2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Teamformer: Scalable Heterogeneous Multi-Robot Team Formation

Noah Boehme and Geoffrey Hollinger

***Abstract*— Accounting for heterogeneity among robots and tasks adds additional complexity to multi-robot task allocation. While existing task allocation methods effectively handle heterogeneity among robots and tasks, they do not scale well in the number of different robots and tasks. To address this gap, we formulate the Team Formation Markov Decision Process (TF-MDP) for training *Teamformer*: a scalable, decentralized transformer policy for dynamically forming heterogeneous teams of robots to complete diverse tasks. Combining the TF-MDP with the autoregressive capability of transformers enables *Teamformer* to scale linearly in the number of robots, tasks, and combinations of different heterogeneous robots. Simulations demonstrate *Teamformer* generalizing to combinations of 100 different types of robots and tasks. Hardware experiments using Georgia Tech's Robotarium show *Teamformer* decentrally coordinating up to 20 heterogeneous robots for task completion.**

I. Introduction

Many real-world problems, such as warehouse management [1] or environmental monitoring [2], benefit from, or even necessitate, coordinating heterogeneous Multi-Robot Systems (MRS). Incorporating heterogeneity in MRS allows robots to specialize towards certain tasks, enabling increased performance by leveraging diversity among robots [3], and empowers different robots to coordinate to complete tasks requiring diverse skills, like assembly tasks [4]. While coordinating heterogeneous MRS is beneficial, effectively solving heterogeneous Multi-Robot Task Allocation (MRTA) requires robots to reason about tasks they can complete with their own capabilities, as well tasks they can complete by coordinating with different robots.

To solve heterogeneous MRTA, prior works commonly use classical optimization methods or learning-based approaches. Classical optimization methods constrain robots to meet task requirements, but are limited in their ability to scale to many heterogeneous robots and tasks [5]–[7]. Learning-based approaches have demonstrated better scalability in the number of robots and tasks, but still do not address how to scale in the number of distinct robots and tasks [8]–[10].

To address the challenge of scaling in the number of robots, tasks, and distinct heterogeneous classes of both, we introduce the *Teamformer* framework. The *Teamformer* framework relies on solving a Team Formation Markov Decision Process (TF-MDP) to dynamically form heterogeneous teams and then performs task allocation and motion planning for each team (Figure 1). The TF-MDP is a decentralized MDP with a state space that 1) linearly scales in the number of robots, tasks, and different robot capabilities and 2) captures the heterogeneity among robots and tasks to produce high-level team formation actions. To solve the TF-MDP we use Deep Reinforcement Learning (DRL) and the *Teamformer* policy: a transformer-based policy. The *Teamformer* policy naturally handles the changing state space size of the TF-MDP, since transformers handle variable sequence length inputs [11], [12], and operates decentrally on each robot. Additionally, by using the autoregressive capability of transformers, the TF-MDP action space grows linearly in the number of distinct robot and task classes, instead of exponentially. This allows *Teamformer* to not only scale in the number of robots and tasks, but also the number of distinct heterogeneous classes.

Our main contributions are: 1) the TF-MDP that accounts for robot and task heterogeneity in the state and action space, 2) the *Teamformer* framework that scales in the number of heterogeneous robots and tasks, and 3) experiments demonstrating *Teamformer*'s ability to generalize to combinations of up to 100 different robots and tasks and real-world experiments showing sim-to-real transfer using *Teamformer* on up to 20 heterogeneous robots in the Georgia Tech Robotarium [13].

II. Related Works

Heterogeneous MRTA: Existing heterogeneous MRTA methods frequently use classical optimization techniques (e.g., MILP, QP). To model heterogeneity among robots and tasks, prior work commonly use binary vectors, to indicate the presence or absence of a robot capability or task requirement [5]–[7]. Treating the binary vectors as constraints in an optimization problem ensures robots are assigned to relevant tasks. Both [5] and [6] consider centralized MRTA, limiting their ability to scale, and only demonstrate their methods operating on two and three, respectively, different types of robots. For decentralized MRTA, [7] simulate on up to 500 robots completing 10 tasks, but only consider three different types of robots. For a more fine-grained representation of heterogeneity among robots and tasks, [14] extend prior work by using continuous and binary vectors. However, their method is only demonstrated on five different types of robots.

In recent years, learning-based methods have been used to account for heterogeneity in MRTA [15], [16]. To learn heterogeneous MRTA, the heterogeneity among robots and tasks can be included as a feature to a neural network. This technique has been used for heterogeneous MRTA among aerial and ground vehicles [9], robots with four different sensing modalities [10], and robots that complete tasks at variable speeds, with 10 or seven different speed levels, respectively [8], [17]. By inputting heterogeneity to the neural network, the network is able to learn what tasks should

*This work is funded in part by ONR award N00014-23-1-2171

*Noah Boehme and Geoffrey Hollinger are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR 97331, USA. {boehmen, geoff.hollinger}@oregonstate.edu

