IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Graph Neural Network for Decentralized Multi-Robot Goal Assignment

Manohari Goarin, *Graduate Student Member, IEEE*, and Giuseppe Loianno, *Member, IEEE*

***Abstract*—The problem of assigning a set of spatial goals to a team of robots plays a crucial role in various multi-robot planning applications including, but not limited to exploration, search and rescue, or surveillance. The Linear Sum Assignment Problem (LSAP) is a common way of formulating and resolving this problem. This optimization problem aims at assigning the tasks to the robots minimizing the sum of costs while respecting a one-to-one matching constraint. However, communication restrictions in real-world scenarios pose significant challenges. Existing decentralized solutions often rely on numerous communication interactions to converge to a conflict-free and optimal solution or assume a prior conflict-free random assignment. In this paper, we propose a novel Decentralized Graph Neural Network approach for multi-robot Goal Assignment (DGNN-GA). We leverage a heterogeneous graph representation to model the inter-robot communication and the assignment relations between goals and robots. We compare in simulation its performance to other decentralized state-of-the-art approaches. Specifically, our method outperforms popular state-of-the art approaches in strictly restricted communication scenarios and does not rely on any initial conflict-free guess compared to two other algorithms. Finally, the DGNN-GA is also deployed and validated in real-world experiments.**

***Index Terms*—Task and motion planning, integrated planning and learning, deep learning methods.**

I. Introduction

MULTI-ROBOT systems have gained popularity in recent years due to their ability to speed up the execution of several tasks compared to single robot solutions, while concurrently offering additional resilience to robot failures. Multi-robot planning problems have been widely studied for exploration, flocking or formation control. They can be solved using centralized approaches [1], [2], [3], where a central computer calculates the solution for all robots, or decentralized approaches [4], [5], in which each robot computes its own policy given some local information about its environment and neighbors. While centralized approaches find optimal solutions, decentralization despite challenging becomes necessary for application to real-world scenarios with communication restrictions and to avoid single point of failures.

Manuscript received 25 October 2023; accepted 7 February 2024. Date of publication 28 February 2024; date of current version 21 March 2024. This letter was recommended for publication by Associate Editor J. Wang and Editor H. Kurniawati upon evaluation of the reviewers' comments. This work was supported in part by DARPA YFA under Grant D22AP00156-00, in part by NSF CPS under Grant CNS-2121391, in part by Qualcomm Research, Nokia, and in part by NYU Wireless. *(Corresponding author: Manohari Goarin.)*

The authors are with the Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: mg7363@nyu.edu; loiannog@nyu.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3371254, provided by the authors.
Digital Object Identifier 10.1109/LRA.2024.3371254

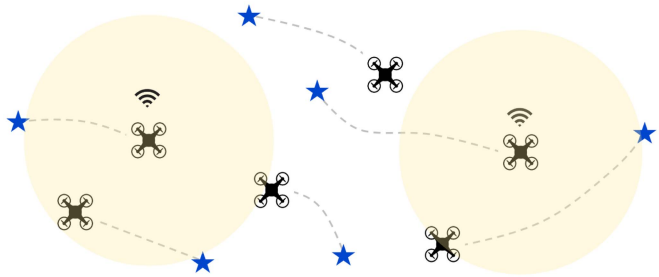

Fig. 1. Multi-robot Goal Assignment problem scenario with communication ranges for two robots represented by circles.

The Linear Sum Assignment Problem (LSAP) is a frequent problem in many of these applications. An optimal goal assignment as shown in Fig. 1 must minimize a given cost metric, such as the total distance traveled or time required for completion, while satisfying a one-to-one matching constraint. An efficient assignment is crucial to leverage the collaborative capabilities of multi-robot systems, expedite task completion, and reduce conflicts and collision risks.

Optimal solutions exist and are provided by well-known centralized algorithms such as the Hungarian algorithm [6]. However, these centralized methods are often impractical in real-world scenarios because they still rely on the global knowledge of the team's state. This is generally not available due to environment constraints or sparse communication among the agents, and results in a single point of failure. Therefore, this pushed to the design of decentralized solutions. For comprehensive reviews, the readers can refer to [7], [8], [9]. Many works explore decentralized optimization-based algorithms and market-based algorithms which often require numerous communication exchanges to guarantee the convergence to a consensus. Recently, deep learning methods as in [10], [11] have shown promising results in solving the LSAP using a Graph Neural Network (GNN) on a bipartite graph where edges connect the agent nodes to their possible task assignments. These techniques leverage the structure knowledge of the problem and optimize the information sharing between the nodes. However, they have not been designed to model and solve multi-robot systems problems with communication restrictions, and thus require full communication between all nodes of the graph.

In this work, we propose a novel approach for decentralized multi-robot LSAP using a GNN. We adopt a heterogeneous graph structure to model both the inter-robot communication links and the assignment relations between robots and goals. We design a GNN that achieves competitive performances by learning how to optimize the shared information. It demonstrates good zero-shot generalizability to large teams and greater

robustness in highly restricted communication scenarios compared to existing methods.

The main contributions of this paper are the following

- We design a novel GNN architecture, namely DGNN-GA, acting on a heterogeneous graph model to solve the decentralized LSAP for multi-robot teams.
- We analyze the performance and zero-shot generalizability of the proposed DGNN-GA with respect to the number of agents and the communication topology. Additionally, we investigate the influence of the number of communication rounds and its impact on the trade-off between performance and communication burden. We also conduct an ablation study to discern the roles of the different neural network architecture's components.
- We evaluate our solution in simulation and real-world settings for a general multi-robot planning objective minimizing the total distance squared traveled. We compare our solution with four other decentralized methods, namely a Local Hungarian algorithm (LH) [12], the D-CAPT pair-wise switching algorithm [4], the Consensus-Based Auction Algorithm (CBAA) [13], and the Decentralized-Based Hungarian Algorithm (DHBA) [14]. We show that DGNN-GA outperforms CBAA with an equal number of communication exchanges and improves the performance compared to DHBA when the number of communication rounds is minimal, while devoid of the strong assumption of an initial conflict-free guess contrary to LH and D-CAPT.

