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

Graph Neural Planning and Predictive Control for Multi-Robot Communication-Constrained Unlabeled Motion Planning

Manohari Goarin[1], Yang Zhou[1], and Giuseppe Loianno[2]

***Abstract*— The multi-robot unlabeled motion planning problem of concurrently assigning robots to goals and generating safe trajectories is central in many collaborative tasks. Recent Graph Neural Network methods offer scalable decentralized solutions but rely on simplified dynamics and simulation environments, overlooking key challenges of real-world deployment such as dynamic feasibility and communication constraints. To address these gaps, we propose a hierarchical framework that combines a Graph ATtention Planner (GATP) with a decentralized Nonlinear Model Predictive Controller (NMPC). GATP provides intermediate subgoals through multi-robot cooperation, and the NMPC enforces safety under nonlinear dynamics and actuation constraints. We evaluate our framework in both simulation and real-world quadrotor experiments. Thanks to attention mechanisms and minimal communication requirements, we demonstrate improved generalization to larger teams, robustness to communication delays up to 200 ms and practical feasibility with decentralized on-board inference.**

I. Introduction

In recent years, multi-robot systems have attracted significant attention due to their ability to speed up task execution compared to single robot solutions, while concurrently offering additional resilience to robot failures. Cooperative multi-robot planning has been extensively studied for exploration, surveillance, search and rescue, or warehouse automation [1], [2]. By coordinating their actions, robots can solve these tasks with increased energy and time efficiency. The unlabeled motion planning problem offers a unifying formulation for many of these applications when the robots are homogeneous and interchangeable. It is a joint assignment and trajectory planning problem, with the objective of cooperatively reaching a set of goals while minimizing the total distance and time to travel and avoiding collisions.

While centralized methods can find optimal solutions, the computational burden of large robot teams make them impractical, motivating the development of decentralized methods. Among recent works, learning-based methods and especially Graph Neural Networks (GNNs) have shown strong potential to solve multi-robot collaborative tasks [3]–[11] and notable works have applied them to the decentralized unlabeled motion planning problem [12]–[15]. GNNs are inherently decentralized through their message-passing architecture and can exploit the structural information of the team topology to learn solutions that are close to optimal while scaling to large robot teams. However, these GNN-based methods rely on simplified dynamics in simulation environments and overlook key challenges for real-world deployment such as communication constraints, trajectory smoothness, and guaranteed safety under actuation limits.

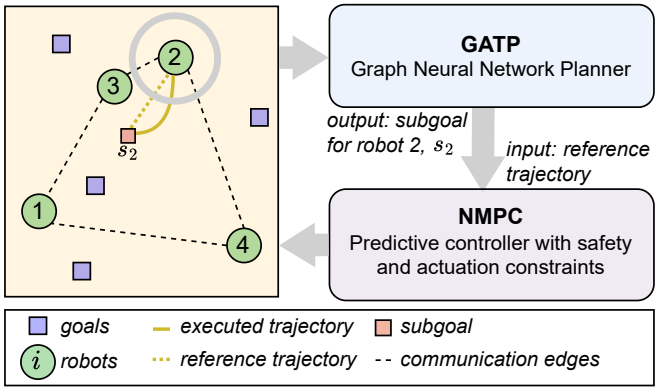


Fig. 1: **Hierarchical Architecture for cooperative and safe unlabeled motion planning:** a Graph ATtention Planner (GATP) that exchanges information over the robot communication graph and provides subgoals; a Nonlinear Model Predictive Control (NMPC) that tracks these subgoals with safety and actuation constraints.

To address these challenges in unlabeled motion planning, we propose a hierarchical architecture that integrates a GNN for high-level prediction of subgoals with a decentralized Nonlinear Model Predictive Control (NMPC) for trajectory execution toward these subgoals (see Figure 1). This hierarchy leverages the complementary strengths of both components: the GNN captures collaborative behavior through inter-robot communication to guide the robots toward their goals, and the NMPC guarantees safety and smooth motion under nonlinear dynamics and actuation constraints. The planner provides intermediate goals to the robots over a future horizon, and continuously updates them according to the robots' position changes during NMPC execution. This dynamic-agnostic high-level planning formulation allows low frequency replanning and direct deployment in real-world scenarios. Moreover, our GNN architecture relies on minimal communication, improving robustness to communication delays. We summarize our contributions as follows

- We propose a hierarchical framework combining a Graph ATtention Planner (GATP) and an NMPC for multi-robot collaborative and safe unlabeled motion planning with minimal communication.

[1]The authors are with New York University, NY 10012, USA. email: {mg7363, yz5794}@nyu.edu.
[2]The author is with the University of California Berkeley, Department of Electrical Engineering and Computer Sciences, Berkeley, CA 94720, USA. email: loiannog@eecs.berkeley.edu.
This work was supported by the DARPA YFA Grant D22AP00156-00, the DEVCOM ARL grant SARA W911NF-24-2-0057, and the NSF CPS Grant CNS-2603416.

- We perform an ablation study on our GNN architecture and benchmark it against Graph Convolution Networks (GCN) commonly used in prior works. We compare their coverage performance and generalization across various team sizes.
- We test our GATP-NMPC in simulation with 10 quadrotors in two application scenarios, circle formation and zone coverage. We evaluate its coverage time performance under increasing communication delays and show its robustness for delays under 200 ms.
- We implement a ROS 2 decentralized inference algorithm and deploy our framework in real-world experiments with 4 quadrotors. We quantify the inference times and communication delays to demonstrate its practical feasibility.

II. Related Works

Multi-robot motion planning is particularly challenging in the unlabeled setting, where robots are interchangeable and the joint assignment–planning problem is PSPACE-hard [16]. Centralized solutions exist, such as C–CAPT [17] that employs the Hungarian algorithm for assignment [18] and constant-velocity trajectory generation with collision avoidance guarantees under certain conditions. However, they are often impractical in real-world scenarios since they rely on global knowledge of the team's state, which is typically unavailable due to communication constraints. Moreover, they scale poorly as the number of robots increases. These limitations have motivated the development of decentralized approaches, presented in this section.

