EGNN-based Topology Control in Wireless Mobile Infrastructure on Demand with Shared Access Restrictions

Mariana del Castillo
mdelcastillo@fing.edu.uy
Universidad de la República
Montevideo, Uruguay

Alejandro Ribeiro
aribeiro@seas.upenn.edu
University of Pennsylvania
Philadelphia, USA

Federico Larroca
flarroca@fing.edu.uy
Universidad de la República
Montevideo, Uruguay

ABSTRACT

Mobile Infrastructure on Demand (MIoD) involves deploying a mobile ad-hoc network through a team of network agents to provide communication infrastructure for another group of mobile agents serving a specific application. This study aims to determine the optimal locations for these network agents. Previous approaches have focused on maximizing graph connectivity or directly utilizing communication indicators. However, these methods often overlook the shared nature of the wireless communication medium, leading to suboptimal results. In this study, we incorporate shared access restrictions into our optimization model, addressing the inherent challenges in wireless communication. By leveraging the natural equivariance to translations and rotations of communication indicators, we employ E(n)-Equivariant Graph Neural Networks (EGNNs) to approximate the dependency of our indicator on node positions. We then use a gradient ascent algorithm to find optimal positions for the network agents. Our methodology demonstrates superior performance compared to traditional approaches, as evidenced by a higher figure of merit for the final configurations. These findings highlight the critical importance of considering the shared nature of the wireless medium for effective topology control in MIoD systems. The key contributions of this work include the incorporation of shared access restrictions into the optimization model, allowing for a more accurate representation of the problem, and the proposal of an EGNN-based Black Box Optimization approach to solve the resulting topology control problem.

CCS CONCEPTS

• **Networks** → *Ad hoc networks*; • **Theory of computation** → **Nonconvex optimization**; • **Mathematics of computing** → *Graph algorithms*; • **Computing methodologies** → **Robotic planning**; *Multi-agent systems*; *Neural networks*; **Mobile agents**.

KEYWORDS

Multi-agent System, Ad-Hoc networks, Robotics, Black box optimization

ACM Reference Format:
Mariana del Castillo, Alejandro Ribeiro, and Federico Larroca. 2024. EGNN-based Topology Control in Wireless Mobile Infrastructure on Demand with Shared Access Restrictions. In *Proceedings of the 3rd GNNet Workshop: Graph Neural Networking Workshop (GNNet '24), December 9–12, 2024, Los Angeles, CA, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3694811.3697821

1 INTRODUCTION

In the present day, technologies like Wi-Fi, LTE, and 5G create the illusion of ubiquitous broadband communication. There are numerous scenarios where this holds true, facilitated by meticulous planning and deployment of telecommunications infrastructure to ensure high-quality communication.

Nonetheless, numerous application scenarios exist where the aforementioned statement is not true, and the infrastructure is insufficient or even nonexistent. An example is monitoring marine animals in vast bodies of water (such as large lakes or the open sea). For such tasks, drones equipped with cameras are deployed to traverse and film various areas, while an operator on the ground observes the live footage and captures high-resolution photos of points of interest, typically when spotting specimens of the animals being monitored [20, 21]. The extremely limited connectivity in these remote coastal areas, often situated in relatively isolated regions, necessitates point-to-point communication between the drone and the ground operator. This results in a range of only hundreds of meters at best, constituting one of the main limitations of this application. Deploying additional drones to serve as 'repeaters' between the filming drone and the coastal operator could potentially extend this range.

The example illustrates the concept of a *Wireless Mobile Infrastructure on Demand* [15], which involves deploying a mobile ad-hoc network to provide communication infrastructure for a specific application. Furthermore, it highlights the network's composition of two distinct types of mobile nodes: those performing specific tasks (such as capturing footage of monitored areas) and those ensuring continuous connectivity between them. We will refer to the former as *task agents* and the latter as *network agents*. It is important to emphasize that the task agents are mobile nodes that do not necessarily follow planned movements; rather, they often need to adjust their positions dynamically. For instance, in scenarios like monitoring marine fauna, operators may require filming drones to reposition to track specific individuals. The role of the network agents is to maintain connectivity for the task agents, who operate in a mobile manner without predefined routes.

This is precisely the problem we consider here; i.e. where to locate the network agents given the task agents' positions. To this end, the first step is to define an objective function so that these

positions maximize a certain indicator. A relatively straightforward approach is to choose a proxy of the ad-hoc network's performance, for instance, the connectivity of the underlying communication graph [25]. The intuition is that the more connected a network is, the better its performance. However, as we confirm in our simulations, this is not necessarily true and gains can be obtained by directly considering the pertinent network performance as an indicator (e.g. the obtained throughput of the task agents).

