

Graph Sampling for Scalable and Expressive Graph Neural Networks on Homophilic Graphs

Haolin Li, Haoyu Wang, Luana Ruiz

Department of Applied Mathematics and Statistics, Johns Hopkins University
{hli230, hwang320, lrubini1}@jh.edu

Abstract—Graph Neural Networks (GNNs) excel in many graph machine learning tasks but face challenges when scaling to large networks. GNN transferability allows training on smaller graphs and applying the model to larger ones, but existing methods often rely on random subsampling, leading to disconnected subgraphs and reduced model expressivity. We propose a novel graph sampling algorithm that leverages feature homophily to preserve graph structure. By minimizing the trace of the data correlation matrix, our method better preserves the graph Laplacian trace—a proxy for the graph connectivity—than random sampling, while achieving lower complexity than spectral methods. Experiments on citation networks show improved performance in preserving Laplacian trace and GNN transferability compared to random sampling.

Index Terms—graph signal processing, graph sampling, graph neural networks, transferability, homophily.

I. INTRODUCTION

GNNs are deep neural networks tailored to network data which have shown great empirical performance in several graph machine learning tasks [1]–[4]. This is especially true in graph signal processing problems—such as recommender systems on product similarity networks [5], or attribution of research papers to scientific domains [6]—in which GNNs’ invariance and stability properties [7], [8] play a key role.

Yet, in practice most successful applications of GNNs are limited to graphs of moderate size. The sheer size of many modern networks, typically in the order of several millions, frequently makes these models impractical to train. Good results have been seen by leveraging the GNN’s transferability property [9], [10], which states that a GNN with fixed weights produces similar outputs on large enough graphs belonging to the same “family”, e.g., the same random graph model. This property allows training the GNN on a graph of moderate size, and transferring it for inference on the large graph.

The transferability of GNNs is closely related to their convolutional parametrization, and is a consequence of graph convolutions converging on sequences of graphs that converge to a common graph limit [11]. Under certain assumptions on the type of limit, and on how the graphs converge to (or are sampled from) them, it is possible to obtain non-asymptotic error bounds inversely proportional to the sizes of the graphs. Such bounds are then used to inform practical considerations, such as the minimum graph size on which to train a GNN to meet a maximum transference error. Once this is determined, the training graph is obtained by sampling a subgraph of the appropriate size at random from the large graph.

Learning GNNs on randomly subsampled graphs works reasonably well on average but, for models trained on small samples, there is large variance in performance, and the worst-case performance can be quite low; see Figure 3. While this is a natural consequence of subsampling any type of data, on graphs issues are exacerbated by the fact that random subgraphs often have disconnected components and isolated nodes. This leads to rank reduction in graph matrix representations, which in turn affects the expressive power of GNNs [12]. Indeed, the loss of rank is explicit for the graph Laplacian, in which the multiplicity of the zero eigenvalue indicates the number of connected components in the graph.

Sampling algorithms better at preserving matrix rank—such as spectral algorithms, e.g. [13], [14]—or connectivity—such as local algorithms, e.g., breadth-first search [15]—exist, however, spectral methods have high computational complexity, and local methods focus too much on specific regions of the graph and fail to capture global structure.

In this paper, we identify a property that allows graphs to be sampled efficiently without restricting to local regions: feature homophily. Specifically, let $G = (V, E)$ be a graph with node features $X \in \mathbb{R}^{|V| \times d}$. This graph is feature-homophilic if, given that (i, j) is an edge in G , the normalized features $\hat{X}[i, :]$ and $\hat{X}[j, :]$ are close. Our first contribution is to introduce a novel definition of feature homophily based on the graph Laplacian. Then, we show that, by sampling nodes to minimize the trace of the correlation matrix XX^T , it is possible to improve the trace of the graph Laplacian, which provides a measure of connectivity of the subsampled graph.