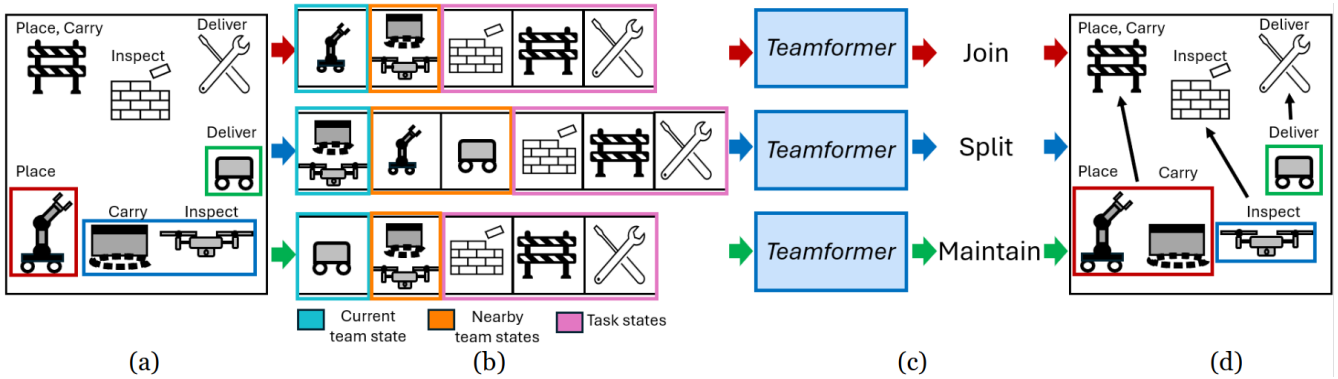


Fig. 1: Overview of MRTA using the *Teamformer* framework. a) Each robot is in a team, indicated by a colored box. b) Each team has an overall state, containing the current team state, locally sensed team states, and task states. c) Using each team's overall state, the *Teamformer* policy decentrally forms teams to meet task requirements. d) The *Teamformer* framework allocates tasks and motion plans. Best seen in color.

be assigned to a given robot based on its heterogeneity. However, similar to non-learning methods, these prior works only consider a limited number of distinct robots and tasks.

Scalable MRTA: While some classical optimization methods are capable of scaling to many robots and tasks [7], [18], [19], in general, the exponential increase in planning space for MRTA renders them computationally infeasible. Neural networks have been shown to generalize well to unseen data, allowing them to be trained in small scenarios and executed in large ones [17], [20]–[26] and can learn cooperative behavior by employing the centralized training, decentralized execution (CTDE) paradigm [20]. Recent work using graph neural networks (GNNs) [24], [27] or sequence models [17], [20], [21] for MRTA have shown the ability to execute in scenarios more than 100× larger than trained on. For example, [20] design a single deep set network [28] for coordinated target tracking. They train on four agents tracking four targets and decentrally execute on 1000 agents tracking a 1000 targets. However, to scale they use a masking heuristic on targets such that only the k closest targets are included in the state. Similar to our method, [21] uses an attention-based sequence model for MRTA and A* as a motion planner. In a simulated warehouse setting, their policy executes on 1000 agents performing 1000 tasks. However, these approaches assume no heterogeneity among robots and lack the ability to handle the combinatorial nature of heterogeneous MRTA.

III. Problem Formulation

Consider N robots that must decentrally cooperate to complete T tasks. Each robot has capabilities (e.g., sensors) taken from the set of capabilities, C, and can communicate with other robots within a sensing radius, r_s. Each task in T has a set of required capabilities, referred to as requirements. A given task can have 1 to $|C|$ requirements. To complete tasks, robots must use local communication to consider the possible combinations of capabilities, form teams whose capabilities meet task requirements, and move to completable tasks. The objective of the N robots is to complete all tasks in the shortest amount of time.

To achieve this objective, we formulate the TF-MDP. The TF-MDP is represented by the tuple $(G, s_i, a_i, P, R, \gamma)$, where G is the set of heterogeneous teams ($G \subseteq N$), s_i is the state for the i-th team, a_i is the action of the i-th team, P is the state transition probability distribution, R is the reward function, and γ is the discount factor. Unlike a standard multi-robot MDP, that uses each robot's state to produce an action, the TF-MDP uses a team state to produce a team-level action. Solving the TF-MDP results in a decentralized policy, $\pi_G(a_i|s_i)$, that each team uses.

States: Each team in G considers two states: 1) local team state and 2) task state. The local team state consists of the location and capabilities of all teams within r_s, including the current team. The team state for the i-th team is defined as: $p_i = [x_j, y_j, \mathbf{c}_j], \forall j \in L$, where x_j and y_j are the location of the j-th nearby team, $\mathbf{c}_j$ is a binary vector of the present capabilities of the j-th nearby team, and L is the set of teams within r_s. The task state for each team is made up of the locations of all tasks and the requirements for all tasks. The task state for the i-th team is defined as: $t_i = [x_k, y_k, \mathbf{q}_k], \forall k \in T$, where x_k and y_k are the location of the k-th task and $\mathbf{q}_k$ is a binary vector of the requirements for the k-th task. The overall state of the i-th team is then a concatenation of p_i and t_i: $s_i = [p_i, t_i]$. This state representation allows each team to reason about what combinations of capabilities *can* be created and what combinations of capabilities *need* to be created. Importantly, the state space scales linearly in teams and tasks.

Actions: To form teams, the action space includes a set of join, split, and maintain actions. The join actions include an action for adding every possible combination of capabilities to a team's set of capabilities. Likewise, the split actions include an action for removing every possible set of capabilities from a team. The maintain action results in a team retaining its capabilities. This action space allows a heterogeneous team to form teams with new joint capabilities, using join or split actions, or keep the current team to complete a task. Since the action space considers combinations of capabilities, the size of the action space grows according to $2^{|C|+1}$.

