SPACE-TIME GRAPH NEURAL NETWORKS WITH STOCHASTIC GRAPH PERTURBATIONS

Samar Hadou, Charilaos I. Kanatsoulis, and Alejandro Ribeiro

Department of Electrical and Systems Engineering, University of Pennsylvania, PA, USA

ABSTRACT

Space-time graph neural networks (ST-GNNs) are recently developed architectures that learn efficient graph representations of time-varying data. ST-GNNs are particularly useful in multi-agent systems, due to their stability properties and their ability to respect communication delays between the agents. In this paper we revisit the stability properties of ST-GNNs and prove that they are stable to stochastic graph perturbations. Our analysis suggests that ST-GNNs are suitable for transfer learning on time-varying graphs and enables the design of generalized convolutional architectures that jointly process time-varying graphs and time-varying signals. Numerical experiments on decentralized control systems validate our theoretical results and showcase the benefits of traditional and generalized ST-GNN architectures.

***Index Terms*—** ST-GNNs, GNNs, stability anaysis, graph perturbations, transfer learning

1. INTRODUCTION

Stability analysis of graph neural networks (GNNs) was studied extensively in literature [1–5] and affirms that small changes in input graphs translate into small, bounded changes in the outputs. It also provides a characterization of the design parameters, e.g., the depth and width of GNNs, that preserve stability. The acquired gain, when stability holds, is the ability to transfer learned GNNs between different graphs of the same size without re-training the models. Stability is a notion that is different from GNN transferability, which considers transfer learning to graphs of larger sizes [6–8]. Together, they allow executing GNNs on underlying graphs that are different from those used in training and determine which architectures are more robust to graph changes.

GNNs are aligned with applications in distributed controllers [9–11] and wireless systems [12–14]. Both tasks are challenging, since the underlying graph varies rapidly over time and the signals endure delays while being propagated between agents. To deal with signal delays, we leveraged diffusion equations to introduce space-time graph neural networks (ST-GNNs) in [15]. Specifically, we formulated a diffusion equation over a space-time domain that respects propagation time between nodes. This allowed the proposed space-time convolutions to aggregate each node's present data along with outdated information from its neighbors. During training ST-GNNs learn to weigh the outdated information and offset their negative impact on the model's predictions. Our analysis also proved that the new architecture is stable to graph and time perturbations. However, the challenge of facing dynamic graphs during training is still to be addressed.

The first work to approach this challenge is [16] that studied dynamic graphs modelled as stochastic processes and showed that GNNs are stable to stochastic graph perturbations. However, their analysis focuses on signals that are fixed over time, which is impractical in certain applications. In this paper, we bridge this gap and study both signals and graphs that vary over time. We extend our previous work in ST-GNNs to accommodate time-varying graphs and prove their stability to stochastic graph perturbations. Our contributions can be summarized as follows:

(C1) We prove the stability of STGFs and ST-GNNs to stochastic graph perturbations. Our result implies that ST-GNNs can handle transfer learning to time-varying graphs.

(C2) Motivated by our stability analysis, we introduce generalized STGFs and generalized ST-GNNs that are tailored to jointly process time-varying graphs and time-varying signals and study their spectral properties.

(C3) We conduct experimental examination of ST-GNNs and generalized ST-GNNs distributed controller settings.

Due to the limited space, all the proofs are relegated to an extended version of the paper.

2. SPACE-TIME GRAPH NEURAL NETWORKS

Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of N nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We define discrete-time signals $\mathbf{x}_i(t)$ at each node i, which can be concatenated to a space-time graph signal $\mathbf{X} \in \mathbb{R}^{N \times T}$. Then $\mathbf{SX}$ and $\mathbf{XC}$ define the 1-hop diffusion of the space-time signal in space and time respectively. Note that $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the graph shift operator (GSO), which usually represents the graph adjacency or Laplacian, and $\mathbf{C} \in \{0, 1\}^{N \times N}$ is a circulant matrix that models the time shift operator (TSO). Then we can define the k-hop space-time diffusion as $\mathbf{S}^k \mathbf{X} \mathbf{C}^k$, which allows us to define the space-time graph filter (STGF) as the following convolutional operator

$$\mathbf{Y} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{X} \mathbf{C}^k, \qquad (1)$$

where the coefficients $\{h_k\}_{k=0}^{K}$ and the number of filter taps K are design parameters. The operation in (1) respects the time delays that occur in sharing information between the neighbors. Indeed, the GSO and TSO are jointly applied to ensure that the signals are shifted in space and time before being aggregated over the graphs.

