SADCHER: Scheduling using Attention-based Dynamic Coalitions of Heterogeneous Robots in Real-Time

Jakob Bichler[1], Andreu Matoses Gimenez[1], Javier Alonso-Mora[1]

***Abstract*— We present Sadcher, a real-time task assignment framework for heterogeneous multi-robot teams that incorporates dynamic coalition formation and task precedence constraints. Sadcher is trained through Imitation Learning and combines graph attention and transformers to predict assignment rewards between robots and tasks. Based on the predicted rewards, a relaxed bipartite matching step generates high-quality schedules with feasibility guarantees. We explicitly model robot and task positions, task durations, and robots' remaining processing times, enabling advanced temporal and spatial reasoning and generalization to environments with different spatiotemporal distributions compared to training. Trained on optimally solved small-scale instances, our method can scale to larger task sets and team sizes. Sadcher outperforms other learning-based and heuristic baselines on randomized, unseen problems for small and medium-sized teams with computation times suitable for real-time operation. We also explore sampling-based variants and evaluate scalability across robot and task counts. In addition, we release our dataset of 250,000 optimal schedules: `autonomousrobots.nl/paper_websites/sadcher_MRTA/`**

I. INTRODUCTION

Autonomous multi-robot systems (MRS) are designed for complex, real-world settings, including earthquake disaster response scenarios [1], autonomous construction [2], production assembly processes [3], or search and rescue missions [4]. Interest in MRS and multi-robot task assignment (MRTA) has grown rapidly in recent years [5]. Efficient MRTA algorithms optimize resource usage and minimize operational time [6]. MRS improve performance and enhance system robustness as a multi-robot team is more resilient against individual robot failures and performance bottlenecks [7], [8]. Using sub-teams of robots, i.e., dynamic coalition formation, enables teams to tackle complex tasks that would otherwise be infeasible for a single robot [8]–[10]. In practice, relying on a team of homogeneous robots where each robot possesses all skills can become impractical if task requirements are highly diverse, involving different sensors and actuators [11]. Instead, using heterogeneous robots brings practical and economic advantages, by leveraging existing specialized robots [12] and deploying simpler robots that are more cost-effective to implement and maintain [8] and more robust to failures [6]. MRS operate in dynamic environments where sudden changes, new tasks, unexpected task requirements, robot malfunctions, or moving obstacles can occur [5]. Hence, the ability to adaptively replan in real-time is essential. Modeling precedence constraints, which impose a logical temporal sequence among tasks, further enhances applicability to real-world scenarios [13] where some tasks depend on the completion of prior tasks.

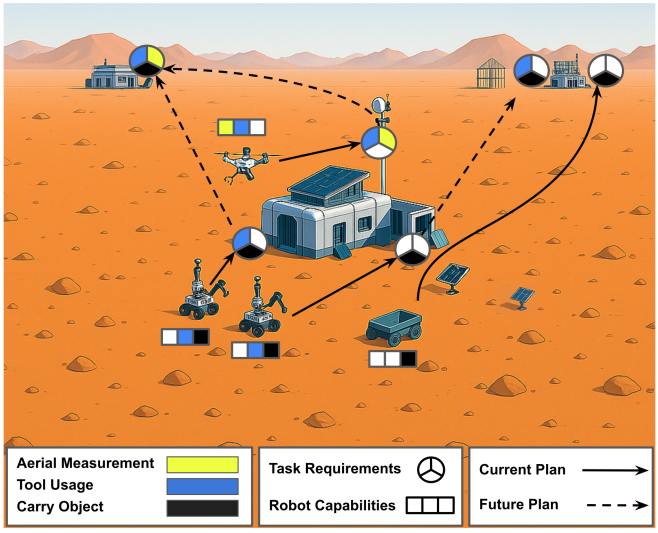

Fig. 1. Illustrative use case of autonomous construction on Mars. Circles represent tasks, color indicates required skills. Robot skills are shown as colored squares. Tasks requiring skills no single robot has (e.g., search for material in top left) must be executed by a synchronized coalition of robots.

Motivated by these challenges this paper proposes Sadcher, a framework for real-time scheduling of heterogeneous multi-robot teams with dynamic coalition formation and precedence constraints. Our main contributions are:

- A learning-based model, combining graph attention networks and transformers, which accounts for robot/task positions, task dependencies and durations, robots' capabilities, and robots' remaining time to complete the current task. This enables advanced spatiotemporal reasoning and generalization.
- A dataset of 250,000 optimally solved small-scale problems, usable as demonstrations for imitation learning or to benchmark against optimal schedules.

II. RELATED WORK

Following the taxonomy introduced in [14] and extended in [15], MRTA problems can be categorized on 4 axes: (1) single- or multi-task robots (ST/MT), (2) single- or multi-robot tasks (SR/MR), (3) instantaneous or time-extended assignment (IA/TA) where TA incorporates information about future tasks and scheduling decisions, (4) interdependence of agent-task utilities, i.e. with cross-dependencies (XD), an

*This paper has been accepted for publication at the 2025 IEEE Int. Symposium on Multi-Robot & Multi-Agent Systems (MRS).

[1]Authors are with the Department for Cognitive Robotics, ME, Delft University of Technology, Delft, Netherlands,

agent's utility for a task depends on the tasks assigned to other agents. This work addresses ST-MR-TA-XD settings.

