RESEARCH | Article in Press

Deep Learning-Based Fast Inference of Team Performance in Heterogeneous Multi-Robot Systems

Mohammed Yusuf Ansari[1] · Eduardo Feo Flushing[1] · Mohammed Ishaq[2] · Mohammed Yaqoob[2] · Gianni A. Di Caro[1]

Received: 17 September 2025 / Accepted: 30 September 2026
© The Author(s) 2026

Abstract

The effectiveness of multi-robot teams depends on various factors, including the size of the team, the capabilities of each individual robot, the synergies between them, and the coordination strategies employed. However, current studies on multi-robot task allocation often assume pre-established teams, which results in a focus on optimizing resources rather than designing teams from a diverse range of robots or selecting the most suitable team from a set of potential configurations. Our research focuses on predicting a team's performance in a given mission based on its composition. This task has significant implications for selecting or designing multi-robot teams, as it helps to optimize performance while minimizing costs. To this end, we propose a novel fast inference method for estimating team performance using supervised learning in multi-robot task allocation situations. We considered a general multi-robot task allocation issue for spatially distributed missions. For evaluation purposes, we used a large dataset of varied problem instances, two different solvers, and five different team performance metrics derived from the solutions computed by these solvers. The results demonstrate that long short-term memory recurrent neural networks, aided by convolutional layers, can accurately predict team performance metrics such as makespans and intra-team communication metrics derived from the solutions obtained by the two solvers. We also highlight the efficacy of transfer learning in fine-tuning models from a simpler sub-optimal solver to a more complex, optimal solver, demonstrating improved accuracy with reduced training set size and computational burden.

Keywords Heterogeneous multi-robot systems · Team performance prediction · Multi-robot task allocation · Deep learning · Robot team composition · Long short-term memory (LSTM) · Transfer learning · Fast inference

Mohammed Yusuf Ansari and Eduardo Feo Flushing contributed equally to this paper.

For G. Di Caro, this publication was partially made possible by the NPRP award 14S-0417-210227 from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors

This Article in Press is shared early to give you faster access to new research. It is citable and carries a permanent DOI. The final edited version will replace it automatically.



https://doi.org/10.1007/s10846-026-02477-8

Published online: 11 November 2026

1 Introduction

The success of a multi-robot team for a specific mission largely depends on team size, the capabilities of the individual robots, the potential synergies among them, and the coordination and planning schemes utilized. The large majority of the works concerning with multi-robot task allocation assume that the team has already been designed to guarantee a given performance or, more simply, it has been predefined before deployment based on available resources. Thus, the focus has traditionally been on making the best use of the available resources in the team rather than on how to select the best resources from a potentially large pool [15]

This work, instead, addresses *how to design* a multi-robot team for tackling a given global mission composed by a number of tasks. In this context, our work investigates the fundamental problem of *inferring team's performance* given a team composition and under a specific allocation scheme for robots to tasks. In other words, we aim to answer the question: *how well a particular team would perform for the given global mission when effectively coordinated?* The ability to infer team performance holds significant implications for the strategic design or selection of multi-robot teams. It can allow to choose teams that can provide an optimized performance while at that same time minimizing costs. For instance, if a team of n robots can do the job better and faster than a different team of $m > n$ robots, this results in an advantage in terms of the final result and use of available resources.

In order to address the above question, as a context of reference we consider general mission scenarios where the multi-robot team is *heterogeneous* and has to tackle a number of tasks that are *spatially distributed*. Each task requires certain sensory-motor skills to be dealt with effectively and efficiently, such that a fundamental aspect for mission success consists in how the individual robots are allocated to the tasks. Moreover, since the tasks are spatially distributed, we assume that the robots are mobile and need to travel from one task location to another. This general scenario defines a *spatial task allocation problem in heterogeneous multi-robot teams*, which was previously introduced in [22, 23] under the acronym STASP-HMR. This type of problem requires jointly tackle task allocation, task scheduling, and path routing challenges, and turns out to be NP-Hard [22]. Given a heterogeneous team, both centralized [23] and decentralized [3, 20] solution approaches have been proposed for finding mission plans.

In this work, we build on STASP-HMR to address the question of selecting the *best* multi-robot team for the mission. In principle, finding the best team would require a search in the space of all possible teams' compositions, each time checking its best performance by solving the associated STASP-HMR model. This turns out to be a very difficult problem, because we need to take into account the specific characteristics of the problem instance in relation to robots' skills for dealing with the tasks, and for each candidate team configuration we would have to solve an associated STASP-HMR instance to obtain the desired measure of performance.

At this aim, instead of directly solving STASP-HMR instances, we propose using *deep learning* to train a predictive model that can quickly estimate team performance for a variety of STASP-HMR missions with similar characteristics. In this way, the deep learning model could effectively be embedded in a search process. In general, the inherent advantages of deep learning, such as its capacity to capture complex patterns and variables' interdependencies, make it an attractive and efficient alternative to other methods, such as solving the underlying multi-robot task allocation/routing problem using approximation algorithms to estimate upper bounds on performance [39], or running simulations.

Training labels for team's performance are obtained by solving a wide range of STASP-HMR problem scenarios. Ideally, an optimal solver shall be used to generate solution labels. However, due to the NP-hardness of the problem, an optimal or near-optimal solver would take a considerable amount of time, imposing a practical limit on of generating large training datasets. Therefore, we mainly used a computationally efficient, yet potentially sub-optimal, distributed solver [3]. Target team performance is computed in terms of *makespan* for completing the given mission, as well as of properties of the network topology resulting from the deployment of the robots. These *networking metrics* are particularly important in a multi-robot team to ensure some degree of local connectivity during mission deployment.

To the best of our knowledge, this is the first work that showcases the potential of deep learning for multi-robot team performance estimation. Specific contributions of the work are as follows:

1. **Real-time LSTM-Based Team Performance Predictor:** We propose a spatio-temporal deep learning architecture that combines *Long-Short Term Memory* (*LSTM*) and *convolutional layers* to predict team performance for heterogeneous multi-robot systems tackling spatially distributed task allocation problems. The proposed models are designed to generalize effectively across unseen grid sizes, grid shapes (e.g., square and rectangular), grid topologies (e.g., inaccessible terrains), and task models (i.e., simple or complex), ensuring broad applicability in diverse real-world scenarios. Additionally, we demonstrate the inference efficiency of the proposed networks over the conventional heuristic approaches.
2. **Representative Team Performance and Connectivity Dataset:** We generate a comprehensive dataset of problem instances labeled with mission makespan and network connectivity metrics using a fast, sub-optimal solver. The data set accounts for various grid sizes and shapes, including square and rectangular grids, as well as different grid topologies to simulate inaccessible real-world scenarios such as uneven or impassable terrains. Furthermore, the data set includes varying team compositions, task models, and spatial correlations of tasks on the grid, ensuring a robust representation of real-world mission scenarios.
3. **Transfer Learning for Near-Optimal Predictions:** We demonstrate the effectiveness of *transfer learning* in achieving near-optimal performance. By initially training the networks with labels from a fast, sub-optimal heuristic solver and fine-tuning them with a small set of labels from an optimal (yet computationally demanding) solver, we obtain a predictor that approaches near-optimal accuracy.
4. **Robust Validation and Sensitivity Analysis:** We validate the proposed models through extensive experiments across diverse problem scenarios and performance metrics. Additionally, we carry out a *sensitivity analysis* under various scenarios and assumptions to assess the reliability and robustness of the team performance predictions.
5. **Scalable Real-Time Applicability:** The resulting predictor enables real-time selection or design of the best-performing multi-robot team, enhancing both scalability and computational efficiency.

The remainder of this paper is structured as follows: Section 2 reviews the related work in spatial task allocation and team performance estimation. Section 3 presents the methodological framework for STASP-HMR, detailing the problem definition, heuristic solvers, team performance and connectivity metrics, and the dataset generation for training neural networks. Section 4 introduces the LSTM-based deep learning framework for team performance estimation. Section 5 discusses the results, including comprehensive experiments and sensitivity analyses. Finally, Section 6 concludes the paper and outlines potential future directions.

2 Related Work

Deep neural networks, and in particular convolutional neural networks, have gained significant traction over the years because of their ability to efficiently learn task-relevant features and patterns directly from the raw data, alleviating the domain knowledge dependence of most of the prior machine learning approaches. Consequently, deep neural networks have seen widespread application across various domains, including medical imaging [7, 9, 40], biomedical signal processing [5, 8], and drug discovery [6, 16]. In the multi-robot domain, deep neural networks have emerged as a common tool in autonomous navigation and path planning [32, 54], swarm robotics [31, 47], and environmental monitoring and exploration [14, 52].

This section provides an overview of learning-based methodologies within multi-robot domains, specifically focusing on deep learning and on task allocation problems. Given the inherent optimization nature of task allocation, we also explore related learning-based approaches within combinatorial optimization.

2.1 Machine Learning in Multi-Robot Domains

In recent years, the use of learning-based approaches in multi-robot applications has significantly increased, with a particular emphasis on deep reinforcement learning (RL). Numerous studies have considered applications across various domains, including as multi-robot path-finding [34, 42, 45], task allocation [1, 24, 27, 37, 43, 51, 55], exploration [25, 28, 46], and information-gathering[48]. These studies typically formulate their corresponding problems as sequential decision-making and aim to create a control policy that identifies the best robot-action pairs to maximize a discounted reward that is presumed to be associated with the team's overall performance. In this respect, deep-learning neural networks allow the extraction of features from the state representations that serve as a guide to decide the best action to select under specific conditions.