We design ST-GNNs as a cascade layers that deploy the STGF in (1) followed by a nonlinear activation function σ. To distinguish the layers' intermediate outputs we use subscript l and superscript f to denote the l-th layer and the f-th feature of the signal respectively. The output of the l-th layer is expressed as

$$\mathbf{X}_l^f = \sigma \Big(\sum_{g=1}^{F} \sum_{k=0}^{K} h_{kl}^{fg} \mathbf{S}^k \mathbf{X}_{l-1}^g \mathbf{C}^k \Big). \qquad (2)$$

We assume that the number of the features F is fixed across all layers. We further refer to the ST-GNN's output, i.e., the output of the last layer L, as $\mathbf{\Phi}(\mathbf{X}; \mathbf{S}, \mathcal{H})$ with $\mathcal{H} = \{h_{kl}^{fg}\}_{k,l}^{f,g}$ being the set of all learnable parameters. Note that, we represent the underlying graph with a fixed GSO $\mathbf{S}$, which is not always the case in practise. In the following section, overcome this limitation we show how (1) can be modified to accommodate time-varying graphs.

3. GENERALIZED SPACE-TIME GNNS

In various applications the graph is changing over time. In multi-agent systems, for instance, the graph dynamics are time varying due to link dropping and re-wiring. As a result, we observe a sequence of GSOs $\{\mathbf{S}_k, \ldots, \mathbf{S}_0\}$. The diffusion of graph signals over this sequence is modeled as

$$\mathbf{S}_k \ldots \mathbf{S}_0 \mathbf{X} =: \mathbf{S}_{k:0} \mathbf{X}.$$

Since STGFs perform a weighted sum of diffused signals as shown in (1), a generalized form of the STGF's output can be derived as

$$\tilde{\mathbf{Y}} = \sum_{k=0}^{K} h_k \mathbf{S}_k \ldots \mathbf{S}_0 \mathbf{X} \mathbf{C}^k = \sum_{k=0}^{K} h_k \mathbf{S}_{k:0} \mathbf{X} \mathbf{C}^k, \qquad (3)$$

with $\mathbf{S}_0$ being the identity matrix $\mathbf{I}$. Equation (1) is obviously a special case with $\mathbf{S}_K = \cdots = \mathbf{S}_1 = \mathbf{S}$. Like ST-GNNs in (2), we define the output of the l-th layer of the generalized ST-GNN as

$$\tilde{\mathbf{X}}_l^f = \sigma \Big(\sum_{g=1}^{F} \sum_{k=0}^{K} h_{kl}^{fg} \mathbf{S}_{k:0} \tilde{\mathbf{X}}_{l-1}^g \mathbf{C}^k \Big). \qquad (4)$$

The following lemma shows the frequency response of (3).

Lemma 1 (Generalized STGF frequency response). *For a space-time graph filter with coefficients $\{h_k\}_{k=0}^{K}$ running over a sequence of K GSOs as in (3), the generalized frequency response is*

$$h(\boldsymbol{\lambda}, \omega) = \sum_{k=0}^{K} h_k e^{jk\omega} \prod_{\kappa=0}^{k} \lambda_\kappa, \qquad (5)$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_K]^\top$, $\lambda_0 = 1$, and λ_k is the analytic variable corresponding to the eigenvalue of $\mathbf{S}_k$.

The proof of Lemma 1 is delegated to the journal version, due to space limitations. We observe that the frequency response is a multi-variate function in the graph-frequency variables $\lambda_k, \forall k$, and the time-frequency variable ω.

In practice, we usually need to transfer generalized ST-GNN to different time-varying graphs that are not observed during training. Therefor a natural question that arises is "under which conditions is transfer learning doable?" We tackle give a definitive answer to this question by analyzing the deviation in the outputs between ST-GNNs and generalized ST-GNNs. Bounded deviations, imply that learned models can be transferred between different sequences of graphs.