In order to directly consider the ad-hoc network's performance, the most significant challenge is the choice of a communication model. The first option is to use a point-to-point model, where the achievable bit rate depends on the signal-to-noise ratio at reception: the higher this ratio, the higher the maximum achievable bit rate. Given a transmission power, this maximum rate transforms into a function of the power attenuation between the nodes. This attenuation, in turn, depends on the distance between the nodes and the obstacles between them [13, 23]. Since an overly precise map of the location where the nodes are deployed is rarely available, a probabilistic model is considered to account for these obstacles.

This constitutes the prevailing approach adopted by the vast majority of topology control methods in mobile ad-hoc networks [11, 15, 22, 24]. However, these models are oversimplistic, neglecting a fundamental aspect of every wireless network: the shared medium. This implies that the attenuation between nodes does not solely determine the attainable bit rate on a given link but is also influenced by neighboring links and their channel usage [4, 16]. Here, interference is not the sole concern; crucially, the medium access inherent to all wireless technologies must be considered. As our simulations demonstrate, ignoring this aspect of the channel produces suboptimal results.

Our first contribution is to explicitly model the access control mechanism of the communication channel through additional constraints on the throughput of each node, which depends on the neighboring nodes. However, this makes finding the optimal positions for the network agents a highly non-convex problem, where even numerical solvers struggle to find an approximate solution. Our second contribution is to propose and evaluate a *Black Box Optimization* approach [2], where given the agents' positions, a trained algorithm provides the network's performance indicator.

To this end we explore an *E(n) Equivariant Graph Neural Networks* (EGNN) [19], which offers two significant advantages. Firstly, its input are both nodes' positions and features, and the output is invariant to rotations and translations of the input positions. This is true for our problem and drastically reduces the number of examples necessary to train the system. Secondly, since one of the algorithm's inputs is the nodes' positions, once trained we can compute the gradient of the output with respect to the network agents' positions. This enables us to use a first-order gradient ascent algorithm to approximate the optimal locations of the network agents. As demonstrated in our simulations, the resulting topology control algorithm improves the network's performance compared to previous proposals. We have furthermore evaluated the algorithm in scenarios unseen during training (even with more agents), and it still perform competitively against non-learning algorithms.

2 BLACK BOX OPTIMIZATION FOR TOPOLOGY CONTROL

2.1 Problem Statement

As previously mentioned, our problem involves two categories of mobile nodes. The first group, known as task agents, necessitates communication among themselves to accomplish certain tasks and will operate under the assumption that such communication is available. To facilitate this, a second group of nodes, referred to as network agents, is deployed solely to act as relays and strategically position themselves to enhance and sustain communication among the task agents. By doing so, the task agents are relieved from considering the impact of their trajectories on their communication abilities with each other, thereby simplifying their operations [15]. Our research focuses on the optimal positioning of the network agents.

More precisely, given a set of T task agents and N network agents, with positions given by $\mathbf{X}_T \in \mathbb{R}^{T\times 2}$ and $\mathbf{X}_N \in \mathbb{R}^{N\times 2}$ respectively (i.e. each row corresponds to the position of an agent in the plane), we address the problem of modifying $\mathbf{X}_N$ to improve the communication between task nodes. For this, we need to define a metric to evaluate the configuration of the entire team in terms of the network performance.

Inspired by the many communication applications of the multi-commodity network flow problem (MNF) [18], in what follows we present a very general formulation of the communication problem, including the notation used throughout the article.

We will consider that communication has to be established between certain pairs of task agents, which we will denote as flows and will be indexed by $k = 1, \ldots, K$. Each flow is thus specified by its source and destination task agent (corresponding to nodes s^k and d^k respectively). For instance, in the example of marine fauna monitoring, all flows would be from the filming drones toward the operator. On the other hand, in a disaster scenario, all members of a rescue team would communicate with each other. We will consider this latter example, and thus if there are T task nodes we will have $K = T \times (T-1)$ flows, but modifications to other examples are straightforward.

Moving on to the wireless channel, assume nodes i and j are positioned at $\mathbf{x}_i$ and $\mathbf{x}_j$ (with i and j in the set $\{1, \ldots, T+N\}$). As we mentioned before, and assuming a fixed transmission power, a reasonable model is that the maximum rate achievable between these nodes depends on their distance through a function which we will denote as $C\left(||\mathbf{x}_i - \mathbf{x}_j||\right)$.