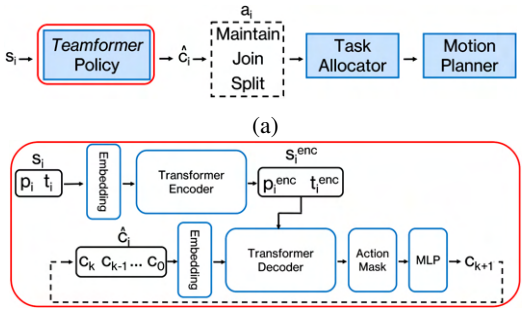


Fig. 2: a) The overall *Teamformer* framework. Given a team's overall state, s_i, the *Teamformer* policy returns a set of desired capabilities, $\hat{\mathbf{c}}_i$, resulting in a join, split, or maintain action. b) Using the team states, p_i, and task states, t_i, within s_i, the *Teamformer* policy autoregressively generates capabilities. Autoregressively generating $\hat{\mathbf{c}}_i$ enables the use of the scalable action space.

Rewards: A team's reward, R_i, is given by:

$$R_i = R_{comp} + \beta \frac{|\mathbf{c}_i^+| - |\mathbf{c}_i^-|}{|\mathbf{q}_{cur}|}, \tag{1}$$

where R_{comp} is a task completion reward, β is a scaling factor, $|\mathbf{c}_i^+|$ and $|\mathbf{c}_i^-|$ are the number of capabilities in the team that meet and do not meet the current assigned task requirements, respectively, and $\mathbf{q}_{cur}$ is the current assigned task requirements. When a team completes a task, R_{comp} is a positive reward α while every other step it is a negative time step penalty δ. The team reward is maximized when a team's capabilities exactly match the assigned task requirements and smaller otherwise. The magnitudes of α, δ, and β dictate the importance of completing tasks, avoiding extra time steps, and forming effective teams, respectively.

IV. Approach

To perform decentralized, scalable MRTA for heterogeneous robots, we split the problem into three subproblems: team formation, task allocation, and motion planning (Figure 2a). This section addresses each subproblem and describes the overall *Teamformer* framework.

A. Team Formation

For forming heterogeneous teams, we introduce the *Teamformer* policy: a decentralized transformer-based policy designed for scalable heterogeneous team formation (Figure 2b). Using DRL and the TF-MDP, the *Teamformer* policy decentrally forms heterogeneous teams based on a given team's overall state, s_i. Using both an encoder and decoder, the *Teamformer* policy learns the complex, combinatorial relationships among teams and tasks.

To use the encoder, a team's overall state must be embedded. Every team and task state within a given team's overall state is embedded into a team or task token. The tokens are then passed to the encoder, where multi-head self attention (MHA) is computed. For heterogeneous team formation, the relationships learned using MHA can be thought of in three ways: 1) how each team relates to nearby teams, 2) how each team relates to relevant tasks, and 3) how tasks relate to the teams. Since the policy is learning what teams to form, the attention mechanism is a natural way to capture the relationships between teams and tasks. A team's encoded overall state, s_i^{enc} is then used as context for the decoder. Conditioned on s_i^{enc}, the decoder predicts a team-level action, a_i, for the i-th team.

While the described *Teamformer* policy can operate on any number of teams and tasks as input, it relies on the original TF-MDP action space, which scales exponentially in the number of capabilities. However, using the autoregressive ability of transformers, the action space can be reduced to grow linearly in $|C|$. The key property of autoregression that allows this is autoregression reduces a joint space to a product of singular spaces [29], [30]. For team formation, instead of predicting the combination of capabilities a team should have all at once – which requires the exponentially growing action space – we obtain a linearly growing action space, in $|C|$, using autoregression to predict a capability at every step of autoregression. Both action spaces can produce the same combination of capabilities, but autoregression enables the space to scale better in C. To autoregressively generate a team action, the scalable action space is introduced.

Scalable Action Space: The scalable action space has an action for each capability in C and an end-of-sequence action. At each step of autoregression, the *Teamformer* policy either predicts the next capability of the current team or ceases autoregression, with action masking ensuring erroneous actions are not taken. The *Teamformer* policy, thus, outputs a sequence of desired capabilities for the current team, denoted as $\hat{\mathbf{c}}$. To identify if a join, split, or maintain action is taken, $\hat{\mathbf{c}}$ must be compared to the current team's capabilities $\mathbf{c}$. If $\hat{\mathbf{c}}$ contains fewer capabilities than $\mathbf{c}$, a split action occurs such that $\mathbf{c} = \hat{\mathbf{c}}$. If $\hat{\mathbf{c}}$ contains more capabilities than $\mathbf{c}$, a join action occurs with a team $l \in L$ that has capabilities $\mathbf{c}_l = \hat{\mathbf{c}} \setminus \mathbf{c}$. When $\hat{\mathbf{c}} = \mathbf{c}$, the team maintains its current capabilities. By iteratively building a team's capabilities, the action space is reduced from an exponential space to a linear space, enabling the *Teamformer* policy to scale linearly in $|C|$.

B. Task Allocation and Motion Planning

The *Teamformer* policy dictates team formation but does not allocate teams to tasks. A task allocator selects which task to perform and a motion planner determines how to get there (Figure 2a).