This heuristic is formalized as a graph sampling algorithm in Algorithm 1. Unlike other graph sampling routines, it does not require sequential node operations, and has complexity $O(d|E|)$, which is substantially cheaper than other algorithms for large $|V|$ and moderate d .

We conclude with an experimental study of the proposed algorithm on homophilic citation networks, in which we compare it with random sampling. We observe that, for the same sampling budget, our algorithm preserves trace better than sampling at random, and leads to better transferability performance in a semi-supervised learning task.

II. PRELIMINARIES

A graph $G = (V, E)$ consists of two components: a set of vertices or nodes V , and a set of edges $E \subseteq V \times V$. Generally,

Algorithm 1 Node Sampling for Feature-Homophilic Graphs**Require:** $G(V, E)$, $|V| = n$; $X \in \mathbb{R}^{n \times d}$; $\gamma \in [0, 1]$ Calculate deletion budget: $n_d \leftarrow \lfloor (1 - \gamma) \cdot n \rfloor$ Calculate node scores: $\vec{s} \leftarrow \text{diag}(XX^T)$ Keep $n - n_d$ nodes with the lowest scores: $\text{idx} \leftarrow \text{argmax}(\vec{s}, \text{descending} = \text{True})[n_d : n]$

Sample graph:

 $\tilde{V} \leftarrow V \cap \text{idx}$ $\tilde{E} \leftarrow \{(u, v) : u, v \in \tilde{V}, (u, v) \in E\}$ $\tilde{X} \leftarrow X[\text{idx}, :]$ **return** $\tilde{G}(\tilde{V}, \tilde{E})$; \tilde{X}

graphs can be categorized as being either directed or undirected based on their edge set E . A graph is undirected if and only if for any two nodes $u, v \in V$, $(u, v) \in E$ also implies $(v, u) \in E$ (and both correspond to the same undirected edge). In this paper, we restrict attention to undirected graphs.

Let $|V| = n$ be the number of nodes and $|E| = m$ be the number of edges in G . Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the corresponding adjacency matrix. The graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_n)$ is the so-called degree matrix. From their definitions, and since G is undirected, we can easily infer that \mathbf{A} and \mathbf{L} are symmetric.

In practice, real-world graphs are associated with node data $x \in \mathbb{R}^n$ called graph signals, where $x[i]$ corresponds to the value of the signal at node i . More generally, graph signals consist of multiple features, in which case they are represented as matrices $X \in \mathbb{R}^{n \times d}$ with d denoting the number of features.

The graph Laplacian plays an important role in graph signal processing (GSP) [16], [17], as it allows defining the notion of total variation of a signal x . Explicitly, the total variation of x is defined as $TV(x) = x^T \mathbf{L} x$ [18]. Let $\mathbf{L} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$ be the Laplacian eigendecomposition, where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues ordered as $\lambda_1 \leq \dots \leq \lambda_n$ and \mathbf{V} is the corresponding eigenvector matrix. For unit-norm signals, it is easy to see that the maximum total variation is λ_n , the largest Laplacian eigenvalue, and the minimum total variation is $\lambda_1 = 0$, which corresponds to the all-ones eigenvector. Therefore, the Laplacian eigenvalues can be interpreted as graph frequencies, and the eigenvectors as these frequencies' respective oscillation modes.

A. Graph Neural Networks

GNNs are deep learning models specifically designed for graph-structured data, where each layer consists of two components: a bank of convolutional filters and a nonlinear activation function. A graph convolutional filter is the extension of a standard convolutional filter to graph data. More specifically, it consists of a shift-and-sum operation of a signal x on the graph G , which is captured by a matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ encoding the sparsity pattern of the graph, i.e., $\mathbf{S}[i, j] \neq 0$ if and only if $(i, j) \in E$ or $i = j$; typical choices are $\mathbf{S} = \mathbf{A}$ or $\mathbf{S} = \mathbf{L}$ [19]. The graph shift operator operates on x as $\mathbf{S}x$.