A. Conventional Methods

Mixed Integer Linear Programming (MILP) offers exact solutions for complex ST-MR-TA-XD problems [16], though its exponential runtime hinders real-time use. The MILP-based CTAS framework [11] explicitly models risk in agent capability for task decomposition and scheduling. Simpler heterogeneous ST-SR scenarios can be addressed with the Tercio algorithm [3], which uses an MILP-based task allocator and a polynomial runtime task scheduler. Auction algorithms like [4], [17] treat tasks as non-atomic – tasks execution can be incremental, not requiring coalition formation. [18] uses auctions to solve heterogeneous ST-MR problems with atomic tasks. Genetic Algorithms offer anytime solutions that balance exploration and exploitation: [19] tackles heterogeneous ST-SR, [20] focusses on coalition formation of homogeneous robots, while [12] can handle heterogeneity and coalition formation. Other optimization metaheuristics applied to heterogeneous ST-MR include Ant Colony Optimization [10] and Particle Swarm Optimization [21]. Greedy formulations like [6] employ construction and improvement heuristics to balance runtime and performance.

B. Learning-based Methods

Deep learning methods promise fast solution generation and good scalability, by offloading most of the computation to the training phase [22]. Reinforcement Learning (RL) does not require a training dataset, but might spend a lot of time on infeasible solutions [23]. RL is used to solve ST-SR problems with mildly heterogeneous robots – robots differ in efficiency but can all perform any task – in [24] and [25]. Other RL methods solve ST-MR problems with dynamic coalition formation, but only for homogeneous robots [26], [27]. Recently, RL has been used to tackle heterogeneous ST-MR problems with dynamic coalition formation in [28]. The authors mitigate some of the problems RL faces through a flash-forward mechanism which allows for decision reordering to avoid deadlocks during training.

Instead of RL, other methods use Imitation Learning (IL) from optimal solutions during training, which requires a (computationally expensive) expert dataset, but benefits from stable training. [23] presents an IL method for mildly heterogeneous robots without coalition formation (ST-SR). Both [29] and [30] address heterogeneous ST-MR problems with a network predicting task assignment rewards and a bipartite matching algorithm yielding task assignments based on these rewards. In [29], coalition formation is only considered if a task fails to be completed by a single robot. There are cases in which a lower cost schedule could be obtained by considering coalition formation for all tasks, e.g., two robots are faster at completing the task than one. [30] improves upon this by always considering coalition formation. Furthermore, they introduce voluntary waiting, which increases performance through enabling better future coalition formation by delaying task assignments. However, [30] omits locations and durations in their network architecture. This implicitly assumes task durations and travel times to be negligible or to match the training distribution.

In this paper, we extend previous IL methods by explicitly modeling robot/task positions, task durations, and robots' remaining time to complete the current task. This enables advanced spatiotemporal reasoning, e.g., synchronizing robot arrivals and anticipating task readiness and robot availability. Additionally, it supports generalization to environments with unseen spatiotemporal distributions.

III. PROBLEM STATEMENT

Notation. Matrices are boldface uppercase (e.g. $\mathbf{M} \in \mathbb{R}^{\mathrm{n}\times\mathrm{m}}$), vectors are boldface lowercase (e.g. $\mathbf{v} \in \mathbb{R}^{\mathrm{d}}$), and scalars are lowercase (e.g. s).

We model a system of N heterogeneous robots, M tasks, and a set of skills $\mathcal{S}$. Each robot is capable of performing a subset of $\mathcal{S}$, and each task requires a subset of $\mathcal{S}$ to be performed at its location for the given task duration.

The N heterogeneous robots with $S_i \subseteq \mathcal{S}$ distinct skills, are modeled as an undirected graph $\mathcal{G}^r = (\mathcal{R}, \mathbf{C})$, where each vertex in $\mathcal{R} = \{\mathbf{r}_i\}^N$ is a robot with d_r dimensions. Robot states $\mathbf{r}_i = [\mathbf{p_i^r}, t_i^r, a_i^r, \mathbf{c_i^r}]$ include position $\mathbf{p_i^r} \in \mathbb{R}^2$, remaining duration at the current task t_i^r, the robot's availability $a_i^r \in \{0,1\}$, and the binary capability vector over the global skill set $\mathbf{c_i^r} \in \{0,1\}^{|\mathcal{S}|}$. $\mathbf{C} \in \{0,1\}^{N\times N}$ represents the network connection among the robots. For simplicity, we assume a fully connected graph, so $C_{i,j} = 1\ \forall i,j$, but the model is designed to accept any connected graph as input.

The M tasks and their respective precedence constraints are represented as a directed acyclic graph $\mathcal{G}^t = (\mathcal{T}, \mathbf{P})$. Each task is a vertex in $\mathcal{T} = \{\mathbf{t}_j\}^M$ with d_t dimensions, and is described by $\mathbf{t}_j = [\mathbf{p_j^t}, t_j^t, \mathbf{r_j^t}, s_j^t]$, with position $\mathbf{p_i^r} \in \mathbb{R}^2$, expected duration t_j^t, required skills $\mathbf{r_j^t} \in \{0,1\}^{|\mathcal{S}|}$ and status $\mathbf{s_j^t} \in \{0,1\}^3$. The status indicates whether tasks are ready, assigned, or incomplete, e.g., $\mathbf{s}_j^t = [1,0,1]$ represents a task that is ready to be scheduled, currently not assigned, and incomplete. Precedence constraints are encoded in the edges $\mathbf{P}^{M\times M}$, where $P_{i,j} = 1$ means the i-th task is a predecessor of the j-th task. The j-th task is only ready to be scheduled if all its preceding tasks have been completed. A task can only commence when all required skills are covered by the dynamically formed coalition of robots assigned to it. This can be denoted as $\mathbf{c_C} \succeq \mathbf{r_j^t}$ where $\mathbf{c_C}$ is the element-wise sum of robot capabilities $\mathbf{c_i^r}$ of assigned robots and $\succeq$ is the element-wise greater-or-equal operator. The tasks require tightly coupled coalitions [20] - all robots have to be present at the task location for the entire execution duration. Furthermore, we introduce an idle task t_{M+1} that robots can choose to increase overall performance by delaying assignments until a better coalition can be formed.