However, node i will not be able to use this maximum rate to communicate with node j, since the medium has to be shared with all its neighbors. All wireless systems include protocols to avoid collisions; e.g. in Wi-Fi whenever a node needs to transmit, it first checks if the channel is in use, in which case it waits a random time before trying again. We thus introduce a variable $0 \leq \tau_{ij} \leq 1$ which indicates the proportion of time that node i transmits to a neighboring node j. Furthermore, r_{ij}^k will represent the communication throughput from node i to node j used for flow k. Naturally, the sum over all flows of this rate for any pair of nodes cannot exceed the maximum rate times the proportion of time

the transmitting node uses to communicate with that particular neighbor; i.e. $\sum_k r_{ij}^k \leq \tau_{i,j} C(||\mathbf{x}_i - \mathbf{x}_j||)$.

Finally, we will denote as a^k the net flow transmitted at node $i = s^k$ corresponding to the source of flow k. Ideally, we want these values to be maximized, meaning that the network has reached a configuration where flows are operating at their maximum capacity. However, since several flows are operating in the network we need to enforce a sense of fairness [14] among them, and we thus maximize a utility function $\sum_k U(a^k)$, where $U(\cdot)$ is a concave function. For this implementation, we use the logarithm function.

All in all, we have the following maximization problem

$$\max_{\{a^k\},\{r_{ij}^k\},\{\tau_{ij}\}} \quad \sum_{k=1}^{K} U(a^k) \tag{1a}$$

$$\text{s.t.} \quad a^k \leq \sum_{j=1}^{T+N} r_{ij}^k - r_{ji}^k, \forall\, k \text{ and } i = s^k, \tag{1b}$$

$$0 = \sum_{j=1}^{T+N} r_{ij}^k - r_{ji}^k, \forall\, i \notin \{s^k, d^k\}, \tag{1c}$$

$$\sum_{j \in \mathcal{N}_i} \tau_{ij} + \sum_{j \in \mathcal{N}_i} \sum_{l \in \mathcal{N}_j} \tau_{jl} \leq 1, \forall i, \tag{1d}$$

$$\sum_k r_{ij}^k \leq \tau_{ij} \times C(||\mathbf{x}_i - \mathbf{x}_j||)\,,\ \forall\, i, j, \tag{1e}$$

$$0 \leq \tau_{ij}\ \forall\, i; j \in \mathcal{N}_i. \quad \tau_{ij} = 0\ \forall\, i; j \notin \mathcal{N}_i. \tag{1f}$$

Note that constraint (1b) indicates that the difference between the throughput transmitted and received by source node s^k is precisely a^k, the total throughput for flow k. On the other hand, and as enforced by constraints (1c), nodes acting purely as relays have a total balance of zero. Regarding our addition of constraints to the classical MNF formulation that model the shared nature of the wireless medium, (1d) limits the proportion of time that node i can transmit by considering its neighboring nodes $\mathcal{N}_i$ (more precisely, the total proportion of time used by node i plus its neighbors, should never exceed 1). Naturally, both the channel capacity function $C(\cdot)$ and the condition under which two nodes are considered neighbors are related. For the former, and following [10], we will consider

$$C(||\mathbf{x}_i - \mathbf{x}_j||) = e^{-\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||}{\alpha}\right)^{\beta}}, \tag{2}$$

where α and β are parameters that may be modified to consider different propagation scenarios and technologies (in our simulations, we used $\alpha = 10$ and $\beta = 2$). Note that the maximum obtainable capacity is one, and (2) should thus be interpreted as a normalized capacity. Furthermore, we highlight that our proposal is still valid for any other expressions of $C(\cdot)$, as long as it depends on the distance between the nodes.

Following the typical configuration in wireless communications, we will consider that two nodes are neighbors (and thus cannot transmit simultaneously) if the received power is larger than a certain threshold. We will implicitly model this as a minimum rate, so that two nodes are neighbors whenever the rate is above a certain threshold $C_{\min}$ (in our simulations we used $C_{\min} = 0.01$).

For a set of positions of task and network agents $\mathbf{X}_T$ and $\mathbf{X}_N$, we will consider the result of problem (1) as the figure of merit of that particular configuration of the network, which we will denote as $P(\mathbf{X}_T, \mathbf{X}_N)$. Since our objective is to find the optimal position of the network agents given the task agents' positions, the problem is then

$$\mathbf{X}_N^* = \arg\max_{\mathbf{X}_N} \quad P(\mathbf{X}_T, \mathbf{X}_N). \tag{3}$$

2.2 Learning Algorithm

It is possible to attempt a direct optimization of the network agents' position problem (3). However, this problem is ill-conditioned due to the neighboring definition and the shared access constraints. This resulted in even state-of-the-art non-convex optimization tools (in particular we tried [6]) failing to provide reasonable results, not converging or simply taking too long. On the other hand, problem (1) is actually concave for a given configuration $\mathbf{X}_T, \mathbf{X}_R$ of nodes, so it is relatively straightforward to find $P(\mathbf{X}_T, \mathbf{X}_R)$.