The task allocator has two steps: preprocessing and assignment. At the beginning of an episode, the preprocessing step solves a Traveling Salesperson Problem (TSP) to find an ordering to visit tasks [31]. Using a task order 1) ensures deadlock, i.e. two or more teams that need to join to complete tasks but are waiting for each other at different tasks, is avoided and 2) upper bounds the total makespan by the original TSP cycle cost, given an optimal team formation policy π_G^*. The assignment step is performed when a team has no assignment and after its team formation action has

been taken. The task allocator either assigns a team to the next task in the ordering that a team can contribute towards or performs a rewiring step such that the team is assigned to a task it can complete that decreases the calculated TSP cycle cost while still ensuring deadlock is avoided.

Once a team is assigned a task, the motion planner moves a team towards its assigned task. While A* [32] is used as the motion planner during training, the overall framework (Figure 2a) is designed for easily changing motion planners, such that any motion planner that moves a team towards its assigned task is viable (V-E).

V. Experiments

This section introduces the Team Formation Environment (TFE) and training of the *Teamformer* policy. To evaluate how well the *Teamformer* policy scales in the number of distinct robots and tasks, we compare against a state-of-the-art scalable deep set policy [20], an attention-based GNN policy [33], a heuristic team formation policy, and an optimal centralized time-extended MRTA solver. We perform hardware experiments demonstrating the *Teamformer* framework's ability to decentrally coordinate heterogeneous robots. Additionally, an ablation study is performed where previously unknown tasks are introduced to the problem as *Teamformer* is operating.

A. Team Formation Environment and Training

The TFE simulates heterogeneous robots and tasks in a discrete world. During training, the TFE generates environments containing four robots and 1-4 tasks, with randomly assigned capabilities and requirements. Each robot has capabilities randomly sampled from C without replacement. While the *Teamformer* framework can be used for robots with any number of capabilities we trained with each robot having a single capability as this yields the largest combinatorial space. Each task has a randomly generated set of requirements based on what capabilities are present. During training, the environment is a 10×10 grid, with robot locations generated around each other while task locations are selected randomly. The locations of robots and tasks are normalized before being input to the policy. Robots begin in teams of varying size, with teams being able to move up, down, left, or right, and only occupying one space at a time. The join action can be taken when a team is within a radius r_j of another team and results in two teams joining, when they are in the same space, or the team taking the join action motion planning towards the second team otherwise. The split action results in two teams being located at the same location. A task is completed when a team is located at the task and meets all requirements. An episode is terminated when all tasks are completed or a max number of time steps is reached.

We train the *Teamformer* policy in the TFE using Proximal Policy Optimization (PPO) [34], implemented by Stable Baselines [35]. To learn cooperative behavior more easily, we use the CTDE training paradigm. The critic uses all team and task states to predict a better estimate of the value, while the actor operates decentrally on each team's overall state. For efficient batching, a team's overall state is padded to be the same size for all teams and padded tokens are masked in the encoder and decoder. Since the action space changes with the number of capabilities, we train three *Teamformer* policies, with a total of four, eight, and 100 distinct capabilities, respectively.

B. Comparison Methods

The *Teamformer* policy is compared to two other learned team formation policies using a deep set architecture with attention [20] and an attention-based GNN adapted from [24], [27]. Both deep set models and GNNs handle variable size inputs and have been shown to work well as scalable multi-robot policies (see Section II). However, deep set architectures and GNNs do not inherently have a method for autoregression, resulting in the use of the original TF-MDP action space. Since the original action space becomes large quickly, comparison policies with only four and eight distinct capabilities are trained. Unlike the *Teamformer* policy, which explores its action space training on four heterogeneous robots, the deep set and GNN policy with eight distinct capabilities must train on eight heterogeneous robots to explore the entire action space.

Besides the learned comparison policies, *Teamformer* is compared against a heuristic team formation policy and a centralized time-extended MRTA solver. The heuristic policy takes the maintain action until a team is located at its assigned task. Once at a task, teams take the join action, when possible, and the maintain action otherwise, until the task requirements are met. Once the task is complete, teams split into individual robots and continues the process. The centralized solver, implemented using OR-Tools [36], attempts to find the optimal allocation of tasks, such that the total travel cost of all robots is minimized while ensuring all task requirements are met. Unlike the learned policies, which can fail to complete tasks, both the heuristic and centralized solver are guaranteed to complete all tasks given enough run time.

C. Simulations

We evaluate the performance of *Teamformer* and the comparisons methods across varying numbers of robots, capabilities, tasks, and number of requirements per task. For each test, a set number of robots are randomly placed and capabilities are evenly distributed among them, meaning no capability is more present than others. Each task can have up to $|C|$ requirements. The higher the number of requirements per task, the harder the problem becomes, since more combinations of capabilities must be considered. For a team formation policy trained with n max capabilities and tasks with at most k requirements per task, the total number of combinations, Ω, is described by:

$$\Omega = \sum_{i=1}^{k} \binom{n}{i} \tag{2}$$

The first test compares the learned and heuristic policies to the centralized solver. The test is performed in a 10×10 environment with four robots, each with a unique capability,

completing four tasks. This is the only test the centralized solver is able to find the optimal allocation of robots to tasks in a reasonable amount of time.

The second set of tests evaluates *Teamformer*'s ability to scale in the number of robots and tasks. With $|C| = 4$, each method is used to complete 20 tasks with four robots in a 10 $\times$ 10 environments and 100 tasks with 20 robots in 20 $\times$ 20 environments. Similarly, with $|C| = 8$, each method is used to complete 20 tasks with eight robots in 10 $\times$ 10 environments and 100 tasks with 24 robots in 20 $\times$ 20 environments. Tests marked with $*$ indicate when the centralized solver ran out of time and returned a feasible solution.