II. Related Works

Task or goal assignment is widely used in various multi-robot applications [9], [15] such as exploration [16], formation control [17] and search and rescue [18]. This section focuses on decentralized optimization-based, market-based, and centralized or decentralized learning-based approaches.

Optimization-based Methods: Several works like [14], [19], [20] propose decentralized versions of the Hungarian algorithm. After $\mathcal{O}(N^3)$ iterations, equivalent to $\mathcal{O}(N^3)$ communication rounds (N the number of agents) the robots are guaranteed to converge to an optimal solution, albeit with considerable communication requirements. Alternatively, heuristic approaches [21] and genetic algorithms [22] are also employed. Especially, the authors in [21] introduce a greedy algorithm for one-to-one assignment, guaranteeing a 2-approximation to the optimal with $\mathcal{O}(N)$ communication rounds. Multi-robot path planning applications also leverage multiple iterative methods that rely on online improvements of the assignment during the robots' motion, with one-hop communication round at each time step. In particular, D-CAPT [4] uses a pair-wise switching policy, and the authors in [12] apply the Hungarian algorithm locally every time a subset of robots communicate. These methods achieve near-optimal assignments in practice and are communication efficient. However, they rely on an initial conflict-free guess.

Market-based Methods: These approaches are mostly auction-based methods [13], [17], [23]. Each robot shares its list of task preferences along with bids, and the winning bid is determined once the team reaches a consensus. For example, the Consensus-Based Auction Algorithm (CBAA) [13] represents a baseline for multiple other auction algorithms due to its communication efficiency. CBAA guarantees to converge to a suboptimal conflict-free assignment after $\mathcal{O}(Nd)$ iterations (or communication rounds), where d represents the diameter (i.e., the longest path connecting two robots) of the network, but needs much less in practice.

Learning-based Methods: These approaches have been widely explored to address multi-robot task assignment and trajectory planning problems using imitation learning [24], [25], [26] or reinforcement learning [27], [28], [29]. Notably, Graph Neural Networks (GNNs) gained significant attention due to their intrinsic ability to model and process graph-structured data, composed of nodes and edges, using graph filters [30], [31], [32]. Naturally decentralized, they offer an effective representation for multi-robot systems with robot nodes and communication edges [33]. Through a message-passing framework, robots share information with their neighbors and make predictions locally. Successful applications include collaborative perception [25], [34], [35], [36], flocking [26], [37], path planning [24], [36] or formation control [38] problems.

Learning-based approaches have also been used to tackle the LSAP as demonstrated in works such as [39] or [10], [11], which employ GNNs on a bipartite graph model constructed from agent and task nodes. The GNN is trained in a supervised-learning fashion to predict the classes of nodes or edges, demonstrating competitive performances with compact architectures. However, these methods do not address multi-robot problems with communication restrictions which are typical in real-world settings for multi-robot systems. Conversely, in this paper, we leverage the GNN benefits to solve the challenging general LSAP problem in decentralized settings while considering communication restrictions.

III. Problem Definition

In the following, we denote column vectors with bold notation like $\mathbf{h}$, matrices with capital notation like $\mathbf{A}$, and scalars with unbold notation like c. Let $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$ be the set of robots, and $\mathcal{G} = \{g_1, g_2, \ldots, g_N\}$ be the set of goals. We assume an equal number of robots and goals (i.e., N). All robots are homogeneous and possess identical communication capabilities. We denote with c_{ij} the cost of the assignment of the robot r_i to the goal g_j, which is arbitrarily chosen depending on the application. The term s_{ij} is the assignment label between r_i and g_j that equals to 1 if r_i is assigned to g_j, and 0 otherwise. The variables i and j will be used to refer respectively to robots' indices and goals' indices. The LSAP is formulated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}s_{ij}, \qquad (1)\\ \text{subject to} \quad & \sum_{i=1}^{N} s_{ij} = 1, \quad \forall j \in \{1,\ldots,N\},\\ & \sum_{j=1}^{N} s_{ij} = 1, \quad \forall i \in \{1,\ldots,N\},\\ & s_{ij} \in \{0,1\}, \quad \forall (i,j) \in \{1,\ldots,N\}^2. \end{aligned}$$

To solve this centralized problem in a decentralized fashion, we build an heterogeneous graph represented in Fig. 2 from the network of robots and goals. The graph is composed of robot nodes $\{r_i\} \in \mathcal{R}$ and goal nodes $\{g_j\} \in \mathcal{G}$. We define two types of edges: $\mathcal{E}^{ass} = \{e_{ij}^{ass}\}_{1\leq i,j\leq N}$ is the set of assignment edges connecting each robot to its possible assigned goals,

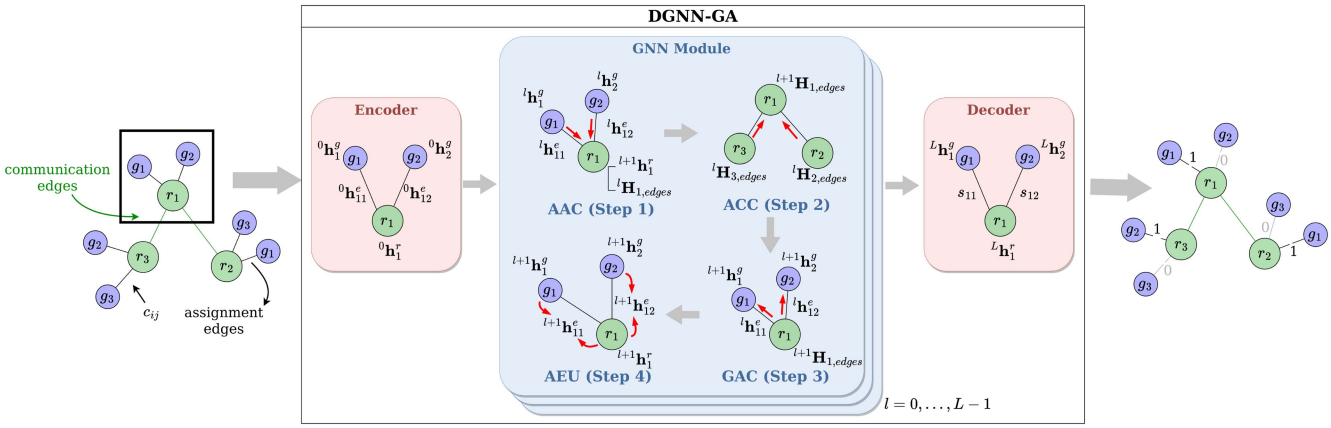

Fig. 2. DGNN-GA architecture, sample case with three robots and three goals with the GNN applied on robot r_i with $i = 1$ and its local graph. The information aggregation at each step is illustrated with red arrows.

