

Learning to Identify Graphs from Node Trajectories in Multi-Robot Networks

Eduardo Sebastián

Thai Duong

Nikolay Atanasov

Eduardo Montijano

Carlos Sagüés

Abstract—The graph identification problem consists of discovering the interactions among nodes in a network given their state/feature trajectories. This problem is challenging because the behavior of a node is coupled to all the other nodes by the unknown interaction model. Besides, high-dimensional and nonlinear state trajectories make it difficult to identify if two nodes are connected. Current solutions rely on prior knowledge of the graph topology and the dynamic behavior of the nodes, and hence, have poor generalization to other network configurations. To address these issues, we propose a novel learning-based approach that combines (i) a strongly convex program that efficiently uncovers graph topologies with global convergence guarantees and (ii) a self-attention encoder that learns to embed the original state trajectories into a feature space and predicts appropriate regularizers for the optimization program. In contrast to other works, our approach can identify the graph topology of unseen networks with new configurations in terms of number of nodes, connectivity or state trajectories. We demonstrate the effectiveness of our approach in identifying graphs in multi-robot formation and flocking tasks.

I. INTRODUCTION

The study of networked systems is essential in many disciplines like brain network imaging [1]–[3], genetics [4], [5], power networks [6], social networks [6]–[9], environmental monitoring [7], [10], [11], or general large-scale physically interconnected systems [12], [13]. The interactions among entities play a central role in understanding networked systems and motivates an extensive effort to identify the interactions (i.e., the graph topology) from data [14]–[16]. For example, graph topology identification is crucial for modeling multi-robot interactions [17] in learning collaborative behaviors from observations or demonstrations [18]–[20]. In this paper, we aim to develop an algorithm to identify the underlying graph topology that best describes the behavior of a networked system given its node trajectories.

A widely used approach for graph topology identification is graph signal processing [21]. Diffusion-based methods [6], [11] assume that the node signals diffuse through the edges following heat kernels. The proposed approaches are posed as minimization problems over the heat kernels’ parameters

E. Sebastián, E. Montijano and C. Sagüés are with the RoPeRt group, at DIIS - I3A, Universidad de Zaragoza, Spain (e-mails: {esebastian, emonti, csagues}@unizar.es).

T. Duong and N. Atanasov are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mails: {tduong, natanasov}@ucsd.edu).

This work has been supported by NSF CCF-2112665 (TILOS) and via Spanish projects PID2021-125514NB-I00, PID2021-124137OB-I00 and TED2021-130224B-I00 funded by MCIN/AEI/10.13039/501100011033, by ERDF A way of making Europe and by the European Union NextGenerationEU/PRTR, DGA T45-23R, and Spanish grant FPU19-05700 and EST22/00253.

and the Laplacian matrix. However, the constraints on Laplacian matrix are non-convex, and an a priori dictionary of functions is required to find the kernels’ parameters. Another line of research focuses on the reconstruction of a graph shift operator by means of its eigenvalue and eigenvector pairs, under perfect observations and known eigenvectors [10], [22] or linear time-invariant single-input single-output nodes [23]. A common issue is how to encode the constraints of adjacency and Laplacian matrices [24], which lead to NP-hard problems. Learning a Laplacian can be recast to a L1 norm minimization by exploiting a smoothness assumption [25], [26]. This perspective has been extended to develop fast online algorithms [27], [28] which benefit from the fact that the L1-norm and the Laplacian constraints can be rewritten as a vectorized multiplication and an indicator function, leading to unconstrained convex problems [29]. Nevertheless, these algorithms assume scalar node signals, which is not the usual case in general networked systems.

A promising alternative to graph signal processing is machine learning algorithms that use attention [30] and self-attention [31] mechanisms. Self-attention discovers the relationships among elements of a sequence by computing an attention map, and its structure can be combined with linear layers and activation functions to encode nonlinear behaviors. Furthermore, attention layers allow for a time-varying size of one of the input dimensions, e.g., the number of robots in a robotic team. Regarding multi-robot systems, self-attention layers can be found in recent path planning [32], [33] or task scheduling [34] applications. It is worth mentioning that there are learning techniques related to graphs which cannot be applied to our problem because they assume a known graph: relation prediction, graph regression, clustering [35], and graph neural network solutions [36], [37].

We formulate the problem of identifying the graph topology of a networked system from state trajectories of its nodes (Section II). Our main contribution is a graph topology identification approach (Section III) that captures high-dimensional node features in a strongly convex optimization problem for weighted adjacency matrix optimization. In contrast with learning-based techniques, our approach exploits principled strongly convex optimization to generate weighted adjacency matrices with guarantees of convergence. Compared to graph signal processing techniques, our approach relies on a self-attention-based neural networks to represent high-dimensional node features for the adjacency matrix optimization problem. The neural networks are trained to balance between the loss function and the regularizers in the objective function so that it best captures the sparsity

of the graph. The approach is validated through multi-robot formation and flocking experiments in Section IV, and the the benefits of our proposal are discussed in Section V.

II. PROBLEM FORMULATION

Consider a networked system characterized by an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. The set of nodes is $\mathcal{V} = \{1, \dots, n\}$, with $n > 1$ the number of nodes. The interactions among nodes are represented by the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ represents the intensity of the interactions among nodes, and it is such that $[\mathbf{W}]_{ij} = w_{ij} \in \mathbb{R}_{>0}$ if $(i, j) \in \mathcal{E}$ and $[\mathbf{W}]_{ij} = 0$ otherwise. We assume that there are no self-loops, i.e., $w_{ii} = 0$. The set of neighbors of node i is $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. Since \mathcal{G} is undirected, $(i, j) \in \mathcal{E}$ implies that $(j, i) \in \mathcal{E}$ and $w_{ij} = w_{ji}$. We define the edge density as $\rho(\mathcal{G}) = |\mathcal{E}|/n^2$, with $|\mathcal{E}|$ the number of edges. The (weighted) Laplacian is $\mathbf{L} = \text{diag}(\mathbf{W}\mathbf{1}_n) - \mathbf{W}$, where $\mathbf{1}_n$ is the column vector of ones of size n .