The final set of tests assess *Teamformer*'s generalization ability to scenarios requiring many heterogeneous robots to coordinate. The *Teamformer* policy with 100 distinct capabilities is tested on 100 heterogeneous robots completing 20 tasks with 10, 20, 50, and 90 requirements per task randomly sampled from $|C|$, ensuring all capabilities are present in at least one task.

Each test is performed over 50 randomized trials, with the number of failed trials and average inference time per trial indicated in Table I. All trials were performed on a NVIDIA Geforce RTX 2070.

D. Simulation Results

Comparison to Optimal: In the first test case (Figure 3), the *Teamformer* framework is competitive, within 1.5 time steps, to the makespan of the centralized solver and outperforms all other baselines by 0.4–6.4 time steps. While the ordering of tasks for the learned and heuristic policies is identical, the *Teamformer* policy already demonstrates its superior performance, even in small scenarios. Additionally, none of the learned team formation policies failed, which is expected since each policy was trained in these environments.

Scaling in Robots and Tasks: Analyzing the first test when $|C| = 4$ (Figure 4a), all learned policies generalize to five times more tasks than trained on, with the exception of the deep set and GNN policies failing to complete one trial each. The *Teamformer* policy completes all trials and does so 3%, 42%, 32%, and 21% faster than the baselines, respectively. In the second test (Figure 4b), each learned policy generalizes to five times the number of robots and 25 times the number of tasks trained on. Results for the centralized solver are omitted since it could not find a feasible solution given a 60 second run time. Similar to before, the *Teamformer* policy demonstrates superior ability to generalize by not failing any trials and outperforms all baselines.

While *Teamformer* performs the best over both test cases, the deep set policy's performance is similar. Since the number of distinct capabilities is small, and thus the original TF-MDP action space is relatively small, the benefit of using the scalable action space is not as pronounced. However, as the number of capabilities increases, the *Teamformer* framework clearly shows its enhanced scalability.

For both tests with eight distinct capabilities (Figure 5), the *Teamformer* policy maintains its ability to generalize across the number of robots and tasks, while the deep set and GNN policies struggle to generalize. The *Teamformer* policy, which was only trained on four heterogeneous robots, generalizes so well because the scalable action space enables frequent sampling of each action. This allows the policy to effectively learn team formation based on team capabilities and task requirements. Despite the deep set and GNN policies being trained on eight heterogeneous robots, they see a drastic increase in failed trials because their action space has increased by 2^4 compared to the previous trials. Effectively learning such a large action space is difficult, yielding increased failures and task completion times. Comparing against the heuristic policy and centralized solver, the *Teamformer* policy consistently outperforms them by $35 - 45\%$.

Scaling in Capabilities: The final test case showcases *Teamformer*'s ability to generalize up to 100 heterogeneous robots completing tasks with up to 90 requirements each (Figure 6). In this test case, the combinatorial team formation space is massive, yet the *Teamformer* policy effectively navigates the space and does not fail any trial. As the number of requirements per task increases, the number of steps to complete all tasks increases for the heuristic policy, yet decreases for the *Teamformer* policy. This is likely due to the *Teamformer* policy not using the split action as the tasks become more tightly coupled, since splitting a team in such scenarios does not result in completing tasks in parallel.

Computational Cost: In general, the *Teamformer* policy has the highest inference time. This is caused by 1) the increasing number of team and task states, which affects the $O(n^2)$ cost of MHA, and 2) the increased number of autoregression steps, up to the total number of robots in each trial, in each time step. The computational cost could be reduced by caching tokens and computing linear MHA [37].

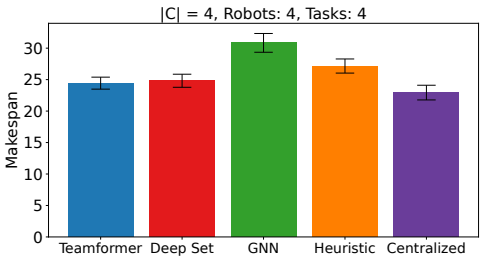


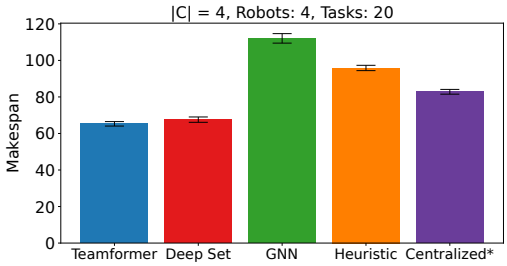
Fig. 3: Mean makespan over 50 trials with standard error of the mean for four heterogeneous robots completing four tasks. While *Teamformer* is decentralized, it is still competitive with the optimal solution.

E. Sim-to-Real Transfer

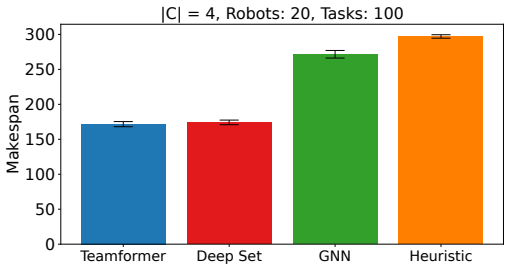
For sim-to-real transfer and hardware validation of *Teamformer*'s decentralized execution, we use the Georgia Tech Robotarium [13]. Despite the *Teamformer* policy being trained in a discrete environment using A* as a motion planner, it is capable of operating in a continuous environment with a continuous controller because the *Teamformer* policy makes no assumption on what type of environment it is operated in and has the minimal assumption that a robot's

TABLE I: The number of failed trials and average inference time to complete a trial for each test. N and T indicate the number of robots and task, respectively, in the test. $|R|$ indicates the number of requirements per task. Inference times show one standard deviation. Trials with * indicate the centralized solver reached its run time limit and returned a feasible solution.

