theta(log\pi_\theta(\tau_i|a_i, s_t) + log\pi_\theta(a_i|s_t)) \quad (7)$$

In Equation 7, the advantage term, A_t is estimated by subtracting a "baseline" from the total future reward calculated in Equation 2. We calculate the "baseline" using the reward generated for the same task-allocation problem from multiple batches executed in multiple sequential rounds in the Multi-Round Environment. Each element of the batch solves the same scheduling problem and the environment is updated to account for the task-allocation of the previous round, updating the agent models. The gradients were calculated from Equation 7 to updated the model weights.

Due to the combinatorial nature of the task scheduling problem, plus the stochasticity in human proficiency, learning a helpful value function as a baseline for computing the advantage term is non-trivial. Instead, we investigate two more accessible and efficient alternatives:

- *Step-based Baseline:* During gradient estimation, the baseline value subtracted is set as the average return value across training episodes in the current batch.
- *Greedy Rollout Baseline:* Greedy Rollout Baseline uses, $\pi_{greedy}(A|S)$, a deterministic greedy version of the HybridNet scheduler, to collect rewards in the environment. Its weights, θ_{greedy}, are updated periodically by copying the weights from the current learner, $\pi_\theta(A|S)$.

V. EXPERIMENTAL RESULTS

A. Data Generation

We generate scheduling problems with deadline and wait constraints under different scales. For all scales, the deadline constraints are randomly generated for approximately 25% of the tasks from a range of $[1, 5N]$ where N is the number of tasks. Approximately 25% of the tasks have wait constraints, and the duration of non-zero wait constraints is sampled from $U([1, 10])$. Task durations are clamped to 10 to 100.

1) Small Scale: The small data set has 9 to 11 tasks with 2 robots and 2 humans in a team. We generated 2000 Training Problems and 200 Test Problems.

2) Medium Scale: The medium data set has 18 to 22 tasks with 2 robots and 2 humans in a team. We generated 2000 Training Problems and 200 Test Problems to inspect the scalability of our trained model.

3) Large Scale: The large data set is defined as problems with 36 to 44 tasks chosen at random with 2 robots and 2 humans in a team. We have generated 200 Test Problems to evaluated the HybridNet performance with zero training problems (i.e., zero-shot transfer to from the smaller scale datasets to the Large Scale dataset)..

To simulate the stochastic learning of human agents, for each Data Set noise is introduced to the Human Agent models by simulating the natural distribution of the c, k, β parameters of Equation 1. This allows for each Data Set to simulate Deterministic and Stochastic Human Performance. The stochastic model is clipped to fall within the specified range of task durations.

B. Benchmarking

We benchmark HybridNet against the following 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 agent.
- *Genetic Algorithm*: An Evolutionary Optimization Algorithm that uses Post-Processing on the Schedule Generated by EDF [21]. Genetic algorithm creates new schedules based on the initial schedule through iterative randomized mutations by swapping task allocations and task orders [4]. Each generation selects the top performing schedules, sorted on feasibility and total schedule completion time, and used as the baseline for creating new mutations. The Genetic Algorithm was run for 10 generation with 90 baseline schedules, 10 task allocation and 10 task order swapping mutations.

Furthermore, we evaluate the functionality of the Recurrent Schedule Propagator by comparing it against the following HybridNet variant:

- *HetGAT*: We implement a HetGAT Scheduler based on the Encoder of HybridNet. After each task-agent pair assignment, instead of using the LSTM Cells to update the task, agent and state embeddings, it directly interacts with the environment to model the consequences of action with a new heterogeneous graph and re-computes those information from it.

We evaluate HybridNet on three metrics: 1) Proportion of problems solved; 2) Adjusted makespan: determined by the average of the makespan of feasible schedules and the maximum possible makespan of the infeasible schedules; and 3) Runtime statistics. Runtime statistics for training and execution is compared for HybridNet and HetGAT Scheduler to model their computational complexity. Because HetGAT Scheduler relies on interactive scheduling through the environment after every task-agent pair allocation, we only train and evaluate it for Deterministic Human Performance.

C. Model Details

We implement HybridNet and HetGAT using PyTorch [35] and Deep Graph Library [36]. The HybridNet Encoder used in training/testing is constructed by stacking three multi-head

TABLE I
EVALUATION RESULTS: ADJUSTED MAKESPAN AND FEASIBILITY WITH DETERMINISTIC HUMAN TASK PROFICIENCY COMPARING BENCHMARKS WITH HYBRIDNET TRAINED ON SMALL AND MEDIUM SCALES, WITH SCHEDULES SAMPLED FROM SIZES 8 AND 16