Node i is characterized by a state $\mathbf{x}_i(t) \in \mathcal{X}_i \subseteq \mathbb{R}^s$ at a discrete time $t \in \mathbb{N}$, where \mathcal{X}_i is the space of admissible states of dimension $s \in \mathbb{N}$. Each node obeys unknown discrete-time dynamics,

$$\mathbf{x}_i(t+1) = f_i(\mathbf{x}_i(t), \mathbf{x}_{\mathcal{N}_i}(t)), \quad (1)$$

where $\mathbf{x}_{\mathcal{N}_i}(t) = \{\mathbf{x}_j(t)\}_{j \in \mathcal{N}_i}$ is the state of the neighbors of node i at time t . Let $\mathbf{x}_i^d(t) = [\mathbf{x}_i(t-d), \dots, \mathbf{x}_i(t)]$ be the trajectory formed by the last d states of node i at instant t . We define the tensor $\mathbf{X}(t) = [\mathbf{x}_1^d(t), \dots, \mathbf{x}_n^d(t)]$ as the $n \times s \times d$ collection of trajectories at instant t . We assume that these trajectories are available at each instant, and the associated \mathbf{W} is time-invariant. The aim of this work is to develop an approach to identify \mathbf{W} given $\mathbf{X}(t)$.

We assume that the evolution of the node states is such that the graph state is smooth. Roughly speaking, the smoothness assumption implies that the neighbors of node i have more similar values of $\mathbf{x}_i(t)$ compared to non-neighboring nodes. Formally, the total variation of the graph state is

$$\varphi(\mathbf{x}(t)) = \mathbf{x}^\top(t)(\mathbf{L} \otimes \mathbf{I}_s)\mathbf{x}(t), \quad (2)$$

where $\mathbf{x}(t) = [\mathbf{x}_1^\top(t), \dots, \mathbf{x}_n^\top(t)]^\top$, \otimes is the Kronecker product, and \mathbf{I}_s is the identity matrix of dimension s . Then, we define graph state smoothness as follows.

Definition 1. *The state trajectories of graph \mathcal{G} are smooth if $\varphi(\mathbf{x}(t))$ satisfies $\varphi(\mathbf{x}(t)) < \sigma$, with $0 < \sigma \ll \varrho$ and $\varrho \in \mathbb{R}_{>0}$ an upper bound on the total variation of the graph state.*

We assume that the evolution of the node trajectories given by Eq. (1) is such that graph state smoothness holds, which is the case for many multi-agent and multi-robot problems such as flocking, cooperative exploration, opinion dynamics, and consensus [38]–[41]. The particular values for σ and ϱ depend on the application. Under these assumptions, the goal of the paper is formulated as follows.

Problem 1. *Given smooth node state trajectories $\mathbf{X}(t)$ obtained from a graph \mathcal{G} with unknown dynamics $f_i(\bullet) \forall i$ and edges \mathcal{E} , find the weighted adjacency matrix \mathbf{W} of \mathcal{G} .*

III. LEARNING TO IDENTIFY GRAPHS

In this section, we propose an approach to solve Problem 1 that combines (i) a fast strongly convex optimization algorithm with global convergence guarantees and (ii) a self-attention encoder that transforms high-dimensional node states to one-dimensional features and learns the regularization parameters that match the graph sparsity pattern. We describe the optimization algorithm in Section III-A and develop the self-attention encoder in Section III-B. In Section III-C, we describe our overall approach for graph topology learning from node trajectories.

A. Accelerated graph learning from smooth signals

In a networked system, the behavior of one node is influenced by the entire network by means of local interactions with neighboring nodes via Eq. (1). Hence, to identify the existence of interactions among two nodes we need to consider, at least initially, the state information from all the nodes. In this section, we describe an optimization method [28] for scalar states ($s = 1$) that finds the underlying graph that best describes the state trajectories of the nodes. When state \mathbf{x}_i is scalar, the trajectory $\mathbf{X}(t)$ is an $n \times d$ matrix.

First, we define $\mathbf{Y}(t) \in \mathbb{R}^{n \times n}$ as the Euclidean distance matrix among the node states such that $[\mathbf{Y}(t)]_{ij} = \|\mathbf{x}_i^d(t) - \mathbf{x}_j^d(t)\|_2^2 \forall i, j \in \mathcal{V}(t)$. Then,

$$\varphi(\mathbf{x}(t)) = \text{trace}(\mathbf{X}^\top(t)\mathbf{L}\mathbf{X}(t)) = \frac{1}{2} \|\mathbf{W} \circ \mathbf{Y}(t)\|_1, \quad (3)$$

where \circ denotes element-wise product and $\|\cdot\|_1$ is the L1 norm. We omit the time dependence going forward to simplify the notation because all operations refer to the same instant t . Based on Eq. (3), finding \mathbf{W} translates into solving a convex inverse problem [25]:

$$\min_{\mathbf{W}} \|\mathbf{W} \circ \mathbf{Y}\|_1 - \alpha \mathbf{1}_n^\top \log(\mathbf{W}\mathbf{1}_n) + \beta \|\mathbf{W}\|_F^2 \quad (4a)$$