Our primary objective is different from these previous works. While other approaches aim to optimize team actions for a specific task and a given team, our goal is to estimate the team's potential performance under optimized conditions. We aim to achieve this without directly solving the associated optimization problem. In other words, we focus on predicting how effective a team could be if their actions were optimized. We are addressing a complementary problem, which pertains to team design and selection rather than to its use. At the same time, our learning-based approach helps us avoid the computational complexity that comes with solving optimization problems.

In contrast to previous applications of neural networks in multi-robot decision problems, we use recurrent neural networks to incorporate team characteristics alongside mission data. Our choice of the neural network architecture is motivated by the variable-length, non-fixed input team size.

Specifically, our approach uses LSTM networks [26], which also have a widespread application in various multi-robot problems. For instance, in tasks such as robot navigation and trajectory prediction, where the input data unfolds over time, LSTM architectures capture temporal dependencies and patterns. The work in [50] utilized LSTM to accurately predict robot dynamic movement within an environment, exploiting the sequential nature of spatial-temporal data. Similarly, [2] employed LSTM networks to determine 3D trajectories that are energy efficient and ensure the coverage of large physical structures. In [36], an LSTM-based network is used for robot path planning in a dynamic obstacle avoidance environment.

Besides using deep learning approaches to enable a team of robots to achieve a certain task, we identified a few works where deep learning is used to predict certain team properties from an outsider's perspective. Closer to our work, [53] uses a deep neural network to predict congestion delays in large multi-robot teams motivated by real-world challenges in logistics automation. Although they aim to predict values associated with future spatio-temporal locations, they assume that planned paths have already been computed and, therefore, are provided as input. Our approach differs significantly as we focus on anticipating team performance before the team has decided on specific actions, avoiding the necessity of solving associated optimization problems. Similar to their methodology, we utilize grid graphs to represent the mission and employ a Convolutional LSTM (ConvLSTM) to learn the spatial features of the mission [44]. To elaborate, the convolutional layers capture local spatial relationships within the task grids, while the LSTM models temporal dependencies arising from interactions among agents within a team. ConvLSTM was selected over transformer-based alternatives such as Vision Transformers (ViTs) due to its stronger spatial inductive bias and superior data efficiency in limited-data settings. Unlike conventional vision tasks, generating labeled training instances in our framework requires solving task allocation problems using solvers, making dataset generation computationally expensive and constraining the amount of available training data. As noted in the original ViT study [18], transformer-based architectures generally require substantially larger datasets to learn effective spatial representations, whereas convolutional architectures can exploit spatial structure more efficiently. Furthermore, the proposed framework is intended to be rapidly retrained and adapted to changing mission conditions, including different robot teams, environments, and task distributions. Consequently, we prioritized an architecture that achieves strong predictive performance on small-to-moderate spatial grids while remaining practical for repeated retraining. Although ViTs may offer advantages for larger

spatial representations and substantially larger training datasets, investigating such architectures falls outside the scope of the present work.

In a recent study [10], a team of researchers employed a supervised learning framework to develop a predictive model for assessing subteam performance in routing tasks for specific coalitions. However, their analysis was limited to specific types of robots, and performance predictions were solely based on the number of robots of each type. In contrast, our approach considers a broader range of team compositions, where each robot type can be described in terms of its task performance. This results in a significantly expanded set of features. Therefore, we focused more on the dataset generation and model training aspects to effectively capture the complex relationships within the broader data distribution.

2.2 Machine Learning for Optimization

Recently, many studies have studied ways to analyze and solve combinatorial optimization problems using machine learning [13]. Approaches that aim at directly solving the optimization problem can be widely divided into two groups: step-by-step and end-to-end [30]. In step-by-step approaches, the goal is to build or improve a solution iteratively, where the learning component is usually combined with an algorithm that decides on an improvement strategy for the current solution [33]. On the other hand, in end-to-end approaches, the model learns a mapping function from a problem instance to a feasible solution. RL frameworks are commonly used in end-to-end approaches to build domain-specific greedy heuristics [11, 17]. The resulting policies are used to construct good sub-optimal solutions by means of operations on the solution structure [35]. Other end-by-end approaches use supervised learning, e.g., Pointer Networks [49], to construct solutions based on the predicted occurrence of certain elements in the optimal solution. Related to multi-robot task allocation that we address in this paper, many works have studied the learning-based approaches to Vehicle Routing Problems (VRPs) [30], which deals with defining the optimal routes for visiting a set of customers using a fleet of vehicles. VRPs are closely related to multi-robot task allocation problems [38] and are amongst the most widely studied combinatorial optimization problems in operations research.

Both step-by-step and end-to-end approaches have been proposed for addressing VRPs. Step-by-step approaches are known for their better generalization capabilities and ability to produce higher-quality solutions. In contrast, end-to-end approaches prioritize speed, sometimes at the expense of solution quality. Nevertheless, both approaches are supported by deep neural networks, which have demonstrated their effectiveness in extracting features from the complex structure of problem instances, further enhancing their utility in solving VRPs.

Our approach falls under the end-to-end framework, which involves learning a mapping function that goes from input to solution. However, in our case, the output of the learned function is not the actual solution to the problem. Rather, the output is a singular value that reflects the objective function value of the associated optimization problem. Only a few studies aim to develop a mapping function from an instance of an optimization problem to an output that deviates from the optimal solution. In a recent study [57], supervised learning is used to identify the best solver for a given traveling salesman problem (TSP). The authors solved 6,000 TSP instances using different solvers and employed a convolutional neural network (CNN) to predict the performance of a solver in terms of running time. Although their primary focus was solver selection, the successful application of deep neural networks as feature extractors for combinatorial optimization problem instances highlights their potential in this domain. In evolutionary computation, supervised learning is used to approximate the fitness evaluation of candidate solutions to improve the convergence speed. In these works, the role of the neural network is to assess the fitness when there is no analytical fitness function by which the solutions are evaluated, or even if it exists, it is computationally expensive to evaluate [29]. Supervised learning has also gained traction in solving stochastic optimization problems. In[12], Bengio *et al.* used supervised learning to predict the quality of candidate facility allocations in a two-stage stochastic capacitated facility location problem. They generated a dataset of 50K random

locations for ten facilities that they solved using a MILP. They showed that the learning-based solver provided good quality solutions in orders of magnitude faster than off-the-shelf state-of-the-art solvers.

3 Methodological Framework of STASP-HMR: Problem, Solvers, Metrics, and Dataset

This section introduces the Spatial Task Allocation and Scheduling Problem in Heterogeneous Multi-Robot Teams (STASP-HMR) [22], which serves as the underlying optimization framework used throughout this work. We first describe the problem formulation and the representation of mission plans. Next, we present the solution methods used to generate mission plans, namely the CoLoSSI [4] heuristic and the MILP formulation. The resulting mission plans are then used to derive team-performance measures, including mission makespan and communication-network metrics. Finally, we describe how these metrics are used to generate the datasets that serve as training labels for the proposed neural-network models.

STASP-HMR, introduced in [22, 23], provides a general framework for jointly addressing task allocation, routing, and scheduling in heterogeneous multi-robot teams. Its broad and flexible formulation encompasses a wide range of spatially distributed multi-robot missions and therefore serves as a suitable benchmark for studying team-performance estimation.

3.1 The Problem: STASP-HMR

The STASP-HMR considers a set of robots $A = \{1, \dots, m\}$ and a set of spatially distributed tasks $\mathcal{T} = \{1, \dots, n\}$. Similar to [22], we consider missions in which task locations are embedded in a two-dimensional grid environment. Robots can move between neighboring grid cells and service tasks located at those cells. The STASP-HMR explicitly considers heterogeneous teams, where agents may exhibit different levels of performance when servicing the same task. This heterogeneity is captured by the task-efficacy model $\varphi : A \times \mathcal{T} \mapsto \mathbb{R}$, which relates the amount of effort devoted by an agent to the progress achieved on a task. Thus, larger values of $\varphi(i, k)$ indicate that robot i is more effective at executing task k.

A feasible solution to the STASP-HMR is a mission plan that completes all tasks in $\mathcal{T}$. Following [22], a mission plan is represented as $P = \{P_k \mid k \in A\}$, where $P_k = \langle a_k, p_k, s_k \rangle$ denotes the plan associated with agent k. Here, $a_k \subseteq \mathcal{T}$ is the set of tasks assigned to the agent, p_k defines the order in which those tasks are executed, and s_k specifies the amount of service time allocated to each task. A mission plan does not directly encode robot trajectories. Instead, it specifies task allocations, task routes, and service schedules. Given a mission plan, an agent s route can be converted into a sequence of task waypoints. In the grid-world abstraction considered here, the corresponding execution trace is reconstructed by connecting consecutive waypoints along route p_k through grid paths. In a physical deployment, these waypoints would instead be realized by a lower-level motion planning and control layer that generates trajectories satisfying the robot s kinematic and dynamic constraints. Since STASP-HMR assumes discretized mission time, execution traces are represented as sequences of grid locations indexed by mission time steps. Let w_{ik} denote the total amount of time that agent i devotes to task k. A task is considered *completed* when the total service provided by the team is at least equal to the task's required workload. Formally, $\sum_{i \in A} \varphi(i, k) w_{ik} \geq C_k$, where C_k denotes the initial workload requirement of task k.

An optimal solution to the STASP-HMR is a feasible mission plan that minimizes the mission makespan, i.e., the amount of time required by the team to complete all tasks. Accordingly, the optimization problem can be written as

$$
\begin{aligned}
P^\star = \arg\min_{P} \quad & \text{Makespan}(P) \\
\text{s.t.} \quad & \sum_{i \in A} \varphi(i, k) w_{ik} \geq C_k, \quad \forall k \in \mathcal{T},
\end{aligned}
$$

together with the allocation, scheduling, and routing constraints that define feasible mission plans [22]. The complete MILP formulation is reported in [22]. The STASP-HMR is NP-hard [22] and therefore presents computational challenges for solving it to optimality.