Model-based unlabeled motion planning. These methods often solve the assignment problem explicitly and combine it with robot and task-dependent trajectory optimization techniques. Switching-based strategies [17], [19]–[22] allow robots to iteratively refine their goal selection through local interactions with their neighbors, requiring little communication, but often suffering from slow or suboptimal convergence. Other methods rely on decentralized optimization-based and auction-based algorithms [23]–[26]. They can achieve better assignments, but typically require numerous communication rounds before converging to an optimal solution, and degrade in restricted communication settings [27]. In addition, explicit assignment can lead to abrupt changes in direction resulting in inefficient trajectories.

Learning-based unlabeled motion planning. Learning-based methods, including reinforcement learning [28]–[32] and unsupervised learning [33], have been explored to enhance the scalability and efficiency of unlabeled motion planning. Notably, GNNs gained significant attention due to their intrinsic ability to model and process graph-structured data, composed of nodes and edges, using graph filters [34]. Naturally decentralized, they offer an effective representation for multi-robot systems with robot nodes and communication edges. Through a message-passing framework, robots share information with their neighbors and make predictions locally. They have been applied successfully to various multi-robot applications like path planning [3], [4], coverage [5], [6], [10], flocking [5], [7], collaborative perception [8], target tracking [9], or exploration [10], [11]. By optimizing information sharing between robots, GNNs can learn near-optimal solutions with limited communication [27].

In the context of unlabeled motion planning, GNN-based methods have been proposed to learn end-to-end the concurrent goal assignment and trajectory planning problem [12]–[15]. Collision avoidance is typically handled either as a soft constraint within a reinforcement learning framework [13]–[15], or by applying a safety filter to the GNN outputs [12]. However, these approaches rely on simplified dynamics in simulation and directly predict velocity commands, which may cause abrupt accelerations and assume dynamic feasibility on real robots. Moreover, incorporating collision avoidance as a penalty term does not guarantee safety, while applying a post hoc safety filter can alter the network outputs and degrade performance. Finally, these GNN-based methods require multi-hop communication and 4 to 5 layers, which impose a heavy communication burden and make the system more vulnerable to delays in real-world settings.

Building on these works, this paper proposes a hierarchical GATP–NMPC framework that maintains trajectory optimality and safety guarantees under nonlinear dynamics and actuation limits. In addition, our GNN relies on only 2 layers of communication which is more robust to practical delays, and is directly deployable in real-world. To the best of our knowledge, we are the first to validate our GNN-based unlabeled motion planner in real-world settings with fully decentralized, on-board inference on the robots.

Real-world Communication challenges for GNNs. A major challenge when deploying GNNs on multi-robot systems is communication. While most learning-based works assume reliable message exchanges, real-world deployments are subject to communication delays, drops, and asynchronous updates [35]. A handful of studies evaluate their GNNs in physical multi-robot experiments, and typically rely on off-board centralized inference [4], [36]. In contrast, the authors in [37] introduce a decentralized ROS 2 framework to execute inference directly on-board. By testing standard velocity-planning GNN architectures, they notice that communication constraints, typically communication delays, lead to non-negligible performance degradation. In this paper, we analyze the impact of increasing communication delays on our task performance and show that our GNN is robust to bounded delays under ~ 200 ms.

III. Problem Formulation

In the following, all variables are time-dependent, but we drop the t for better clarity. We denote column vectors with bold notation like $\mathbf{x}$, matrices with capital notation like $\mathbf{A}$, and scalars with unbold notation like d. We consider a team of N identical robots $i \in \{1, ..., N\}$ and a set of N goals $j \in \{1, ..., N\}$. Each robot follows nonlinear dynamics:

$$\forall i, \quad \dot{\mathbf{x}}_i = f(\mathbf{x}_i, \mathbf{u}_i),$$

with $\mathbf{x}$ the state of robot i, including its position $\mathbf{p}_i$ and velocity $\mathbf{v}_i$, and $\mathbf{u}_i$ its control input. The unlabeled motion planning objective is to compute minimum-time and

collision-free trajectories from some initial positions to the goal positions as follows:

1) **Coverage objective.** Robots are interchangeable, not pre-assigned to goals and need to cover all goals as fast as possible. The task is complete when each goal is within a coverage threshold c of some robot:

$$\forall j, \quad \min_i \|\mathbf{p}_j - \mathbf{p}_i\|_2 < c.$$

2) **Pairwise safety.** For all times t and robot pairs (i, i'),

$$\|\mathbf{p}_i - \mathbf{p}_{i'}\|_2 > d_{\text{safe}},$$

with d_{safe} the safety distance to maintain.

In a decentralized setting, robots have limited sensing and communication capabilities. We consider that each robot i can sense its P_g closest goals and P_r closest robots and estimate their relative positions. We define its observation as

$$\mathbf{o}_i = \begin{bmatrix} \mathbf{p}_i^\top & \mathbf{g}_{i,1}^\top \cdots \mathbf{g}_{i,P_g}^\top & \mathbf{r}_{i,1}^\top \cdots \mathbf{r}_{i,P_r}^\top \end{bmatrix}^\top,$$

with $\mathbf{g}_{i,j}$ the relative position of goal j with respect to i, and $\mathbf{r}_{i,i'}$ the relative position of robot i' with respect to i. $\mathcal{O} = \{\mathbf{o}_i \mid i \in \{1, ..., N\}\}$ is the set of all robots' observations. $\mathcal{R}_i$ is the set of all P_r robots sensed by robot i.

Additionally, a robot i can communicate with its M closest neighbors, and we note this neighborhood $\mathcal{N}_i$.