4. STABILITY TO STOCHASTIC GRAPH PERTURBATIONS

To analyze the deviation between ST-GNNs and genaralized ST-GNNs, we first define a model for time-varying graphs. In particular, we study a sequence of random graphs $\{\mathcal{G}_k\}_{k=1}^{K}$ sampled from a nominal graph $\mathcal{G}$ according to the following criteria.

Definition 1 (Random Edge Sampling). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define a random graph $RES(\mathcal{G}, p)$ with a realization $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$ such that an edge $(i, j) \in \mathcal{E}$ exists in $\mathcal{E}_k$ with probability p, i.e.,*

$$\mathbb{P}[(i,j) \in \mathcal{E}_k] = p, \ \forall (i,j) \in \mathcal{E}. \qquad (6)$$

Definition 1 implies that $\mathcal{G}_k$ is constructed by sampling edges from the nominal graph with probability p. From Definition 1, it follows that the relation between the nominal GSO $\mathbf{S}$ and the random GSO $\mathbf{S}_k$ can be written as

$$\mathbf{S}_k = \mathbf{S} + \mathbf{E}_k, \qquad (7)$$

where $\mathbf{E}_k$ models the deviation between the two GSOs. In (7), The size of the deviation $\mathbf{E}_k$ is controlled by the edge-drop probability $1 - p$. Higher probabilities account for dropping more edges, which in turn results in higher perturbation size $\|\mathbf{E}_k\|_2$. Probability p can be viewed as a measure of similarity between the nominal graph $\mathcal{G}$ and a random realization $\mathcal{G}_k$.

With relation (7) in place, we next aim to compute the difference in STGF's outputs when running over either $\mathbf{S}$ or the sequence $\mathbf{S}_{K:0}$ with each GSO having been constructed according to Definition 1. To achieve this, we introduce Lemma (2), which defines a smoothness measure of the multi-variate filter in (5).

Lemma 2. *Consider the generalized frequency response $h(\boldsymbol{\lambda}, \omega)$ in Lemma 1, and let $\boldsymbol{\lambda}_1 = [\lambda_{11}, \ldots, \lambda_{1K}]^T$ and $\boldsymbol{\lambda}_2 = [\lambda_{21}, \ldots, \lambda_{2K}]^T$ be two multivariate graph-frequency vectors. The variability in the frequency response is then quantified as*

$$h(\boldsymbol{\lambda}_1, \omega) - h(\boldsymbol{\lambda}_2, \omega) = \nabla_{\boldsymbol{\lambda}}^T h(\boldsymbol{\lambda}_{1,2}, \omega) \cdot (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2), \qquad (8)$$

where $\nabla_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda}_{1,2}, \omega) = \left[\frac{\partial h(\boldsymbol{\lambda}^{(1)}, \omega)}{\partial \lambda_1}, \ldots, \frac{\partial h(\boldsymbol{\lambda}^{(K)}, \omega)}{\partial \lambda_K} \right]^T$ is the Lipschitz gradient, and $\boldsymbol{\lambda}^{(k)} = [\lambda_{11}, \ldots, \lambda_{1k}, \lambda_{2(k+1)}, \ldots, \lambda_{2K}]^T$ contains the first k entries of $\boldsymbol{\lambda}_1$ along with the last $K - k$ entries of $\boldsymbol{\lambda}_2$.

The proof of Lemma 2 follows the proof of Lemma 1 in [16]. The Lipschitz gradient in (8) can be seen as a general measure of filter-response variability, which is usually quantified by the derivatives in univariate filters. Using the Lipschitz gradient, we define a class of smooth filters, namely generalized integral Lipschitz filter, in the following definition.

Assumption 1. *[Generalized Integral Lipschitz filter] An STGF with the generalized frequency response in Lemma 1 is generalized integral Lipschitz if there exists a constant $C_L > 0$ such that for any graph-frequency vectors $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ it holds that for all ω*

$$\|\nabla_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda}_{1,2}, \omega)\|_2 \leq C_L, \qquad (9)$$

$$\|\boldsymbol{\lambda}_1 \odot \nabla_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda}_{1,2}, \omega)\|_2 \leq C_L, \qquad (10)$$

where $\nabla_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda}_{1,2}, \omega)$ is the Lipschitz gradient defined in Lemma 2, and $\odot$ is the Hadamard product.