Given any choice of \mathbf{S} , the graph convolution is defined as $y = \sum_{k=0}^{K-1} h_k \mathbf{S}^k x$, where h_0, \dots, h_{K-1} are the filter

coefficients or taps. More generally, for $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times f}$ we can define the convolutional filterbank [4]

$$Y = \sum_{k=0}^{K-1} \mathbf{S}^k X \mathbf{H}_k \quad (1)$$

where $\mathbf{H}_k \in \mathbb{R}^{d \times f}$, $0 \leq k \leq K - 1$.

The ℓ th layer of a GNN is given by [4]

$$X_\ell = \sigma \left(\sum_{k=0}^{K-1} \mathbf{S}^k X_{\ell-1} \mathbf{H}_{\ell k} \right) \quad (2)$$

with $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ an entry-wise nonlinearity (e.g., the ReLU or sigmoid). At layer $\ell = 1$, X_0 is the input data, and the last layer output X_L is the output Y of the GNN. For succinctness, in the following we will represent the whole L -layer GNN as a map $Y = \Phi(X, G; \mathcal{H})$ with $\mathcal{H} = \{\mathbf{H}_{\ell k}\}_{\ell, k}$.

B. Transferability of GNNs

The mathematical property that allows training GNNs on small graph subsamples of larger graphs is their transferability. Explicitly, GNNs are transferable in the sense that when a GNN with fixed weights \mathcal{H} is transferred across two graphs in the same “family”, the transference error is upper bounded by a term that decreases with the graph size. Typical transferability analyses show this by defining graph “families” as graphs coming from the same random graph model, or converging to a common graph limit. Here, we consider families of graphs identified by the same graphon, which can be seen as both a generative model and a limit model for large graphs.

A graphon is a bounded, symmetric, measurable function $\mathbf{W} : [0, 1]^2 \rightarrow [0, 1]$ [20], [21]. Graphs can be sampled from \mathbf{W} by sampling nodes u_1, \dots, u_n from $[0, 1]$, and sampling edges (u_i, u_j) with probability $\mathbf{W}(u_i, u_j)$. The graph limit interpretation is more nuanced but has to do with the fact that, on sequences of graphs converging to \mathbf{W} , the densities of certain “motifs”, e.g., triangles, also converge. For graphs associated with the same graphon, we have the following transferability theorem.

Theorem II.1 (GNN transferability, simplified [22]). *Let Φ be a GNN with fixed coefficients, and G_n, G_m graphs with n and m nodes sampled from a graphon \mathbf{W} . Under mild conditions, $\|\Phi(G_n) - \Phi(G_m)\| = \mathcal{O}(n^{-1} + m^{-1})$ w.h.p.*

In this paper, we will use the transferability property of GNNs, together with a novel graph sampling algorithm, to train GNNs on small graph subsamples and ensure they scale well to large graphs.

C. Expressive Power of GNNs

While GNNs achieve remarkable performance in many graph machine learning tasks, they have fundamental limitations associated with their expressive power [23], [24]. In GSP problems specifically, the expressivity of a GNN is constrained by the expressivity of the graph convolution, which in turn is constrained by the rank of the graph shift operator [12]. This is demonstrated in the following proposition.