3.2 The Solvers: CoLoSSI and MILP

Given the NP-hardness of the STASP-HMR , using out-of-the-box solvers can result practically intractable for large instances. To cope with this challenge, a solution method based on the combination of standard solvers and heuristics was proposed [19, 22], that was able to retain *anytime* properties and formal guarantees on solution quality while speeding up computations. In addition, decentralized [20] and distributed approaches [3] have also been proposed, each presenting its own set of tradeoffs.

In this study, for solving the problem instances (i.e., generating mission plans) and computing associated training labels, we have selected the Cooperative and Load-Balancing Task Allocation and Scheduling based on Sequential Single-Item Auctions (CoLoSSI) [3, 4] as a relatively fast, yet sub-optimal distributed approach, and the centralized, optimal approach described in [22], that computes near-optimal solutions using the MILP . As explained later in the paper, most labels have been generated using CoLoSSI precisely to generate a large training dataset.

CoLoSSI implements a sequential single-item auction approach to perform efficient task allocation to all agents. The algorithm begins by randomly selecting an auctioneer from a team of agents, with the remaining agents acting as bidders. Tasks are broadcast, and agents calculate bid values based on task proximity and expected completion time. The auctioneer assigns tasks based on the minimum bid, and agents adjust their plans accordingly. The key features of CoLoSSI include a bidding function designed to promote cooperation and post-auction processes to balance workloads and improve agent communication.

More specifically, at each auction round, every agent computes a bid for each available task based on the current mission state and its local plan. Let C^j denote the completion map at auction round j and let a'_k denote the set of tasks already awarded to agent k. The bid computed by agent k for task i accounts for four factors: (i) the residual effort required to complete the task, $C^j(i)/\varphi(k,i)$, where $\varphi(i,k)$ captures the ability of agent k to service task i; (ii) the current workload assigned to the agent, measured by the number of previously awarded tasks $|a'_k|$; (iii) the distance between the candidate task and the agent's current route, $\text{dist}(p_k, i)$; and (iv) the duration of the agent's current schedule, $\sum_{l \in a_k} s_k(l)$. The auctioneer assigns the task to the agent with the minimum bid and the corresponding mission plan is updated.

As described in the previous section, a feasible solution to the STASP-HMR is a mission plan that completes all tasks while satisfying the workload requirements. The output of CoLoSSI is one such feasible mission plan P. Since CoLoSSI is a heuristic auction-based solver, the resulting mission plan is generally sub-optimal and therefore does not necessarily minimize the mission makespan.

Although CoLoSSI efficiently generates high-quality feasible mission plans, repeatedly solving STASP-HMR instances remains computationally demanding for large scenarios, as shown in Fig. 5. The mission plans generated by CoLoSSI are used here to compute the mission makespan and the communication-network metrics described in Section 3.3; these quantities serve as target labels for the learning models. The computational burden associated with repeatedly generating such labels motivates the proposed learning-based approach, in which a neural network is trained to directly estimate team-performance metrics without repeatedly solving the underlying task-allocation problem.

On the opposite side of the spectrum, we also selected an optimal solution approach [22] leveraging mixed-integer linear programming, specifically utilizing the CPLEX mathematical solver. It is important to highlight that finding optimal solutions with optimal approaches involves significantly high computational costs due to the NP-Hard nature of the associated problem. For the STASP-HMR , computing optimal plans for instances with a few hundred tasks and few agents can sometimes take hours.

Neither of these methods allows for repeated evaluation of the effectiveness of a specific team. Our primary objective is to train a deep learning model using instances solved offline, enabling its utilization alongside search-based approaches for evaluating multiple compositions. Deep learning models offer the advantage of constant inference time, significantly reducing computational requirements compared to the solvers considered in this study.

3.3 Team Metrics: Mission Makespan and Network Connectivity

As already pointed out, the main metric of team performance is *mission makespan*, which is a direct measure of the efficiency of the selected team composition.

In addition, we also consider the fact that some measures of *network connectivity* within the team should be taken into account when assessing team performance. More precisely, when a multi-robot team is deployed in a possibly large area, and the mission is composed of spatially distributed tasks, the robots must traverse the environment to arrive at the locations of the tasks. In communication-restricted environments, which models several practical scenarios of interest, a fixed communication infrastructure is not available. As a result, wireless communications – commonly used in real-world deployments – become unreliable and prone to failures due to radio range limitations. In a real-world deployment, the lack of communication can be problematic, impeding local coordination and planning, as well as the overall monitoring of the system.

The solvers considered in this work, like most task allocation frameworks, do not enforce specific levels of network connectivity. Mission plans are designed to minimize the time required to complete all tasks without considering the resulting network topology while executing the mission.

Yet, given the trajectories induced by the mission plan and a communication model, it is possible to assess the level of communication friendliness by simulating the execution of the plan or using metrics from graph theory. At this aim, we have defined four metrics of team performance related to network connectivity, M_0, M_1, M_2, M_3, which are explained below.

Assuming a simplified *disk model* for the communication based on a predefined wireless radio range, in this work, we hypothesize that, given a mission scenario, we can predict the intra-team communication level of the mission plan that would result from a given task allocation scheme (i.e., from CoLoSSI, or the optimal solver). To test this hypothesis, we consider various metrics derived from the deployed trajectories of the team when executing a given mission plan. In practice, these metrics correlate to the communication quality of the mobile wireless ad hoc network formed by the teams after deployment. It is important to note that we aim to understand how well the learning model can predict other values associated with the team's deployment. Therefore, our choice of metrics mainly serves for evaluation purposes. The study of their practical use as communication quality predictors is outside the scope of this work. Note that our approach can be expanded to other metrics related to the topology of the communication graph formed by the agents during the mission execution.

For a given mission plan, we extract the locations at which agents will be deployed at any time step and construct a sequence of M undirected communication graphs G_t for $0 < t \leq M$, where M is the mission makespan for the specific plan, and there's an edge between agent i and j if the location of the tasks executing at time t by these two agents permits communication between i and j. For simplicity reasons, we assume that two agents can communicate if the distance between the location of the tasks they execute is within a certain threshold (i.e., the radio range).

Using the extracted communication graphs G_t, we compute four different values related to how well-connected the team is during the mission. The metrics are referred to as M_j, for $0 \leq j \leq 3$ and described in Table 1.

3.4 Dataset Generation for Team Performance Estimation

We consider a cell-based grid decomposition of the (continuous) mission area, where each cell corresponds either to a task location or an inaccessible zone. In turn, this cell-based decomposition defines a traversability graph, where each cell corresponds to a vertex and has edges to all its eight cells in the Moore neighborhood. Given

Table 1 Intra-team Communication metrics. Each metric is computed over all G_t, $0 < t \leq M$

Metric	definition
M_0	Average Connected Component Size
M_1	Average Node Degree
M_2	Average Betweeness Centrality
M_3	Average Algebraic Connectivity

such a spatial task decomposition and a team of robot agents A, the parameters that define a problem instance are the following:

3.4.1 Grid Size

We consider square and non-square grids of size $M \times N$, with $0 \leq M, N \leq 20$. The grid size defines the total number of tasks, thus impacting the mission makespan. Figure 1 (a-c, e) show 20×20 scenarios. The color of the cell/task indicates the amount of time one specific agent in the team requires to complete it, as it is explained below. Figure 1 (d) shows one scenario with size 20×5.

3.4.2 Topological Features

Agents can move through adjacent tasks/cells that share at least one vertex in the graph associated with the grid. Yet, in many real-world scenarios, robot teams may encounter several regions on task grids that cannot be traversed due to obstacles or terrain challenges. To this end, we also generate instances containing contiguous groups of grid cells that are not associated with any task and are not accessible by the agents, henceforth referred to as *inaccessible areas*. Figure 1 (c) shows one scenario with three inaccessible areas depicted in black. Including these areas allows us to generalize to scenarios with non-trivial traversability. We expect instances with inaccessible areas would induce larger mission makespans as agents require larger trajectories to move throughout the grid.

3.4.3 Team Composition

Besides the characteristics of the environment, a core aspect of a problem instance is the composition of the agent team A. As mentioned in Section 3.1, each team member is characterized by its performance at doing the tasks, which in turn is modeled through the agent's task model φ. The values in φ specify the fraction of the workload of each task that the agent can service during a single time step. Note that the task model can also be seen as the amount of time the agent requires to complete the task by itself ($\frac{1}{\varphi}$).

In our dataset we consider teams ranging from 5 up to 20 agents. When defining the team composition, we consider different heterogeneity levels, where the heterogeneity level is defined as the number of different task models among all agents.

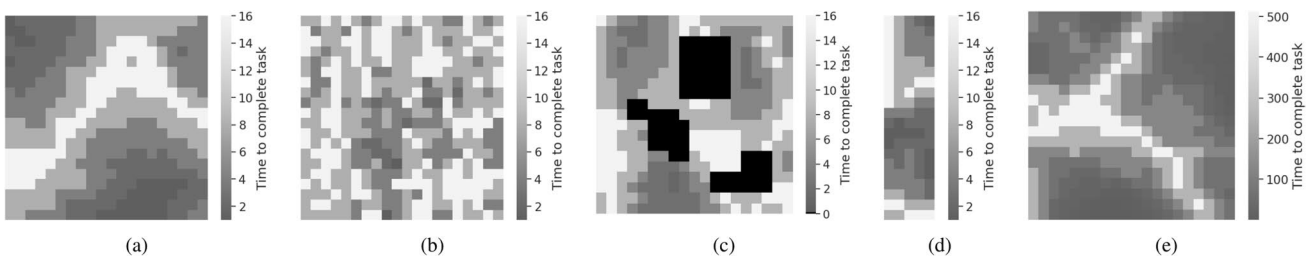


Fig. 1 Examples of task models with different spatial correlation

3.4.4 Task Models

We have considered two types of task models: standard and complex. In standard task models, robot agents exhibit one out of five different performance levels (randomly determined) at doing each task, ranging from completely inefficient ($\varphi \triangleq \frac{1}{16}$) up to highly efficient ($\varphi \triangleq 1$). On the other hand, in complex task models, the range of values is largely increased ($\frac{1}{512} \leq \varphi \leq 1$).

