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

Haolin Li and Luana Ruiz
Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, USA

***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's rank than random sampling while achieving lower complexity than spectral methods. Experiments on citation networks show improved performance in preserving graph rank and GNN transferability compared to random sampling.**

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

I. Introduction

Graph neural networks (GNNs) are deep neural networks tailored to network data which have shown great empirical performance in several graph machine learning tasks [**?**], [**?**], [**?**], [**?**]. This is especially true in graph signal processing problems—such as recommender systems on product similarity networks [**?**], or attribution of research papers to scientific domains [**?**]—in which GNNs' invariance and stability properties 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 [**?**], [**?**], 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 subsequently transferring it for inference on the large graph.

The transferability of GNNs is closely related to their convolutional parametrization, and is a consequence of the fact that graph convolutions converge on sequences of graphs converging to a common graph limit [**?**]. 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 subsamples, there is large variance in performance, and the worst-case performance can be quite low; see Figure **??**. While this is a natural consequence of subsampling any type of data, on graphs these issues are exacerbated by the fact that the random node-induced subgraphs usually have disconnected components and isolated nodes. This leads to loss of rank, which in turn affects the expressive power of GNNs [**?**]. Graph sampling algorithms better at preserving matrix rank—such as spectral algorithms, e.g. [**?**], [**?**]—or connectivity—such as local algorithms, e.g., breadth-first search—exist, however, spectral methods have high computational complexity, and local methods can be myopic, focusing too much on specific regions of the graph.

In this paper, we identify a property of graphs that allows them to be sampled more efficiently and without restricting to local regions: feature homophily. More specifically, let $G = (V, E)$ be a graph with node features $X \in \mathbb{R}^{|V| \times d}$. This graph is said to be feature-homophilic if, given that (i, j) is an edge in G, the node features $X[i, :]$ and $X[j, :]$ are close. Our first contribution is to introduce a novel mathematical definition of feature homophily based on the graph Laplacian. Then, we show that, by sampling nodes that minimize the trace of the correlation matrix XX^T, it is possible to improve the trace of the graph Laplacian, which is directly related to graph rank, on homophilic graphs. This heuristic is formalized as a graph sampling algorithm in Algorithm **??**. Unlike other graph sampling routines, it does not require sequential node additions/deletions, and has complexity $O(|V||E|)$, which is substantially cheaper than other graph sampling 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, 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 $m = |E|$ be the number of edges in G. Then, the graph adjacency matrix is the $n \times n$ matrix $\mathbf{A}$ with entries

$$\mathbf{A}[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The Laplacian matrix or graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1})$ 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) [?], [?], as it allows defining the notion of total variation of a signal x. Explicitly, the total variation of x is defined as $TV(x) = x^T \mathbf{L} x$ [?]. 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 $\lambda_1 \leq \ldots \leq \lambda_n$ 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 a class of deep learning models specifically designed for graph-structured data. GNNs are layered architectures 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. The notion of graph shift 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 $\mathbf{S} = \mathbf{L}$ [?]. 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 $h_0, \ldots, h_{K-1}$ 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 [?]

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

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

Equipped with the definition of graph convolution, we write the ℓth layer of a GNN as [?]

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

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a pointwise nonlinearity (e.g., the ReLU or sigmoid) acting nodewise on the graph. At layer $\ell = 1$, X_0 is the input data X, 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 $\mathcal{H} = \{\mathbf{H}_{\ell k}\}_{\ell, k}$.

Transferability. 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 $\mathbf{W} : [0,1]^2 \to [0,1]$ [?], [?]. Graphs can be sampled from $\mathbf{W}$ by sampling nodes $u_1, \ldots, u_n$ 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 [?]). *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.

Expressivity. While GNNs achieve remarkable performance in many graph machine learning tasks, they have fundamental limitations associated with their expressive power [?], [?]. 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 [?]. This is demonstrated in the following proposition.