The condition in (9) indicates that the frequency response $h(\boldsymbol{\lambda}, \omega)$ does not change faster than linear since we have

$$\begin{aligned}
|h(\boldsymbol{\lambda}_1, \omega) - h(\boldsymbol{\lambda}_2, \omega)| &= |\nabla_{\boldsymbol{\lambda}}^T h(\boldsymbol{\lambda}_{1,2}, \omega) \cdot (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)| \\
&\leq \|\nabla_{\boldsymbol{\lambda}}^T h(\boldsymbol{\lambda}_{1,2}, \omega)\|_2 \cdot \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\|_2 \\
&\leq C_L \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\|_2.
\end{aligned}$$

The first inequality follows from triangle inequality while the first and third lines are direct application of Equations (8) and (9) respectively. On the other hand, the condition in (10) states that the variability in the frequency response should be low at higher eigenvalues. In other words, filters that meet (10), albeit stable, have a near-flat response at higher frequencies and therefore have poor discriminibilty between high graph frequencies. Under the two conditions, Theorem 1 characterizes the stability properties of STGFs.

Theorem 1 (STGF Stability). *Consider an STGF with coefficients* $\{h_k\}_{k=0}^{K}$ *running over a sequence of K identical GSOs $\mathbf{S}$ with output $\mathbf{Y}$ as in (1). Consider a sequence of K realizations of $RES(\mathcal{G}, p)$ with GSOs $\{\mathbf{S}_k\}_{k=1}^{K}$ under which the filter output $\tilde{\mathbf{Y}}$ is captured in (3). Let $\mathbf{S}_k = \mathbf{S} + \mathbf{E}_k$, for $k = 1, \ldots, K$, and $\mathbf{S}_0 = \mathbf{I}$. If Assumption 1 holds, the expected difference between the filter outputs satisfies*

$$\mathbb{E}[\|\tilde{\mathbf{Y}} - \mathbf{Y}\|_F^2] \le C(1-p)\|\mathbf{X}\|_F^2 + \mathcal{O}\left((1-p)^2\right), \quad (11)$$

where $C = \alpha N C_L^2$, and the scalar α is either the maximum node degree if $\mathbf{S}$ is the adjacency or 2 if it is the Laplacian.

Theorem 1 affirms that the bound depends on some graph and filter parameters as well as the probability of edge drop $1 - p$. It comes with no surprise that N and $1 - p$ contribute to the stability constant since holding either of them fixed and increasing the other leads to dropping more edges while constructing the random graph realization $\mathcal{G}_k$. Thus the distance between the GSOs $\mathbf{S}$ and $\mathbf{S}_k$ increases and affects the stability of the model. We also observe that filters with lower C_L are more robust to perturbations. The constant C_L describes the variability in the filter response (cf. (9)) and smaller values of C_L indicates that the filter response changes slowly. This implies that the filter smoothness is key to provoke stability.

To analyze the stability of ST-GNNs, we consider a class of nonlinearities σ that is Lipschitz continuous as stated in Assumption 2.

Assumption 2. *The nonlinearities σ are Lipschitz-continuous functions with a Lipschitz constant $C_\sigma > 0$ such that, for any two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, we have*

$$\|\sigma(\mathbf{x}_1) - \sigma(\mathbf{x}_2)\|_2 \le C_\sigma \|\mathbf{x}_1 - \mathbf{x}_2\|_2. \quad (12)$$

This assumption is satisfied by the activation functions usually used in GNNs, e.g., ReLU and hyperbolic tangent (tanh) functions. When Assumptions 1 and 2 hold, the stability of ST-GNNs is determined by Theorem 2.

Theorem 2 (ST-GNN Stability). *Consider an L-layer ST-GNN $\mathbf{\Phi}(\mathbf{X}; \mathbf{S}, \mathcal{H})$ with F features per layer and F input and output features. Let the STGF with coefficients $\{h_{kl}^{fg}\}_{kl}^{fg}$ at each layer satisfy Assumption 1 and have a unit norm, i.e., $|h(\boldsymbol{\lambda}, \omega)| \le 1$. Also, consider the same ST-GNN runs over a sequence of K realizations of $RES(\mathcal{G}, p)$ with output $\tilde{\mathbf{\Phi}}(\mathbf{X}; \mathbf{S}, \mathcal{H})$. If the nonlinear activation functions satisfy Assumption 2, the expected difference between the outputs is bounded by*