IV. Methodology

A. Hierarchical Planning and Control Overview

Our hierarchical framework is depicted in Fig. 1. From an initial set of robot and goal positions, we can model a graph connecting robot nodes with communication edges. Each robot is connected to its M closest neighbors ($M = 2$ on the figure). We define the multi-robot graph as $\mathcal{G} = \{\mathcal{O}, \mathbf{A}\}$. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ describes the topology of the graph with its coefficients being

$$A_{ii'} = \begin{cases} 1 & \text{if } i' \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}$$

The framework is decomposed in two main parts:

1) Our GNN-based planner GATP, applied on each robot i. It takes as input the local graph $\mathcal{G}_i = \{\mathbf{o}_i, \mathbf{A}_{:i}\}$ with $\mathbf{A}_{:i}$ the i's column of $\mathbf{A}$, and outputs a subgoal position command s_i. s_i is an intermediate goal point for the robot to reach within the prediction horizon T_p of the planner.
2) A Nonlinear Model Predictive Controller with a prediction horizon T_c. Each robot i's NMPC tracks a straight linear minimum jerk reference trajectory [38] to the subgoal s_i while respecting hard safety and dynamic constraints. This module allows smooth and safe motion of the robots toward their subgoals.

GATP continuously updates the NMPC reference depending on the evolution of the robots' positions during execution.

B. Graph Neural Network for high-level planning

We design our GNN architecture as a Graph Attention Network (GAT) [39] which improves expressiveness by learning adaptive importance weights for each node's neighbors. We add Multi Layer Perceptrons (MLPs) update functions between layers, allowing each node to fuse its initial embedding with aggregated neighbor information. This mechanism preserves the permutation invariance property of the GNN and enhances node differentiation.

At a robot node i, the GNN architecture can be decomposed into three steps: **the Encoder**, **the Attention-based Message-passing module**, and **the Decoder**. We note $\mathbf{h}_i^l$ the embedding of node i at layer l, and ϕ_* MLPs with two linear layers and a $\mathrm{LeakyReLu}$ activation function in between.

1) **The Encoder** transforms the robots' observations into embedding vectors of size F

$$\mathbf{h}_i^0 \leftarrow \phi_{enc}(\tilde{\mathbf{o}}_i), \tag{1}$$

where $\tilde{\mathbf{o}}_i$ is the normalized observation $\mathbf{o}_i$ with respect to the environment spatial dimension D_w, obtained by dividing $\mathbf{p}_i$ by D_w and $\mathbf{g}_{i,j}$, $\mathbf{r}_{i,i'}$ by $2D_w$.

2) **The Attention-based Message-passing module** applies L layers (or communication rounds) of 1-hop neighbor information aggregation and embedding update. At each layer l, first, attention coefficients $e_{ii'}^l$ are computed between node i and each of its neighbors i'

$$\forall i' \in \mathcal{N}_i, \quad e_{ii'}^l = \mathbf{a}^l(\mathbf{W}^l\mathbf{h}_i^l, \mathbf{W}^l\mathbf{h}_{i'}^l). \tag{2}$$

The attention function $\mathbf{a}^l$ is a weight vector of size $2F$ followed by a $\mathrm{LeakyReLu}$. $\mathbf{W}^l$ is a weight matrix of size $F \times F$. These attention coefficients are normalized throughout the neighborhood using a softmax function

$$\forall i' \in \mathcal{N}_i, \quad \tilde{e}_{ii'}^l = \mathrm{softmax}_{i'}(e_{ii'}^l). \tag{3}$$

Finally, a weighted sum using the coefficients $\tilde{e}_{ii'}^l$ followed by a nonlinear activation function combines the neighbors' embeddings

$$\bar{\mathbf{h}}_i^l = \tanh\left(\sum_{i' \in \mathcal{N}_i} \tilde{e}_{ii'}^l \mathbf{W}^l \mathbf{h}_{i'}^l\right). \tag{4}$$

We apply multi-head attention by performing K attention mechanisms in parallel. These heads k are combined using the $\max$ function

$$\hat{\mathbf{h}}_i^l = \max_{k \in \{1,...,K\}} \bar{\mathbf{h}}_i^{l,k}. \tag{5}$$

Once this message-passing is performed, we update node i's embedding by combining its initial encoded embedding $\mathbf{h}_i^0$ with the aggregated information $\hat{\mathbf{h}}_i^l$

$$\mathbf{h}_i^{l+1} \leftarrow \phi_{update}(\mathbf{h}_i^0 \mid\mid \hat{\mathbf{h}}_i^l). \tag{6}$$

This fusion strategy improves expressiveness and node differentiation.

3) **The Decoder** predicts an output $\tilde{s}_i$ for i from its final embedding $\mathbf{h}_i^L$

$$\tilde{s}_i = \tanh(\phi_{dec}(\mathbf{h}_i^L)), \quad \in [-1, 1]^n. \tag{7}$$

The Cartesian dimension n is 2 or 3 depending on whether the environment is 2D or 3D. Then this output is converted into a relative subgoal command by scaling and clamping it to its maximum magnitude, i.e, its maximum desired distance S_p from i

$$s_i = \min\left(1, \frac{S_p}{||\ \tilde{s}_i\ ||_2}\right)\tilde{s}_i, \tag{8}$$

where $S_p = v_{max}T_p$ represents the spatial horizon of the planner, given a desired time horizon T_p and a desired maximum speed v_{max} of the robots.