We propose instead a Black Box Optimization (BBO) approach, where we approximate $P(\mathbf{X}_T, \mathbf{X}_R)$ through a differentiable function $\Phi(\mathbf{X}_T, \mathbf{X}_R)$, and then use this function to better position each communication node using a first-order gradient ascent method. By constructing a dataset of input configurations and solutions to (1), we can train a Neural Network (NN) model that approximates the dependency of (1)'s solution on the agents' configurations. This trained NN will serve as our $\Phi(\mathbf{X}_T, \mathbf{X}_R)$.

An important consideration for understanding why a gradient approach might be sufficient for this application is that we are continuously tracking a team of mobile agents. The dynamics involved might limit how frequently the task agents can update their positions. Thus, moving in the direction of improvement is the best course of action. Furthermore, since the task agents configurations are constantly changing, aiming for an absolute maximum might be futile.

In particular, we have chosen a Graph Neural Network (GNN) [12] because the topology of the problem allows for a good representation of our data as a graph. Furthermore, scalability properties of GNNs provide guarantees of its results when applied to larger networks [17], where even solving (1) becomes computationally expensive, an aspect we will explore in the simulations section.

Furthermore, it is important to note that, since all network modeling depends on the relative distances between nodes, problem (1) is invariant to translations, rotations, and reflections on the nodes' positions. Moreover, if properly tagged in the nodes' signal, the solution is also invariant to the task and network agents' permutation. This last feature is enforced in a standard GNN model, but not the invariance to the nodes' positions. Naturally, we may expose the learning algorithm to data samples which are rotations and translations of a given configuration (as a sort of data augmentation), but this approach would require large computational capabilities, and complete invariance would still not be assured.

In order to leverage this geometric *a priori* in our approximation, we will consider the so-called E(n) Equivariant Graph Neural Network (EGNN) [19]. This architecture stacks L Equivariant Graph Convolutional Layers (EGCL), where the input to each such layer are two vectors per node i in the graph: a node representation $\mathbf{h}_i^l$ and the node coordinates $\mathbf{x}_i^l$.

The following expressions are used to compute the output of each layer, corresponding to new coordinates and vector representation

for each node.

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, e_{ij}\right) \tag{4}$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right) \tag{5}$$

$$\mathbf{m}_i = \sum_{j\in\mathcal{N}_i}\mathbf{m}_{ij} \tag{6}$$

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right) \tag{7}$$

Equation (4) shows the dependency of the model with the squared distance between nodes (where e_{ij} is the weight of the edge between nodes i and j), which provides the equivariance to transformations in the nodes' position that preserve the distance. Functions ϕ_e and ϕ_h are the edge and node operations respectively, and are usually implemented as Multi-layer Perceptrons (MLPs). Furthermore, equation 5 is not used for our implementation as positions are not updated between EGNN layers.

2.3 Implementation

To implement our learning problem, we first model it in terms of an EGNN. Firstly, and quite naturally, each agent corresponds to a node in the graph, and the coordinates input of each node ($\mathbf{x}_i^0$) in the first layer corresponds to their position (i.e., the row of $\mathbf{X}_T$ or $\mathbf{X}_N$). Lastly, the input node's signal $\mathbf{h}_i^0$ is one-hot encoded, indicating whether the node corresponds to a task or network agent. The agents' positions were also saved as part of the graph definition. The chosen architecture consisted of two EGCL layers. Since the problem is a regression, the last stage consists of a global mean readout followed by a linear layer. Details of the implementation can be found in the publicly available code.[1]

To train the model and obtain $\Phi(\mathbf{X}_T, \mathbf{X}_N)$, we first generated 68.000 pairs of $((\mathbf{X}_T, \mathbf{X}_N), P(\mathbf{X}_T, \mathbf{X}_N))$ samples. Nodes' positions were random and uniformly generated in a square of sides' length equal to 20% more than the distance corresponding to the rate $C_{\min}$. In all cases there are 3 task agents, in half of the samples there are 2 network agents, and in the other half only one. For each configuration, the corresponding value of $P(\mathbf{X}_T, \mathbf{X}_N)$ was computed using CVXPy [1, 7] and Mosek [3]. A validation dataset to monitor the training process is generated under the same assumptions as the training set. Both PyTorch Geometric [9] and PyTorch Lightning [8] libraries were used to implement and train the model, while the EGNN layer implementation [19] was modified to allow wider MLPs.