$$\text{s.t. } w_{ii} = 0, w_{ij} \geq 0, \forall i, j \in \mathcal{V}, \quad (4b)$$

where $\alpha, \beta > 0$ are tuning parameters such that α penalizes the possibility of isolated nodes and β encourages graph sparsity. To solve (4) efficiently and with global convergence guarantees, Saboksayr and Mateos [28] proposed a dual reformulation of (4). Since the graph is undirected, we define $\mathbf{w} = \text{vech}(\mathbf{W})$ as the half-vectorization of \mathbf{W} . Let $\mathbf{y} = \text{vec}(\mathbf{Y})$ be the vectorization of \mathbf{Y} and \mathbb{I} be the indicator function such that $\mathbb{I}\{\mathbf{w} \geq 0\} = 0$ and ∞ otherwise, where the operation \geq is applied element-wise. Finally, $\mathbf{S} \in \{0, 1\}^{n \times \frac{n(n-1)}{2}}$ is the operator that satisfies $\mathbf{W}\mathbf{1}_n = \mathbf{S}\mathbf{w}$. This leads to the unconstrained optimization problem

$$\min_{\mathbf{w}} \mathbb{I}\{\mathbf{w} \geq 0\} + 2\mathbf{w}^\top \mathbf{y} + \beta \|\mathbf{w}\|_2^2 - \alpha \mathbf{1}_n^\top \log(\mathbf{S}\mathbf{w}). \quad (5)$$

Eq. (5) can be solved by accelerated dual proximal gradient methods with guarantees of global convergence, proved by Beck and Teboulle [29] in the general case, and by Saboksayr and Mateos [28] in the specific case of Eq. (5). In particular, we exploit the analytical expressions derived in [28], summarized in Algorithm 1.

Algorithm 1 Accelerated dual proximal gradient method for graph identification

1: **Inputs:** \mathbf{y}, \mathbf{S}
 2: **Initialization:** $\omega_0 = \lambda_0 = \lambda_{-1} \sim \mathcal{U}(\frac{n(n-1)}{2})$, $\tau_0 = 1$
 3: **Parameters:** $K > 0$, $\alpha > 0$, $\beta > 0$, $L = \frac{n-1}{\beta}$, $\epsilon > 0$
 4: **for** $k \in \{0, \dots, K-1\}$ **do**
 5: $\mathbf{w}_k = \max\left(\mathbf{0}, \frac{\mathbf{S}^\top \omega_k - 2\mathbf{y}}{2\beta}\right)$
 6: $\mathbf{u}_k = \frac{1}{2} \left(\mathbf{S}\mathbf{w}_k - L\omega_k + \sqrt{(\mathbf{S}\mathbf{w}_k - L\omega_k)^2 + 4\alpha L\mathbf{1}_n} \right)$
 7: $\lambda_k = \omega_k - L^{-1}(\mathbf{S}\mathbf{w}_k - \mathbf{u}_k)$
 8: $\tau_{k+1} = \frac{1 + \sqrt{1 + 4\tau_k^2}}{2}$
 9: $\omega_k = \lambda_k + \frac{\tau_k - 1}{\tau_{k+1}}(\lambda_k - \lambda_{k-1})$
 10: **if** $\|\lambda_k - \lambda_{k-1}\| / \|\lambda_{k-1}\| < \epsilon$ **then break**
 11: **end for**
 12: **Output:** \mathbf{w}_k

The algorithm iterates until convergence or a certain number of iterations $K \in \mathbb{N}$ is reached. The current iteration is denoted by k and λ_k denotes the Lagrange multipliers of the dual optimization problem of Eq. (5). Algorithm 1 alternates between weighted adjacency matrix and Lagrange multipliers' updates. Once the algorithm terminates, the graph adjacency matrix is recovered from \mathbf{w}_k . Interestingly, the stability of the training process is not affected by the max operator in line 5 of Algorithm 1. At the boundary $\frac{\mathbf{S}^\top \omega_k - 2\mathbf{y}}{2\beta} = \mathbf{0}$, the gradient can be smooth by, e.g., using an approximated value like in Pytorch [42].

Besides the global convergence [28], which implies robustness against initializations, Algorithm 1 is efficient to compute so, at each instant, sufficient iterations can be run to ensure convergence to the weighted adjacency matrix that best describes the node state trajectories defined over the Euclidean space.

B. Self-attention encoder

To use Algorithm 1 for graph identification, the node state trajectories must be encoded to a feature space of dimension $s = 1$. On the other hand, the cost function in Eq. (5) is taken with respect to the distance between state trajectories in Euclidean space, so the solution might not reflect the best topology given $\mathbf{X}(t)$ because the interactions may be determined by state proximity in a different space. Therefore, the encoding must be such that the Euclidean distance is the one that best reflects the distances among node trajectories. In addition, the encoding must handle a time-varying number of nodes and adjust to changes in the graph connectivity. To address all these points, in this section we develop a neural network encoder using self-attention [30] to extract suitable features from the node state trajectories.