Given that we consider instances where tasks are embedded in a 2D grid, we can visualize a task model as a 2D heat map, as shown in Fig. 1. Figure 1 (a-d) show four standard task models considered in the dataset. Note that the maximum time to complete a task under these models is 16 time units, while the minimum time is 1 time unit. Figure 1(e) shows a complex task model. Note that in these models, the maximum time to complete a task is 512 time units, and the minimum time remains 1 time unit.

3.4.5 Spatial Correlation in Task Models

Another aspect considered when generating task models is their degree of spatial correlation. This correlation is related to the nature of changes in agent performance throughout the grid. When the spatial correlation is high, these changes are smooth. This implies that when an agent is allocated to a task well suited, it is expected that it is well suited for the surrounding tasks, too, thus facilitating planning. Previous work [21] shows that mission makespans are shorter in missions with correlated task models. Figure 1 (a) shows a task model with high spatial correlation, while Fig. 1 (b) depicts a task model with low spatial correlation. Note that task models with low spatial correlation tend to be more noisy, while with high spatial correlation, tasks tend to be grouped based on the efficiency level at which the agent can handle them. In [21], we used a method based on fractal image generation to generate 2D distributions with varying levels of spatial correlation (i.e., low, and high spectral coefficient values) to generate the task models. We showed that spatial correlation significantly impacts the team's performance.

We generated 120,000 problem instances with different properties based on the above parameters. Table 2 summarizes the parameters used for the dataset. We use CoLoSSI to compute a task allocation and retrieve the mission makespan for all instances (Fig. 2). Using the trajectories described by the task allocation, we also compute values for the network metrics based on the location of the robots at each time step. We also attempted to compute the optimal task allocation for all instances using a MILP solver. However, given the computational costs, we only obtained optimal solutions for a small subset of the benchmark dataset. We used these optimally solved instances in the experiments in Section 5.4. Both solvers run on an Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.30GHz with 16GB of RAM and a 1-hour limit of CPU time.

4 Deep Learning of Team Performance Estimation

This section outlines our *supervised learning*-based approach for predicting team performance. We detail the neural network architectures, their configurations, hyperparameters, loss functions, and implementation details to ensure reproducibility.

Table 2 Dataset generation parameters

Parameter	Values
Grid Size ($M \times N$)	$M \times N, 5 \leq M, N \leq 20$
Number of Agents ($\|A\|$)	5 up to 20
Heterogeneity Level	Number of different agents: 1 up to $\|A\|$
Task Model (values)	Standard: $[\frac{1}{16}, 1]$, Complex: $[\frac{1}{512}, 1]$
Task Model (correlation)	Uncorrelated & High Correlation

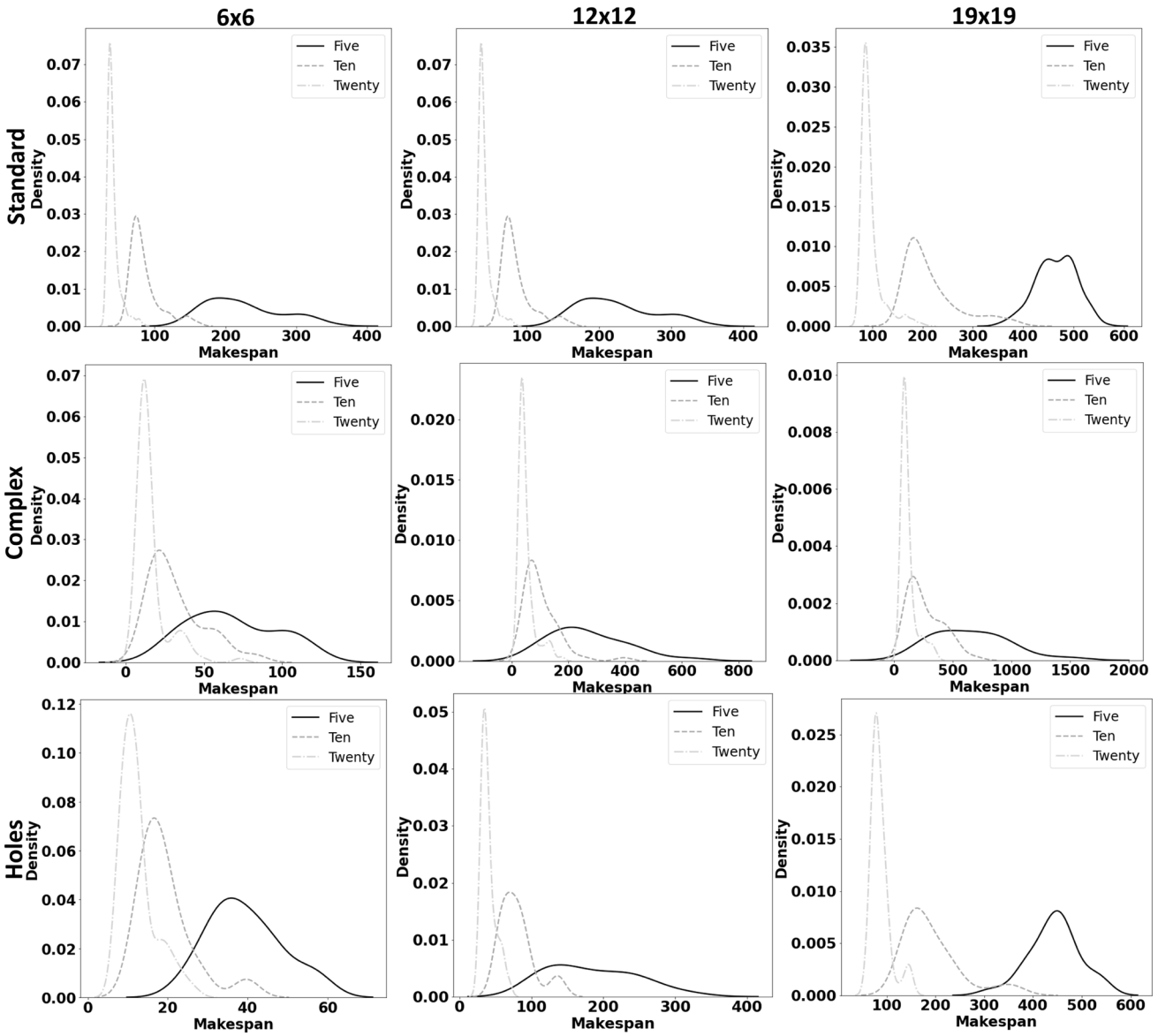


Fig. 2 Makespan distribution of 6x6, 12x12, and 19x19 task grids with team sizes of 5, 10, and 20 across the three datasets

4.1 Deep Learning Architectures for Performance Prediction

We tackle the problem of learning to predict team performance using a *Long short-term memory* network (LSTM). LSTMs are a form of *recurrent neural networks* (RNN), which in turn are a generalization of feed-forward neural networks with internal memory. Unlike feed-forward neural networks, RNNs can process sequential inputs using their internal state (memory). This property allows the usage of RNNs for ordinal or temporal problems, such as language translation, natural language processing, speech recognition, and image captioning. However, RNNs have shortcomings in tasks that require learning long-term temporal dependencies. This limitation is because the gradient of the loss function in RNNs decays exponentially with time (vanishing gradient problem) [41]. LSTMs overcome this main limitation of general RNNs [56].

In particular, an LSTM is a specific RNN architecture that models temporal sequences and their long-range dependencies more accurately than conventional RNNs. The central role of an LSTM model is performed by a

memory cell known as a *cell state* that updates its state over time. Specialized gates can add or remove information from the cell state. There are three types of gates: forget gate, input gate, and output gate. The forget gate decides what information should be discarded or kept in the cell. The input gate updates the cell state, and the output gate decides the next hidden state. LSTMs utilize these memory gates to forget task-irrelevant information while retaining the task-relevant information in the hidden states generated by the output gate, thereby providing a better memory span than RNNs [56].

We use LSTMs for predicting the performance $\hat{f}_{A,E}$ of a team of agents A operating in a 2-D grid environment E. More generally, LSTMs can process sequences of variable length while maintaining an internal memory that captures dependencies among sequence elements. This property is particularly useful in our setting, where team performance depends on the combined contributions of multiple agents. In our formulation, each agent is associated with a task grid, and the agent-specific task grids are processed sequentially by the LSTM. The hidden and cell states provide a compact representation of the information accumulated from previously processed agents, allowing the network to retain and leverage dependencies among their respective contributions when estimating overall team performance.

Specifically, at each timestep the LSTM processes the task grid of a single agent together with the internal state summarizing the task grids of previously processed agents. Although the resulting sequence does not correspond to a temporal evolution, the recurrent memory mechanism enables information from earlier agents in the sequence to influence the processing of later agents. Consequently, the network can capture interactions among agent contributions and generate a nuanced prediction of team performance.

Because the inputs to the network are agent-specific task grids defined over a 2-D environment, we employ ConvLSTM cells [44], which combine the sequence-modeling capabilities of LSTMs with convolutional operations that capture spatial structure. The convolutional layers are particularly well suited for learning the 2-D spatial efficacy distributions encoded in the task grids, as well as the spatial correlations they contain. As shown in Section 5.1, the use of ConvLSTMs significantly improves team-performance estimation.