	Method	$\|C\| = 4$		$\|C\| = 8$		$\|C\| = 100$			
		4N,20T	20N,100T	8N,20T	24N,100T	$\|R\| = 10$	$\|R\| = 20$	$\|R\| = 50$	$\|R\| = 90$
Failures	*Teamformer*	0	0	0	0	0	0	0	0
	Deep Set	1	1	26	12	–	–	–	–
	GNN	1	3	21	42	–	–	–	–
Inference Time	*Teamformer*	0.81 ± 0.11	7.81 ± 2.26	1.5 ± 0.2	13.13 ± 1.29	15.7 ± 1.32	17.4 ± 1.55	17.92 ± 1.22	18.06 ± 1.99
	Deep Set	0.16 ± 0.03	0.4 ± 0.03	0.35 ± 0.05	0.67 ± 0.16	–	–	–	–
	GNN	0.55 ± 0.09	2.07 ± 0.34	0.86 ± 0.14	4.48 ± 0.24	–	–	–	–
	Heuristic	0.31 ± 0.03	4.84 ± 0.3	0.73 ± 0.08	7.32 ± 0.32	6.34 ± 0.53	10.85 ± 0.47	14.98 ± 0.7	19.25 ± 1.6
	Optimal	61.47 ± 0.8*	–	63.49 ± 2.25*	–	–	–	–	–

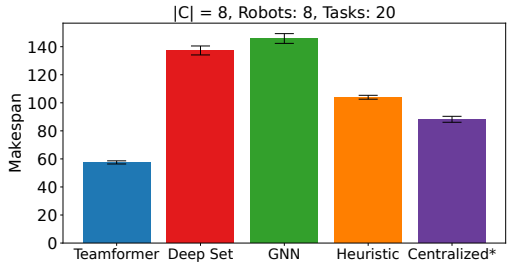


(a)

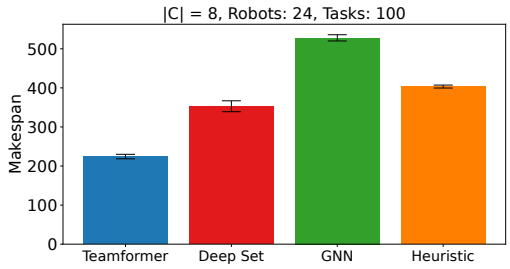


(b)

Fig. 4: Mean makespan over 50 trials with standard error of the mean with $|C| = 4$. a) Four robots completing 20 tasks. b) 20 robots completing 100 tasks. As the problem size increases, *Teamformer* shows superior performance to the baseline methods.



(a)



(b)

Fig. 5: Mean makespan over 50 trials with standard error of the mean with $|C| = 8$. a) Eight robots completing 20 tasks. b) 24 robots completing 100 tasks. With 256 possible team configurations, *Teamformer* demonstrates substantial improvement over competing methods.

motion planner will move it towards its assigned task. Additionally, the high-level team formation actions produced by the *Teamformer* policy enable easier sim-to-real transfer, since the *Teamformer* policy does not rely on training on difficult-to-simulate robot dynamics [38]. The *Teamformer* policy with four distinct capabilities was decentrally run on four heterogeneous robots to complete three tasks (Figure 7), and the *Teamformer* policy with 100 distinct capabilities was decentrally run on 20 heterogeneous robots to complete five tasks (Figure 8). The inner ring represents the robot's individual capability, while the outer ring represents the team the robot is in. When a team meets all task requirements, the outer ring matches the task outline color.

F. Ablation Study

To evaluate how the *Teamformer* framework adapts to previously unknown tasks during execution, we perform tests with four heterogeneous robots initially assigned to complete 20 tasks, with 20 more tasks appearing at a random time step. The tasks are added to the current task ordering by either resolving the TSP (Figure 9a) or by using a nearest neighbor (NN) heuristic (Figure 9b), where new tasks are inserted into the task order based on the order of the nearest task. All policies demonstrate the ability to adapt to the new tasks, demonstrating how the overall framework can adapt even when the individual team formation policies differ. Additionally, the *Teamformer* policy achieves the fastest time to complete all tasks and does not fail a single trial.

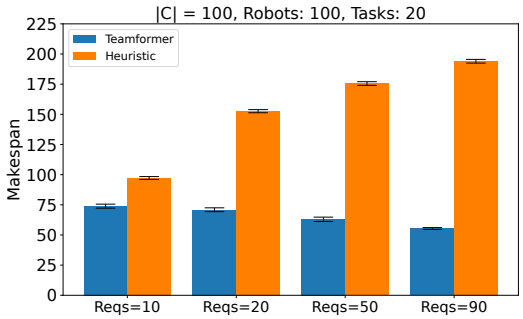
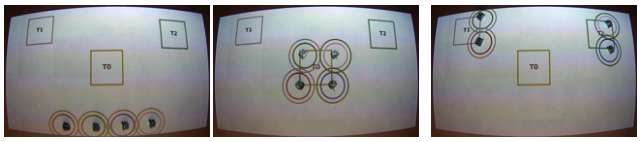


Fig. 6: Mean makespan over 50 trials with standard error of the mean with $|C| = 100$, with 10, 20, 50, and 90 requirements per task. *Teamformer* generalizes to large, complex problems demonstrating state-of-the-art scalability in heterogeneous MRTA.