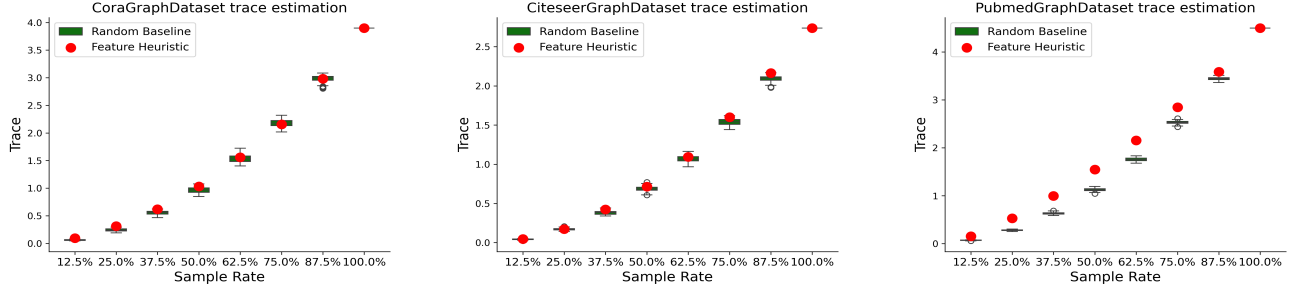


Fig. 1. Adjusted Laplacian trace versus graph subsampling rate. The adjusted trace is the subsampled graph Laplacian trace normalized by the number of sampled nodes. Boxplots indicate the mean and standard error of the trace obtained from 50 rounds of random node subsampling; red dots are the trace of subgraphs generated using our sampling heuristic (Algorithm 1). Trace preservation is almost always better for our algorithm compared with the average of random subsampling, except for Cora at 75% sample rate.

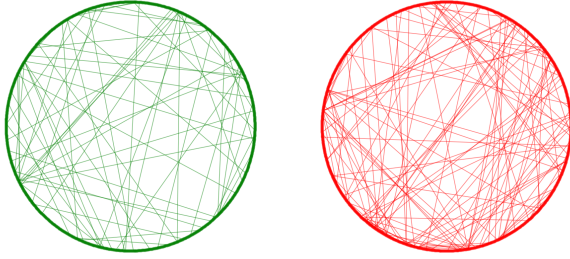


Fig. 2. Example of randomly sampled subgraph (green) and subgraph sampled using Algorithm 1. Both graphs have 800 nodes and were sampled from the PubMed citation network, which has 19,717 nodes.

Proposition II.2 (Expressivity of Graph Convolution). *Let G be an n -node symmetric graph with rank- r graph shift operator \mathbf{S} , $r < n$, and $x \in \mathbb{R}^n$ an arbitrary graph signal. Consider the graph convolution $\hat{y} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k x$. Let $\mathcal{Y} \subset \mathbb{R}^n$ be the subspace of signals that can be expressed as $y = \hat{y}$ for some h_0, \dots, h_{K-1} . Then, $\dim(\mathcal{Y}) \leq r + 1$.*

Proof. See the extended version, available here. \square

In other words, the space of signals that can be represented with a graph convolution shrinks with the rank of the graph shift. Rank preservation is thus an important consideration when sampling subgraphs for GNN training. We will focus on the more tractable problem of preserving the Laplacian trace, which can be seen as a continuous proxy of its rank.

III. FEATURE HOMOPHILY AND HOMOPHILY-BASED SAMPLING

We start by introducing the notion of feature homophily. In order to make this definition compatible for graphs of different sizes and with different features, we first need to normalize $X \in \mathbb{R}^{n \times d}$ along both the feature and node dimensions. Explicitly, let $\bar{\mu} = (\mu_1, \dots, \mu_d)^T \in \mathbb{R}^d$ be the mean feature vector and $\bar{\sigma} = (\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_d})^T \in \mathbb{R}^d$

the standard deviation vector. We define the normalized graph feature matrix as:

$$\hat{X} = (X - \mathbf{1}_n \bar{\mu}^T) \odot \left(\frac{1}{\sqrt{d}} \cdot \mathbf{1}_n \bar{\sigma}^T \right). \quad (3)$$

$$\text{I.e., } \hat{X}[i, j] = \frac{X[i, j] - \mu_j}{\sqrt{d} \sigma_j}.$$

Definition III.1 (Feature Homophily). Let G be a graph with Laplacian matrix \mathbf{L} , and let \hat{X} be the corresponding normalized feature matrix (3). The feature homophily of graph G is defined as:

$$h_G = \frac{1}{n} \cdot \text{tr}(-\mathbf{L} \hat{X} \hat{X}^T). \quad (4)$$

Since \mathbf{L} is positive semidefinite and the trace of the outer product is the product of traces, it is ready to see, by Cauchy-Schwarz, that $h_G \leq 0$ for any undirected graph G . The larger the feature homophily h_G , i.e., the closer it is to 0, the higher the alignment of the data X (or, more precisely, of its principal components) with the low-frequency eigenvectors of \mathbf{L} —which account for most of the graph’s global structure such as its connected components and clusters/communities. Thus, the data is informative with regards to the graph. On the other hand, highly negative values of h_G indicate strong alignment of \hat{X} with high-frequency eigenvectors, which tend to be noisier and less descriptive of the graph structure.