The encoder architecture is illustrated in Fig. 1. First, a fully connected network projects a node state to a feature state of dimension $s = 1$. The input is $\mathbf{x}_i^d(t)$, so the network is applied to every node state and trajectory. The result is a matrix $\mathbf{X}'(t)$ that is $n \times d$ as required by the optimization problem in Eq. (5). After that, a self-attention

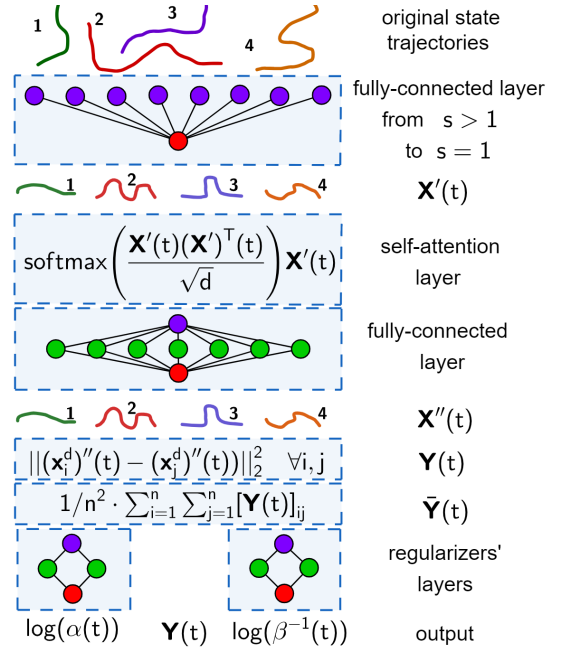


Fig. 1: The state trajectory encoder consists of four blocks: (i) a fully connected layer that encodes each individual state to a feature state of dimension $s = 1$; (ii) a self-attention layer that finds the relationships among features; (iii) a fully connected series of layers that find $\mathbf{Y}(t)$; and (iv) two separate fully connected layers to find the regularization parameters $\alpha(t)$ and $\beta(t)$ for the subsequent graph identification optimization.

layer processes $\mathbf{X}'(t)$ according to the relationships found in the attention map $\mathbf{X}'(t)(\mathbf{X}'(t))^T(t)/\sqrt{d}$, using the operation $\text{softmax}(\mathbf{X}'(t)(\mathbf{X}'(t))^T(t)/\sqrt{d})\mathbf{X}'(t)$ with $\mathbf{X}'(t)$ the query, key and value matrices. Finally, another fully connected network processes every $(\mathbf{x}_i^d)'(t)$ to obtain a new matrix $\mathbf{X}''(t)$ that is $n \times d$, composed by the individual trajectories $(\mathbf{x}_i^d)''(t)$. These feature trajectories are then used to compute the distance matrix $\mathbf{Y}(t)$. The encoder has an additional module that uses the mean of $\mathbf{Y}(t)$, $\bar{\mathbf{Y}}(t) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n [\mathbf{Y}(t)]_{ij}$, to compute the parameters $\alpha(t)$ and $\beta(t)$ of Algorithm 1. The logarithm is considered because sparsity is determined by the difference in the order of magnitude between $\alpha(t)$ and $\beta(t)$.

The encoder network, thanks to the self-attention structure, handles graphs of different configurations in the number of nodes and the intensity of interactions. Furthermore, it not only projects the state trajectories to a convenient feature space, but also provides the parameters for Algorithm 1. Thus, the encoder can be trained to adjust the connectivity depending on the state trajectories. For instance, in a multi-robot flocking task like the one used for evaluation in Section IV, the robots can depart from a spread initial condition and gather in a more compact formation. We remark that the design of the self-attention encoder is not limited to the proposed architecture. Depending on the complexity of the task, the encoder can be increased in depth and number of parameters to ensure that the dimensionality reduction captures all the behaviors of the multi-robot team.

C. Learning to identify graphs

The combination of the fast convex optimization algorithm described in Section III-A and the encoder detailed in Sec-

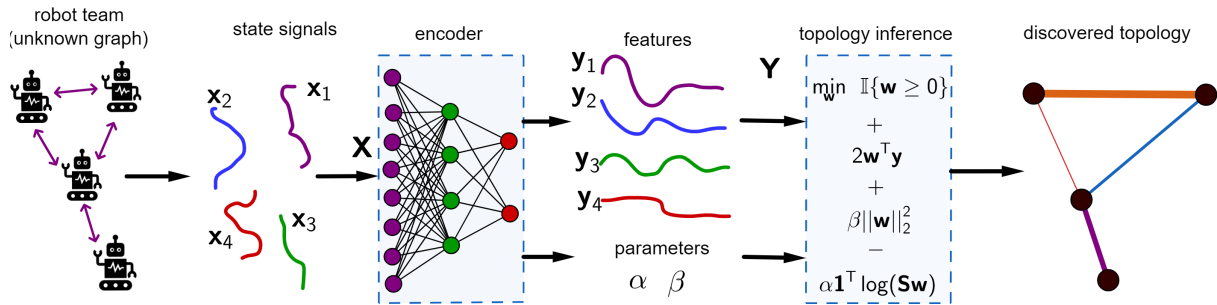


Fig. 2: A time-varying graph with unknown connectivity and node dynamics generates a dataset of trajectories (left). A self-attention encoder generates node trajectories in a feature space and computes the regularization parameters. These outputs are the input for an adjacency matrix optimization problem (middle). A fast strongly convex optimization algorithm identifies the weighted adjacency matrix that best describes the observed node trajectories (right).

tion III-B leads to a learning architecture for graph topology identification from node state trajectories, presented in Fig. 2. The self-attention encoder receives the state trajectories of the multi-robot team. The outputs of the encoder are the one-dimensional feature trajectories of the multi-robot team and the α and β regularizers of Algorithm 1. Algorithm 1 is then executed using the feature trajectories as input, providing the discovered topology. To train the model, we use the following loss function:

$$\mathcal{L}(t) = |\mathbf{w}_k(t) - \hat{\mathbf{w}}| \mathbf{1}_{n(n-1)/2} + |\mathbf{w}_k^*(t) - \hat{\mathbf{w}}^*| \mathbf{1}_{n(n-1)/2}. \quad (6)$$