C. Safe Predictive Control for Trajectory Execution

We formulate a nonlinear optimization problem over the prediction horizon T_c for a robot i in continuous time

$$\underset{\mathbf{u}_i}{\arg\min} \quad \int_{t_0}^{t_0+T_c} \left(\|\mathbf{x}_i - \mathbf{x}_i^*(s_i)\|_Q^2 + \|\mathbf{u}_i - \mathbf{u}_i^*\|_R^2 \right) dt \tag{9a}$$

$$\text{s.t.} \quad \forall t, \quad \dot{\mathbf{x}}_i = f(\mathbf{x}_i, \mathbf{u}_i), \tag{9b}$$

$$\mathbf{x}_i \in \mathcal{X},\ \mathbf{u}_i \in \mathcal{U}, \tag{9c}$$

$$G_{ii'}(\mathbf{x}_i, \mathbf{x}_{i'}, \mathbf{u}_i) \geq 0,\ \forall i' \in \mathcal{R}_i, \tag{9d}$$

where Eq. (9a) represents the quadratic objective function minimizing the distance to the min-jerk reference trajectory $\mathbf{x}_i^*(s_i)$ guiding the robot toward its subgoal s_i with a maximum desired speed v_{max}, and the reference control inputs $\mathbf{u}_i^*$. Eq. (9b) is the nonlinear dynamics of the robot, Eq. (9c) the state and input constraints, and Eq. (9d) safety constraints maintaining the safety distance d_{safe} between i and the closest robots it can sense $\{i' \in \mathcal{R}_i\}$. In this paper, we choose to formulate $G_{ii'}$ as Control Barrier Function (CBF) constraints which can provide strong safety guarantees under actuation constraints [40]. The NMPC prediction horizon T_c is lower or equal than GATP prediction horizon T_p.

V. Experimental Setup

A. GATP Training Setup

GATP is trained with imitation learning. The centralized expert employs the Hungarian Algorithm [18] to assign the robots to the goals minimizing the total distance traveled. The optimal subgoals s_i^* are calculated by discretizing straight trajectories toward the assigned goals with a step size S_p. During training, the robots follow simple simulation dynamics under a mixed GATP/expert policy

$$\forall i, \forall t, \quad \mathbf{p}_i(t+1) = \begin{cases} s_i(t) & \text{with a probability } \beta \\ s_i^*(t) & \text{with a probability 1-}\beta \end{cases}$$

We adopt a scheduled sampling scheme, gradually increasing β to replace expert subgoals with the GATP predictions. The planning spatial horizon is set at $S_p = 4$ m, which defines the maximum magnitude of the predicted subgoals. Since the GNN outputs are normalized, different horizons can still be applied at test time. The choice of S_p was empirically tuned to balance convergence speed and stability near the goals: larger horizons lead to oscillations around the targets, while smaller horizons result in slower learning and convergence.

We build a dataset of 10000 graphs of $N = 10$ agents with random initial and goal positions in an environment of 20×20 meters, and rollout trajectories of $N_t = 70$ timesteps. We train our GNN with centralized inference using the Deep Graph Library [41] to minimize the mean squared error between the normalized predicted subgoals $\tilde{s}_i$ and the normalized expert subgoals $\tilde{s}_i^*$, summed over trajectories and averaged across robots per batch. The loss is formulated as

$$\mathcal{L} = \frac{1}{NB}\sum_i \sum_t w_i(t)\ ||\ \tilde{s}_i(t) - \tilde{s}_i^*(t)\ ||_2^2,$$

with B the batch size. $w_i(t)$ is an adaptive weight that increases when the error is small, to refine the predictions' accuracy and stabilize the planning behavior around the goal locations. It is obtained as

$$w_i(t) = 1 + \alpha_1 \exp\left(-\alpha_2\ ||\ \tilde{s}_i(t) - \tilde{s}_i^*(t)\ ||_2 \right),$$

where the parameters α_1 and α_2 are hand-tuned.

In the context of restricted communication scenarios in the real world, and to reduce the GNN sensitivity to possible delays during deployment, we limit the message-passing module to 1-hop aggregations, $L = 2$ layers, and $M = 2$ maximum neighbors each robot can communicate with. We restrict the observation of the robots to $P_g = 5$ closest goals and $P_r = 3$ closest robots, use $K = 3$ heads of attention, and a feature dimension $F = 64$. In the following subsections, the training runs and comparisons were conducted using a batch size of $B = 200$ graphs and 80 epochs on a 12th-generation Intel CPU I9-12900H. We increase the probability β incrementally from 0.5 to 1.0 every 20 epochs, and fix the loss parameters $\alpha_1 = 5.0$ and $\alpha_2 = 4.0$. The learning rate is set at $6\text{e}{-4}$ and $1\text{e}{-4}$ on the last 20 epochs.

B. Quadrotor dynamics

In our experiments, we demonstrate and deploy our framework with quadrotors. The state and control inputs of a quadrotor i can be described as

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_i^\top & \mathbf{v}_i^\top & \mathbf{q}_i^\top & \boldsymbol{\omega}_i^\top \end{bmatrix}^\top, \mathbf{u}_i = \begin{bmatrix} u_{i0} & u_{i1} & u_{i2} & u_{i3} \end{bmatrix}^\top,$$

where $\mathbf{p}_i \in \mathbb{R}^3$ and $\mathbf{v}_i \in \mathbb{R}^3$ are respectively the position and linear velocity of the quadrotor in the inertial frame, $\mathbf{q}_i \in \mathbb{R}^4$ the rotation in quaternions from the quadrotor's body frame to the inertial frame, $\boldsymbol{\omega}_i \in \mathbb{R}^3$ the angular velocity in the body frame, and $\{u_{ik} \in \mathbb{R}, k \in [0, ..., 3]\}$ the motor thrusts of the quadrotor. The dynamic equations are as presented in [42] and can be written in a control-affine form as follows

$$\dot{\mathbf{x}}_i = f'(\mathbf{x}_i) + g(\mathbf{x}_i)\mathbf{u}_i.$$

The safety constraints in Eq. (9d) from the NMPC formulation are based on Exponential Control Barrier Functions (ECBFs) which can provide forward invariance guarantees of the safe set for the quadrotors' higher-order dynamics. We refer the readers to [40] for further details.