The following proposition provides a lower bound on $\text{tr}(\mathbf{L})$ in terms of the feature homophily h_G .

Proposition III.2 (Lower Bound on $\text{tr}(\mathbf{L})$).

$$\text{tr}(\mathbf{L}) \geq -\frac{nh_G}{\text{tr}(\hat{X} \hat{X}^T)} \quad (5)$$

Proof. See the extended version, available here. \square

Note that if the graph G has high feature homophily, the right-hand side of (5) is small and the lower bound on $\text{tr}(\mathbf{L})$ is approximately vacuous. For heterophilic graphs, h_G has higher magnitude, so the lower bound is further away from zero.

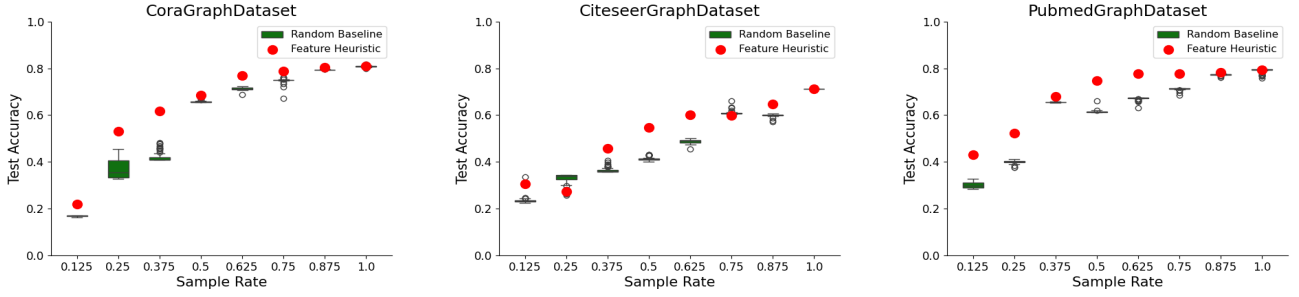


Fig. 3. Test accuracy achieved by GNN on full graph versus training graph subsampling rate. Boxplots indicate the spread of the accuracy realized by GNNs trained on 50 random node-induced subgraphs; red dots are the test accuracy of GNNs trained on subgraphs produced by our heuristic (Algorithm 1). The feature-based heuristic yields better results across the board, except for CiteSeer at 25% sample rate.

A. Sampling Heuristic for Homophilic Graphs

Although simple, the result in Proposition III.2 has important implications for feature-homophilic graph sampling. Assume we start removing nodes from G according to the diagonal entries of XX^T sorted in decreasing order, so that the denominator on the right-hand side of (5) becomes progressively smaller. We prioritize removing nodes corresponding to the largest diagonal elements because, in order to tighten the lower bound of $\text{tr}(\mathbf{L})$, we should enforce the denominator $\text{tr}(\hat{X}\hat{X}^T)^2$ to be as small as possible, given that the graph is homophilic. Hence, deleting from largest to smallest makes sense. This idea is formalized in Algorithm 1.

When $d < n$, as is often the case in practice, Algorithm 1 offers lower computational complexity than graph sampling algorithms which maximize the graph Laplacian trace directly, as demonstrated by Proposition III.3.