Robots start at location $\mathbf{p}_i^{\text{start}}$ and end at $\mathbf{p}_i^{\text{end}}$. The cost function aims to minimize the makespan, defined as the latest arrival time of any robot at $\mathbf{p}_i^{\text{end}}$ after completing its tasks:

$$\min \max_{i\in\{1,\dots,N\}} \left(t_i^{\text{finish}} + \tau\left(\mathbf{p}_i^{\text{finish}}, \mathbf{p}_i^{\text{end}}\right)\right) \tag{1}$$

where t_i^{finish} is the time robot i finishes its final task, which is computed as the sum of its execution times, idling times, and travel times. $\tau\left(\mathbf{p}_i^{\text{finish}}, \mathbf{p}_i^{\text{end}}\right)$ is the travel time from the location of the last finished task $\mathbf{p}_i^{\text{finish}}$ to the end location $\mathbf{p}_i^{\text{end}}$. Travel times can be estimated using Euclidean distance or path planning algorithms that take obstacles into account.

IV. METHOD

The Sadcher framework consists of a neural network based on attention mechanisms to predict assignment rewards for robots to tasks that is agnostic to the size of the input graphs, i.e., can handle arbitrary numbers of robots and tasks. A relaxed bipartite matching algorithm extracts task assignments based on the predicted reward. During runtime, the method asynchronously recomputes assignments at decision steps, i.e., when robots finish tasks or new tasks are announced.

A. Network Architecture

The high-level network structure is depicted in Fig. 2 and is similar to [30], but extended with a distance multilayer perceptron (MLP) that informs the network about relative distances between robots and tasks and separate heads for predicting rewards for "normal" tasks and the idle action.

The key components of the network are graph attention encoders (GAT) [31], transformer blocks [32], and reward MLPs that project latent embeddings into a reward matrix.

1) Graph Attention (GAT) Encoder Blocks: After mapping robot features $\mathbf{r}_i$ and task features $\mathbf{t}_j$ into d-dimensional embeddings, the embedded robot and task features are processed by separate GATs to capture local information-rich latent representations of the input graphs. GATs process a set of node features, incorporating information from neighboring nodes based on an adjacency matrix. While the robot features are processed as a fully connected graph, assuming all-to-all attention, the task GAT leverages the encoded precedence constraints in the adjacency matrix to understand the temporal task logic. A single head of the GAT computes attention weights $\alpha_{i,j}$ between node i and its neighbors j, based on a projected feature vector $\mathbf{h}' = \mathbf{h}\mathbf{W}^h$ (where $\mathbf{h}$ is the input node feature and $\mathbf{W}^h$ is a learned weight matrix):

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(a([\mathbf{h}'_i || \mathbf{h}'_j])))}{\sum_{k \in \mathcal{N}i \cup i} \exp(\text{LeakyReLU}(a([\mathbf{h}'_i || \mathbf{h}'_k])))} \tag{2}$$

here, a is a learnable linear transformation, $||$ denotes concatenation and $\mathcal{N}_i$ is the set of neighbors of node i. A Leaky ReLU [33] in combination with a softmax function over the neighbors of node i yields the final $\alpha_{i,j}$. The resulting $\alpha_{i,j}$ represent the relative importance of node j to node i, enabling context-aware feature propagation. Spatiotemporally related tasks or robots with complementary skills will attend more strongly to each other. The output $\mathbf{h}_i^{GAT}$ for a single head at node i is a sum of a self-loop contribution and the transformed neighbor contributions:

$$\mathbf{h}_i^{\text{GAT}} = \alpha_{i,i}\mathbf{h}'_i + \text{LeakyReLU}\left(\sum_{j \in \mathcal{N}_i, j \neq i} \alpha_{i,j}\mathbf{h}'_j\right) \tag{3}$$

In the GAT encoder blocks, we apply multi-head GAT, concatenating the outputs of Z_{GAT} independent heads and applying residual connections and layer normalization. The GAT encoder consist of L_{GAT} such layers and outputs $\mathbf{h}^{\text{GAT\_R}}$ for robots and $\mathbf{h}^{\text{GAT\_T}}$ for tasks respectively.

2) Transformer Encoder Blocks: Following the GAT encoders, the representations $\mathbf{h}^{\text{GAT\_R}}$ and $\mathbf{h}^{\text{GAT\_T}}$ are processed by independent transformer encoders, to build the global context of robots and tasks. Each transformer block applies multi-head self-attention (MHA) [32] on the input $\mathbf{h}$:

$$\alpha_z = \text{Softmax}\left(\frac{(\mathbf{W}_z^Q\mathbf{h})(\mathbf{W}_z^K\mathbf{h})^\top}{\sqrt{d}}\right)(\mathbf{W}_z^V\mathbf{h}) \tag{4}$$

$$\text{MHA}(\mathbf{h}) = \big(\alpha_1||\alpha_2||\ldots||\alpha_Z\big)\mathbf{W}^O \tag{5}$$

where d is the key dimensionality. MHA computes this operation in parallel for Z_T heads to generate the final outputs $\mathbf{h}^{\text{T\_R}}$ for robots and $\mathbf{h}^{\text{T\_T}}$ for tasks respectively.