A comparison between the obtained $\Phi(\mathbf{X}_T, \mathbf{X}_N)$ and the actual $P(\mathbf{X}_T, \mathbf{X}_N)$ is shown in Fig. 1. In this particular example, the position of the three task agents is fixed in the square's vertices (indicated by the red stars) and we compute the resulting functions as we move the network agent through a grid. We highlight that the approximation $\Phi(\mathbf{X}_T, \mathbf{X}_N)$ provided by the EGNN compares very favorable to the actual $P(\mathbf{X}_T, \mathbf{X}_N)$ as computed by CVXPy even in this corner case. Since $\Phi(\mathbf{X}_T, \mathbf{X}_N)$ is differentiable with respect to the input $\mathbf{X}_N$, we propose to move the network agents following the gradient $\nabla_{\mathbf{X}_N}\Phi(\mathbf{X}_T, \mathbf{X}_N)$, resulting thus in a first-order gradient ascent algorithm. Figure 2 overlays this gradient on the same example as that of Fig. 1, represented as an arrow for each corresponding $\mathbf{X}_N$ (recall that in this example there is a single network agent). Naturally, as in all gradient-based methods applied to functions with multiple local maxima, achieving the optimum is not guaranteed. We will nevertheless show through simulations that this method performs better in terms of the resulting $P(\mathbf{X}_T, \mathbf{X}_N)$ than other topology control algorithms, that do not take into account the access control in the wireless network.

[1] https://github.com/mdelcalaru/EGNN-TopologyControl

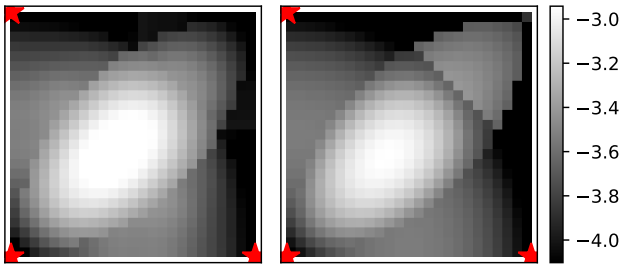

Figure 1: Example of a trained EGNN (left) and the values of $P(\mathbf{X}_T, \mathbf{X}_N)$ obtained through CVXPy (right). There are three fixed task agents (indicated with red stars) and a single network agent, and the heatmap depicts the value of $P(\mathbf{X}_T, \mathbf{X}_N)$ as a function of the network agent's position. Note how the EGNN produces good approximation.

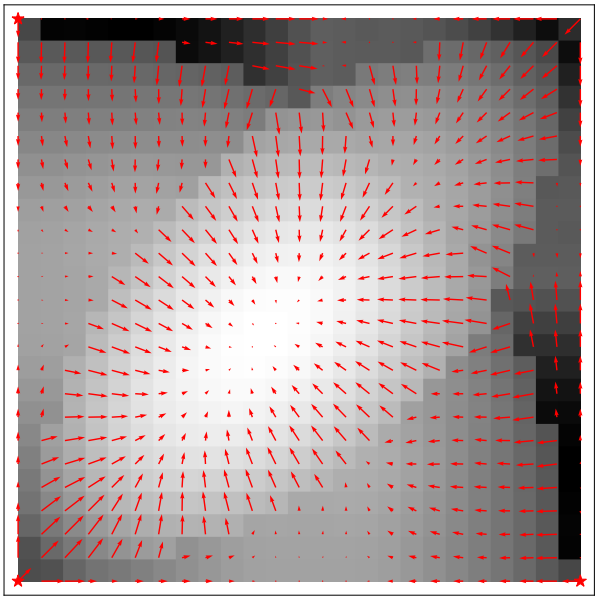

Figure 2: The gradient $\nabla_{\mathbf{X}_N}\Phi(\mathbf{X}_T, \mathbf{X}_N)$ for the same example as in Fig. 1. The proposed topology control method is that the network agent follows this gradient, constituting a first-order gradient ascent algorithm.

3 SIMULATIONS

In this section, we evaluate our proposal and compare it with two approaches. First, we consider a method based purely on the communication graph, aiming to maximize its connectivity [25]. Specifically, this algorithm selects the positions of network agents that maximize the second smallest eigenvalue of the Laplacian matrix, where the edge weights represent the rate between nodes. We will refer to this method as `max-`λ_2. Second, we examine the effect of ignoring the medium access constraints in problem (1). We will denote this approach as `No-SA-MNF` (No Shared Access MNF). This optimization problem is derived from problem (1), by removing constraints (1d) and (1f), as well as eliminating τ_{ij} from equation (1e). As a result, (3) becomes directly solvable. This model, which is more applicable to wired networks, forms the basis of several topology control algorithms. In particular, we consider the proposal detailed in [5], which, similar to our approach, performs a gradient ascent but is derived directly from the optimization problem. For our method, we utilize the gradient of the trained $\Phi(\mathbf{X}_T, \mathbf{X}_N)$, as introduced in the previous section.