Proposition III.3 (Complexity of Algorithm 1). *Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$, and let $X \in \mathbb{R}^{n \times d}$ be the corresponding node feature matrix. For any sampling budget γ , the complexity of Algorithm 1, including computation of h_G , is $O(dm)$. If G is known to be feature-homophilic, computation of h_G can be bypassed and the complexity simplifies to $O(dn)$.*

Proof. See the extended version, available here. \square

Importantly, the complexity of Algorithm 1 is dominated by the complexity of calculating the feature homophily, which only has to happen once prior to execution to determine if the graph is homophilic. The complexity is further independent of the sampling budget, as the diagonal elements of XX^T only have to be computed and sorted once.

On sparse graphs ($m \ll n^2$) with moderate feature dimension d , Algorithm 1 is cheaper than direct maximization of $\text{tr}(\mathbf{L})$, which requires $O((1 - \gamma)n^2)$ computations— $O(n)$ node degree computations $(1 - \gamma)n$ times, as node degrees change each time a node is removed from G . In fact, this is an underestimation, as it does not factor in the cost of breaking ties across nodes with the same degree. Algorithm 1 is also notably cheaper than spectral algorithms such as [13], [14] which are inspired by E-optimal sampling and, without

exhaustive search, require greedy routines with complexity at least $O(\gamma nm)$. Another advantage of Algorithm 1 is that it does not require sequential execution, unlike maximization of $\text{tr}(\mathbf{L})$ and [13], [14], which cannot be parallelized.

B. Connections with Orthogonality, Leverage Scores Sampling, and Further Discussion

Our feature homophily definition provides insights into graph alignment and orthogonality. More specifically, h_G can be interpreted as an inner product between the Laplacian \mathbf{L} and the feature correlation matrix $\hat{X}\hat{X}^T$. High homophily (i.e., $h_G \approx 0$) implies that the graph Laplacian and the feature correlation matrix are almost orthogonal.

The feature homophily score can also be rewritten as a weighted ℓ_1 -norm of the adjacency matrix, where the weight of each edge A_{ij} is the squared Euclidean distance between feature vectors X_i and X_j [25]. This formulation connects homophily with graph sparsity, suggesting that our sampling heuristic could serve as a structured sparsification technique for large-scale graphs similar to graph sparsification techniques based on effective resistances [26].

The sampling method we propose in Algorithm 1 has interesting connections with leverage scores sampling approaches used in regression models. In classical settings, leverage scores quantify the influence of data points on the regression outcome. However, unlike traditional leverage score sampling, which retains influential, high-leverage points, our approach removes the nodes with the largest scores.

To formalize this connection, recall that leverage scores sampling is based on the diagonals of the hat matrix $\mathbf{H} = X(X^T X)^+ X^T$. Since \mathbf{H} is positive semidefinite, its diagonal is always nonnegative. Using the singular value decomposition (SVD) of X , $X = U\Sigma V^T$, we can decompose the hat matrix as $\mathbf{H} = U\Sigma V^T(V\Sigma^2 V^T)^+ V\Sigma U^T = U_r U_r^T$, thus the images of both X and \mathbf{H} lie in the span of U . Examining the diagonal of $XX^T = U\Sigma^2 U^T$, we observe that the node scores computed in our algorithm are effectively leverage scores weighted by the singular values of X . Our approach removes nodes that contribute disproportionately to the feature covariance structure, regularizing the selection.

This criterion aligns with the goal of preserving Laplacian rank: since the multiplicity of the zero eigenvalue counts the

number of connected components, removing nodes that form separate components or have low connectivity minimally affects the overall structure. Specifically, if a graph initially has n nodes and k connected components (so that $\text{rank}(\mathbf{L}) = n - k$), removing an isolated node decreases both n and k by one, preserving the rank at $n - k$. This suggests that our method implicitly filters out nodes that contribute little to the global structure while retaining the essential connectivity properties of the original graph.

IV. EXPERIMENTAL RESULTS

In this section, we present an empirical analysis of our sampling algorithm on citation networks. Specifically, we compare the test accuracy achieved by GNN models trained on subgraphs sampled using our heuristic and the random node sampling baseline on the full size graph.