In the loss function, $\hat{\mathbf{w}}$ refers to the vectorized form of the ground-truth weighted adjacency matrix \mathbf{W} . Moreover, $(\bullet)^*$ denotes the adjoint of a graph. More precisely, \mathbf{W}^* is the weighted adjacency matrix such that $[\mathbf{W}^*]_{ij} = 0$ if $[\mathbf{W}]_{ij} \neq 0$, $[\mathbf{W}^*]_{ij} > 0$ if $[\mathbf{W}]_{ij} = 0$, and $\mathbf{W}^* \mathbf{1}_n = \mathbf{W} \mathbf{1}_n$. We compute each $[\mathbf{W}^*]_{ij} > 0$ as $[\mathbf{W} \mathbf{1}_n]_i / n_i^*$, where n_i^* is the number of non-zero elements of the i -th row of \mathbf{W}^* . Thus, \mathbf{w}_k^* and $\hat{\mathbf{w}}^*$ refer to the adjoints of the identified and ground-truth weighted adjacency matrices. The use of the difference of adjoint graphs is to avoid degenerate solutions. For instance, if the ground-truth graph is very sparse, the training might tend to overfit to a graph with no edges unless the adjoint difference is part of the loss. We consider that each iteration of Algorithm 1 is a training step.

Finally, one consideration is in order. Algorithm 1 and optimizations (4)-(5) all provide the optimal graph topology in the state smoothness sense. However, there is one reason for using Algorithm 1 instead of (4) and (5) beyond the fast convergence. The training of the proposed neural network requires to backpropagate the gradients of the loss function with respect to the output through the complete neural network. In our case, the gradients are propagated backward through the steps of Algorithm 1 and the self-attention encoder. Our proposed approach benefits from the fact that these steps are analytical equations rather than an optimization problem as (4) and (5). They are solved by gradient based methods, so, to backpropagate through them, we would need to the gradients of the optimization gradients, which are computationally intractable and numerically unstable. Besides, (4) has constraints and (5) as a non-differentiable indicator function, leading to additional difficulties for the training of the neural network.

IV. EVALUATION

We conduct two types of experiments¹. The first experiment (Section IV-A) considers a multi-robot formation task to verify the ability to extrapolate to other formations and number of robots. The second experiment (Section IV-B) considers a flocking task to study how the learned neural network extrapolates to other flocking initializations, graph-densities and emergent topologies.

A. Graph identification in multi-robot formation tasks

Following the setup proposed in many graph identification works [6], [7], [22], [24], [26]–[28], [43]–[45], we consider a multi-robot formation problem where the position of the robots is given by an Erdős-Rényi random graph [46]. For the training of the neural network, a single graph is instantiated with edge probability $p = 0.2$ and a number of robots $n = 50$, to study the performance of the proposed approach in a sparse and large multi-robot network. We note that the edge density of the graph $\rho(\mathcal{G}) = p$. Different from other works, instead of considering scalars, the robot state is determined by their 2D position, i.e., $s = 2$. The states are generated using the following Gaussian random process:

$$\mathbf{x}_x(t) \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{L}}^\dagger + \sigma \mathbf{I}_n) \text{ and } \mathbf{x}_y(t) \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{L}}^\dagger + \sigma \mathbf{I}_n), \quad (7)$$

where $\hat{\mathbf{L}}^\dagger$ is the pseudoinverse of the ground-truth Laplacian, $\sigma = 0.1$ is the noise level that simulates potential uncertainty in the position of the robots, and $\mathbf{x}_x(t)$ and $\mathbf{x}_y(t)$ refer to the x -position and y -position of the robots at instant t . The covariance is chosen in this way to force that the position of the robots at each instant is correlated to their neighbors, following the state-of-the-art [7], [28] (see [7] for further details). We generate $d = 10000$ samples for training. The other training hyperparameters are detailed in Appendix I. To assess the performance of our approach, we randomly generate 18 test sets composed by 20 Erdős-Rényi random graphs each, generated from the combination of the following hyperparameters: $n = \{10, 20, 30, 40, 50, 60\}$ and $p = \{0.1, 0.2, 0.4\}$. All of them are formed by $d = 10000$ samples. We set $K = 2000$ to ensure convergence. To assess performance, we use the Mean Absolute Error (MAE) between the ground-truth and the identified weights

¹<https://eduardosebastianrodriguez.github.io/LIGMRS/>

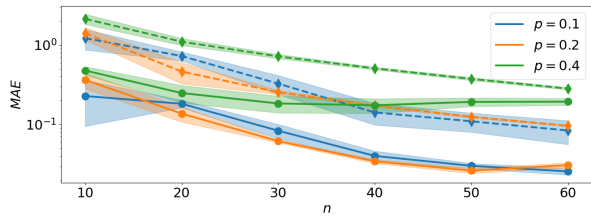


Fig. 3: MAE as a function of the number of robots and edge probability. Each configuration is run 20 times, computing the mean and standard deviation. Diamond dashed lines are the state-of-the-art [28] ($\alpha = 0.2$ and $\beta = 0.0001$), whereas the circle solid lines are ours.

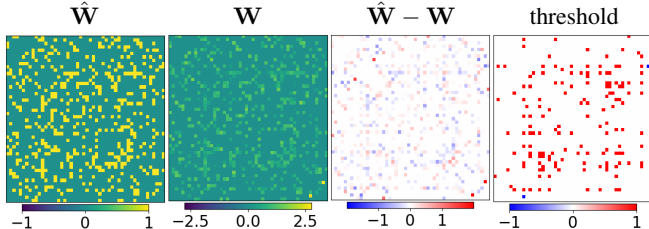


Fig. 4: After the training, the learned neural network correctly identifies the ground-truth graph. The intensity of the colors in the discovered graphs is lower because there are a few outlier weights with a greater value compared to the ground-truth. The right panel shows the difference between identified and ground-truth weighted adjacency matrices, but with a threshold $|\hat{W}_{ij} - W_{ij}| < 10^{-5}$.