(a) (b) (c)

Fig. 7: *Teamformer* decentrally operating on 4 heterogeneous robots. a) All robots in individual teams. b) The single team splitting. c) The split teams completing the final tasks.

VI. Conclusion

This work introduces the Team Formation MDP and *Teamformer* framework. Unlike previous work that considers limited heterogeneity among robots and tasks, *Teamformer* considers up to 100 robot capabilities and task requirements, resulting in a large combinatorial space. Combining the TF-MDP state space with the scalable action space enables *Teamformer*'s state and action spaces to scale linearly in the number of robots, tasks, and heterogeneous classes. Simulation and hardware results verify *Teamformer*'s ability to scale in the number of heterogeneous robots and tasks. Future work can extend the *Teamformer* framework to more explicitly account for continuous heterogeneity (i.e. differing speeds or carrying capacities) and uncertainty in task location and requirements.

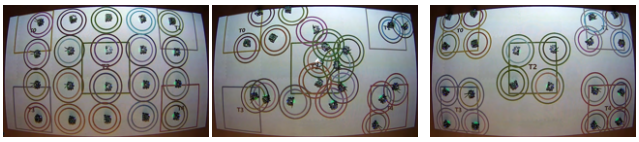
Acknowledgments

Google Gemini 2.5 Pro was used to help create the decision variables and constraints for the centralized solver.

References

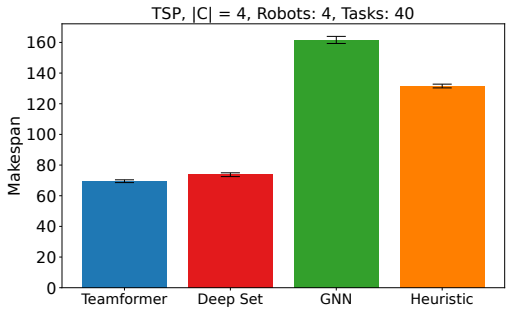
[1] E. Ackerman, "Boston Dynamics' Handle Teams Up With Mobile Robots on Warehouse Logistics - IEEE Spectrum," Mar. 2020. [Online]. Available: https://spectrum.ieee.org/boston-dynamics-otto-motors-warehouse-logistics

[2] C. Kunz, C. Murphy, H. Singh, C. Pontbriand, R. A. Sohn, S. Singh, T. Sato, C. Roman, K. Nakamura, M. Jakuba, R. Eustice, R. Camilli, and J. Bailey, "Toward extraplanetary under-ice exploration: Robotic steps in the Arctic," *Journal of Field Robotics*, vol. 26, no. 4, pp. 411–429, Apr. 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.20288

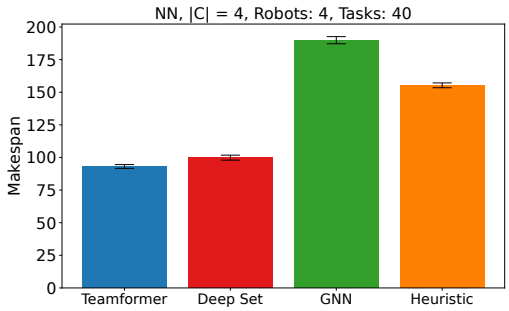


(a) (b) (c)

Fig. 8: *Teamformer* decentrally operating on 20 heterogeneous robots. a) All robots in individual teams. b) Teams moving and joining. c) All teams formed at their tasks.



(a)



(b)

Fig. 9: Mean makespan over 50 trials with standard error of the mean with four heterogeneous robots initially performing 20 tasks, with 20 more tasks being allocated using a) the TSP solver or b) a nearest neighbor heuristic. *Teamformer* effectively adapts to tasks appearing during execution as well as a changing task order.

[3] A. J. Shafer, M. R. Benjamin, J. J. Leonard, and J. Curcio, "Autonomous cooperation of heterogeneous platforms for sea-based search tasks," in *OCEANS 2008*, Sept. 2008, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/document/5152100

[4] Y. Xiao, J. Hoffman, T. Xia, and C. Amato, "Learning Multi-Robot Decentralized Macro-Action-Based Policies via a Centralized Q-Net," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 10 695–10 701. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9196684

[5] S. Mayya, R. K. Ramachandran, L. Zhou, V. Senthil, D. Thakur, G. S. Sukhatme, and V. Kumar, "Adaptive and Risk-Aware Target Tracking for Robot Teams With Heterogeneous Sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5615–5622, Apr. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9726858

[6] G. Notomista, S. Mayya, Y. Emam, C. Kroninger, A. Bohannon, S. Hutchinson, and M. Egerstedt, "A Resilient and Energy-Aware Task Allocation Framework for Heterogeneous Multirobot Systems," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 159–179, Feb. 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/

document/9531464

[7] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, Apr. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7819471

[8] 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. [Online]. Available: https://doi.org/10.1007/s10514-021-09997-2

[9] J. Peng, H. Viswanath, and A. Bera, "Graph-Based Decentralized Task Allocation for Multi-Robot Target Localization," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 10 676–10 683, Nov. 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10706007