The proposed LSTM model predicts team performance metrics from flattened task efficacy grids of the agents stacked sequentially. The model consists of one LSTM layer followed by four dense layers. The LSTM layer consists of 256 internal units and an input size equal to 400 (i.e., flattened 20x20 agent task efficacy grids). Smaller task efficacy grids of the agents are padded with 0s and flattened to ensure a uniform input size of 400 to the LSTM model. This layer is followed by a dropout layer ($p = 0.2$). The dropout property sub-samples the neurons of a layer during training, preventing model over-fitting. Three dense layers follow the LSTM layer, having 256, 126, and 64 units. The dense layers utilize the ReLu activation function. Finally, the output is a dense layer with a single neuron activated by a linear function. The output represents the target measure of team performance. We use a sequence of dense layers to add depth to the network and gradually condense the feature information extracted by the LSTM layer into team performance while enabling the model to learn intricate non-linear regression curves. Note that the prediction problem is a *regression* one, since makespan covers a large range of discrete values, and networking metrics are continuous.

Flattening task efficacy grids to make them compatible with the LSTM model proposed earlier may lead to the loss of spatial information embedded in the grid. To incorporate and learn the spatial pattern in the task efficacy grids of agents, we propose a ConvLSTM model with a structure similar to the LSTM model. The critical difference is that the LSTM layer is replaced by the ConvLSTM layer, which has 32 kernels of dimensions 3x3. For the ConvLSTM, we pass 2-D grids of agents (20x20) as input to the model. Additionally, a batch normalization layer is added after flattening for training stability. A series of dense layers follow the ConvLSTM block as in the LSTM architecture. We do not use additional ConvLSTM layers in our architecture to minimize computational costs and inference times, to showcase effectiveness and efficiency over conventional methods like CoLoSSI. Figure 3 provides a comprehensive visual summary of both neural network architectures. We predict five different metrics (i.e., makespan and four networking metrics) using both architectures. Note that the proposed architectures

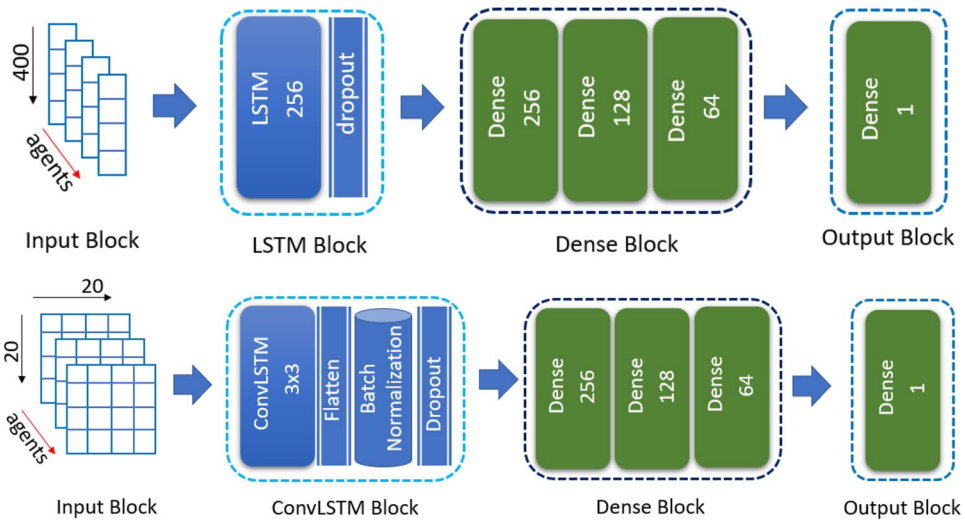


Fig. 3 Architecture of the LSTM model (top) and ConvLSTM (bottom)

can be extended to predict other metrics of interest related to measuring team performance by training with different ground truths.

4.2 Loss functions

This section briefly explains the loss function used to evaluate the performance of the proposed neural networks and drive the learning process for team performance estimation.

Mean Absolute Error (MAE): MAE quantifies the average absolute difference between the estimated team performance and CoLoSSI computed ground truth values, providing a straightforward and interpretable measure of the network's accuracy in predicting team performance. MAE is defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

where y_i represents the actual ground truth team performance, $\hat{y}_i$ is the predicted value, and n is the number of samples.

Percentage Mean Absolute Error (%MAE): One limitation of MAE is that it only captures the average deviation from the ground truth, disregarding the magnitude of the ground truth values. This limitation becomes apparent in scenarios where, for instance, an absolute error of 10 points with a ground truth of 20 signifies a markedly inferior estimation compared to a similar error when the ground truth is 200. To mitigate this, %MAE systematically normalizes the absolute error relative to the scale of the ground truth values. By expressing the error as a percentage, %MAE provides a standardized and interpretable loss metric across diverse scenarios (i.e., scales of ground truth). %MAE is defined as follows:

$$\%MAE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|y_i - \hat{y}_i|}{|y_i|}\right) \tag{2}$$

We utilize MAE as the loss function for training and validation sets. Weights of the network that result in minimal MAE on the validation set are saved for evaluation on the hold-out test set. We utilize the % MAE metric to evaluate the test set. %MAE accounts for the scale of the team performance metric, which may vary significantly with task grid size and number of agents in a team, making it easier to interpret across different datasets and team sizes.

Table 3 Performance of LSTM Model on different metrics

Metric	Val MAE	Train MAE	Test %MAE	SD Test %MAE
Makespan	6.52	5.75	8.59	7.09
M0	0.09	0.29	6.80	4.71
M1	0.09	0.24	7.05	4.67
M2	0.07	0.22	6.36	4.77
M3	0.06	0.14	6.26	3.37

Table 4 Performance of ConvLSTM Model on different metrics

Metric	Val MAE	Train MAE	Test %MAE	SD Test % MAE
Makespan	4.05	4.59	5.47	5.14
M0	0.04	0.04	3.09	2.69
M1	0.02	0.02	1.83	1.80
M2	0.02	0.02	2.13	2.86
M3	0.01	0.01	1.1	1.76

4.3 Implementation Details

The neural network architectures were designed, trained, and evaluated using the Keras framework (Version 2.3.0) with Tensorflow (v2.4.1) as the backend. The workstation utilized to train the network is an Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz CPU (48 CPU Cores) with 4 Nvidia Tesla K80 GPUs with 12GB RAM. We quantify the discrepancy between the network's estimated team performance and the ground truth using the MAE loss function, which is minimized using the Adam optimizer with a learning rate of 0.0005. For robust evaluation of our models and to overcome data biases, we employ five-fold cross-validation (100 epochs per fold). All reported results in Section 5 are the average of the five folds.

5 Results and Discussion

This section analyzes the performance of the LSTM and ConvLSTM models and explores their generalizability in several scenarios, highlighting their feasibility and effectiveness for estimating team performance on grids with varying team composition and topological features.

5.1 Performance Comparison of Networks Predictions vs. CoLoSSI

We have evaluated the performance of the LSTM and the ConvLSTM model on five different metrics (makespan, four networking metrics Table 1). Tables 3 and 4 show the training set (train) MAE, validation set (val) MAE, testing set (test) %MAE, and standard deviation of test %MAE across the five-folds for the LSTM and ConvLSTM models, respectively. The observed error values suggest that both LSTM and ConvLSTM models can learn the different team performance metrics. The LSTM model performs decently on all five metrics, achieving 8.59 %MAE for the makespan and a slightly smaller %MAE error on the networking metrics. These results also indicate that the LSTM cells in the first layer of the model can extract dependencies regarding the agent's contributions even from the flattened 1D representations of the square grids. Comparing the ConvLSTM model with the LSTM model, we see that the ConvLSTM model obtains a significant decrease in test %MAE for the team performance metrics. Specifically, ConvLSTM provides ≈3% improvement for makespan and M0, ≈4% improvement for M2, and ≈5% improvement for M1 and M3, in terms of %MAE. Furthermore, the performance of the ConvLSTM is relatively consistent across the five-folds, resulting in ≈2% lower standard deviation of test %MAE compared

to the LSTM. The superior performance of the ConvLSTM architecture can be attributed to the convolutional kernels embedded in the initial ConvLSTM layer. These 2D kernels effectively capture the spatial distribution of the efficacies of the different robots in the team and extract the relevant features necessary for predicting the makespan and the networking metrics.

5.2 Generalization of ConvLSTM Over Different Grids and Dataset Sizes

One central goal of learning is to obtain generalization of results beyond the training data. In our case, we want to be sure that the proposed models can generalize on environments presenting different characteristics.

As a first test, we consider *unseen square grid dimensions not present in the train set.* To conduct this study, we trained the model without instances of 7x7, 13x13, and 18x18 grids. Table 5 shows the performance of the LSTM and ConvLSTM models on the unseen grid sizes. Results suggest that the LSTM model cannot generalize over unseen grid sizes because of the high % MAE on 7x7, 13x13, and 18x18 grids. This behavior of the model can be due to the type of input the LSTM layer receives. Since the input to the model is a sequence of 1-D arrays (robot contributions), the model cannot extract the spatial information embedded within the grids. The spatial information and the robot's contributions could assist the model in generalizing over the unseen grids.

On the other hand, the results in Table 5 show that ConvLSTM achieves results comparable to those reported in the previous experiment (Table 4). We observe that for small grid sizes (e.g., 7x7), the test % MAE for ConvLSTM is slightly higher than before. This observation on the 7x7 grid is due to the small makespan values, leading to a large % MAE due to the normalization factor as shown in Eq. 2. Given the superior generalizability of ConvLSTM, we exclusively utilize this architecture for the remaining experiments conducted in this section.