In all the simulations, the final value of $\mathbf{X}_N$ is determined by the three algorithms starting from the same initial configuration. The resulting positions are evaluated through $P(\mathbf{X}_T, \mathbf{X}_N)$ (i.e. by solving (1)) and the corresponding flows' throughput (i.e. $\{a^k\}$ for $k = 1, \ldots, K$). In particular, for each example, we will randomly generate several initial configurations, and report the difference between the results obtained by our method and the other two algorithms (relative to the result obtained by our method).

Let us consider a first simple scenario, consisting of three task agents and a single network agent. As expected, and due to the topology of the problem, all algorithms obtain similar configurations, positioning the network agent approximately in the center of the three task agents; see an example in Fig. 3.

Dynamic simulations were conducted in which three task agents moved randomly, while the network agent attempted to relocate for better performance. In these simulations, the new position of each task agent was chosen within a radius of its previous position. Even with just five steps in the gradient ascent direction, the error relative to the optimal configuration was under 3%. Results are shown in Fig. 4, where our method was compared against the `max-`λ_2 approach over 10 different simulations, each running for 50 iterations of task agent position updates. The difference in performance is negligible, so the statistics for the `max-`λ_2 are omitted. Nevertheless, this experiment further demonstrates the effectiveness of our approach, highlighting that, while we applied our methodology to only five steps in the gradient ascent direction, the `max-`λ_2 computation requires a full optimization. What we want to emphasize is that performing these five gradient ascent steps is more than 10 times faster computationally than executing the complete `max-`λ_2 optimization, and this difference will only increase with the number of agents.

We acknowledge that this experiment is conducted for a simple example. However, obtaining the absolute maxima involves evaluating (1) over a grid of points, which limits the types of configurations for which this evaluation makes sense. Despite this limitation, the experiment demonstrates the utility of a gradient approach.

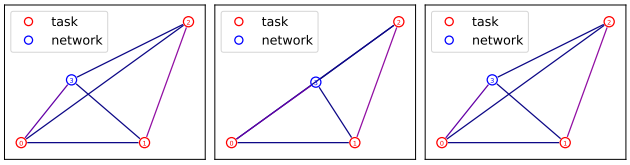

Figure 3: Example results for a scenario with 3 task agents and a single network agent corresponding to our algorithm (left), `max-`λ_2 (middle) and `No-SA-MNF` (right). All three methods obtain similar results.

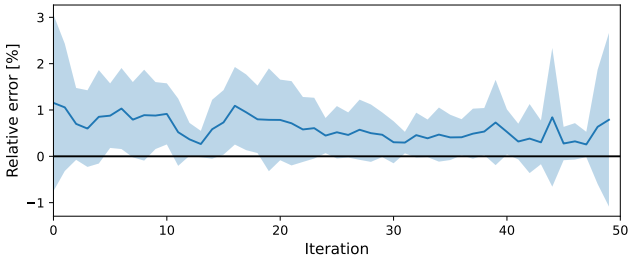

Figure 4: Relative error of the configuration obtained after 5 gradient ascent steps compared to the optimal configuration. The mean and variance of this error are shown for 10 simulations in a dynamic scenario with 3 task agents and 1 network agent. Task agents moved randomly, while the network agent's position was updated using our proposed gradient ascent method.

Another interesting case arises when we add a second network agent. Here, we start to observe the effects of including access constraints. Figure 5 illustrates an example of the final positions obtained by the three algorithms. Both `max-`λ_2 and `No-SA-MNF` tend to place both network agents similarly to the previous scenario, approximately in the center of the three task agents. However, such a configuration would result in both agents simply sharing the medium equally, effectively wasting the potential of the new network agent in terms of performance. This is confirmed in Fig. 6, which demonstrates that the vast majority of the values of both a^k and $P(\mathbf{X}_T, \mathbf{X}_N)$ are significantly increased in our method.

Finally, we tested the trained $\Phi(\mathbf{X}_T, \mathbf{X}_N)$ in configurations not seen during training to verify the transferability capacity of the EGNN. For instance, we chose teams of eight task agents and five network agents. Results are shown in Fig. 7, where we can see that our method still obtains significantly better results in most of the examples. It is important to consider that the `max-`λ_2 uses the current configuration for optimization, while our method was not trained with configurations such as this. Therefore, it is remarkable that our implementation mostly outperforms the others when we look at the values of $P(\mathbf{X}_T, \mathbf{X}_N)$. For a^k, it is important to note that in a network configuration, certain flows will decrease as overall performance improves (strongly depending on the notion of fairness enforced by the chosen $U(\cdot)$ in (1)).