3) Reward Prediction: The normalized relative distances between robot i and task j are passed through the distance head MLP_D to compute the distance feature $d_{i,j}$:

$$d_{i,j} = \text{MLP}_\text{D}\left(\text{Normalize}(\|\mathbf{p}_i^R - \mathbf{p}_j^T\|_2)\right) \tag{6}$$

While task and robot positions are part of the raw input features in $\mathcal{G}^r$ and $\mathcal{G}^t$, this explicit distance term provides the network with direct access to spatial proximity.

We construct feature vectors $\mathbf{f}_{i,j}$ for each robot-task pair by concatenating the local (GAT) and global (transformer) representation of robot i and task j with the distance term $d_{i,j}$, so $\mathbf{f}_{i,j} \in \mathbb{R}^{4\times \text{d}_\text{k}+1}$:

$$\mathbf{f}_{i,j} = \mathbf{h}_i^{\text{GAT\_R}} \parallel \mathbf{h}_j^{\text{GAT\_T}} \parallel \mathbf{h}_i^{\text{T\_R}} \parallel \mathbf{h}_j^{\text{T\_T}} \parallel d_{i,j} \tag{7}$$

This information-rich representation is then passed through the reward head MLP_R to compute the task assignment reward $R_{i,j}$ to assign robot i to task j. The idle reward R_i^{IDLE} is computed by passing $\mathbf{f}_{i,j}$ to the idle head MLP_I and summing the outputs across all tasks for each robot i:

$$R_{i,j}^{\text{task}} = \text{MLP}_\text{R}\left(f_{i,j}\right), \qquad R_i^{\text{IDLE}} = \sum_{j=1}^{M} \text{MLP}_\text{I}(f_{i,j}) \tag{8}$$

MLP_I can be understood as learning per-task signals that encourage a robot to wait when short-term idling is advantageous (e.g., a nearby task will become ready soon). The final predicted reward $\mathbf{R}$ contains the task rewards $R_{i,j}^{\text{task}}$ for all pairs of robots i and tasks j, concatenated with the idle rewards R_i^{IDLE} for each robot i, so $\mathbf{R} \in \mathbb{R}^{\text{N}\times(\text{M}+1)}$.

B. Task Assignment through Bipartite Matching

The final reward $\mathbf{R}$ can be interpreted as the edge rewards between robots $\mathcal{R}$ and tasks $\mathcal{T}$ at a given timestep, encoding the full complexity of the current problem state. To extract task assignments at this timestep, we employ a relaxed bipartite matching formulation (no strict one-to-one matching). The constraints ensure valid assignments: (10) prevents robots from being assigned more than one task, (11) guarantees that each task's required skills are fully

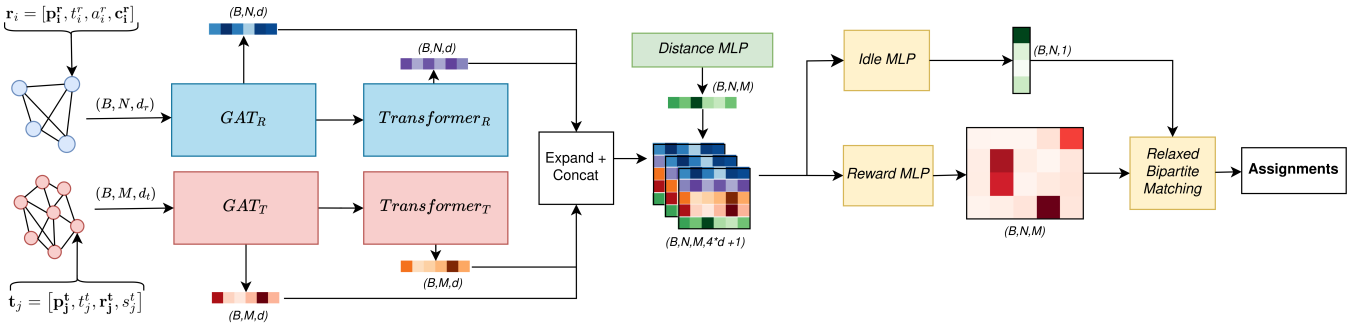

Fig. 2. Sadcher architecture overview. Robot and task graphs are processed by graph attention and transformer encoders and concatenated with distance features. The reward matrix is estimated by the Idle and Reward MLPs and final task assignments are extracted using relaxed bipartite matching. B: batch size, N: number of robots, M: number of tasks, d_r: robot input dimension, d_t: task input dimension, d: latent dimension.

covered by the assigned coalition using the element-wise inequality $\succeq$, and (12) enforces that only idle robots and ready tasks are matched. The bipartite matching finds the optimal assignment matrix $\mathbf{A}^* \in \mathbb{R}^{\mathrm{N}\times(\mathrm{M}+1)}$ that maximizes the selected edge reward encoded in $\mathbf{R}$:

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \sum_{i,j} A_{i,j} R_{i,j} \tag{9}$$

subject to:

$$\sum_{j=0}^{M+1} A_{i,j} \leq 1, \qquad \forall i \in \mathcal{R} \tag{10}$$

$$\sum_{i=0}^{N} A_{i,j}\, \mathbf{c}_i^r \succeq \mathbf{c}_j^t, \qquad \forall j \in M \tag{11}$$

$$A_{i,j} = 0, \qquad \forall i,j : i \notin \mathcal{R}_{\text{idle}} \vee j \notin \mathcal{T}_{\text{ready}} \tag{12}$$

This formulation prevents deadlocks since no coalition can be assigned to a task that it cannot execute. However, it allows for redundant assignments, so after computing $\mathbf{A}^*$, we remove robots that do not contribute unique required skills, starting from the robot with the highest travel time to the task. Additionally, we implement a pre-moving strategy: If robot r_i is assigned the idle task t_{M+1}, it moves towards the task with the highest reward $t^i_{\text{highest}} = \arg\max_{1\leq j\leq M} R_{i,j}$, without being formally assigned to it. This does not concatenate the tasks into a fixed schedule for the robot, since assignments are recomputed at decision steps. The robot is likely to be assigned to t^i_{highest} at the next decision step, so pre-moving can reduce the delay to task start if r_i would have been the last coalition member to arrive at t^i_{highest}.