$$\mathbb{E}\left[\|\tilde{\mathbf{\Phi}}(\mathbf{X}; \mathbf{S}, \mathcal{H}) - \mathbf{\Phi}(\mathbf{X}; \mathbf{S}, \mathcal{H})\|_F^2\right] \\ \le C(1-p)\|\mathbf{X}\|_F^2 + \mathcal{O}\left((1-p)^2\right), \quad (13)$$

where $C = \alpha N L^2 C_L^2 C_\sigma^{2L} F^{2L}$ with α being the same constant in Theorem 1.

Besides all the factors that control the stability of STGFs, Theorem 2 includes some design factors, e.g., the number of features F, the number of layers L, and the Lipschitz constant of the nonlinearities C_σ. It is worth noting, however, that the bound grows exponentially with the number of layers, which suggests that deep networks are less robust to perturbations. This result is in line with other works that recommend the use of GNNs with only a few layers [17]. The authors in [18] even show that a single-layer GNN with multi-tap filters has a performance that is comparable, or even superior, to that of multi-layer single-tap GNNs. For our considered tasks, we have also observed that a small number of layers is sufficient to ensure stability without sacrificing performance/expressivity. Detailed study of the expressive power of GNNs can be found in [19].

5. NUMERICAL SIMULATIONS

We consider the flocking problem in [20], where the objective is to coordinate a swarm of agents to move together at the same velocity and avoid collisions. The problem has a closed-form optimal solution (see [21]). However, this solution requires access to data from all the agents and therefore does not allow decentralized settings. On the other hand, solutions that rely on gathering information from local neighborhood are harder to find. ST-GNNs can close this gap by learning a decentralized policy using imitation learning.

To construct a dataset, each agent is associated with a time sequence of positions $\mathbf{p}_i \in \mathbb{R}^{2 \times T}$ and velocities $\mathbf{v}_i \in \mathbf{R}^{2 \times T}$ as node features. The agent is also associated with optimal accelerations $\mathbf{u}_i \in \mathbb{R}^{2 \times T}$ as node predictions, which are pre-computed with a centralized algorithm. The node features are computed at each time step t via the system dynamics as soon as the model predicts the acceleration $\hat{\mathbf{u}}_i(t-1)$, as described in [20]. The agents form a communication network that varies over time. The underlying graph $\mathcal{G}(t)$ contains an edge between nodes i and j if and only if they are within the communication range of each other, i.e., $\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|_2 \le R$. Since the graph depends on the present positions of the agents, we cannot forecast the graph sequence before training/execution. For our experiments, the dataset contains 400 examples for training, 80 for validation, and another 80 for testing.

We train an ST-GNN that consists of one layer with $K = 3$, $F = 64$ and $\tanh$ activation function. We then implement a local layer at each node to predict the agent accelerations $\hat{\mathbf{u}}_i(t)$. We use ADAM [22] with learning rate 5×10^{-4} and forgetting factors $\beta_1 = 0.9$ and $\beta_2 = 0.999$ in order to optimize the MSE between the optimal $\mathbf{u}_i(t)$ and the predicted $\hat{\mathbf{u}}_i(t)$. We train our model over 30 epochs and used validation input-output pairs to choose the one that minimizes the velocity variation among the agents,

$$\text{cost} = \sum_{i=1}^{N} \left\| \mathbf{v}_i - \frac{1}{N} \sum_{j=1}^{N} \mathbf{v}_j \right\|_F^2,$$

averaged over the validation dataset.