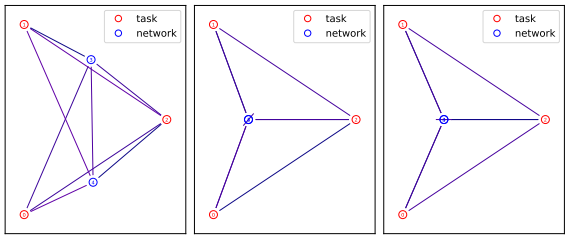

Figure 5: Example results for a scenario with 3 task agents and two network agents corresponding to our algorithm (left), `max-`λ_2 (middle) and `No-SA-MNF` (right). The latter two place both network agents together, resulting in both evenly sharing the wireless medium and thus wasting the new network agent, since they do not consider the access constraints imposed by the wireless communication system.

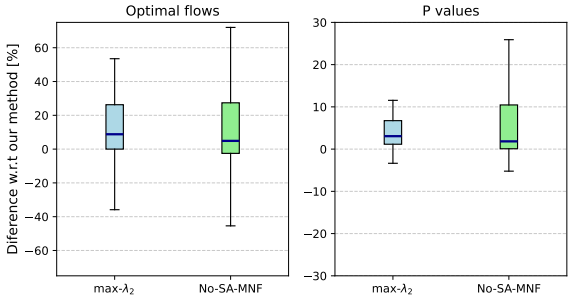

Figure 6: Performance statistics for 50 simulations on the scenario with 3 task agents and two network agent. Each datum in the boxplot is the relative difference (with respect to our method) in terms of a_i^k (left) and $P(\mathbf{X}_T, \mathbf{X}_N)$ (right). As illustrated by Fig. 5, our method typically obtains better results.

Thus, we find that some net flows a^k exhibit worse performance, although the overall $P(\mathbf{X}_T, \mathbf{X}_N)$ is increased (specially when compared with `No-SA-MNF`). Furthermore, our solution gracefully scales with the number of agents, taking a much shorter time to compute its output positions. It is expected that the performance of this methodology will improve when trained for larger configurations.

4 CONCLUSIONS

We proposed a novel approach to topology control for a team of mobile agents tasked with providing communication infrastructure for a separate team of task-performing agents. Our method integrates wireless medium access control into a multicommodity network flow problem, resulting in a highly non-convex optimization challenge. To exploit the symmetries in the problem, we employ an E(n) Equivariant Graph Neural Network (EGNN) to approximate our figure of merit and design a first-order gradient ascent algorithm. This approach is computationally efficient and scalable across various configurations, including scenarios with multiple agents. Our

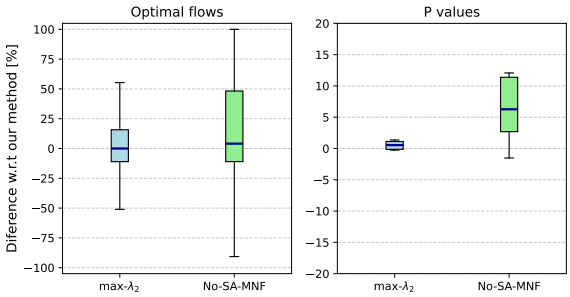

Figure 7: Performance statistics for 10 simulations with 8 task agents and 5 network agents. Each boxplot datum shows the relative difference (compared to our method) in net flows a_i^k (left) and $P(\mathbf{X}_T, \mathbf{X}_N)$ (right). Although the scenario was not seen during training, our system still produces better outcomes in terms of $P(\mathbf{X}_T, \mathbf{X}_N)$.

simulations demonstrate that our topology control algorithm significantly outperforms previous methods in enhancing network performance.

Despite these promising results, our study has several limitations. These include the modeling of both the communication channel and the access restrictions, as well as the fact that the computation of the solution is centralized, which may limit scalability in distributed systems. Future research could explore more sophisticated channel models, further refine the graph representation to enhance our methodology, decentralize the computation for scalability, and investigate larger agent configurations in both training and testing.

REFERENCES

[1] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2018. A rewriting system for convex optimization problems. *Journal of Control and Decision* 5, 1 (2018), 42–60.