Trace preservation. First, we investigate the ability of our heuristic to preserve Laplacian trace in three homophilic citation networks. In agreement with the lower bound derived in Proposition III.2, in Figure 1 we see that across all datasets, our sampling method results in subgraphs with larger adjusted $\tilde{tr}(\mathbf{L}) = \text{tr}(\mathbf{L})/n$. This pattern is especially visible for PubMed, as exemplified by the 800-node subsamples in Figure 2. Clearly, the graph sampled using Algorithm 1 is more connected than its random counterpart.

GNN training. Next, we investigate the transferability of GNNs trained on subgraphs sampled using our heuristic and subgraphs and random node-induced subgraphs. Specifically, we train a GNN model on graphs of size given by the x -axis of Figure 3, and report their test accuracy on the full size graph.

Experiment details. For each dataset and sample rate, we chose hidden dimension in $\{64, 128\}$, number of layers in $\{1, 2, 3\}$, number of epochs in $\{200, 300\}$, learning rate and weight decay in $\{0.001, 0.0001\}$, GCN model type in $\{\text{GCN}, \text{SAGE}\}$ with ReLU activations, all optimized using Adam [27]. All of the graphs are symmetrized beforehand, turning directed edges into undirected ones.

In Figure 3, the box plots represent the mean and standard error of the accuracy achieved over 50 rounds, and the red dots the test accuracy achieved by our model. Our sampling heuristic is substantially better, except for CiteSeer at sample rate 25%. Notably, at a sample rate of 62.5%, our heuristic beat random by around 10% for all three datasets.

Acknowledgements. We thank Gonzalo Mateos for insightful comments on the connections of our method with graph orthogonality and leverage scores sampling.

REFERENCES

- [1] Marco Gori, Gabriele Monfardini, and Franco Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE international joint conference on neural networks, 2005*. IEEE, 2005, vol. 2, pp. 729–734.
- [2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th Int. Conf. Learning Representations*, Toulon, France, 24–26 Apr. 2017, Assoc. Comput. Linguistics.
- [3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Inform. Process. Syst.*, Barcelona, Spain, 5–10 Dec. 2016, NIPS Foundation.
- [4] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, pp. 1034–1049, 2018.
- [5] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability and transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660–682, 2021.
- [6] Will Hamilton, Zhitaoying, and Jure Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.
- [7] L. Ruiz, F. Gama, A. G. Marques, and A. Ribeiro, "Invariance-preserving localized activation functions for graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 127–141, 2020.
- [8] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 5680–5695, 2020.
- [9] L. Ruiz, L. F. O. Chamon, and A. Ribeiro, "Graphon neural networks and the transferability of graph neural networks," in *34th Neural Inform. Process. Syst.*, Vancouver, BC (Virtual), 6–12 Dec. 2020, NeurIPS Foundation.
- [10] R. Levie, W. Huang, L. Bucci, M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *J. Mach. Learning Res.*, vol. 22, no. 272, pp. 1–59, 2021.
- [11] L. Ruiz, L. F. O. Chamon, and A. Ribeiro, "Graphon signal processing," *IEEE Trans. Signal Process.*, vol. 69, pp. 4961–4976, 2021.
- [12] Luana Ruiz, Ningyuan Teresa Huang, and Soledad Villar, "A spectral analysis of graph neural networks on dense and sparse graphs," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 9936–9940.
- [13] S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, pp. 6510–6523, 2015.
- [14] Aamir Anis, Akshay Gadde, and Antonio Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, 2016.
- [15] Yeganeh Alimohammadi, Luana Ruiz, and Amin Saberi, "A local graph limits perspective on sampling-based gnns," 2023.
- [16] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.
- [17] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.
- [18] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.
- [19] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, pp. 4117–4131, Aug. 2017.
- [20] C. Borgs, J. T. Chayes, L. Lovász, V. T. Sós, and K. Vesztegombi, "Convergent sequences of dense graphs I: Subgraph frequencies, metric properties and testing," *Adv. Math.*, vol. 219, no. 6, pp. 1801–1851, 2008.
- [21] L. Lovász, *Large Networks and Graph Limits*, vol. 60, American Mathematical Society, 2012.
- [22] L. Ruiz, L. F. O. Chamon, and A. Ribeiro, "Transferability properties of graph neural networks," *arXiv:2112.04629 [eess.SP]*. Submitted to *IEEE TSP*, 2021.
- [23] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *7th Int. Conf. Learning Representations*, New Orleans, LA, 6–9 May 2019, pp. 1–17, Assoc. Comput. Linguistics.
- [24] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna, "On the equivalence between graph isomorphism testing and function approximation with gnns," *Advances in neural information processing systems*, vol. 32, 2019.
- [25] Vassilis Kalofolias, "How to learn a graph from smooth signals," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2016, pp. 920–929.
- [26] Daniel A. Spielman and Nikhil Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011.
- [27] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations*, San Diego, CA, 7–9 May 2015, Assoc. Comput. Linguistics.