C. Training through Imitation Learning

1) Training Data Generation: We generate 250,000 small-scale problem instances (8 tasks, 3 robots, 3 skills) with fully randomized configurations: each robot is assigned 1–3 skills, each task requires 1–3 skills, task locations and robot start/end depot lie in $[0, 100] \times [0, 100] \subset \mathbb{R}^2$ and are randomly sampled. Execution times are drawn uniformly from $[50, 100]$, precedence constraints are acyclic and generated between random task pairs, and robot travel speed is assumed to be 1 unit per timestep. To solve these scenarios optimally, we extend the exact MILP formulation of [16] with precedence constraints. Due to its exponential time complexity, both in the number of robots and tasks, only small instances can be solved in a reasonable time to generate a training dataset. We omit modelling of stochastic travel times, as our framework handles deviations via real-time replanning and does not need conservative safety margins.

2) Optimal Reward Extraction: To train the network to imitate the optimal behavior, we extract "ground-truth" reward matrices $\mathbf{O_k} \in \mathbb{R}^{\mathrm{N}\times(\mathrm{M}+1)}$. The optimal schedules are sliced into K decision points T_k^{dec}, corresponding to timesteps when a task finishes and the robots require re-assignment. At each T_k^{dec} the optimal reward is calculated based on the time difference between T_k^{dec} and the finish time of task j with discount factor $\gamma \in (0, 1]$:

$$O_{k,i,j} = \gamma^{\left(T_j^{\text{finish}} - T_k^{\text{dec}}\right)} o_{k,i,j} \tag{13}$$

where $o_{k,i,j} = 1$ if robot i is assigned to task j in the optimal solution and the decision point occurs before the task's start time ($T_k^{\text{dec}} < T_{i,j,\text{start}}$); otherwise, $o_{k,i,j} = 0$.

We handle the idle action t_{M+1} in the same way: If the time between a robot's last finish time and its next start time exceeds the travel time between the corresponding tasks, we treat this interval as an explicit idle assignment in the optimal schedule and compute its reward using the above formulas.

By design, this reward encoding captures the optimal decision logic: The next selected task will have the highest reward, with decreasing rewards for later tasks. Given the sequence of optimal rewards $\mathbf{O_k}$ over all decision steps T_k^{dec}, the bipartite matching algorithm outputs the exact solution.

3) Training Details : We modify the loss $\mathcal{L}$ from [30] by applying the inverse mask $(1 - \mathbf{X_k})$ to the second term:

$$\mathcal{L} = \|\mathbf{X_k} \circ (\mathbf{R_k} - \mathbf{O_k})\|_1 + \lambda \|(1 - \mathbf{X_k}) \circ (\mathbf{R_k} - \mathbf{O_k})\|_1 \tag{14}$$

where $\circ$ denotes the element-wise product operator, $\mathbf{O_k}$ is the optimal reward, $\mathbf{R_k}$ is the predicted reward and $\mathbf{X_k} \in \mathbb{R}^{\mathrm{N}\times(\mathrm{M}+1)}$ is a feasibility mask with $X_{i,j} = 1$ if robot i is available and task j is ready, else $X_{i,j} = 0$. The first term encourages accurate prediction of feasible rewards, while the second discourages high values for infeasible ones. λ balances the two terms: Intuitively, accurate feasible

predictions are more important than suppressing infeasible ones, as the bipartite matching will select high reward tasks. We use the ADAM optimizer [34] to train the network.

V. EXPERIMENTS AND RESULTS

We evaluate makespan and computation time metrics for six algorithms, averaged over 500 unseen problem instances with randomized task locations/durations, skill requirements, robot capabilities, and precedence constraints. Experiments are conducted on a consumer machine with an AMD Ryzen 7 4800H CPU and NVIDIA GeForce GTX 1650 GPU.

A. Compared Algorithms

1) Baselines: There exist few algorithms in literature that address heterogeneous ST-MR-TA-XD with precedence constraints, often without weights or datasets [30]. We chose one available algorithm for each of the approaches commonly followed (optimization, learning, heuristic search):
(1) An MILP formulation based on [16], adding precedence constraints and omitting stochastic travel times, which provides optimal solutions with formal guarantees. A decentralized RL framework HeteroMRTA [28], adapted for precedence constraints by masking out tasks that have incomplete predecessors during action selection. We compare against (2), the single solution variant HeteroMRTA, where agents choose the highest-probability task at decision steps, and (3), the sampling variant S-HeteroMRTA (Boltzmann weighted-random action selection) which returns the best makespan solution across 10 runs per instance. We also implement and compare (4), a greedy heuristic that assigns robots to tasks based on reducing the remaining skill requirements the most and breaking ties based on travel time (shortest first).

2) Sadcher Variants: (5) The Sadcher framework predicts robot-task rewards deterministically as described in Section IV. We also benchmark (6), a S-Sadcher variant, which samples reward matrices from a normal distribution centered around the deterministic output then used by the bipartite matching. This introduces stochastic variations in the schedules. As for S-HeteroMRTA we run this process 10 times per instance and select the best-performing rollout.