In the next experiment, we train ConvLSTM on a dataset containing *grids of different rectangular dimensions*. The model achieves test % MAE of 6.61 and SD test % MAE of 5.38 on this rectangular grids dataset, providing comparable performance ($\approx$ one percent higher % MAE) to the square grids dataset. The slightly higher % MAE on the rectangular grid dataset could be attributed to a diverse distribution of agent efficacies, which was restricted to a square shape in the square grid dataset. Nevertheless, the results indicate that the ConvLSTM can predict makespan on grids of different rectangular shapes.

Another crucial aspect of neural network generalizability is their performance under *limited training dataset scenarios*. This is relevant for the case of estimating team performance and networking metrics because the ground-truth solvers have exponential time complexity, making it computationally challenging to accumulate datasets with a large number of instances for the bigger grid sizes. We validate the generalizability and performance variation of ConvLSTM architecture on a dataset containing both square and rectangular grids by controlling the number of instances corresponding to each grid in the training set. Tables 6, 7, and 8 show the performance of the ConvLSTM on the makespan and networking metrics by restricting the instances per grid size to 1020, 500, and 100, respectively. In all three cases, train and val MAE losses are comparable, suggesting minimal overfitting. Analysis of the networking metrics M0-M3 reveals that % MAE and SD % MAE increase by 1-1.5% as the number of instances decreases from 1020 to 100, suggesting that the ConvLSTM model can provide decent performance in data-constrained scenarios.

We also notice from the results in Tables 7, and 8 that % MAE and SD % MAE decrease (i.e., improved performance) as the dataset is shrunk to 500 instances per grid size and then increases after a further reduction to 100

Table 5 Evaluation of neural networks on makespan metric on novel grid sizes that were not utilized during training

	7x7 Grid		13x13 Grid		18x18 Grid	
Model	%MAE	SD %MAE	%MAE	SD %MAE	%MAE	SD %MAE
LSTM	63.55	26.75	44.15	18.50	29.60	15.3
ConvLSTM	9.58	7.37	4.47	3.49	6.48	4.34

Table 6 Evaluation of ConvLSTM on Makespan and networking metric on Square and Rectangular grids, each with 1020 instances

Metric	Val MAE	Train MAE	Test % MAE	SD Test % MAE
Makespan	7.23	6.88	7.54	10.02
M0	0.05	0.04	3.83	3.26
M1	0.03	0.03	2.63	2.37
M2	0.04	0.03	3.43	3.72
M3	0.02	0.02	1.97	2.36

Table 7 Evaluation of ConvLSTM on makespan and networking metric on square and rectangular grids, each with 500 instances

Metric	Val MAE	Train MAE	Test %MAE	SD Test % MAE
Makespan	5.31	5.05	6.82	6.05
M0	0.05	0.03	4.08	3.50
M1	0.03	0.02	2.78	2.50
M2	0.04	0.03	3.60	3.71
M3	0.03	0.02	2.54	3.03

Table 8 Evaluation of ConvLSTM on Makespan and networking metric on Square and Rectangular grids, each with 100 instances

Metric	Val MAE	Train MAE	Test % MAE	SD Test % MAE
Makespan	6.52	4.01	8.25	7.28
M0	0.06	0.02	4.75	4.25
M1	0.04	0.02	3.64	3.23
M2	0.05	0.02	4.74	4.87
M3	0.04	0.03	3.54	3.39

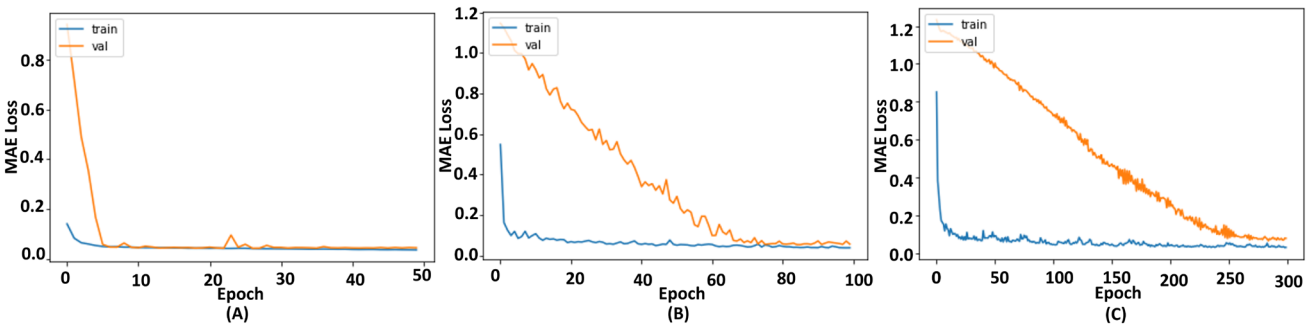


Fig. 4 Losses of ConvLSTM model when training with (A) 1020, (B) 500, and (C) 100 instances per grid in square and rectangular dataset

instances per grid size. This anomaly can be attributed to the random selection of instances for each grid size, potentially allowing for "easier" (i.e., with simple efficacy distribution) to dominate the dataset. Nonetheless, we observe % MAE and SD % MAE of 8.25% and 7.28, respectively, for the dataset with 100 instances, indicating acceptable performance in data-constrained settings.

Other important factors for neural networks in data-constrained scenarios include convergence and overfitting. Figure 4 shows the variations in training and validation losses across datasets with 1020, 500, and 100 instances. We observe that ConvLSTM does not overfit that data because of the convergence of the training and validation losses towards the end of training. However, it can be deduced that as the dataset size decreases, it takes more epochs for the network to converge, requiring longer training time and higher computational resources.

Table 9 Evaluation of ConvLSTM on makespan and network metrics on square grid sizes with inaccessible areas and 1000 instances per gird

Metric	Val MAE	Train MAE	Test % MAE	SD Test % MAE
Makespan	7.38	6.32	10.06	9.68
M0	0.04	0.02	10.47	8.37
M1	0.01	0.009	13.62	14.64
M2	0.005	0.003	43.72	49.92
M3	0.008	0.004	24.68	27.04

Table 10 Evaluation of ConvLSTM on Makespan and network metrics on square grid sizes with complex efficacy distribution

Metric	Val MAE	Train MAE	Test % MAE	SD Test % MAE
Makespan	21.92	15.36	16.34	16.64
M0	0.05	0.031	12.32	10.6
M1	0.03	0.02	19.12	20.9
M2	0.006	0.004	57.24	93.86
M3	0.01	0.01	38.58	53.32

Table 9 presents the performance of ConvLSTM on square grids containing inaccessible areas. We observe that the performance of ConvLSTM in terms of % MAE for the dataset with inaccessible areas is $\approx 2\times$ worse than the dataset containing complete square grids (as shown in Table 4). Nevertheless, ConvLSTM provides acceptable team performance estimation within $\approx$10% window of the CoLoSSI makespans. Interestingly, the performance hit on the networking metrics (i.e., M0-M3) is significantly worse. This performance degradation is attributed to the inaccessible areas significantly affecting the networking metric ground truth values. We conjecture the underlying reason is that inaccessible areas require the model to have a global understanding of the entire grid to capture the impact of these areas in the networking metric estimation.

5.3 Generalization of ConvLSTM over Different Task Models

Another pivotal aspect of this work is to examine the limits of the proposed neural network architectures for estimating team performance by varying the complexity of the agent efficacy distribution. Table 10 presents the results of the ConvLSTM architecture using a dataset comprising square grids with complex task models ($\frac{1}{512} \leq \varphi \leq 1$). Analysis of the train and validation losses for the makespan metric shows up to $3\times$ higher losses than other datasets. This suggests that learning over complex task models is a difficult task. Despite higher MAE losses, ConvLSTM performs well for the makespan, M0, and M1 metrics (i.e., within 20% of the ground truth). Unfortunately, test % MAE for M2 and M3 metrics are significantly higher, suggesting insufficient generalizability of the ConvLSTM architecture for these metrics. Additionally, the standard deviation for M2-M3 metrics is significantly higher, indicating inconsistent performance of the network across the different folds. Examining the M2 and M3 distribution in the dataset revealed a characteristic long-tail distribution pattern. We conjecture that this is because robot trajectories in mission plans with complex task models result in sparse trajectories. Consequently, this sparsity results in a wider range of values for the node centrality (M2) and algebraic connectivity (M3) metrics compared to the average connected component size (M0) and node degree (M1). These insights highlight the importance of robust priors for LSTM-based architectures, particularly when confronted with long-tail distributed metrics.

5.4 Estimation of Optimal Makespan using Transfer Learning

In this section, we investigate the utility of using *transfer learning* for learning a better model. In principle, we would like to train our model using a dataset where the labels are obtained using an optimal (or near-optimal)

solver. Unfortunately, due to the NP-hardness of the problem at hand, collecting extensive labeled examples from an optimal solver would require significant time and computational resources. Instead, using a heuristic and relatively fast solver, such as CoLoSSI, it is possible to build a relatively large labeled dataset, yet not featuring optimal labels.

However, with transfer learning we might obtain the desired result of training a near-optimal predictor: we first train with a large yet sub-optimal dataset, then we fine-tune it with a small dataset of optimal labels, and eventually we get a near-optimal prediction model. Here we conduct an experiment to precisely demonstrate (i) that neural networks can approximate optimal team performance, and (ii) that transfer learning can be effectively employed to transfer learned features from a simpler sub-optimal solver to train the model for optimal solvers. By doing so, we aim to reduce the required training set size and computational burden while maintaining performance estimation accuracy.