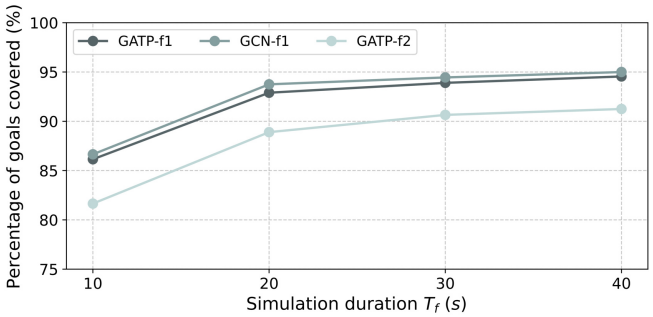


Fig. 2: GATP coverage performance analysis with 10 robots.

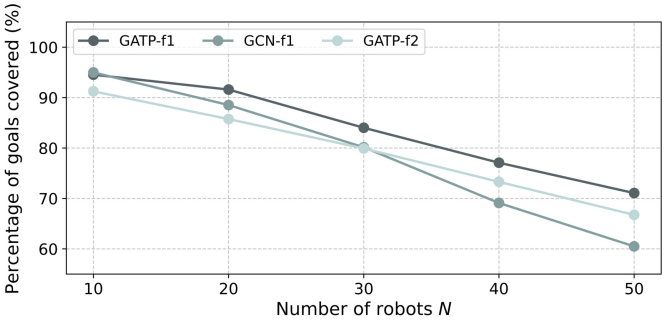


Fig. 3: GATP generalization analysis to larger teams with a simulation duration of $T_f = 40$s.

VI. Results and Analysis in Simulation and Real-World

A. GATP coverage performance analysis

First, we evaluate the planning performance of GATP using point-like robots and show the benefits of attention and our MLP fusion strategy (Eq. (6)) on coverage performance and generalization to larger teams. We define coverage performance as the percentage of goals reached in a given time T_f. The spatial horizon is still fixed at $S_p = 4$ m. We report in Figs. 2 and 3 the total coverage percentage over 200 scenarios, for different simulation durations T_f from 10 to 40 s and different team sizes from $N = 10$ to 50 robots. We test 3 different GNN architectures for comparison:

- **GATP-f1**, our architecture as presented in section IV-A, with f_1 the update function from Eq. (6): $f_1(i,l) = \phi_{update}(\mathbf{h}_i^0 \,||\, \hat{\mathbf{h}}_i^l)$.
- **GCN-f1**, a Graph Convolutional Network as employed in [12]–[14], similar to GATP-f1 but with 2 layers of 1-hop convolutional filters instead of attention layers.
- **GATP-f2**, a variation of GATP-f1 that uses a different update function $f_2(i,l) = \mathbf{h}_i^l + \phi_{update}(\hat{\mathbf{h}}_i^l)$ that combines the aggregated information with the previous node embedding instead of the initial encoded embedding. This function was used in [13].

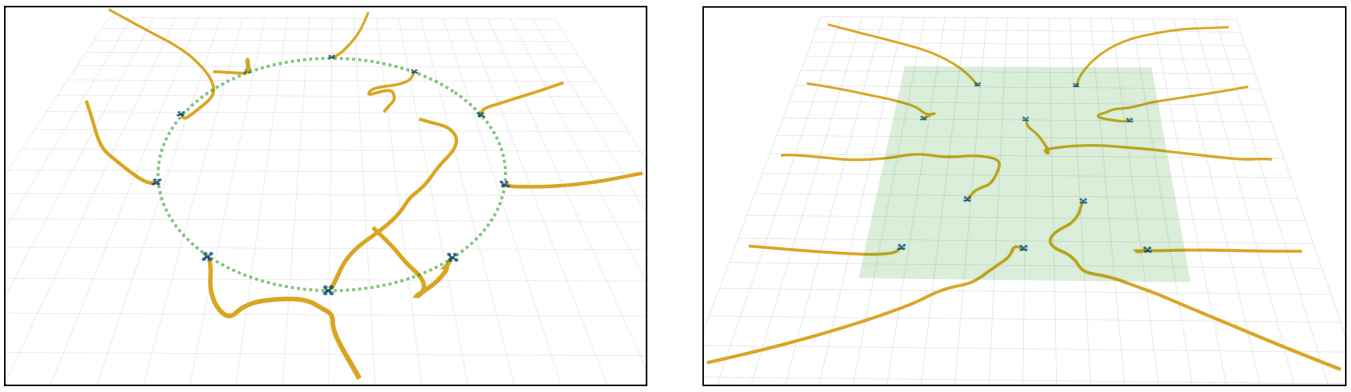
First, we analyze the coverage performance on 10 robots with increasing simulation durations (Figure 2). GCN-f1 and GATP-f1 achieve comparable results. With a limited time of $T_f = 10$ s, GATP-f1 reaches 86.15% of the goals compared to 86.65% for GCN-f1, despite relying on only 1-hop neighbor aggregation and 2 communication rounds. In $T_f = 40$ s, GATP-f1 reaches 94.55% of coverage against 95.00% for GCN-f1. These results exhibit the ability of GNNs to learn efficient heuristics through optimized information exchange. Both architectures outperform GATP-f2 which completes at most 81.45% in 10 s and 91.25% in 40 s. The greater performance of GCN-f1 and GATP-f1 can be attributed to the update function f_1 that re-injects the initial node embedding after each layer via a learnable MLP. This strategy enhances node differentiation and reduces assignment conflicts.