In our first experiment, we train an ST-GNN on a fixed graph and test it over a sequence of random graphs sampled according to definition 1. Contrary to [20], we train our model on the average of the graph sequence. But since we cannot forecast the graphs before training as we mentioned before, we use the agent's positions computed with the optimal accelerations to compute the graph sequence and, in turn, their average. During execution, we sample a graph sequence from the average graph with a sampling probability $p \in \{1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7\}$. Note that the sampled graph sequences no longer represent the agent's proximity since we decide
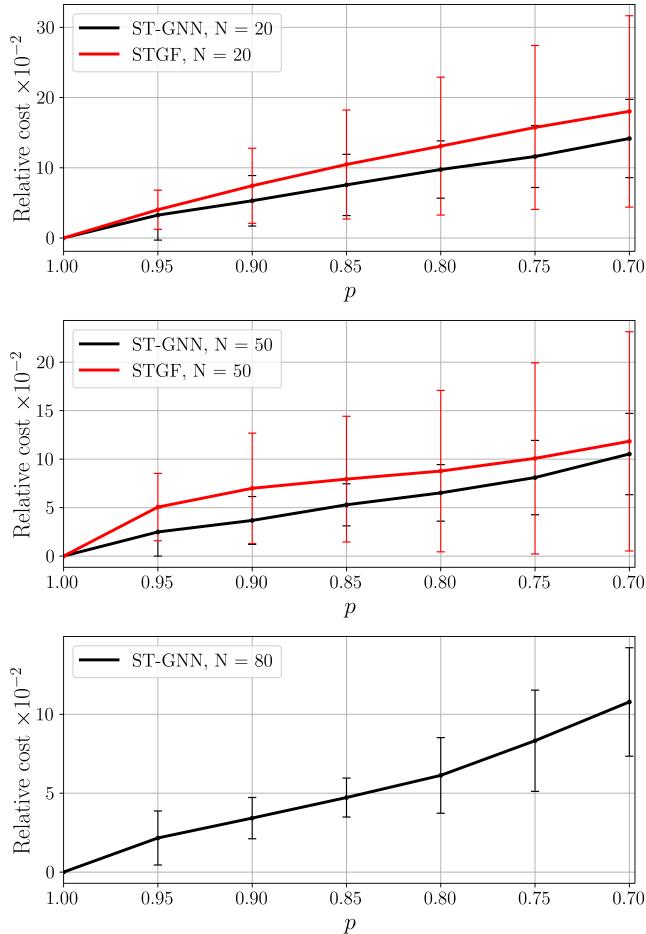
Fig. 1: Relative cost induced by stochastic perturbations with different sampling probabilities p for $N = 20$ (top), $N = 50$ (middle), and $N = 80$ (bottom).

(a) $t = 0$s

(b) $t = 1.99$s

Fig. 2: Snapshots of two test examples: (top) $N = 20$ and (bottom) $N = 50$. The dots represent the agents, the gray lines illustrate the underlying graphs, and the dark arrows show the agents' velocity.

6. CONCLUSIONS

The paper studied the stability of ST-GNNs to stochastic graph perturbations and showed that we can transfer learned models over time-varying graphs. Our analysis also enabled us to design generalized ST-GNNs that can be trained over time-varying graphs. We concluded that the stability bound is controlled by some graph, filter, and design parameters. The bound increases linearly with the graph size as well as the size of graph perturbations, represented by the edge-drop probability. It also depends on the Lipschitz constant of STGFs and the nonlinearities. Among all those parameters, only the Lipschitz constant of the filters cannot be decided before training since the filter is learnable. The stability can though be enhanced by forcing the filter to have a low Lipschitz constant during training, which is left as a future work.

7. REFERENCES

[1] S. Verma and Z.-L. Zhang, "Stability and generalization of graph convolutional neural networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1539–1548.

[2] N. Keriven, A. Bietti, and S. Vaiter, "Convergence and stability of graph convolutional networks on large random graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21512–21523, 2020.

[3] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5680–5695, 2020.

[4] H.-S. Nguyen, Y. He, and H.-T. Wai, "On the stability of low pass graph filter with a large number of edge rewires," in *IEEE*

on the fixed graph before training and we sample randomly from it at execution. Fig. 1 shows that the relative cost increases linearly with the edge-drop probability $1 - p$, matching the result of Theorem 2. We also train an STGF with $K = 4$ and $F = 32$ and verify the linear bound of Theorem 1. We, however, observe that ST-GNNs show less relative cost compared to STGFs even though we used a lower number of features with the latter. We attribute this observation to the $\tanh$ activation function being a nonexpansive mapping (i.e., we have $C_\sigma \leq 1$).