$\frac{1}{n^2} \sum_i \sum_j^n |[\hat{W}]_{ij} - [W]_{ij}|$. We compare our proposal with the state-of-the-art algorithm in [28], which is the closest to our problem assumptions, tuning the parameters according to the procedures detailed in [25]–[27]. Since this algorithm only allows scalar trajectories, we apply their algorithm once per state dimension, computing the average graph.

Fig. 3 shows the MAE for the different number of robots and edge probabilities. Our proposed approach surpasses the state-of-the-art in one order of magnitude for all the configurations. It is seen how the learned neural network generalizes to different number of robots, achieving a similar performance in terms of MAE. Therefore, the training has been able to learn a good encoding of the state trajectories and identifies a good α and β . We emphasize that the training is conducted with just one graph. The learned neural network does not generalize to $\rho(\mathcal{G}) = p = 0.4$ since only a single graph with $\rho(\mathcal{G}) = p = 0.2$ has been used for training.

Figs. 4, 5 and 6 show some qualitative results. The weights of the existing edges are accurately identified in graphs with a similar number of robots than in training (Fig. 4), different numbers of robots (Fig. 5) and edge densities (Fig. 6). The intensity of the colors in the discovered graphs is lower than in the ground-truth graphs because there are a few outlier weights with a greater value compared to the ground-truth, so the color scales of \mathbf{W} and $\hat{\mathbf{W}}$ are different. Since the $\hat{\mathbf{W}}$ from the Erdős-Rényi graphs are such that $[\hat{\mathbf{W}}]_{ij} = \{0, 1\}$ and to verify that our approach is successful at capturing the existence or absence of edges, the right panels of Figs. 4, 5 and 6 show the difference between identified and ground-truth weighted adjacency matrices, but thresholding to zero all the elements such that $|\hat{W}_{ij} - W_{ij}| < 10^{-5}$. Note that this threshold is to remove edges whose order of magnitude is far from the weights of an existing edge. With a correct threshold, the existence or absence of edges is correctly

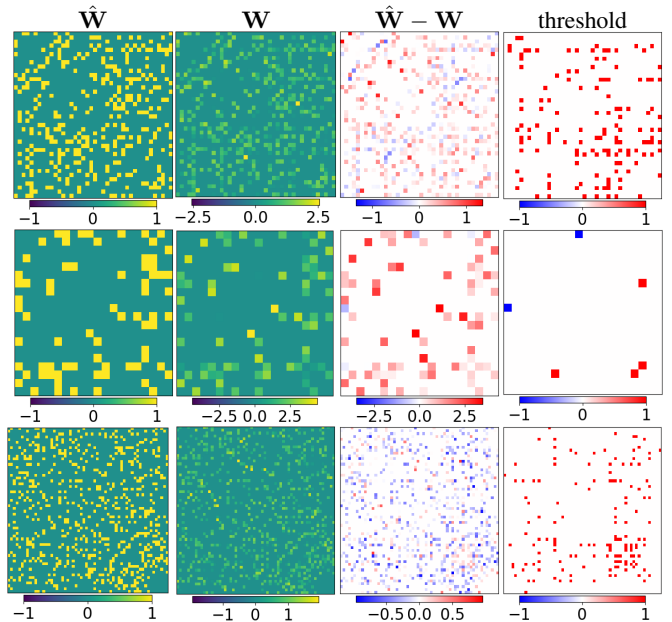


Fig. 5: Qualitative results obtained by our approach for different number of robots. Our approach is accurate in the identification of the weights irrespective of the number of robots.

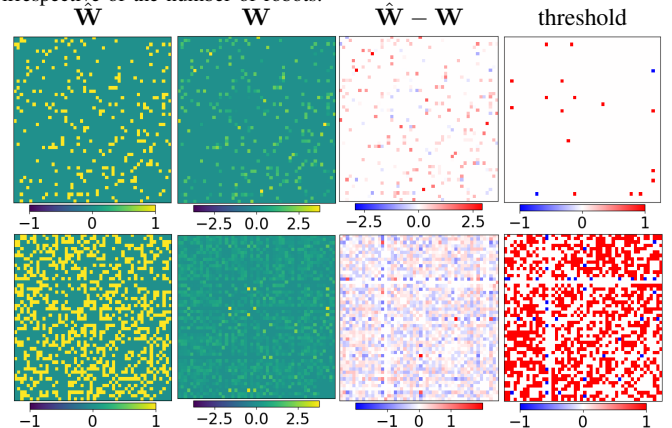


Fig. 6: Our approach moderately generalizes to other edge densities during evaluation because the training set only consists of a single graph with $p = 0.2$. Top row shows a case with $p = 0.1$, and the bottom row shows a case with $p = 0.4$.

identified for all configurations, except for the case of $\rho(\mathcal{G}) = p = 0.4$, where the number of edges is underestimated because only a single graph with $\rho(\mathcal{G}) = p = 0.2$ has been used for the training of the neural network.