B. Training-Domain Evaluation

We evaluate the algorithms on 500 randomized problem instances of the training domain size (8 tasks, 3 robots, 3 precedence constraints). Results are shown in Fig. 3 and 4.

1) Makespan: The MILP formulation provides optimal makespans, establishing a baseline for comparing the average relative gaps of other methods. S-Sadcher (gap: 3.8%) and Sadcher (gap: 6.8%) are the best-performing non-optimal algorithms. HeteroMRTA performs worst (gap: 21.5%), but sampling reduces the optimality gap to 10.8%, leveraging its RL policy, which follows a sampling strategy during training. In the pairwise comparison in Fig. 4, Sadcher achieves a lower makespan for 403 of 500 instances (80.6%, binomial test: $p \approx 2 \times 10^{-45}$). S-Sadcher outperforms S-HeteroMRTA on 389 of 500 instances (77.8%, binomial test: $p \approx 3 \times 10^{-37}$). Greedy reaches an average gap of 20.4%.

2) Computation Time: For dynamic scenarios with real-time requirements, the time per assignment decision (t_{dec}) is crucial. MILP cannot compute instantaneous assignments, but only globally optimal schedules. S-HeteroMRTA and S-Sadcher roll out the full scenario to select the best assignments. Therefore, these three algorithms, do not yield a time per decision, but only for full solution construction (t_{full}). Due to its simplicity, the greedy algorithm computes the fastest (t_{dec}: 0.080 ms; t_{full}: 1.7 ms). HeteroMRTA (t_{dec}: 9.1 ms; t_{full}: 0.20 s) is faster than Sadcher (t_{dec}: 22 ms; t_{full}: 0.57 s), which needs to solve the relatively expensive bipartite matching for each decision. S-HeteroMRTA computes full solutions in 0.96 s, S-Sadcher in 5.7 s, and MILP in 76 s. In the worst case, MILP takes up to 12 minutes, rendering it infeasible for real-time applications, even on small problems.

3) Precedence Constraints: The Sadcher model demonstrates an understanding of task dependencies by prioritizing the assignment of predecessor tasks. This improves performance by unlocking successors earlier and enabling better global schedules. On average, the model assigns ready predecessor tasks approximately 1.7 times more frequently compared to the baselines. (S-) HeteroMRTA and Greedy cannot make this informed decision, but selecting tasks with incomplete predecessors is prevented through masking.

C. Out-of-Domain Generalization

To evaluate generalization, we scale the number of robots $N \in \{3, 5, 20\}$, and the task number $M \in [6, 250]$ (see Fig. 5) and compare the makespan gap relative to Sadcher (trained on $N{=}3$, $M{=}8$). With a 1-hour cutoff, the MILP solver fails to find solutions beyond 10 tasks and 7 robots within this limit. For smaller problem sizes, it finds the optimal makespans, outperforming Sadcher by 6-16%.

For $N{=}3$ robots, S-Sadcher and Sadcher are the strongest non-optimal methods across all M. S-HeteroMRTA outperforms Greedy for $M{\leq}60$ and HeteroMRTA finds the highest makespans overall. Although Sadcher's relative performance is best for small M, it outperforms Greedy by more than 4% and (S-) HeteroMRTA by more than 9% for $M{=}250$.

For $N{=}5$ robots, S-Sadcher remains the best learning-based method across all M. S-HeteroMRTA performs better than Sadcher for $M{\leq}9$, but is surpassed by Sadcher beyond that, and by Greedy for $M{\geq}70$. HeteroMRTA outperforms Greedy for $7{\leq}M{\leq}10$. Greedy reaches a 2% gap for $M{=}200$.

For $N{=}20$ robots, relative performance changes significantly, with degradation of the learning-based methods: S-Sadcher is the best-performing method only for $M{\leq}70$, beyond that, Greedy becomes superior. S-HeteroMRTA consistently beats Sadcher, yet only outperforms Greedy for $M{\leq}50$. HeteroMRTA surpasses Sadcher for $M{\geq}150$, and the performance gap for $M{\leq}100$ between the two is smaller compared to scenarios with fewer robots.

Overall, Sadcher excels on smaller robot teams across all task counts, but its performance decreases with more robots. We hypothesize that increasing the number of tasks while keeping the number of robots fixed is similar to solving multiple smaller subproblems sequentially, where local