whereas $\mathcal{E}^{com} = \{e_{ij}^{com}\}_{i,j \in 1,..,N}$ is the set of communication edges between robots. These depend on the communication availability between two robots (i.e., if they are within each other's communication range).

We denote $\mathcal{E}^{ass}(r_i)$ the set of assignment edges connected to r_i. Additionally, we define $\mathcal{N}_c(r_i)$ the set of neighboring robots that can communicate with r_i. Similarly, $\mathcal{N}_a(r_i)$ and $\mathcal{N}_a(g_j)$ represent the set of neighboring goals that can be assigned to robot r_i and the set of neighboring robots that can be assigned to goal g_j respectively as shown in Fig. 2. To characterize the connectivity of the communication topology of a team, we define the communication adjacency matrix $\mathbf{A}$, of size $N \times N$ whose components are

$$A_{ij} = \begin{cases} 1 & \text{if } r_i \text{ and } r_j \text{ can communicate} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

and the corresponding communication density $D_{com} = \frac{\sum A_{ij} - N}{N^2 - N}$. D_{com} is equal to zero if all robots are isolated, and to 100% if all robots can communicate with each other.

IV. DGNN-GA Architecture

In this section, we introduce the proposed DGNN-GA architecture design. It is decomposed in three modules (see Fig. 2): the **Encoder**, the **GNN module**, and the **Decoder** detailed below. For the purpose of predicting assignments, the optimization problem is solved via an edge binary classification framework. The DGNN-GA predicts a label for each assignment edge (see Fig. 2) namely 1 if a node is assigned to a specific goal and 0 otherwise. The nodes and assignment edges are embedded with feature vectors. We denote $\mathbf{h}_i^r$, $\mathbf{h}_j^g$, $\mathbf{h}_{ij}^e$ respectively for robot node r_i, goal node g_j, and edge e_{ij}^{ass}. In the following, the notation ϕ_* refers to multi-layer perceptron (MLP) functions with two linear layers and a ReLU activation function in between. Initially, the nodes' embeddings are zero vectors, and the assignment edges' equal to their corresponding cost: $\forall i, j,\ \mathbf{h}_{ij}^e = c_{ij}$.

Encoder: The encoder at the robot r_i transforms its assignment edges' costs into embedding vectors of size F

$$\forall e_{ij}^{ass} \in \mathcal{E}^{ass}, j \in \{j_1, \ldots, j_{M_i}\},\ {}^0\mathbf{h}_{ij}^e \leftarrow \phi_{enc}\left(c_{ij}^e\right), \tag{3}$$

where $M_i = |\mathcal{N}_a(r_i)|$. The superscript 0 indicates the initial value of the embeddings before the GNN Module is applied.

GNN Module: The GNN module is composed of a series of L (layers) node and edge convolutions. Through a message-passing framework, information is shared between neighbors L times. In other words, L corresponds also to the number of communication rounds among the robots.

One convolution at a node level is composed of a message function that creates a message from nodes and edges' embeddings ($\mathbf{m}_*$ or $\mathbf{M}_*$), an average function that is permutation invariant to aggregate the messages from other nodes and edges together ($\bar{\mathbf{h}}_*$ or $\bar{\mathbf{H}}_*$), and an update function that updates the nodes' embedding. Moreover, DGNN-GA uses generalized convolutions with nonlinear functions, which were used in diverse applications to improve the performance of a GNN [40], [41], [42]. At one layer l and at a robot node r_i level, we apply the following steps. The superscript l indicates the value of the embeddings at layer l.

Step 1: An agent assignment convolution (AAC) aggregates information from the goal nodes connected to r_i and updates r_i's embedding

$$\forall g_j \in \mathcal{N}_a(r_i),\ \mathbf{m}_{ji} = \phi_{ac1}\left({}^l\mathbf{h}_j^g || {}^l\mathbf{h}_{ij}^e\right), \tag{4}$$

$$\bar{\mathbf{h}}_i^r = \frac{1}{|\mathcal{N}_a(r_i)|} \sum \mathbf{m}_{ji}, \tag{5}$$

$${}^{l+1}\mathbf{h}_i^r \leftarrow \phi_{ac2}\left({}^l\mathbf{h}_i^r || \bar{\mathbf{h}}_i^r\right). \tag{6}$$

The notation $||$ refers to the concatenation operation at the channel dimension. Additionally, the assignment edges' information is concatenated at the node r_i's level in another embedding vector ${}^l\mathbf{H}_{i,edges}$ of size $F \times M_i$

$$\begin{aligned} {}^l\mathbf{H}_{i,edges} &= \begin{bmatrix} {}^l\mathbf{h}_{ij_1,edges} & \ldots & {}^l\mathbf{h}_{ij_{M_i},edges} \end{bmatrix} \\ &= \begin{bmatrix} \phi_{ac3}\left({}^l\mathbf{h}_i^r || {}^l\mathbf{h}_{ij_1}^e\right) & \ldots & \phi_{ac3}\left({}^l\mathbf{h}_i^r || {}^l\mathbf{h}_{ij_{M_i}}^e\right) \end{bmatrix}, \end{aligned} \tag{7}$$

where $g_{j_1}, \ldots, g_{j_{M_i}} \in \mathcal{N}_a(r_i)$. $\mathbf{H}_{i,edges}$ helps differentiate information related to different goals.