B. Graph identification in multi-robot flocking tasks

The next experiment evaluates our approach when it is trained with a variety of graph edge densities. We study our proposed approach in a multi-robot flocking problem, where the state is defined by the 2D position the robots, so $s = 2$. Robot trajectories are generated using the controller proposed in [38], parameterized as detailed in Appendix I. The training set is generated with a sparsity pattern determined by the desired inter robot distance $\rho = 0.7\text{m}$ and communication radius $r_{comm} = 1.2\text{m}$, simulated through 6s with a sample time of 0.04s, and where the robots are uniformly spawned in a square of $5 \times 5\text{m}$. The trajectories are then split in sub-trajectories of $d = 10$, each of them associated to a

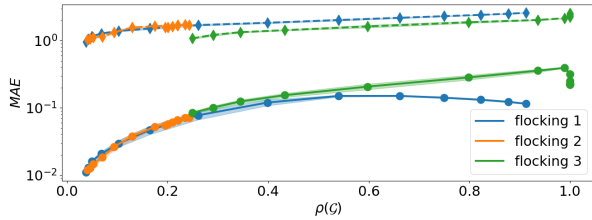


Fig. 7: MAE as a function of the edge density and flocking configurations. Each configuration is run 10 times, computing the mean and standard deviation. Diamond dashed lines are the state-of-the-art [28] ($\alpha = 0.1$ and $\beta = 10^{-5}$), whereas the circle solid lines are ours.

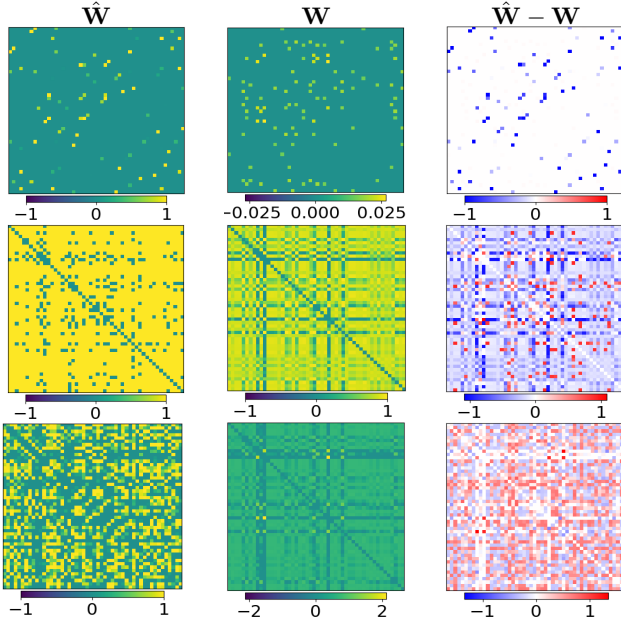


Fig. 8: Qualitative results of our approach for different flocking configurations and edge densities, extracted from the three flocking examples. The relative value among edges of the discovered weights is accurately identified, according to the color scales.

graph resulting from the average of the d samples. This leads to a training set of graphs with edge densities from 5% to 95%. The test cases are generated with the following configurations, one trajectory each: (1) $\rho = 0.7m$ and $r_{comm} = 1.2m$; (2) $\rho = 1.0m$ and $r_{comm} = 1.2m$; and (3) $\rho = 0.7m$, $r_{comm} = 1.2m$ and a compact initialization in a square $2 \times 2m$. The desired position of the flock is always $x^* = y^* = 0.0m$. The training of the neural network takes 150000 steps. Every 500 steps, one graph with its associated sub-trajectories is uniformly randomly picked, initializing the optimization module each time.

The results in Fig. 7 show how the MAE evolves as a function of the edge density and flocking configuration. For all the cases, our approach outperforms the state-of-the-art by more than one order of magnitude. Compared to the formation problem, the difference now is greater because the dynamics of the robots under the flocking controller are nonlinear. Moreover, in the state-of-the-art algorithms, α and β are static, so the output is not able to adjust to changes in edge density. In contrast, our approach adjusts the optimization regularizers and provides the latent feature trajectories that best fit the flocking task, concluding that our approach has learned to adapt the parameters of the optimization

method to the observed edge probability/density. Compared to the formation task, the learned neural network achieves a good performance for the different edge densities because the training set is diverse in this aspect. The MAE in Fig. 7 is specially good in very sparse networks ($\rho(\mathcal{G}) < 0.25$). As observed in Fig. 8, the relative value among the discovered weights is correctly identified, whereas the value of these weights with respect to the ground-truth \hat{W} vary depending on the edge density of the graph. Looking at Fig. 8, for sparse, medium and dense graphs the identified weights tend to be lower, similar and greater than in the ground-truth graph. This also explains why the MAE in Fig. 7 grows with the edge density. Besides, the neural network predicts a greater number of cliques, as in the middle row of Fig. 8. These behaviors require further investigation. As future work, We also plan to conduct ablation studies to analyze the impact of the dataset edge density variety in the generalization capabilities of our approach.

V. CONCLUSION

This work proposed a novel approach for graph topology identification combining a self-attention encoder with a fast analytical convex optimization algorithm. Our method provides a neural network model that learns to identify general graph topologies using only state trajectories of the nodes. Our approach is accurate and flexible, surpassing the state-of-the-art in multi-robot graph identification with different node configurations, node number, and edge densities. The applications to multi-robot systems problems are diverse. For instance, it can be used to identify the topology in multi-agent demonstrations to constrain the learning of multi-robot control policies or to detect failures in the communications among warehouse robots. In addition, by considering agents instead of robots, the proposed approach can be directly applied to general networked systems like brain imaging, genetics or social interactions.