Training	Methods	Small		Medium		Large	
		Makespan	Feasibility (%)	Makespan	Feasibility (%)	Makespan	Feasibility (%)
-	EDF	239.31	73.00	1109.85	15.00	2535.89	1.00
-	Genetic Algorithm	302.42 ± 0.77	74.10 ± 0.30	1180.07 ± 2.54	16.60 ± 0.70	2542.79 ± 0.06	1.00 ± 0.00
Step-based	HetGAT 8	257.20 ± 0.18	86.29 ± 0.08	751.27 ± 1.29	50.17 ± 0.14	2123.96 ± 5.66	17.12 ± 0.27
	HetGAT 16	249.69 ± 0.30	86.51 ± 0.09	723.57 ± 0.94	50.29 ± 0.11	2081.65 ± 5.45	17.15 ± 0.16
Greedy	HetGAT 8	261.15 ± 0.09	85.59 ± 0.10	784.32 ± 0.52	53.28 ± 0.17	2017.25 ± 2.16	23.98 ± 0.14
	HetGAT 16	255.70 ± 0.23	86.05 ± 0.15	765.79 ± 0.96	53.41 ± 0.08	1983.73 ± 1.59	23.84 ± 0.01
Step-based	HybridNet Small 8	260.22 ± 0.15	86.93 ± 0.10	770.48 ± 1.07	59.11 ± 0.35	2005.80 ± 2.33	30.65 ± 0.39
	HybridNet Small 16	**252.57 ± 0.49**	**87.08 ± 0.10**	746.35 ± 0.52	60.89 ± 0.36	1953.65 ± 3.76	33.24 ± 0.61
Greedy	HybridNet Small 8	266.74 ± 0.31	84.65 ± 0.32	758.96 ± 2.27	61.09 ± 0.43	2049.32 ± 3.73	28.74 ± 0.45
	HybridNet Small 16	258.17 ± 0.45	85.13 ± 0.20	723.35 ± 1.70	63.68 ± 0.49	1973.15 ± 2.91	32.46 ± 0.40
Step-based	HybridNet Medium 8	-	-	722.85 ± 0.61	64.69 ± 0.29	2010.86 ± 1.97	30.86 ± 0.45
	HybridNet Medium 16	-	-	697.40 ± 2.04	66.25 ± 0.51	1944.72 ± 4.10	33.88 ± 0.49
Greedy	HybridNet Medium 8	-	-	692.01 ± 3.69	68.33 ± 0.66	2011.78 ± 5.08	30.58 ± 0.87
	HybridNet Medium 16	-	-	**659.01 ± 0.89**	**71.00 ± 0.45**	**1936.97 ± 4.68**	**34.66 ± 0.74**

TABLE II
EVALUATION RESULTS: ADJUSTED MAKESPAN AND FEASIBILITY WITH STOCHASTIC HUMAN TASK PROFICIENCY

Methods	Small		Medium		Large	
	Makespan	Feasibility (%)	Makespan	Feasibility (%)	Makespan	Feasibility (%)
EDF	227.81 ± 6.17	75.65 ± 1.21	1071.02 ± 20.65	17.30 ± 1.12	2524.92 ± 8.95	1.15 ± 0.23
Genetic Algorithm	283.79 ± 10.39	77.45 ± 2.05	1149.42 ± 12.14	19.55 ± 1.31	2541.20 ± 3.54	1.05 ± 0.15
HybridNet Small	**298.81 ± 0.96**	**79.54 ± 0.52**	881.16 ± 2.89	48.89 ± 1.09	**2141.80 ± 5.12**	**23.51 ± 0.96**
HybridNet Medium	-	-	**859.99 ± 4.82**	**51.94 ± 1.32**	2174.57 ± 8.53	22.31 ± 0.94

TABLE III
EVALUATION RESULTS: RUNTIME PERFORMANCE ON SINGLE PROBLEM

	Methods	HetGAT8	HybridNet8	HybridNet16
Training Time (s)	Small	184.52 ± 18.00	19.97 ± 0.91	-
	Medium	354.77 ± 38.31	22.40 ± 6.52	-
Evaluation Time (s)	Small	22.91 ± 5.85	10.94 ± 0.99	18.95 ± 3.53
	Medium	70.12 ± 8.67	14.77 ± 1.42	22.30 ± 7.55
	Large	123.76 ± 32.32	18.84 ± 7.38	27.78 ± 16.52

HetGAT layers (the first two use concatenation, and the last one uses averaging). The feature dimension of hidden layers = 64, and the number of heads = 8. The Recurrent Propagator utilizes a LSTMCell of size 32 followed by a fully-connected layer and a softmax layer. We set $\gamma = 0.99$, batch size = 8 and used Adam optimizer [37] with a learning rate of 2×10^{-3}, and a weight decay of 5×10^{-4}. We employed a learning rate decay of 0.5 every 4000 epochs. We evaluate the models using a batch size of 8 and 16. For the Multi-Round Environment, the infeasible reward coefficient $C_i = 2.0$ and total round number = 4. Both training and evaluation were conducted on a Quadro RTX 8000 GPU.

D. Evaluation Results

Table I shows the evaluation performance with Deterministic Human Proficiency in different scales. The Deterministic Human Proficiency means that during training and evaluation, human learning curve is known and execution is deterministic for every agent. In Table I, "Small" and "Medium" term after model name denotes the data scale the model was trained on and the number following it denotes the batch size for schedule sampling. The results show that HybridNet outperforms both EDF and Genetic Algorithm in adjusted makespan and percentage of feasibility. HybridNet trained on Small scale problems generalizes for both Medium and Large scale problems with similar or slightly worse performance than HybridNet trained on Medium. HybridNet and HetGAT performs similarly on all scales. This shows that HybridNet is capable of learning high performance policies by leveraging the Recurrent Schedule Propagator and without requiring interaction with the Environment.