Step 2: An Agent Communication Convolution (ACC) aggregates information from neighboring agent nodes about their

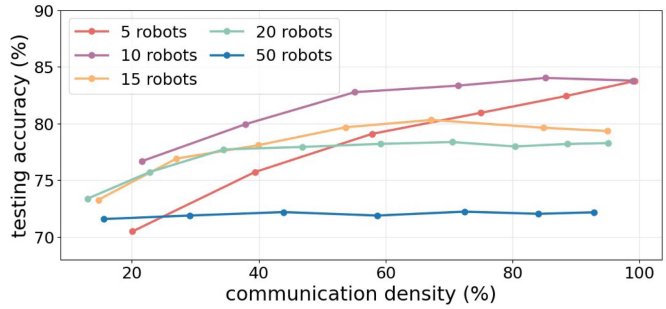

Fig. 3. Testing results when training with $N = 10$ and $L = 5$.

possible assignments

$$\forall r_{i'} \in \mathcal{N}_c(r_i),\ \mathbf{M}_{i'i} = f_{map}\left({}^{l}\mathbf{H}_{i',edges}\right), \tag{8}$$

$${}^{l+1}\mathbf{H}_{i,edges} \leftarrow \bar{\mathbf{H}}_{i,edges} = \frac{1}{|\mathcal{N}_c(r_i)|}\sum \mathbf{M}_{i'i}. \tag{9}$$

where f_{map} is a mapping function that retrieves information from ${}^{l}\mathbf{H}_{i',edges}$ about common goals of interests between r_i and $r_{i'}$.

Step 3: A Goal Assignment Convolution (GAC) updates the embedding of each g_j connected to r_i given $\mathbf{H}_{i,edges}$

$$\forall g_j \in \mathcal{N}_a(r_i),\ {}^{l+1}\mathbf{h}_j^g \leftarrow \phi_{gc}\left({}^{l}\mathbf{h}_j^g || {}^{l+1}\mathbf{h}_{ij,edges}\right). \tag{10}$$

Step 4: Assignment edges' embeddings are updated (AEU)

$${}^{l+1}\mathbf{h}_{ij}^e \leftarrow \phi_{eu}\left({}^{l}\mathbf{h}_{ij}^e || {}^{l+1}\mathbf{h}_i^r || {}^{l+1}\mathbf{h}_j^g\right), \tag{11}$$

Decoder: The decoder at r_i's level predicts the assignment edges' labels from their final embeddings and is designed as

$$\forall e_{ij}^{ass} \in \mathcal{E}^{ass},\ s_{ij} = \text{sigmoid}\left(\phi_{dec}\left({}^{L}\mathbf{h}_{ij}^e\right)\right). \tag{12}$$

V. Results and Analysis

A. GNN Training

The DGNN-GA is trained in a supervised fashion, imitating the Hungarian algorithm [6] in distributed fashion. To speed up the training, the model is trained on a graph where goal nodes are not duplicated for different robots and the communication adjacency matrix is used to enforce the communication restrictions in the convolutions. Then the model is deployed and executed as shown in Fig. 2. We build training datasets of 22000 graphs, each with a fixed number N of robots and goals and a fixed number L of communication rounds. The assignment costs are randomly initialized and communication topologies uniformly distributed between 20% (sparse communication) and 100% (full communication). We also employ 50 test sets of 1000 graphs each, for different team sizes (i.e., 5, 10, 15, 20, 50) and multiple communication densities (between 15% and 99%). For training, the number of assignment edges is limited to the top 5 for each goal to reduce the size of the graph without impacting the overall performance, since the optimal solution is likely to be found within the top 5 possibilities.

To achieve a binary classification task, we choose to minimize the Balanced Cross-Entropy (BCE) loss function

$$L_c = -\alpha \mathbf{y}^* \log \hat{\mathbf{y}} - (1-\alpha)(1-\mathbf{y}^*)\log(1-\hat{\mathbf{y}}), \tag{13}$$

TABLE I
Minimum Testing Accuracy in % Depending on the Number of Robots for Training N_T and Testing N_E

N_T \ N_E	5	6	7	8	9	10
5	79.03	77.51	76.31	72.29	66.09	70.47
10	71.04	73.35	74.85	76.78	75.70	76.68
15	67.78	69.96	70.56	72.78	72.05	73.26
20	68.08	69.43	69.58	72.33	71.79	73.38
50	66.21	68.27	67.85	69.67	69.87	71.57

where $\mathbf{y}^*$ is the flattened optimal assignment matrix $\mathbf{S}^*$ given by the Hungarian algorithm, and $\hat{Y}$ the flattened assignment matrix $\hat{\mathbf{S}}$ predicted by the DGNN-GA. We choose $\alpha = 0.9$ to account for the imbalance in the distribution of zero and one labels. Moreover, inspired by [10], we use an additional loss to help the neural network learn a one-to-one matching, satisfying the constraints in eq. (1)

$$L_m = \frac{1}{2}\left(\left|\left|\mathbf{1}_N - \sum_i \text{row}_i(\hat{\mathbf{S}})\right|\right| + \left|\left|\mathbf{1}_N - \sum_i \text{col}_i(\hat{\mathbf{S}})\right|\right|\right)$$
$$+\frac{1}{2}\left(\left|\left|\mathbf{1}_N - \begin{bmatrix}\|\text{row}_1(\hat{\mathbf{S}})\|\\ \vdots \\ \|\text{row}_N(\hat{\mathbf{S}})\|\end{bmatrix}\right|\right| + \left|\left|\mathbf{1}_N - \begin{bmatrix}\|\text{col}_1(\hat{\mathbf{S}})\|\\ \vdots \\ \|\text{col}_N(\hat{\mathbf{S}})\|\end{bmatrix}\right|\right). \tag{14}$$

The total loss to minimize is therefore

$$L = \beta L_c + (1-\beta)L_m, \tag{15}$$

where β weights the importance given to one loss with respect to the other. We empirically set $\beta = 0.8$. The lower is β the more the DGNN-GA will prioritize finding a one-to-one matching rather than an optimal solution. Finally, the accuracy metric chosen to evaluate the classification performance is the Binary F1-score, which is well-suited for class-imbalanced data. In the following subsections, all training runs and comparisons were conducted using the random seed 1215, a batch size of 200 graphs and 40 epochs on a 12$^{\text{th}}$ generation Intel CPU I9-12900H.