APPENDIX I HYPERPARAMETERS

The training of the neural networks evaluated in Section IV is parameterized by a learning rate $\mu = 0.001$ and uses Adam [47]. Algorithm 1 is parameterized by $\epsilon = 10^{-5}$. The encoder of the neural network in Section IV-A is given by a first fully-connected network of layer dimensions $[2, 5, 10, 5, 1]$ with \tanh activation functions and a second fully connected network of layer dimensions $[1, 2, 1]$ with \tanh activation functions except the last one, which is linear. The layers for α and β are of dimensions $[1, 2, 1]$ with \tanh activation functions except the last layer, which is a sigmoid. The encoder of the neural network in Section IV-B is given by a first fully-connected network of layer dimensions $[2, 4, 4, 1]$ with \tanh activation functions. The layers for α and β are of dimensions $[1, 2, 1]$ with \tanh activation functions except the last layer, and the output is multiplied by a scalar $b = 3$. The flocking follows the dynamics proposed in [38], with parameters: $\epsilon = 0.1$, $a = 5.0$, $b = 5.0$, $h = 0.2$, $c_1 = 0.4$, $c_2 = 0.8$.

REFERENCES

- [1] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana, "Estimating time-varying brain connectivity networks from functional MRI time series," *NeuroImage*, vol. 103, pp. 427–443, 2014.
- [2] W. Huang, T. A. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A graph signal processing perspective on functional brain imaging," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 868–885, 2018.
- [3] E. Nozari, R. Planas, and J. Cortés, "Structural characterization of oscillations in brain networks with rate dynamics," *Automatica*, vol. 146, p. 110653, 2022.
- [4] A. Julius, M. Zavlanos, S. Boyd, and G. J. Pappas, "Genetic network identification using convex programming," *IET Systems Biology*, vol. 3, no. 3, pp. 155–166, 2009.
- [5] M. Nabi-Abdolyousefi and M. Mesbahi, "Network identification via node knockout," *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3214–3219, 2012.
- [6] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.
- [7] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.
- [8] R. Shafiqpour, S. Segarra, A. G. Marques, and G. Mateos, "Network topology inference from non-stationary graph signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 5870–5874.
- [9] E. Montijano, G. Oliva, and A. Gasparri, "Distributed estimation and control of node centrality in undirected asymmetric networks," *IEEE Trans. on Automatic Control*, vol. 66, no. 5, pp. 2304–2311, 2020.
- [10] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. on Signal and Information Processing over Networks*, vol. 4, no. 3, pp. 481–496, 2017.
- [11] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 484–499, 2017.
- [12] M. Timme, "Revealing network connectivity from response dynamics," *Physical Review Letters*, vol. 98, no. 22, p. 224101, 2007.
- [13] D. Napolitano and T. D. Sauer, "Reconstructing the topology of sparsely connected dynamical networks," *Physical Review E*, vol. 77, no. 2, p. 026103, 2008.
- [14] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.
- [15] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.
- [16] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.
- [17] F. Rossi, S. Bandyopadhyay, M. T. Wolf, and M. Pavone, "Multi-agent algorithms for collective behavior: A structural and application-focused atlas," *arXiv preprint arXiv:2103.11067*, 2021.
- [18] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-swarm: Decentralized close-proximity multi-robot control using learned interactions," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 3241–3247.
- [19] T. Z. Jiahao, L. Pan, and M. A. Hsieh, "Learning to swarm with knowledge-based neural ordinary differential equations," in *Int. Conf. on Robotics and Automation*. IEEE, 2022, pp. 6912–6918.
- [20] E. Sebastián, T. Duong, N. Atanasov, E. Montijano, and C. Sagiús, "LEMURS: Learning distributed multi-robot interactions," in *IEEE Int. Conf. on Robotics and Automation*, 2023, pp. 7713–7719.
- [21] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.
- [22] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.
- [23] S. Shahrapour and V. M. Preciado, "Topology identification of directed dynamical networks via power spectral analysis," *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2260–2265, 2014.
- [24] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, "Blind identification of graph filters," *IEEE Transactions on Signal Processing*, vol. 65, no. 5, pp. 1146–1159, 2016.
- [25] V. Kalofolias, "How to learn a graph from smooth signals," in *Artificial Intelligence and Statistics*, 2016, pp. 920–929.
- [26] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 2826–2830.
- [27] S. S. Saboksayr, G. Mateos, and M. Cetin, "Online graph learning under smoothness priors," in *IEEE European Signal Processing Conference*, 2021, pp. 1820–1824.
- [28] S. S. Saboksayr and G. Mateos, "Accelerated graph learning from smooth signals," *IEEE Signal Processing Letters*, vol. 28, pp. 2192–2196, 2021.
- [29] A. Beck and M. Teboulle, "A fast dual proximal gradient algorithm for convex minimization and applications," *Operations Research Letters*, vol. 42, no. 1, pp. 1–6, 2014.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [31] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.
- [32] N. Kamra, H. Zhu, D. K. Trivedi, M. Zhang, and Y. Liu, "Multi-agent trajectory prediction with fuzzy query attention," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 530–22 541, 2020.
- [33] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533–5540, 2021.
- [34] Z. Wang and M. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.
- [35] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.
- [36] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, vol. 38(5), pp. 1–12, 2019.
- [37] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems*, vol. 187, p. 104816, 2020.
- [38] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.
- [39] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 4775–4782.
- [40] D. Acemoglu and A. Ozdaglar, "Opinion dynamics and learning in social networks," *Dynamic Games and Applications*, vol. 1, pp. 3–49, 2011.
- [41] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on dynamic average consensus: The problem, its applications, and the algorithms," *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40–72, 2019.
- [42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
- [43] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825–841, 2017.
- [44] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2016.
- [45] E. Pavez, H. E. Egilmez, and A. Ortega, "Learning graphs with monotone topology properties and multiple connected components," *IEEE Transactions on Signal Processing*, vol. 66, no. 9, pp. 2399–2413, 2018.
- [46] P. Erdos and A. Rényi, "On the evolution of random graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, no. 1, pp. 17–60, 1960.
- [47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2015.