Second, we analyze the generalization of the GNN architectures to larger teams of up to 50 robots. For lower computation cost and faster training, the GNNs are trained with 10 robots but are directly transferrable to larger team sizes thanks to the decentralized nature of GNNs. On Figure 3, we analyze the evolution of the coverage percentage as we increase the number of robots for a fixed simulation duration of 40 s. Overall, the more robots we add, the more the GNNs degrade in performance because of the small number of robots used for training. However, GATP-f1 outperforms both GCN-f1 and GATP-f2. Although GCN-f1 demonstrated similar performance to GATP-f1 with 10 robots, GATP-f1 and GATP-f2 are more generalizable to larger teams with a smaller decrease in performance. From 10 to 50 robots, GATP-f1 experiences a performance drop of 23.5%, compared to 24.5% for GATP-f2 and 34.5% for GCN-f1. We conclude that attention mechanisms improve generalization. By dynamically weighting neighbors' information, the network can capture more complex interactions and account for the varying importance of different neighbors.

B. GATP-NMPC performance analysis under increasing communication delays

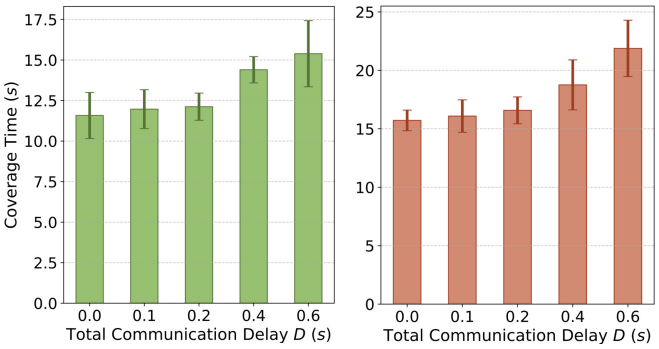
In this section, we test our hierarchical GATP-NMPC in simulation with quadrotors. We run 10 robot nodes in a ROS 2 environment, each running its own NMPC but with a centralized GATP node, only for this simulation experiment, to avoid overloading our computer with parallel inferences. The planner runs at 2 Hz and the NMPC at 100 Hz. We set the GATP time horizon at $T_p = 2$ s and the desired maximum speed of the robots at $v_{max} = 2$ m/s, which is equivalent to a planning spatial horizon of $S_p = 4$ m. The desired safety distance between robots is $d_{safe} = 0.5$ m. We use acados [43] to solve the NMPC with SQP-RTI and a prediction horizon of $T_c = 1.5$ s.

We demonstrate the efficacy of our framework in two example tasks: (a) a **circle formation task**, where robots start from a set of random positions and must form a circle, and (b) a **zone coverage task**, where robots must uniformly cover a given zone of the environment. Two simulation runs illustrate these tasks in Figure 4. Additional examples for different tasks can be found in the video. The performance metric for these experiments is the **coverage time**, defined as the total time required for the team to reach all goals, with a coverage threshold of $c = 0.2$ m per goal.



(a) Circle Formation (b) Zone Coverage

Fig. 4: Simulation with 10 quadrotors for a circle formation task and a coverage task. The green circle is the desired circle to form, and the green area is the environment zone to cover. The quadrotors are in blue, and their trajectories are yellow.



(a) Circle Formation (b) Zone Coverage

Fig. 5: Performance analysis under increasing communication delays on both tasks.

First, the GATP–NMPC framework successfully guides the quadrotors to their goal locations in both tasks, producing smooth and safe trajectories through the seamless integration of the graph-based planner with the NMPC. The high-level planner leverages local information to minimize the distance traveled by the robots, helping to deconflict the multi-robot system and facilitate NMPC feasibility.

In the real world, significant delays may arise from inter-robot communication, which the message-passing module depends on, and can lead to planning updates based on outdated observations. To model this effect, we assume that all robots experience the same communication delay d per communication round (i.e., GNN layer). The total communication delay is then $D = Ld$. We test different values of D and analyze its impact on coverage time performance for both tasks. The inference time also contributes to the total delay but is negligible compared to the communication effect in practice. We run each task 10 times with identical initial and goal positions, and report the mean and standard deviation of the coverage time in Fig. 5. Variations across runs are due to additional uncertainties related to ROS 2 processes and numerical approximations in the NMPC. In both scenarios, we notice some performance degradation for large communication delays $D = 0.4$ s and $D = 0.6$ s. However, the framework is robust to delays under 0.2 s, with an average increase of coverage time of 0.44 s (3.8%) on the circle formation task, and 0.86 s (5.5%) on the zone coverage task. The performance degradation becomes impactful when the delay is comparable to the planning update interval of 0.5 s. At such delays, the outputs are too outdated to remain valid for effective planning. With $D = 0.6$ s, the coverage time increases by 31.8% in the circle formation task and by 39.2% in the zone coverage task. The effect is more pronounced in the last scenario, where robots must travel longer distances and thus accumulate larger errors over time.

In conclusion, our GATP architecture with $L = 2$ layers for high-level, low-frequency planning is robust enough to bounded communication delays of up to 0.2 s (i.e., $d = 0.1$ s per layer). Using more layers (e.g., 4–5 as in previous works [12]–[14]) could potentially improve planning performance, but would also make the system significantly more sensitive to delays, since with 0.1 s per layer the total communication latency would already reach $D = 0.4$ to 0.5 s.

C. System Implementation and Deployment in Real-World

We deploy our solution in an indoor $10\,\mathrm{m} \times 6\,\mathrm{m} \times 4\,\mathrm{m}$ testbed with Vicon[1] localization and 4 custom quadrotors based on [44] and equipped with Qualcomm® Snapdragon™ VOXL® 2[2]. We demonstrate the applicability of our GATP–NMPC framework in a formation task, where robots sequentially form different shapes (see Figure 6 and the attached multimedia material). Taking into account our environmental dimensions, we set the desired maximum speed of the robots at $v_{max} = 0.5$ m/s, the planner time horizon at $T_p = 1.5$ s, and the NMPC horizon $T_c = 1$ s. Each robot senses its $P_g = 4$ closest goals and $P_r = 2$ closest neighbors. Both the planner and the controller run onboard each quadrotor at 1 Hz and 160 Hz, respectively. 1 Hz was sufficient for this experimental setup given the small flying area and the robots' velocity.