B. Performance and Generalizability Analysis

First, we train the model with $N = 10$ robots and $L = 5$ convolution rounds. After 2 hours, we achieve a training classification accuracy of 81.02%. We then evaluate the GNN on the test sets and obtain the results shown in Fig. 3.

The average testing accuracy on all possible communication densities is between 70% and 82% for 5 to 50 robots. Even with a communication density 20% or lower, the DGNN-GA is able to achieve an accuracy greater than 70%, which demonstrates that the heuristic learned by our DGNN-GA is robust to sparsely connected networks which is an extremely important property in multiple real-world settings and deployment cases. Moreover, the DGNN-GA also demonstrates zero-shot generalizability to larger teams of robots, up to 50, with an overall minimum accuracy of 70.47% and a fixed number of communication rounds $L = 5$.

To further analyze the generalizability properties of our solution, we train the DGNN-GA with different fixed numbers

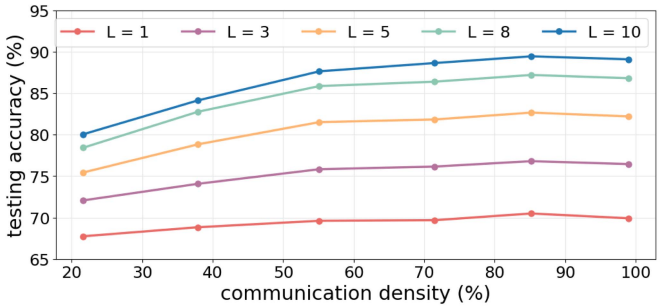

Fig. 4. Influence of L on the testing accuracy for $N = 10$.

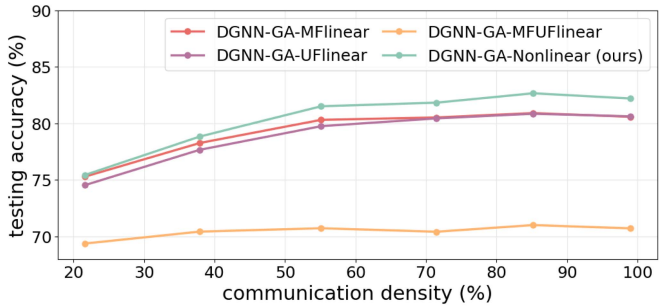

Fig. 5. Ablation study for $N = 10$ and $L = 5$.

of agents, from 5 to 10, and report in Table I the minimum accuracy obtained for each test set with 5 to 50 robots. In most cases, the minimum testing accuracy is obtained for 50 robots with a communication density lower than 20%. For large teams ($N \geq 10$), the minimum performance increases with the number of robots used for training. Therefore, a trade-off can be made between the team size for training and the zero-shot generalizability performance.

C. Influence of the Number of Convolution Rounds

We test several convolutions rounds ($L = 1, 3, 5, 8, 10$) on a 10 robots team. The results are presented in Fig. 4.

The number of layers defines the number of communication exchanges between the neighboring robots. As we can see, the performance increases with L. Therefore, its value strikes a balance between performance and communication burden. In any case, we notice that with only 5 rounds, the network reaches a 75% accuracy for sparse communication networks and more than 82% for well-connected networks.

D. Ablation Study

We conduct an ablation study on our DGNN-GA architecture. Specifically, we compare the testing accuracy obtained if we use linear functions instead of nonlinear MLPs for both update and message functions used in the convolutions performed in steps 1, 2, and 3 presented in Section IV. We compare and analyze our nonlinear DGNN-GA with several variants of the same architecture, where specific modules are replaced to clearly identify their benefits

- DGNN-GA-MFlinear: the message functions are replaced by one linear layer.
- DGNN-GA-UFlinear: the update functions are replaced by one linear layer.
- DGNN-GA-MFUFlinear: both the message and update functions are replaced by linear layers.

For comparison, all architectures are trained with 10 robots. Fig. 5 reports the testing accuracy across the different communication densities using $L = 5$ convolution rounds. Similar results are obtained for other values of L.

As we can see, nonlinear functions improve the expressiveness of our DGNN-GA. The classification performance is significantly better compared to DGNN-GA-MFUFlinear that has fully linear convolutions. DGNN-GA-UFlinear and DGNN-GA-MFlinear perform similarly and lead to a worse but closer performance to our DGNN-GA's performance. We can conclude that at least one nonlinear function is needed to get a satisfying result, and nonlinear functions for both message and update functions provide the best performances.

E. Simulation Results and Comparative Analysis

We evaluate our DGNN-GA in simulation for a common multi-robot goal assignment problem. We run 100 Monte Carlo simulations of duration $T = 10$ s with $N = 5, 10, 15, 20$ robots and goals whose positions are randomly initialized, respecting an initial safety relative distance of 1.0 m, and in variable communication topologies defined by different communication ranges. The cost to minimize in eq. (1) is chosen as the sum of distances traveled squared, thus $c_{ij} = d_{ij}^2$, where d_{ij} is the Euclidian distance between r_i and g_j. Indeed, distance costs are widely used in multi-robot applications, as in [4], [12], [16]. They have shown great utility to minimize the execution time of a mission and to reduce the risk of collision. We assume the robots move at a constant velocity of $v = 2$ m/s.