To this end, we fine-tune and evaluate the ConvLSTM architecture using the labels from the near-optimal CPLEX solver [22], initially pre-trained on CoLoSSI-estimated makespan ground truth. The results in Table 11, show that ConvLSTM can accurately estimate the optimal makespan across various scenarios on a dataset with grid sizes of 6x6, 8x8, 10x10, and 12x12. Training the ConvLSTM architecture from scratch on the optimal makespan ground truth with 1000 instances per grid results in an acceptable test % MAE and SD test % MAE of 13.92 and 13.66, respectively. Remarkably, fine-tuning with only 100 instances per grid, enables the ConvLSTM to attain good accuracy while necessitating 10× less data compared to training from scratch. This indicates that the features extracted by the model trained with the CoLoSSI data significantly contribute to the network's proficiency in estimating optimal makespan with reduced data requirements. Furthermore, a progressive increase in the fine-tuning dataset size leads to a decrement in both test % MAE and SD test % MAE, eventually plateauing at approximately 1000 instances per grid size. Impressively, the optimally performing ConvLSTM model achieves a test % MAE of 9.64 and an SD test % MAE of 9.06, utilizing 1000 instances of optimal makespans per grid size. This exemplifies the model's robustness and potential applicability in practical, data-constrained environments.

5.5 Use of Trained Models for Selecting Team Composition

In the previous section we have shown that we can use the trained networks to well approximate team performance. Therefore, once, trained, we can issue predictions in a short, constant time. On the other hand, if we would run a simulation and use CoLoSSI to obtain a measure of performance, computational requirements would rapidly grow with problem size. This is shown in Fig. 5, which reports the average computation time of CoLoSSI for different grid sizes. Even though the elbow of this curve may vary to some degree depending on the CPU resources of the machine, it is evident from the plot that CoLoSSI requires long computation times and high resources to calculate the team performance. In contrast, estimation using LSTM and ConvLSTM trained models for makespan and networking metrics can be performed in constant time, making them highly efficient for large grid sizes in time- and computation-constrained environments. This efficiency renders them particularly suited for metaheuristic or

Table 11 Performance of the proposed network in a transfer learning scenario, predicting optimal Makespan for 6x6, 8x8, 10x10, and 12x12 grid sizes

Scenario	Val MAE	Train MAE	Test % MAE	SD Test % MAE
Learning Optimal Makespan from scratch	6.32	4.86	13.92	13.66
Transfer Learning Makespan from 1000 instances per grid	4.48	3.84	9.64	9.06
Transfer Learning Makespan from 500 instances per grid	4.68	3.63	9.96	9.10
Transfer Learning Makespan from 250 instances per grid	4.59	2.77	10.34	9.66
Transfer Learning Makespan from 100 instances per grid	5.86	2.03	13.20	12.14

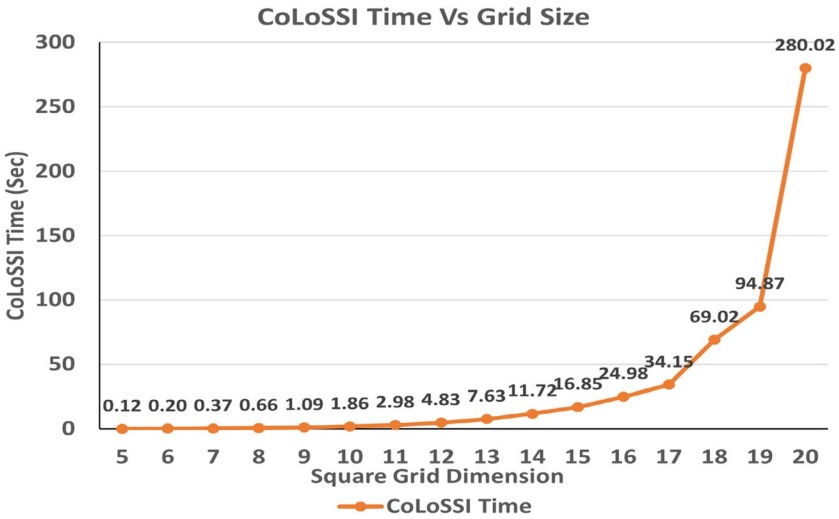


Fig. 5 Average computation time using CoLoSSI

sampling-based approaches that explore the combinatorial space of team compositions to find optimal or near-optimal team configurations.

6 Conclusions

In this study, we propose a new method for estimating team performance in multi-robot task allocation scenarios using neural network-based techniques. While previous works have focused on optimizing pre-defined teams, our approach addresses the challenge of designing teams to maximize mission success while minimizing costs. By predicting team performance based on a specific composition and allocation scheme, our method has significant implications for strategic team design. We use deep learning, specifically LSTM networks with convolutional layers, to capture complex interdependencies among agents and varying team sizes.

In order to test our approach, we used a diverse dataset of problem instances and labeled them based on mission makespan and network connectivity metrics using a fast but sub-optimal solver. Our results demonstrate that deep learning is effective in approximating team performance, with LSTM networks performing well across different problem scenarios and performance metrics. Furthermore, results showed that neural networks can approximate optimal team performance and transfer learning can seamlessly transfer learned features from sub-optimal solvers to train models for optimal solvers. The use of transfer learning allows us to bridge the gap between the computational limitations of getting labeled examples from optimal solvers and the necessity for precise performance estimation. These results lay the foundation for the real-time selection or design of the best multi-robot team for a given task, which offers a promising avenue for future research in this field.

Future work will investigate transformer-based architectures, such as ViTs, to better capture long-range spatial dependencies and assess their effectiveness in larger-scale mission environments. We also plan to explore graph convolutional networks (GCNs) to model richer inter-agent and task relationships that may improve robustness under complex team metrics and mission constraints. Finally, future studies will systematically evaluate scalability by examining larger team sizes, varying degrees of spatial correlation, and scenarios beyond the training distribution to better characterize the generalization limits of the proposed framework.

Acknowledgements This publication was partially made possible by the NPRP award 14S-0417-210227 from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Author Contributions M.Y.A. designed the methodology, conducted the experiments, interpreted the results, wrote the manuscript, and revised it. E.F.F. conceptualized the idea, verified the methodology, wrote the manuscript, and revised it.

M.I ran data simulations, analyzed the data, and verified methodology. M.Y ran data simulations, analyzed the data, and verified methodology. G.D.C conceptualized the idea, verified the methodology, revised the paper, and supervised the research.

Funding Open Access funding provided by Carnegie Mellon University.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

References

1. Agrawal, A., Bedi, A.S., Manocha, D.: RTAW: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 1393–1399. [Online]. Available: https://ieeexplore.ieee.org/document/10161310/
2. Almadhoun, R., Taha, T., Seneviratne, L., Zweiri, Y.: Multi-robot hybrid coverage path planning for 3d reconstruction of large structures, vol. 10, pp. 2037–2050. [Online]. Available: https://ieeexplore.ieee.org/document/9664539/
3. Ansari, I., Mohamed, A., Flushing, E.F., Razak, S.: 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), pp. 213–220 (2020)
4. Ansari, I., Mohammed, A., Ansari, Y., Ansari, M.Y., Razak, S., Flushing, E.F.: Colossi: Multi-robot task allocation in spatially-distributed and communication restricted environments. IEEE Access (2024)
5. Ansari, M.Y.M.S., Mourad, O., Qaraqe, K., Serpedin, E.: Deep learning for ecg arrhythmia detection and classification: An overview of progress for period 2017–2023. Front. Physiol. **14**, 1246746 (2023)
6. Ansari, M.Y., Chandrasekar, V., Singh, A.V., Dakua, S.P: Re-routing drugs to blood brain barrier: A comprehensive analysis of machine learning approaches with fingerprint amalgamation and data balancing. IEEE Access **11**, 9890–9906 (2022)
7. Ansari, M.Y., Mangalote, I.A.C., Masri, D., Dakua, S.P.: Neural network-based fast liver ultrasound image segmentation. In: 2023 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8 (2023)
8. Ansari, M. Y., Qaraqe, M., Charafeddine, F., Serpedin, E., Righetti, R., Qaraqe, K.: Estimating age and gender from electrocardiogram signals: A comprehensive review of the past decade. Artificial Intelligence in Medicine, p. 102690 (2023)
9. Ansari, M.Y., Qaraqe, M., Righetti, R., Serpedin, E., Qaraqe, K.: Unveiling the future of breast cancer assessment: A critical review on generative adversarial networks in elastography ultrasound. Front. Oncol. **13**, 1282536 (2023)
10. Banfi, J., Messing, A., Kroninger, C., Stump, E., Hutchinson, S., Roy, N.: Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance **7**(2), 4464–4471 (2022)
11. Bello, I., Pham, H., Le, Q. V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning **2**, 1–5. [Online] Available: arxiv.org/abs/1611.09940
12. Bengio, Y., Frejinger, E., Lodi, A., Patel, R., Sankaranarayanan, S.: A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Hebrard, E., Musliu, N. (Eds) Springer International Publishing, vol. 12296, pp. 99–111, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-58942-4_7
13. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: A methodological tour d'horizon **290**(2), 405–421. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0377221720306895
14. Bing, Z., Meschede, C., Röhrbein, F., Huang, K., Knoll, A.: C: A survey of robotics control based on learning-inspired spiking neural networks. Front. Neurorobot. **12**, 35 (2018)