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

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

Algorithm 1 GATP inference ROS 2 node on quadrotor i

```
while t < T_f do
    Get observation o_i
    Run encoder
    for l = 1 to L do
        Wait for all neighbors' messages
        if all messages received then
            Aggregate messages
            Update embedding
        end if
    end for
    Run decoder
    Send subgoal command to NMPC
end while
```

We implement a decentralized inference as in Algorithm 1. For each layer, robots wait for all neighbors' messages before proceeding. GATP replans concurrently with the NMPC until the experiment ends.

Our framework transfers directly to the real world thanks to its hierarchical design: the NMPC accounts for nonlinear quadrotor dynamics with actuation constraints while GATP uses position-based observations and commands that are dynamic-agnostic, facilitating sim-to-real transfer and generalization across platforms. For all tested formation shapes, quadrotors successfully reach the goals safely as shown on Figure 6a and in the video. In this experiment, the coverage time performance is similar in simulation and the real world.

We further validate our framework in the presence of obstacles (Figure 6b). Two obstacles were placed along the transitions between shapes 2→3 and 3→4. During 2→3, the robots have to deviate significantly from their reference trajectories to avoid the obstacles. GATP re-planned accordingly, providing a new global solution in which the yellow and blue quadrotors swapped positions compared to the obstacle-free case (Figure 6a). In contrast, during 3→4, obstacle avoidance did not alter the optimal solution, and GATP maintained the same plan.

Finally, we measure the average inference times and communication delays during the experiment (Table I). The total inference time of GATP is only about $1\ \mathrm{ms}$, thanks to its compact architecture and the use of ONNX Runtime[3]. In contrast, the average communication delay per layer is around $26\ \mathrm{ms}$, confirming that delays are mainly due to communication rather than computation. We also observe some delay variability between robots, with a total GATP time per robot ranging from $51\ \mathrm{ms}$ to $134\ \mathrm{ms}$. A $134\ \mathrm{ms}$ of planning time implies that GATP could be run at a maximum frequency of roughly 7 Hz, which is compatible with our low-frequency subgoal planning formulation. Moreover, this maximum total delay falls within the 0.1–$0.2\ \mathrm{s}$ range tested in simulation, where coverage performance was shown to remain mostly unaffected. This justifies the absence of noticeable performance loss between simulation and real-world.

[3]https://onnxruntime.ai/

TABLE I: Inference times and Communication delays

Metric	**Time (ms)**	
Average GATP inference time (comm. excluded)	1.03	
Average communication delay per layer	26.04	
Total GATP time per robot	Robot 1	62.88
	Robot 2	134.29
	Robot 3	106.90
	Robot 4	51.23

VII. Conclusion

In this work, we introduced a novel hierarchical approach that combines a Graph ATtention Planner (GATP) with a decentralized Nonlinear Model Predictive Controller (NMPC) for collaborative and safe unlabeled motion planning under nonlinear dynamics and limited communication. In simulation, we demonstrated improved generalization to larger teams thanks to attention mechanisms, and robustness of our two-layer design against delays up to $200\ \mathrm{ms}$. We deployed decentralized on-board inference on quadrotors, highlighting practical feasibility and effective sim-to-real transfer.

To strengthen coordination between hierarchical modules, future work will investigate coupling planning and control during training by incorporating dynamics knowledge and collision avoidance constraints into the GNN learning process. In addition, we plan to examine more realistic communication conditions to better understand their impact, such as asynchronous updates and variable delays, and extend our real-world validation to larger multi-robot teams.

References

[1] J. K. Verma *et al.*, "Multi-robot coordination analysis, taxonomy, challenges and future scope," *Journal of intelligent & robotic systems*, vol. 102, no. 1, p. 10, 2021.

[2] H. Wang, *et al.*, "Breaking the hierarchy: Taxonomies and survey on multi-robot integrated task and motion planning," *Authorea Preprints*, 2025.

[3] Q. Li, *et al.*, "Graph neural networks for decentralized multi-robot path planning," in *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2020, pp. 11 785–11 792.

[4] X. Ji, *et al.*, "Decentralized, unlabeled multi-agent navigation in obstacle-rich environments using graph neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8936–8943.

[5] Y. Hu, *et al.*, "Graph soft actor–critic reinforcement learning for large-scale distributed multirobot coordination," *IEEE transactions on neural networks and learning systems*, 2023.

[6] W. Gosrich, *et al.*, "Coverage control in multi-robot systems via graph neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8787–8793.

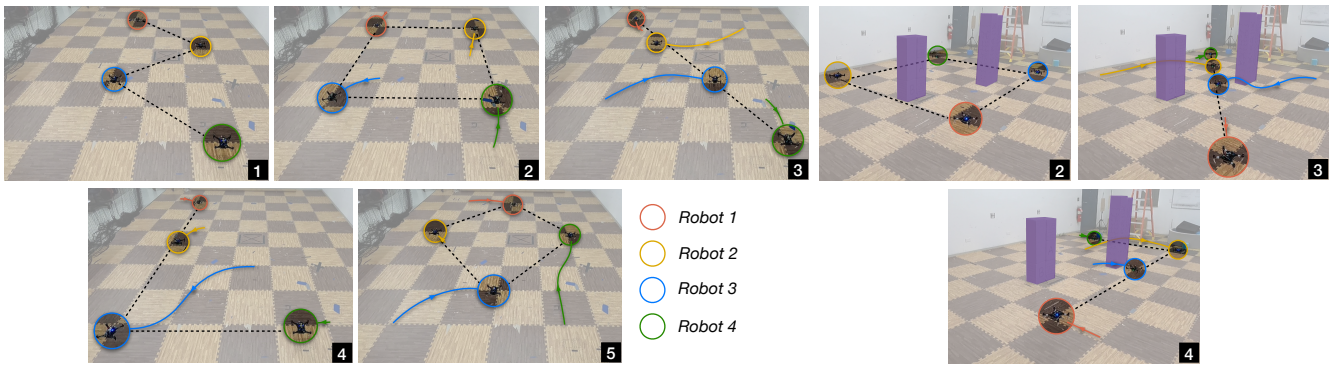
[7] E. Tolstaya, *et al.*, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on robot learning*. PMLR, 2020, pp. 671–682.