We compare our DGNN-GA with respect to four decentralized approaches commonly used in multi-robot goal assignment problems and that have shown great performances with limited communication exchanges

- The pair-wise switching policy (D-CAPT) [4]: Neighboring robots communicate by pair and switch their goals if the sum of their costs decreases. The assignment is improved iteratively during the robots' motion. The approach requires one communication round at each time step and an initial random assignment satisfying the one-to-one matching constraint.
- The Local Hungarian algorithm (LH) [12]: Similar to D-CAPT, the assignment is improved iteratively during the robots' motion. It performs one communication round at each time step and assumes an initial random one-to-one assignment. However, LH applies the Hungarian algorithm on each subset of neighboring robots.
- The Consensus-Based Auction Algorithm (CBAA) [13]: Conversely to LH and D-CAPT, CBAA applies multiple communication rounds at once, without any initial assumption. The robots start moving when a consensus is found. CBAA is a baseline state-of-the-art approach among auction algorithms due to its efficiency. It guarantees to converge to a consensus after $\mathcal{O}(Nd)$ communication rounds, but is much faster in practice.
- The Decentralized Hungarian-Based Algorithm (DHBA) [14]: DHBA also applies multiple communication rounds at once and does not require any initial assumption. In the algorithm, each robot buffers an estimation of the

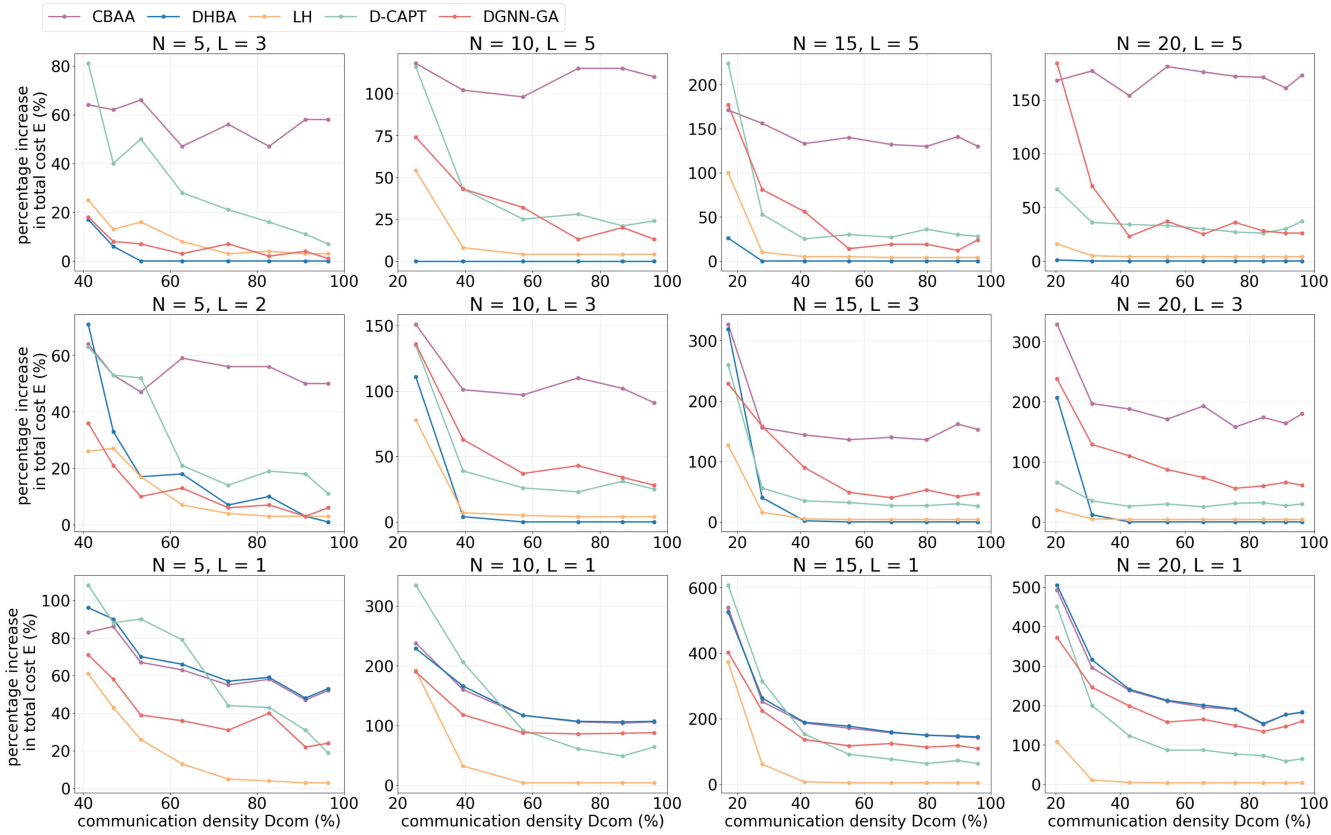

Fig. 6. Assignment costs comparisons using various assignment algorithms while varying the communication density D_{com}, the number of robots N, and the number communication rounds L.

global cost matrix, updates it with the neighbors' cost matrices at each communication round, and applies the Hungarian algorithm. After multiple rounds, all robots' cost matrices are equal and the algorithm converges to the optimal solution.

We compare the average total cost across variable communication densities and different L. Fig. 6 reports the error in $\%$ to the optimal solution, provided by the centralized Hungarian algorithm, i.e. the percentage increase in total cost

$$E(\%) = 100 \times \frac{\sum \hat{s}_{ij} c_{ij} - \sum s^*_{ij} c_{ij}}{\sum s^*_{ij} c_{ij}}, \tag{16}$$

where s^*_{ij} are the components of the optimal assignment matrix $\mathbf{S}^*$ given by the Hungarian algorithm, and $\hat{s}_{ij}$ the components of the predicted assignment matrix $\hat{\mathbf{S}}$ provided by the different decentralized solutions. The associated minimum and maximum values of the third quartile q are reported in Table II as it represents a more intuitive dispersion measure owing to the non-Gaussian data distribution. It corresponds to the value under which 75% of the errors are found. Minimum values are obtained in low connectivity (low L and D_{com}) and maximum values in high connectivity (high L and D_{com}). In general, a high dispersion of E about the average is observed for all methods, reflecting the diversity of the scenarios tested using our simulation setup, especially in low connectivity.