We train a generalized ST-GNN, where the graphs are formed according to the communication networks between the agents. The graphs are constructed based on a communication range of $R = 2m$ as described at the beginning of this section and as in [20]. Thus the graphs are time varying during both training and execution. Fig. 2 illustrates snapshots of the output of a generalized ST-GNN for two test examples with unseen time-varying graphs of sizes $N = 20$ and $N = 50$ nodes. The snapshots were taken at the beginning and end of a time interval of 2s. We observe that all the agents managed to move with the same velocity and in the same direction by the end of the time interval. This indicates that the trained ST-GNN is successfully transferred to unseen graphs, which validate our theoretical results.
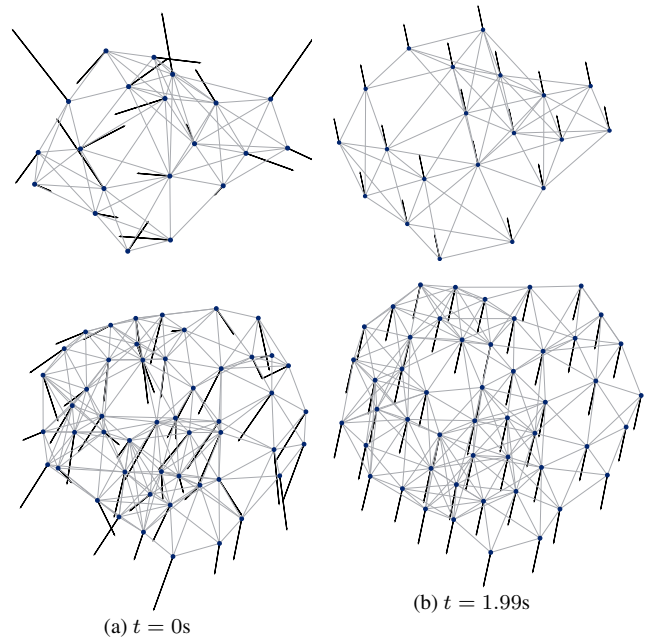
International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 5568–5572.

[5] H. Kenlay, D. Thano, and X. Dong, "On the stability of graph convolutional neural networks under edge rewiring," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 8513–8517.

[6] R. Levie, W. Huang, L. Bucci, M. M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks.," *J. Mach. Learn. Res.*, vol. 22, pp. 272–1, 2021.

[7] L. Ruiz, L. Chamon, and A. Ribeiro, "Graphon neural networks and the transferability of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1702–1712, 2020.

[8] S. Maskey, R. Levie, and G. Kutyniok, "Transferability of graph neural networks: an extended graphon approach," *arXiv preprint arXiv:2109.10096*, 2021.

[9] K. Osanlou, A. Bursuc, C. Guettier, T. Cazenave, and E. Jacopin, "Optimal solving of constrained path-planning problems with graph convolutional networks and optimized tree search," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3519–3525.

[10] 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.

[11] Y. Zhou, J. Xiao, Y. Zhou, and G. Loianno, "Multi-robot collaborative perception with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2289–2296, 2022.

[12] S. He, S. Xiong, Y. Ou, J. Zhang, J. Wang, Y. Huang, and Y. Zhang, "An overview on the application of graph neural networks in wireless networks," *IEEE Open Journal of the Communications Society*, 2021.

[13] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Distributed scheduling using graph neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 4720–4724.

[14] W. Yan, D. Jin, Z. Lin, and F. Yin, "Graph neural network for large-scale network localization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5250–5254.

[15] S. Hadou, C. I. Kanatsoulis, and A. Ribeiro, "Space-time graph neural networks," in *International Conference on Learning Representations*, 2022.

[16] Z. Gao, E. Isufi, and A. Ribeiro, "Stability of graph convolutional neural networks to stochastic perturbations," *Signal Processing*, vol. 188, pp. 108216, 2021.

[17] T. Chen, K. Zhou, K. Duan, W. Zheng, P. Wang, X. Hu, and Z. Wang, "Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[18] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 6861–6871, PMLR.

[19] C. I. Kanatsoulis and A. Ribeiro, "Graph neural networks are more powerful than we think," *arXiv preprint arXiv:2205.09801*, 2022.

[20] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Proceedings of the Conference on Robot Learning*, Oct 2020, vol. 100, pp. 671–682.

[21] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stable flocking of mobile agents part II: dynamic topology," in *Proceedings of the IEEE Conference on Decision and Control*, 2003, vol. 2, pp. 2016–2021.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations ICLR*, 2015.