[2] Stéphane Alarie, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. 2021. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization* 9 (2021), 100011. https://doi.org/10.1016/j.ejco.2021.100011

[3] MOSEK ApS. 2024. *MOSEK Optimizer API for Python 10.2.1.* https://docs.mosek.com/latest/pythonapi/index.html

[4] Yawen Barowski, Saad Biaz, and Prathima Agrawal. 2005. Towards the performance analysis of IEEE 802.11 in multi-hop ad-hoc networks. In *IEEE Wireless Communications and Networking Conference, 2005*, Vol. 1. 100–106 Vol. 1. https://doi.org/10.1109/WCNC.2005.1424483

[5] Ignacio Boero, Igor Spasojevic, Mariana del Castillo, George Pappas, Vijay Kumar, and Alejandro Ribeiro. 2023. Navigation with Shadow Prices to Optimize Multi-Commodity Flow Rates. In *2023 62nd IEEE Conference on Decision and Control (CDC)*. 253–258. https://doi.org/10.1109/CDC49753.2023.10383300

[6] Ju Sun Buyun Liang, Tim Mitchell. 2022. NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning. (2022). arXiv:2210.00973 [cs.LG]

[7] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17, 83 (2016), 1–5.

[8] William Falcon and The PyTorch Lightning team. 2019. *PyTorch Lightning.* https://doi.org/10.5281/zenodo.3828935

[9] Matthias Fey and Jan Eric Lenssen. 2019. *Fast Graph Representation Learning with PyTorch Geometric.* https://github.com/pyg-team/pytorch_geometric

[10] Jonathan Fink. 2011. *Communication for teams of networked robots.* Ph. D. Dissertation. University of Pennsylvania.

[11] Jonathan Fink, Alejandro Ribeiro, and Vijay Kumar. 2012. Robust Control for Mobility and Wireless Communication in Cyber–Physical Systems With Application to Robot Teams. *Proc. IEEE* 100, 1 (2012), 164–178. https://doi.org/10.1109/JPROC.2011.2161427

[12] Fernando Gama, Elvin Isufi, Geert Leus, and Alejandro Ribeiro. 2020. Graphs, Convolutions, and Neural Networks: From Graph Filters to Graph Neural Networks. *IEEE Signal Processing Magazine* 37, 6 (2020), 128–138. https://doi.org/10.1109/MSP.2020.3016143

[13] Andrea Goldsmith. 2005. *Wireless communications.* Cambridge university press.

[14] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998), 237–252.

[15] Daniel Mox, Miguel Calvo-Fullana, Mikhail Gerasimenko, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. 2020. Mobile Wireless Network Infrastructure on Demand. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 7726–7732. https://doi.org/10.1109/ICRA40945.2020.9197460

[16] Shahbaz Rezaei, Mohammed Gharib, and Ali Movaghar. 2018. Throughput Analysis of IEEE 802.11 Multi-Hop Wireless Networks With Routing Consideration: A General Framework. *IEEE Transactions on Communications* 66, 11 (2018), 5430–5443. https://doi.org/10.1109/TCOMM.2018.2848905

[17] Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. 2023. Transferability Properties of Graph Neural Networks. *IEEE Transactions on Signal Processing* 71 (2023), 3474–3489. https://doi.org/10.1109/TSP.2023.3297848

[18] Khodakaram Salimifard and Sara Bigharaz. 2022. The multicommodity network flow problem: state of the art classification, applications, and solution methods. *Operational Research* (2022), 1–47.

[19] Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. 2021. E(n) Equivariant Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 9323–9332.

[20] Gail Schofield, Nicole Esteban, Kostas A. Katselidis, and Graeme C. Hays. 2019. Drones for research on sea turtles and other marine vertebrates – A review. *Biological Conservation* 238 (2019), 108214. https://doi.org/10.1016/j.biocon.2019.108214

[21] Gail Schofield, Kostas A. Katselidis, Martin K. S. Lilley, Richard D. Reina, and Graeme C. Hays. 2017. Detecting elusive aspects of wildlife ecology using drones: New insights on the mating dynamics and operational sex ratios of sea turtles. *Functional Ecology* 31, 12 (2017), 2310–2319. https://doi.org/10.1111/1365-2435.12930 arXiv:https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/1365-2435.12930

[22] James Stephan, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. 2017. Concurrent Control of Mobility and Communication in Multirobot Systems. *IEEE Transactions on Robotics* 33, 5 (2017), 1248–1254. https://doi.org/10.1109/TRO.2017.2705119

[23] David Tse and Pramod Viswanath. 2005. *Fundamentals of wireless communication.* Cambridge university press.

[24] Yuan Yan and Yasamin Mostofi. 2012. Robotic Router Formation in Realistic Communication Environments. *IEEE Transactions on Robotics* 28, 4 (2012), 810–827. https://doi.org/10.1109/TRO.2012.2188163

[25] Michael M Zavlanos, Magnus B Egerstedt, and George J Pappas. 2011. Graph-theoretic connectivity control of mobile robot networks. *Proc. IEEE* 99, 9 (2011).