[10] H. Wu, A. Ghadami, A. E. Bayrak, J. M. Smereka, and B. I. Epureanu, "Impact of Heterogeneity and Risk Aversion on Task Allocation in Multi-Agent Teams," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7065–7072, Oct. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9484733

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[12] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision Transformer: Reinforcement Learning via Sequence Modeling," June 2021, arXiv:2106.01345 [cs]. [Online]. Available: http://arxiv.org/abs/2106.01345

[13] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The Robotarium: Globally Impactful Opportunities, Challenges, and Lessons Learned in Remote-Access, Distributed Control of Multirobot Systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, Feb. 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8960572

[14] H. Ravichandar, K. Shaw, and S. Chernova, "STRATA: unified framework for task assignments in large teams of heterogeneous agents," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 2, p. 38, May 2020. [Online]. Available: https://doi.org/10.1007/s10458-020-09461-y

[15] Y. Cai, X. He, H. Guo, W.-Y. Yau, and C. Lv, "Transformer-based Multi-Agent Reinforcement Learning for Generalization of Heterogeneous Multi-Robot Cooperation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 13 695–13 702. [Online]. Available: https://ieeexplore.ieee.org/document/10802580

[16] 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. [Online]. Available: https://ieeexplore.ieee.org/document/10854527

[17] S. Paul, P. Ghassemi, and S. Chowdhury, "Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks," in *International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 8815–8822. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9812370

[18] S. Choudhury, J. K. Gupta, P. Morales, and M. J. Kochenderfer, "Scalable Online Planning for Multi-Agent MDPs," *Journal of Artificial Intelligence Research*, vol. 73, pp. 821–846, Mar. 2022. [Online]. Available: https://www.jair.org/index.php/jair/article/view/13261

[19] C. Sarkar, H. S. Paul, and A. Pal, "A Scalable Multi-Robot Task Allocation Algorithm," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5022–5027. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8460886

[20] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable Reinforcement Learning Policies for Multi-Agent Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 4785–4791. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9636344

[21] A. Agrawal, A. S. Bedi, and D. Manocha, "RTAW: An Attention Inspired Reinforcement Learning Method for Multi-Robot Task Allocation in Warehouse Environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 1393–1399. [Online]. Available: https://ieeexplore.ieee.org/document/10161310

[22] Z. Zhang, X. Jiang, Z. Yang, S. Ma, J. Chen, and W. Sun, "Scalable Multi-Robot Task Allocation Using Graph Deep Reinforcement Learning with Graph Normalization," *Electronics*, vol. 13, no. 8, p. 1561, Jan. 2024, number: 8. [Online]. Available: https://www.mdpi.com/2079-9292/13/8/1561

[23] A. Khan, V. Kumar, and A. Ribeiro, "Large Scale Distributed Collaborative Unlabeled Motion Planning With Graph Policy Gradients," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5340–5347, July 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9410368

[24] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-Aware Graph Attention Networks for Large-Scale Multi-Robot Path Planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533–5540, July 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9424371

[25] W. Dai, A. Bidwai, and G. Sartoretti, "Dynamic Coalition Formation and Routing for Multirobot Task Allocation via Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 16 567–16 573. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10611244

[26] L. C. D. Bezerra, A. M. G. d. Santos, and S. Park, "Learning Policies for Dynamic Coalition Formation in Multi-Robot Task Allocation," Feb. 2025, arXiv:2412.20397 [cs]. [Online]. Available: http://arxiv.org/abs/2412.20397

[27] M. Goarin and G. Loianno, "Graph Neural Network for Decentralized Multi-Robot Goal Assignment," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4051–4058, May 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10452797

[28] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep Sets," Apr. 2018, arXiv:1703.06114 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1703.06114

[29] M. Wen, J. G. Kuba, R. Lin, W. Zhang, Y. Wen, J. Wang, and Y. Yang, "Multi-Agent Reinforcement Learning is A Sequence Modeling Problem."

[30] T. Van de Wiele, D. Warde-Farley, A. Mnih, and V. Mnih, "Q-Learning in enormous action spaces via amortized approximate maximization," Jan. 2020, arXiv:2001.08116 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2001.08116

[31] S. H. Mulvad, "fast-tsp - A fast TSP solver with Python bindings," July 2022, original-date: 2022-07-27T15:38:14Z. [Online]. Available: https://github.com/shmulvad/fast-tsp

[32] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," *scipy*, May 2008. [Online]. Available: https://proceedings.scipy.org/articles/TCWV9851

[33] S. Brody, U. Alon, and E. Yahav, "How Attentive are Graph Attention Networks?" Jan. 2022, arXiv:2105.14491 [cs]. [Online]. Available: http://arxiv.org/abs/2105.14491

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[35] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[36] L. Perron and F. Didier, "CP-SAT," Feb. 2025. [Online]. Available: https://developers.google.com/optimization/cp/cp_solver/

[37] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-Attention with Linear Complexity," June 2020, arXiv:2006.04768 [cs]. [Online]. Available: http://arxiv.org/abs/2006.04768

[38] H. Choi, C. Crump, C. Duriez, A. Elmquist, G. Hager, D. Han, F. Hearl, J. Hodgins, A. Jain, F. Leve, C. Li, F. Meier, D. Negrut, L. Righetti, A. Rodriguez, J. Tan, and J. Trinkle, "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, p. e1907856118, Jan. 2021. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1907856118