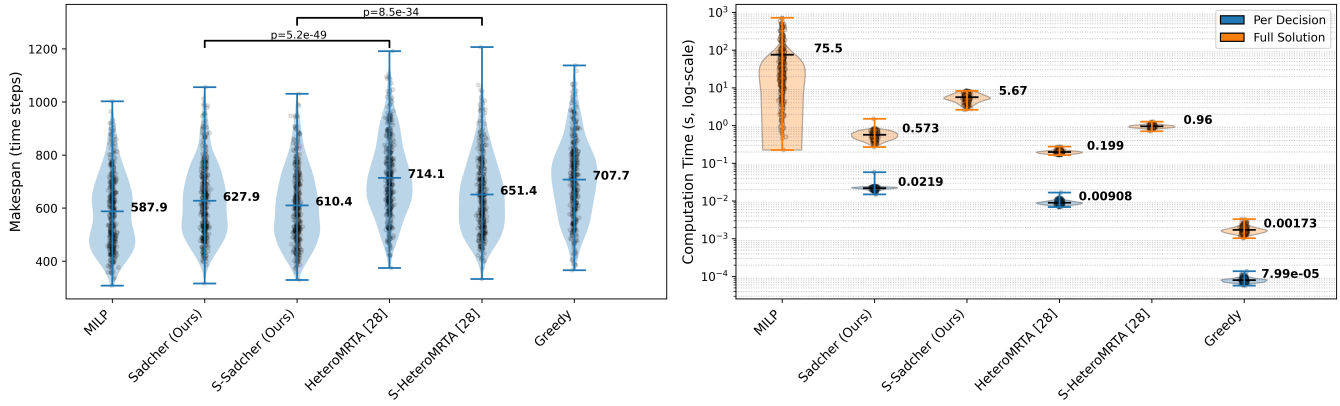

Fig. 3. Comparison on 500 unseen, randomized problem instances (8 tasks, 3 robots, 3 precedence constraints) for makespan (left), and computation time (right). Lower means better performance. Wilcoxon significance levels are annotated for Sadcher compared to HeteroMRTA variants. All other pairwise differences are statistically significant ($p < 0.05$), except between S-HeteroMRTA and Greedy ($p = 0.21$). For algorithms requiring full solution construction, total computation time is reported; for methods returning instantaneous assignments, both time per decision and total time are shown.

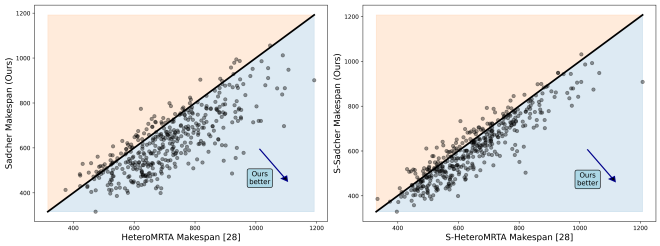

Fig. 4. Pairwise makespan comparison of HeteroMRTA vs. Sadcher (left) and S-HeteroMRTA vs. S-Sadcher (right). Each point is one solved instance; points below the diagonal indicate better performance by (S-)Sadcher.

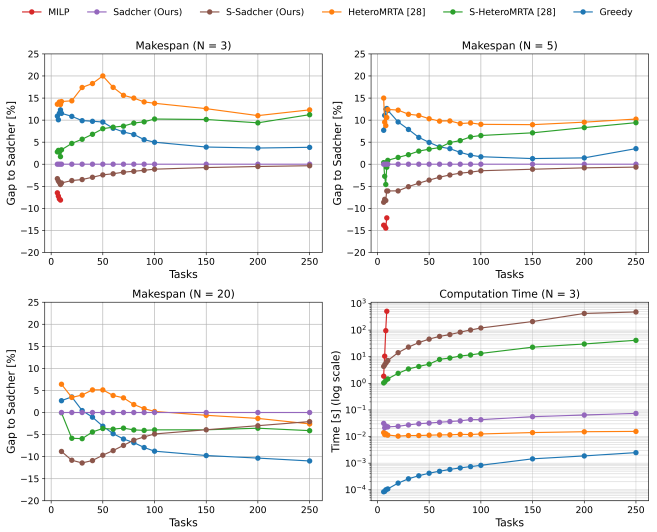

Fig. 5. Relative makespan gap to Sadcher for 3, 5, and 20 robots (top left to bottom left). Bottom right: Computation time for 3 robots (for algorithms requiring full solution construction (Sample-HeteroMRTA, Sample-Sadcher, MILP), total computation time is reported, for methods returning instantaneous assignments (HeteroMRTA, Sadcher, Greedy), time per decision is reported). Task counts $M \in [6, 250]$, with $S = 3$ skills and $M/5$ precedence constraints. Each point shows the mean over 100 runs.

scheduling rules learned during training remain effective. On the other hand, larger teams require different local scheduling strategies that diverge from the distribution Sadcher has seen during training. For high task counts, the greedy algorithm - especially in combination with bigger robot teams - starts beating the learning-based methods. Sampling-based variants (S-Sadcher, S-HeteroMRTA) have a higher impact on smaller problems, where the smaller solution space makes rollouts more likely to yield improvements.

Greedy computes fastest, delivering near-instantaneous decisions ($\leq$3 ms). The computation time of HeteroMRTA is minimally affected by scaling ($\leq$20 ms), while Sadcher is slower and scales worse due to the bipartite matching step ($\leq$ 80 ms per decision). The sampling-based variants require significantly longer computation (up to 450 s for S-Sadcher and 40 s for S-HeteroMRTA for M=250), which makes them impractical for online computation on large problems.

VI. CONCLUSION

In this work, we proposed Sadcher - an IL framework to address real-time task assignment for heterogeneous multi-robot teams, incorporating dynamic coalition formation and precedence constraints. Reward prediction with relaxed bipartite matching yields strong performance with feasibility guarantees. Sadcher outperforms RL-based and heuristic baselines in makespan across small to medium-sized robot teams and a wide range of task counts. For bigger teams, the advantage is lost due to lack of demonstrations. Sadcher can generate assignments in real-time across all tested problem sizes, but the sampling variant S-Sadcher is only real-time for smaller problems. Sadcher relies on a large (computationally expensive) dataset of expert demonstrations for training.

Future work will explore fine-tuning IL policies with RL, which could increase performance on larger problem instances where expert solutions are very expensive or infeasible to obtain. Additionally, extending the dataset with sub-optimal demonstrations for bigger problem instances could improve scalability.

ACKNOWLEDGMENTS

This project has received funding from the European Union through ERC, INTERACT, under Grant 101041863. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] L. Zhang, M. Li, W. Yang, and S. Yang, "Task Allocation in Heterogeneous Multi-Robot Systems Based on Preference-Driven Hedonic Game," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 8967–8972.