Before comparing and discussing the methods, we notice that the errors reported increase significantly over 100% when $N \geq 10$. The reason is that the errors accumulate with the number of robots, especially for low values of L and sparse communication densities ($D_{com} \leq 30\%$), where more conflicts occur and more robots need to be reassigned.

Comparison with LH and D-CAPT: LH converges quickly to the optimal solution when the network is well connected (i.e. $D_{com} \geq 40\%$), contrary to D-CAPT. Indeed, while the pair-wise strategy is efficient, it leads to more frequent reassignments than LH. DGNN-GA provides competitive results compared to D-CAPT with $L = 5$ communication rounds, but predicts worse assignments than LH and D-CAPT for lower L. Indeed, our method DGNN-GA is devoid of any one-to-one initial assumption and thus, as expected, needs more communication rounds to find a consensus. However, in practice, an initial one-to-one assignment represents a strong assumption since it requires some initial global knowledge of the team's state that is not available in many multi-robot applications. On the other hand, our approach, similarly to CBAA and DHBA in restricted communication as discussed in the following, does not always guarantee a conflict-free assignment because the one-to-one matching constraint is enforced as a soft constraint in the training loss. Regarding the dispersion of the errors in Table II, our DGNN-GA provides lower quartiles compared to D-CAPT in both low and high connectivity scenarios. For well-connected networks (high values of L and D_{com}) and $N \leq 15$, 75% of our DGNN-GA's assignments reach the optimal solution.

Comparison with CBAA and DHBA: DGNN-GA outperforms CBAA in every scenario in terms of average percentage increase and dispersion. Therefore, the heuristic learnt by the DGNN-GA, implicitely leveraging the structure knowledge

TABLE II
MINIMUM AND MAXIMUM THIRD QUARTILES $q(\%)$ OF THE PERCENTAGE INCREASE E FOR THE DIFFERENT ASSIGNMENT ALGORITHMS AND TEAM SIZES N

N		**DGNN-GA**	CBAA	DHBA	LH	D-CAPT
5	min	**0**	77	0	4	5
	max	**123**	144	160	72	154
10	min	**0**	130	0	5	25
	max	**298**	385	365	281	491
15	min	**0**	176	0	5	27
	max	**535**	751	750	503	756
20	min	**18**	194	0	4	30
	max	**489**	709	708	146	554

The min and max values are obtained respectively for low and high values of (L,D_{com}).

of the graph, improves the performance compared to CBAA with the same number of communication exchanges. Regarding DHBA method, it outperforms DGNN-GA for $L \geq 3$ and in that case converges to a near-optimal solution ($E \leq 10\%$ and $q = 0\%$) for $D_{com} \geq 40\%$. However, for $L < 3$, the average error of DHBA is similar to CBAA, but worse than DGNN-GA. Moreover, for low values of both L and D_{com}, DHBA generates higher quartiles. Therefore, DGNN-GA is better suited than DHBA for highly restricted communication scenarios. CBAA and DHBA can guarantee the convergence to a conflict-free assignment contrary to our DGNN-GA, but to achieve this goal they need a sufficient number of communication rounds depending on the topology of the network and the number of robots. However, in many real-world scenarios, we need to limit the number of communication rounds, because of time constraints to complete the task, or because of unpractical communication delays or drops due to environment constraints or interferences. In these cases, the conflict-free assignment is not guaranteed anymore for both CBAA and DHBA. In the simulation tests, the comparison is made considering only L communication iterations, and we leverage a conflict resolution module for CBAA, DHBA, and our DGNN-GA in case a consensus is not found. A greedy approach detects and handles locally a conflict between two neighboring robots. The further robot from the goal of interest is reassigned to the second goal with minimum cost. Besides, a random assignment policy was evaluated as a lower bound. Each robot chooses initially a random goal and conflicts are resolved using the same greedy approach. For team sizes from 5 to 20, the average error generated goes from 100% to 230% in high connectivity and explodes from 320% to 2080% in low connectivity.

As a conclusion, our DGNN-GA outperforms CBAA and DHBA in highly restricted number of communication rounds, and does not leverage any assumption of an initial one-to-one assignment available compared to LH and D-CAPT.

F. Real-World Tests

To demonstrate the practical applicability of our solution, we deploy it in real-world settings in an indoor testbed of $10 \times 6 \times 4\, m^3$ with a Vicon[1] system for localization. We employ 3 custom built quadrotors equipped with a Qualcomm Snapdragon™ VOXL 2[2] board and based on our previous work [43]. Three goals are chosen randomly in the flying arena. In the example shown Fig. 7, two scenarios are tested. In the first one, all robots can communicate and in the second one, r_3 can only communicate with r_1. In both cases, each robot effectively computes the optimal assignment.

[1]https://www.vicon.com/

[2]https://www.modalai.com/products/voxl-2?variant=39914779836467

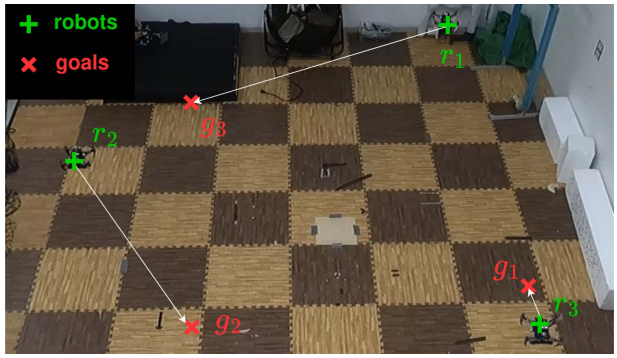

Fig. 7. Real-world experiments with robot positions (green), goal positions (red), trajectories (white).