APPENDIX

Proof of Prop. II.2. From the definition, we can rewrite

$$\mathcal{Y} = \text{span}(\{S^k x : k = 0, \dots, K\})$$

Since \mathbf{S} is symmetric, there exists some $\mathbf{E}, \mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ satisfying $\mathbf{E}\mathbf{E}^T = \mathbf{I}_n$ and $\mathbf{\Sigma}$ diagonal, such that $\mathbf{S} = \mathbf{E}\mathbf{\Sigma}\mathbf{E}^T$. This implies a simple explicit representation for $\mathbf{S}^k = \mathbf{E}\mathbf{\Sigma}^k\mathbf{E}^T$. As a result, \mathbf{S}^k share not only the same rank, also the same coordinates regardless scaling. So for any given $x \in \mathbb{R}^n$, $\text{rank}(\{S^k x : k = 1, \dots, K\}) \leq \text{rank}(\mathbf{S}) = r$. Therefore, we have $\dim(\mathcal{Y}) \leq r + 1$ with x as an additional coordinate candidate. \square

Proof of Prop. III.2. By the definition of feature homophily, and using the Cauchy-Schwarz inequality, we write

$$-nh_G = \text{tr}(\mathbf{L}XX^T) = \langle \mathbf{L}, XX^T \rangle_F \leq \|\mathbf{L}\|_F \cdot \|XX^T\|_F. \quad (6)$$

This means by taking the square of both sides,

$$n^2 h_G^2 \leq \|\mathbf{L}\|_F^2 \cdot \|XX^T\|_F^2 \quad (7)$$

Now we rewrite it in terms of trace

$$\|\mathbf{L}\|_F^2 \cdot \|XX^T\|_F^2 = \text{tr}(\mathbf{L}^2) \text{tr}((XX^T)^2). \quad (8)$$

Because both \mathbf{L} and XX^T are positive semi-definite, we have

$$\text{tr}(\mathbf{L}^2) \text{tr}((XX^T)^2) \leq \text{tr}(\mathbf{L})^2 \text{tr}(XX^T)^2. \quad (9)$$

Finally, combining the above equations, we get the desired result

$$-nh_G \leq \text{tr}(\mathbf{L}) \text{tr}(\hat{X}\hat{X}^T). \quad (10)$$

\square

Proof of Prop. III.3. Algorithm II-A involves two main steps: computation of the homophily $\text{tr}(-\mathbf{L}XX^T)$, and of the node scores $\text{diag}(XX^T)$. Computation of $\text{tr}(-\mathbf{L}XX^T)$ has complexity $O(dm)$, as it is dominated by computing the matrix-matrix multiplication $\mathbf{L}X$ and the subsequent trace computation only requires $O(dn)$. Similarly, the computation of $\text{diag}(XX^T)$ requires $O(dn)$ as we only calculate the diagonal elements of XX^T . \square