[2] W. Gosrich, S. Mayya, S. Narayan, M. Malencia, S. Agarwal, and V. Kumar, "Multi-Robot Coordination and Cooperation with Task Precedence Relationships," May 2023.

[3] 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, pp. 220–239, Feb. 2018.

[4] I. Ansari, A. Mohammed, Y. Ansari, M. Yusuf Ansari, S. Razak, and E. Feo Flushing, "CoLoSSI: Multi-Robot Task Allocation in Spatially-Distributed and Communication Restricted Environments," *IEEE Access*, vol. 12, 2024.

[5] H. Chakraa, F. Guérin, E. Leclercq, and D. Lefebvre, "Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art," *Robotics and Autonomous Systems*, vol. 168, p. 104492, Oct. 2023.

[6] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann, "Multi-Robot Task Allocation and Scheduling Considering Cooperative Tasks and Precedence Constraints," May 2020.

[7] R. K. Ramachandran, J. A. Preiss, and G. S. Sukhatme, "Resilience by Reconfiguration: Exploiting Heterogeneity in Robot Teams," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019.

[8] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Cooperative Robots and Sensor Networks 2015*, A. Koubâa and J. Martínez-de Dios, Eds. Cham: Springer International Publishing, 2015, pp. 31–51.

[9] F. Quinton, C. Grand, and C. Lesire, "Market Approaches to the Multi-Robot Task Allocation Problem: a Survey," *Journal of Intelligent & Robotic Systems*, vol. 107, no. 2, p. 29, Feb. 2023.

[10] W. Babincsak, A. Aswale, and C. Pinciroli, "Ant Colony Optimization for Heterogeneous Coalition Formation and Scheduling with Multi-Skilled Robots," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Dec. 2023, pp. 121–127.

[11] B. Fu, W. Smith, D. Rizzo, M. Castanier, M. Ghaffari, and K. Barton, "Robust Task Scheduling for Heterogeneous Robot Teams under Capability Uncertainty," *IEEE Transactions on Robotics*, June 2021.

[12] P. Muhuri and A. Rauniyar, "Immigrants Based Adaptive Genetic Algorithms for Task Allocation in Multi-Robot Systems," *International Journal of Computational Intelligence and Applications*, vol. 16, p. 1750025, Dec. 2017.

[13] M. Gini, "Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017.

[14] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, Sept. 2004.

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

[16] A. Aswale and C. Pinciroli, "Heterogeneous Coalition Formation and Scheduling with Multi-Skilled Robots," June 2023.

[17] I. Ansari, A. Mohamed, E. F. Flushing, and S. Razak, "Cooperative and load-balancing auctions for heterogeneous multi-robot teams dealing with spatial and non-atomic tasks," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2020.

[18] M. Irfan and A. Farooq, "Auction-based task allocation scheme for dynamic coalition formations in limited robotic swarms with heterogeneous capabilities," in *2016 International Conference on Intelligent Systems Engineering (ICISE)*, Jan. 2016, pp. 210–215.

[19] H. Chakraa, E. Leclercq, F. Guérin, and D. Lefebvre, "A Centralized Task Allocation Algorithm for a Multi-Robot Inspection Mission With Sensing Specifications," *IEEE Access*, vol. 11, 2023.

[20] M. U. Arif, "Robot coalition formation against time-extended multi-robot tasks," *International Journal of Intelligent Unmanned Systems*, vol. 10, pp. 468–481, June 2021.

[21] X.-F. Liu, Y. Fang, Z.-H. Zhan, and J. Zhang, "Strength Learning Particle Swarm Optimization for Multiobjective Multirobot Task Scheduling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 7, pp. 4052–4063, July 2023.

[22] A. Prorok, J. Blumenkamp, Q. Li, R. Kortvelesy, Z. Liu, and E. Stump, "The Holy Grail of Multi-Robot Planning: Learning to Generate Online-Scalable Solutions from Offline-Optimal Experts," July 2021.

[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, Jan. 2022.

[24] S. Paul, P. Ghassemi, and S. Chowdhury, "Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks," May 2022.

[25] B. Altundas, Z. Wang, J. Bishop, and M. Gombolay, "Learning Coordination Policies over Heterogeneous Graphs for Human-Robot Teams via Recurrent Neural Schedule Propagation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022.

[26] F. Deng, H. Huang, L. Fu, H. Yue, J. Zhang, Z. Wu, and T. L. Lam, "A Learning Approach to Multi-robot Task Allocation with Priority Constraints and Uncertainty," in *2022 IEEE International Conference on Industrial Technology (ICIT)*, Aug. 2022, pp. 1–8.

[27] W. Dai, A. Bidwai, and G. Sartoretti, "Dynamic Coalition Formation and Routing for Multirobot Task Allocation via Reinforcement Learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan: IEEE, May 2024, pp. 16 567–16 573.

[28] W. Dai, U. Rai, J. Chiun, Y. Cao, and G. Sartoretti, "Heterogeneous Multi-robot Task Allocation and Scheduling via Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2654–2661, Mar. 2025.

[29] P. Gao, S. Siva, A. Micciche, and H. Zhang, "Collaborative Scheduling with Adaptation to Failure for Heterogeneous Robot Teams," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 1414–1420.

[30] W. J. Jose and H. Zhang, "Learning for Dynamic Subteaming and Voluntary Waiting in Heterogeneous Multi-Robot Collaborative Scheduling," *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," Feb. 2018.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," June 2017.

[33] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," Nov. 2015.

[34] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017.