VI. CONCLUSION AND FUTURE WORKS

In this work, we presented a novel DGNN-GA architecture to solve the decentralized LSAP for multi-robot systems. Inter-robot communication and goal assignments are modeled by one heterogeneous graph and the optimal solution is learned by imitating the expert Hungarian algorithm. The approach optimizes the information sharing process between the robots, leveraging the structure knowledge of the graph, to generate assignment predictions with restricted communication exchanges. We analyzed its zero-shot generalizability for team sizes up to 50 robots. The DGNN-GA was deployed in simulation and real-world for a multi-robot minimum distance traveled squared objective. It demonstrates competitive performances compared to existing approaches with a fixed number of communication rounds. Compared to LH and D-CAPT, it is devoid of the assumption to have an initial one-to-one assignment. It outperforms CBAA in every scenario and DHBA when the number of communication rounds is minimal. DGNN-GA can be used for multiple applications.

In the future, DGNN-GA will be combined with a local motion planner to optimize the robots' motion considering their dynamics and ensuring safe navigation. Moreover, the method will be extended considering more goals than robots.

REFERENCES

[1] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," in *Algorithmic Foundations of Robotics XI*. Berlin, Germany: Springer, 2015, pp. 591–607.

[2] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 856–869, Aug. 2018.

[3] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Auton. Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[4] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 98–112, 2014.

[5] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Auton. Robots*, vol. 33, no. 1, pp. 143–156, 2012.

[6] H. W. Kuhn, "The hungarian method for the assignment problem," *Nav. Res. logistics Quart.*, vol. 2, no. 1/2, pp. 83–97, 1955.

[7] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks*. Berlin, Germany: Springer, 2015, pp. 31–51.

[8] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[9] H. Aziz, A. Pal, A. Pourmiri, F. Ramezani, and B. Sims, "Task allocation using a team of robots," *Curr. Robot. Rep.*, vol. 3, pp. 227–238, 2022.

[10] H. Liu, T. Wang, C. Lang, S. Feng, Y. Jin, and Y. Li, "GLAN: A graph-based linear assignment network," 2022, *arXiv:2201.02057*.

[11] C. Aironi, S. Cornell, and S. Squartini, "Tackling the linear sum assignment problem with graph neural networks," in *Proc. 2nd Int. Conf. Appl. Intell. Inform.*, 2023, pp. 90–101.

[12] D. Panagou, M. Turpin, and V. Kumar, "Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6757–6762.

[13] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[14] S. Ismail and L. Sun, "Decentralized hungarian-based approach for fast and scalable task allocation," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 23–28.

[15] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Comput. Surv.*, vol. 55, no. 10, pp. 1–35, 2023.

[16] J. Faigl, M. Kulich, and L. Přeučl, "Goal assignment using distance cost in multi-robot exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3741–3746.

[17] P. Lusk, X. Cai, S. Wadhwania, A. Paris, K. Fathian, and J. P. How, "A distributed pipeline for scalable, deconflicted formation flying," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5213–5220, Oct. 2020.

[18] M. Pallin, J. Rashid, and P. Ögren, "A decentralized asynchronous collaborative genetic algorithm for heterogeneous multi-agent search and rescue problems," in *Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot.*, 2021, pp. 1–8.

[19] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 932–947, Aug. 2017.

[20] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in *Proc. 23rd Int. Conf. Ind. Eng. Other Appl. Appl. Intell. Syst.*, 2010, pp. 721–730.

[21] Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, "Distributed assignment with limited communication for multi-robot multi-target tracking," *Auton. Robots*, vol. 44, no. 1, pp. 57–73, 2020.

[22] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, and J. W. Herrmann, "Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3770–3776.

[23] M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Auton. Robots*, vol. 44, pp. 547–584, 2020.

[24] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11 785–11 792.

[25] W. Gosrich et al., "Coverage control in multi-robot systems via graph neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8787–8793.

[26] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Proc. Conf. Robot Learn.*, 2020, pp. 671–682.

[27] A. Elfakharany and Z. H. Ismail, "End-to-end deep reinforcement learning for decentralized task allocation and navigation for a multi-robot system," *Appl. Sci.*, vol. 11, no. 7, 2021, Art. no. 2895.

[28] A. Khan et al., "Learning safe unlabeled multi-robot planning with motion constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7558–7565.

[29] D. Wang and H. Deng, "Multirobot coordination with deep reinforcement learning in complex environments," *Expert Syst. with Appl.*, vol. 180, 2021, Art. no. 115128.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," in *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[31] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[32] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 128–138, Nov. 2020.

[33] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized GNN-based policies," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8772–8778.

[34] M. Tzes, N. Bousias, E. Chatzipantazis, and G. J. Pappas, "Graph neural networks for multi-robot active information acquisition," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3497–3503.

[35] Y. Zhou, J. Xiao, Y. Zhou, and G. Loianno, "Multi-robot collaborative perception with graph neural networks," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2289–2296, Apr. 2022.

[36] J. Blumenkamp and A. Prorok, "The emergence of adversarial communication in multi-agent reinforcement learning," in *Proc. Conf. Robot Learn.*, 2021, pp. 1394–1414.

[37] T.-K. Hu, F. Gama, T. Chen, Z. Wang, A. Ribeiro, and B. M. Sadler, "VGAI: End-to-end learning of vision-based decentralized controllers for robot swarms," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 4900–4904.

[38] H. Wang, T. Qiu, Z. Liu, Z. Pu, and J. Yi, "Multi-agent formation control with obstacles avoidance under restricted communication through graph reinforcement learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8150–8156, 2020.

[39] M. Lee, Y. Xiong, G. Yu, and G. Y. Li, "Deep neural networks for linear sum assignment problems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 6, pp. 962–965, Dec. 2018.

[40] E. Tolstaya, J. Paulos, V. Kumar, and A. Ribeiro, "Multi-robot coverage and exploration using spatial graph neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8944–8950.

[41] R. Kortvelesy and A. Prorok, "Modgnn: Expert policy approximation in multi-agent systems with a modular graph neural network architecture," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9161–9167.

[42] H. Zhang, J. Cheng, L. Zhang, Y. Li, and W. Zhang, "H2gnn: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3435–3442, 2022.

[43] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 404–411, Apr. 2017.