15. Chakraa, H., Guérin, F., Leclercq, E., Lefebvre, D.: Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art **168**, 104492. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889023001318
16. Chandrasekar, V., Ansari, M. Y., Singh, A. V., Uddin, S., Prabhu, K. S., Dash, S., Al Khodor, S., Terranegra, A., Avella, M., Dakua, S. P.: Investigating the use of machine learning models to understand the drugs permeability across placenta. IEEE Access (2023)
17. Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., Song, L.: Learning combinatorial optimization algorithms over graphs. Advances in Neural Information Processing Systems **2017** Nips, pp. 6349–6359 (2017)
18. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., N.: Houlsby: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2021)
19. Feo-Flushing, E., Gambardella, L., Di Caro, G.A.: A mathematical programming approach to collaborative missions with heterogeneous teams. In: Proceedings of the 27th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 396–403 (2014)
20. Feo Flushing, E., Gambardella, L. M., Di Caro, G. A.: On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, ser. AAMAS '16. International Foundation for Autonomous Agents and Multiagent Systems, pp. 988–996, event-place: Singapore, Singapore
21. Feo Flushing, E., Gambardella, L. M., Di Caro, G. A.: On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams. In: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 988–996. [Online]. Available: http://dl.acm.org/citation.cfm?id=2937029.2937070
22. Feo-Flushing, E., Gambardella, L. M., Di Caro, G. A.: Spatially-distributed missions with heterogeneous multi-robot teams. IEEE Access **9**, pp. 67 327–67 348 (2021)
23. Flushing, E.F., Gambardella, L.M., Di Caro, G.A.: Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2017)
24. Gao, P., Siva, S., Micciche, A., Zhang, H.: Collaborative scheduling with adaptation to failure for heterogeneous robot teams. In: 2023 IEEE International Conference on Robotics and Automation (ICRA).IEEE, pp. 1414–1420. [Online]. Available: https://ieeexplore.ieee.org/document/10161502/
25. Garaffa, L. C., Basso, M., Konzen, A. A., De Freitas, E. P.: Reinforcement learning for mobile robotics exploration: A survey **34**(8), 3796–3810. [Online]. Available: https://ieeexplore.ieee.org/document/9612713/
26. Hochreiter, S., Schmidhuber, J.: Long short-term memory **9**(8), 1735–1780. publisher: MIT Press
27. Hu, G., Li, H., Liu, S., Zhu, Y., Zhao, D.: NeuronsMAE: A novel multi-agent reinforcement learning environment for cooperative and competitive multi-robot tasks. In: 2023 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/10191291/
28. Hu, J., Niu, H., Carrasco, J., Lennox, B., Arvin, F.: Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning **69**(12), 14 413–14 423. [Online]. Available: https://ieeexplore.ieee.org/document/9244647/
29. Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., Talbi, E.-G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art **296**(2), 393–422. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0377221721003623
30. Li, B., Wu, G., He, Y., Fan, M., Pedrycz, W.: An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem **9**(7), 1115–1138. [Online]. Available: https://ieeexplore.ieee.org/document/9812526/
31. Li, J.: Tan, Y: A two-stage imitation learning framework for the multi-target search problem in swarm robotics. Neurocomputing **334**, 249–264 (2019)
32. Li, Q., Gama, F., Ribeiro, A., Prorok, A.: Graph neural networks for decentralized multi-robot path planning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 11 785–11 792 (2020)
33. Lu, H., Zhang, X., Yang, S.: A learning-based iterative method for solving vehicle routing problems. In: International conference on learning representations (2019)
34. Ma, Z., Luo, Y., Pan, J.: Learning selective communication for multi-agent path finding **7**(2), 1455–1462. [Online]. Available: https://ieeexplore.ieee.org/document/9665278/
35. Mazyavkina, N., Sviridov, S., Ivanov, S., Burnaev, E.: Reinforcement learning for combinatorial optimization: A survey **134**, 105400. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0305054821001660
36. Nair, R.S., Supriya, P.: Robotic path planning using recurrent neural networks. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, pp. 1–5 (2020)
37. Niu, Y., Paleja, R., Gombolay, M.: Multi-agent graph-attention communication and teaming. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, ser. AAMAS '21. International Foundation for Autonomous Agents and Multiagent Systems, pp. 964–973, event-place: Virtual Event, United Kingdom

38. Nunes, E., Manner, M., Mitiche, H.: Gini, M: A taxonomy for task allocation problems with temporal and ordering constraints. Robot. Auton. Syst. **90**, 55–70 (2017)
39. Portugal, D., Rocha, R.: P: Performance estimation and dimensioning of team size for multirobot patrol. IEEE Intell. Syst. **32**(6), 30–38 (2017)
40. Rai, P., Ansari, M.Y., Warfa, M., Al-Hamar, H., Abinahed, J., Barah, A., Dakua, S.P., Balakrishnan, S.: Efficacy of fusion imaging for immediate post-ablation assessment of malignant liver neoplasms: A systematic review. Cancer Med. (2023)
41. Ribeiro, A.H., Tiels, K., Aguirre, L.A., T.: Schön: Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness. In: International conference on artificial intelligence and statistics. PMLR, pp. 2370–2380 (2020)
42. Sartoretti, G., Kerr, J., Shi, Y., Wagner, G., Kumar, T. K. S., Koenig, S., Choset, H.: PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning **4**(3), 2378–2385. [Online]. Available: https://ieeexplore.ieee.org/document/8661608/
43. Seraj, E., Chen, L., Gombolay, M. C.: A hierarchical coordination framework for joint perception-action tasks in composite robot teams **38**(1), 139–158. [Online]. Available: https://ieeexplore.ieee.org/document/9501958/
44. Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., Woo, W.-C.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. Adv. Neural. Inf. Process. Syst. **28** (2015)
45. Song, Z., Zhang, R., Cheng, X.: HELSA: Hierarchical reinforcement learning with spatiotemporal abstraction for large-scale multi-agent path finding. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 7318–7325. [Online]. Available: https://ieeexplore.ieee.org/document/10342261/
46. Tan, A. H., Bejarano, F. P., Zhu, Y., Ren, R., Nejat, G.: Deep reinforcement learning for decentralized multi-robot exploration with macro actions **8**(1), 272–279. [Online]. Available: https://ieeexplore.ieee.org/document/9963690/
47. Tolstaya, E., Gama, F., Paulos, J., Pappas, G., Kumar, V., Ribeiro, A.: Learning decentralized controllers for robot swarms with graph neural networks. In: Conference on robot learning. PMLR, pp. 671–682 (2020)
48. Tzes, M., Bousias, N., Chatzipantazis, E., Pappas, G. J.: Graph neural networks for multi-robot active information acquisition. In: 2023 IEEE International Conference on Robotics and Automation (ICRA).IEEE, pp. 3497–3503. [Online]. Available: https://ieeexplore.ieee.org/document/10160723/
49. Vinyals, O., Fortunato, M., N.: Jaitly: Pointer networks. In: Advances in Neural Information Processing Systems, pp. 2692–2700 (2015)
50. Wang, H., Lu, B., Li, J., Liu, T., Xing, Y., Lv, C., Cao, D., Li, J., Zhang, J., Hashemi, E.: Risk assessment and mitigation in local path planning for autonomous vehicles with LSTM based predictive model **19**(4), 2738–2749. [Online]. Available: https://ieeexplore.ieee.org/document/9440678/
51. Wang, Z., Liu, C., Gombolay, M.: Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints **46**(1), 249–268. [Online]. Available: https://link.springer.com/10.1007/s10514-021-09997-2
52. Yang, C., Peng, G., Li, Y., Cui, R., Cheng, L.: Li, Z: Neural networks enhanced adaptive admittance control of optimized robot-environment interaction. IEEE Trans. Cybernet. **49**(7), 2568–2579 (2018)
53. Yu, G., Wolf, M. T.: Congestion prediction for large fleets of mobile robots. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 7642–7649. [Online]. Available: https://ieeexplore.ieee.org/document/10161554/
54. Yu, J., Su, Y., Liao, Y.: The path planning of mobile robot by neural networks and hierarchical reinforcement learning. Front. Neurorobot. **14**, 63 (2020)
55. Zhang, H., Zhang, X., Feng, Z., Xiao, X.: Heterogeneous multi-robot cooperation with asynchronous multi-agent reinforcement learning **9**(1), 159–166. [Online]. Available: https://ieeexplore.ieee.org/document/10301527/
56. Zhao, J., Huang, F., Lv, J., Duan, Y., Qin, Z., Li, G., Tian, G.: Do rnn and lstm have long memory? In: International Conference on Machine Learning. PMLR, pp. 11 365–11 375 (2020)
57. Zhao, K., Liu, S., Yu, J. X., Rong, Y.: Towards feature-free TSP solver selection: A deep learning approach. In: 2021 International Joint Conference on Neural Networks (IJCNN).IEEE, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9533538/

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Mohammed Yusuf Ansari is a Postdoctoral Fellow in Computer Science at Carnegie Mellon University in Qatar. His research focuses on developing efficient deep learning architectures for biomedical image and signal analysis, with an emphasis on practical AI solutions for resource-constrained healthcare environments.

Mohammed Ishaq is currently pursuing his Ph.D. in artificial intelligence at the University of Melbourne, Australia. His research interests include deep learning, generative AI, computer vision, and developing intelligent AI-driven solutions to real-world problems.

Mohammed Yaqoob is currently pursuing his Ph.D. at the University of Melbourne, Australia, in the field of geomatics. His research interests include machine learning, deep neural networks, computer vision, and their applications to real-world challenges in geospatial computing and intelligent navigation.

Authors and Affiliations

Mohammed Yusuf Ansari[1] · Eduardo Feo Flushing[1] · Mohammed Ishaq[2] · Mohammed Yaqoob[2] · Gianni A. Di Caro[1]

✉ Mohammed Yusuf Ansari
ma1@alumni.cmu.edu

Eduardo Feo Flushing
efeoflus@andrew.cmu.edu

Mohammed Ishaq
m.ansari1@qatar.tamu.edu

Mohammed Yaqoob
yansari@tamu.edu

Gianni A. Di Caro
gdicaro@cmu.edu

[1] Carnegie Mellon University in Qatar, Doha, Qatar

[2] University of Melbourne, Melbourne, VIC, Australia