We provide the runtimes of training and evaluation for HetGAT and HybridNET in Table III. HybridNet is approximately 10 times faster in training compared to HetGAT Model and at least 2 times faster during evaluation for same batch size. EDF and Genetic Algorithm were evaluated through the CPU without GPU acceleration, making it infeasible to accurately compare the performance of the Deep Learning Models to the Traditional Models.

We show that for HybridNet, step-based training has better performance over the greedy baseline, while for HetGAT model, greedy baseline training is better. We also observed that greedy baseline training reached convergence faster than step-based training (4500 epochs vs. 19000 epochs). Further investigation is worthwhile.

Table II shows the evaluation performance with Stochastic Human Proficiency in different scales. The Stochastic Human Proficiency is presented as randomness in both the actual human execution within Multi-Round Environment and uncertainty within the Learning Curve Estimator used for schedule generation. The results show that HybridNet outperforms the EDF and Genetic Algorithm across different data scales. The largest performance gap was observed on large dataset (23.51% vs. 1.15%). Here, HetGAT model is not included as it requires interaction with the environment after every task-agent assignment to observe the outcome, which is not available until the whole schedule is generated and sent to the Stochastic Environment for execution to emulate real-world scenarios.

VI. CONCLUSIONS

We present a deep learning-based framework, called HybridNet, combining a heterogeneous graph-based encoder with a recurrent schedule propagator, for scheduling

stochastic human-robot teams under temporal constraints. The resulting policy network provides a computationally lightweight yet highly expressive model that is end-to-end trainable via reinforcement learning algorithms. We developed a multi-round task scheduling environment for stochastic human-robot teams and conducted extensive experiments, showing that HybridNet outperforms other human-robot scheduling solutions across problem sizes. Future research includes integrating the learning-based human estimator into HybridNet, transfer learning across optimizing different objective functions, and deploying the trained network in a real-world scenario.

REFERENCES

[1] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013. [Online]. Available: https://doi.org/10.5772/57313

[2] C. Heyer, "Human-robot interaction and future industrial robotics applications," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4749–4754.

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

[4] R. Liu, M. Natarajan, and M. Gombolay, "Human-robot team coordination with dynamic and latent human task proficiencies: Scheduling with learning curves," 2020.

[5] M. C. Gombolay, R. A. Gutierrez, S. G. Clarke, G. F. Sturla, and J. A. Shah, "Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams," *Autonomous Robots*, vol. 39, no. 3, pp. 293–312, 2015.

[6] E. Nunes and M. Gini, "Multi-robot auctions for allocation of tasks with temporal constraints," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2110–2116.

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

[8] M. M. Solomon, "On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints," *Networks*, vol. 16, no. 2, pp. 161–174, 1986.

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

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.

[12] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2019.

[13] T. Ma, P. Ferber, S. Huo, J. Chen, and M. Katz, "Online planner selection with graph neural networks and adaptive scheduling," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5077–5084, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5949

[14] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[15] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, Apr. 2017, publisher Copyright: © 2016 Elsevier B.V.

[16] 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.

[17] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, Dec. 1962. [Online]. Available: https://doi.org/10.1007/bf01386316

[18] H. Ren and L. Tang, "An improved hybrid milp/cp algorithm framework for the job-shop scheduling," *2009 IEEE International Conference on Automation and Logistics*, pp. 890–894, 2009.

[19] E. Castro and S. Petrovic, "Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem," *Journal of Scheduling*, vol. 15, no. 3, pp. 333–346, May 2011. [Online]. Available: https://doi.org/10.1007/s10951-011-0239-8

[20] J. Chen and R. Askin, "Project selection, scheduling and resource allocation with time dependent returns," *European Journal of Operational Research*, vol. 193, pp. 23–34, 02 2009.

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

[22] M. C. Gombolay, "Apprenticeship scheduling for human-robot teams," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 2497–2498.

[23] Z. Wang, C. Liu, and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," *Autonomous Robots*, vol. 46, no. 1, pp. 249–268, 2022.

[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *Trans. Neur. Netw.*, vol. 20, no. 1, p. 61–80, jan 2009. [Online]. Available: https://doi.org/10.1109/TNN.2008.2005605

[25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[26] J. Singh, "An algorithm to reduce the time complexity of earliest deadline first scheduling algorithm in real-time system," 2010.

[27] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[28] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *International conference on artificial neural networks*. Springer, 2005, pp. 799–804.

[29] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2021.

[30] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[31] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[32] A. Ycart, E. Benetos, *et al.*, "A study on lstm networks for polyphonic music sequence modelling." ISMIR, 2017.

[33] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[34] R. S. Sutton, S. Singh, and D. McAllester, "Comparing policy-gradient algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, 2000.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[36] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2020.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.