[8] Y. Zhou, *et al.*, "Multi-robot collaborative perception with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2289–2296, 2022.

[9] L. Zhou, *et al.*, "Graph neural networks for decentralized multi-robot target tracking," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2022, pp. 195–202.

[10] E. Tolstaya, *et al.*, "Multi-robot coverage and exploration using spatial graph neural networks." in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8944–8950.

[11] H. Zhang, *et al.*, "H2gnn: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3435–3442, 2022.



(a) Without obstacles

(b) With obstacles

Fig. 6: Real-world Experiments. Each circle represents a robot, the arrow line its trajectory, and the dashed line a desired shape. Color red is used for Robot 1, yellow for Robot 2, blue for Robot 3, green for Robot 4, and purple for obstacles.

[12] A. Khan, *et al.*, "Large scale distributed collaborative unlabeled motion planning with graph policy gradients," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5340–5347, 2021.

[13] S. Muthusamy, *et al.*, "Generalizability of graph neural networks for decentralized unlabeled motion planning," *arXiv preprint arXiv:2409.19829*, 2024.

[14] A. Khan, *et al.*, "Graph policy gradients for large scale unlabeled motion planning with constraints," *arXiv preprint arXiv:1909.10704*, 2019.

[15] T. Wang, *et al.*, "Hierarchical relational graph learning for autonomous multirobot cooperative navigation in dynamic environments," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 3559–3570, 2023.

[16] K. Solovey *et al.*, "On the hardness of unlabeled multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1750–1759, 2016.

[17] M. Turpin, *et al.*, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[18] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[19] H. Bang, *et al.*, "Energy-optimal goal assignment of multi-agent system with goal trajectories in polynomials," in *29th Mediterranean Conference on Control and Automation (MED)*, 2021, pp. 1228–1233.

[20] S. Dergachev *et al.*, "Decentralized unlabeled multi-agent navigation in continuous space," in *International Conference on Interactive Collaborative Robotics*. Springer, 2024, pp. 186–200.

[21] J. Hu, *et al.*, "Convergent multiagent formation control with collision avoidance," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1805–1818, 2020.

[22] D. Panagou, *et al.*, "Decentralized goal assignment and safe trajectory generation in multirobot networks via multiple lyapunov functions," *IEEE Transactions on Automatic Control*, vol. 65, no. 8, pp. 3365–3380, 2019.

[23] P. C. Lusk, *et al.*, "A distributed pipeline for scalable, deconflicted formation flying," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5213–5220, 2020.

[24] Y. Sung, *et al.*, "Distributed assignment with limited communication for multi-robot multi-target tracking," *Autonomous robots*, vol. 44, no. 1, pp. 57–73, 2020.

[25] G. Xu, *et al.*, "Multi-robot task allocation and path planning with maximum range constraints," *arXiv preprint arXiv:2409.06531*, 2024.

[26] D. Morgan, *et al.*, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[27] M. Goarin *et al.*, "Graph neural network for decentralized multi-robot goal assignment," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4051–4058, 2024.

[28] A. Khan, *et al.*, "Learning safe unlabeled multi-robot planning with motion constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7558–7565.

[29] G. E. Setyawan, *et al.*, "Cooperative multi-robot hierarchical reinforcement learning," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 9, 2022.

[30] H. Qie, *et al.*, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," *IEEE access*, vol. 7, pp. 146 264–146 272, 2019.

[31] D. Wang *et al.*, "Multirobot coordination with deep reinforcement learning in complex environments," *Expert Systems with Applications*, vol. 180, p. 115128, 2021.

[32] A. Elfakharany *et al.*, "End-to-end deep reinforcement learning for decentralized task allocation and navigation for a multi-robot system," *Applied Sciences*, vol. 11, no. 7, p. 2895, 2021.

[33] T. Sellers, *et al.*, "Autonomous multi-robot allocation and formation control for remote sensing in environmental exploration," in *Autonomous Systems: Sensors, Processing, and Security for Ground, Air, Sea, and Space Vehicles and Infrastructure 2023*, vol. 12540. SPIE, 2023, pp. 250–266.

[34] Z. Wu, *et al.*, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[35] J. Gielis, *et al.*, "A critical review of communications in multi-robot systems," *Current robotics reports*, vol. 3, no. 4, pp. 213–225, 2022.

[36] Y. Wang, *et al.*, "Multi-robot obstacle-avoidance formation based on graph neural networks and imitation learning," in *China Automation Congress (CAC)*, 2024, pp. 5499–5504.

[37] J. Blumenkamp, *et al.*, "A framework for real-world multi-robot systems running decentralized gnn-based policies," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8772–8778.

[38] D. Mellinger *et al.*, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[39] P. Veličković, *et al.*, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[40] M. Goarin, *et al.*, "Decentralized nonlinear model predictive control for safe collision avoidance in quadrotor teams with limited detection range," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 5387–5393.

[41] M. Y. Wang, "Deep graph library: Towards efficient and scalable deep learning on graphs," in *ICLR workshop on representation learning on graphs and manifolds*, 2019.

[42] A. Saviolo *et al.*, "Learning quadrotor dynamics for precise, safe, and agile flight control," *Annual Reviews in Control*, vol. 55, pp. 45–60, 2023.

[43] R. Verschueren, *et al.*, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[44] G. Loianno, *et al.*, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.