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 $h_0, \ldots, h_{K-1}$. Then, $dim(\mathcal{Y}) \leq r + 1$.*

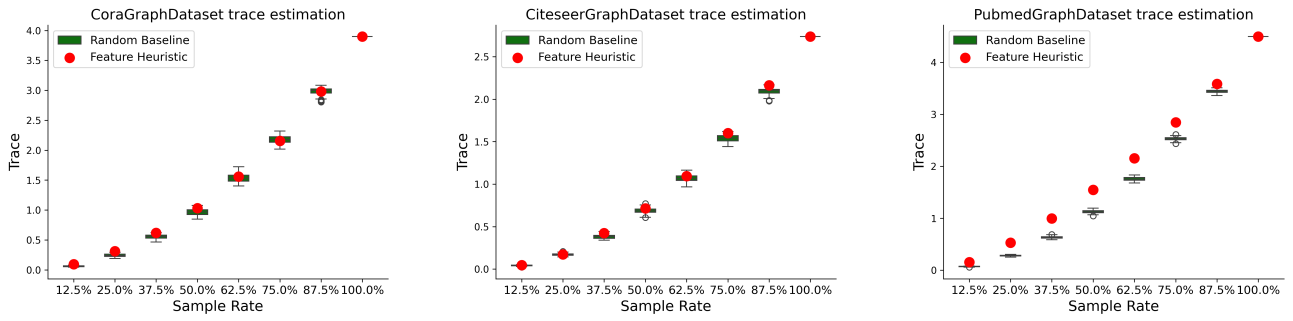

Fig. 1. This figure illustrates how does the trace of Laplacian change with respect to sample rate. Boxplots in the figure indicate the result obtained from random baselines in 50 runs, and the red dots are the adjusted trace of subgraphs generated by our sampling heuristic. Trace obtained by our result is almost always better than the average of random baselines except for Cora at 75%. The adjusted trace is calculated by dividing the number of nodes in the subgraphs.

Proof. See the appendices of the extended version, available here. □

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 hence an important consideration when sampling subgraphs for training GNNs.

III. Feature Homophily and Homophily-Based Sampling

We start by introducing the notion of feature homophily, which is a requirement for our algorithm. In order to make this definition compatible for graphs of different sizes, we first need to normalize $X \in \mathbb{R}^{n\times d}$ along both the feature and node dimensions. Explicitly, let $\vec{\mu} = (\mu_1, \cdots, \mu_d)^T \in \mathbb{R}^d$ be the mean feature vector and $\vec{\sigma} = (\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_d})^T \in \mathbb{R}^d$ the standard deviation vector. We define the normalized graph features $\hat{X}$ as:

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

In other words, $\hat{X}[i,j] = \frac{X[i,j]-\mu_j}{\sqrt{d}\sigma_j}$.

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 (**??**). The feature homophily of graph G is defined as:

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

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 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.

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: $\vec{s} \leftarrow \mathrm{diag}(XX^T)$
  Keep $n - n_d$ nodes with highest score:
      $\mathrm{idx} \leftarrow \mathrm{argmax}(\vec{s},\ \mathrm{descending} = \mathrm{True})[n_d : n]$
  Sample graph:
      $\tilde{V} \leftarrow V \cap \mathrm{idx}$
      $\tilde{E} \leftarrow \{(u,v) : u, v \in \tilde{V}, (u,v) \in E\}$
      $\tilde{X} \leftarrow X[\mathrm{idx}, :]$
  return $\tilde{G}(\tilde{V},\tilde{E})$; $\tilde{X}$
```

The following proposition provides a lower bound on the trace of the graph Laplacian in terms of the feature homophily h_G.

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

$$tr(\mathbf{L})^2 \geq -\frac{h_G}{tr(\hat{X}\hat{X}^T)^2} \tag{5}$$

Proof. See the appendices of the extended version, available here. □

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

Algorithm. Although simple, the result from Proposition **??** has important implications for feature-homophilic graph sampling. Suppose 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 (**??**) becomes progressively smaller. If G is homophilic with $h_G \approx 0$, the decrease in the denominator $tr(\hat{X}\hat{X}^T)$ is likely to lead to an increase of the lower bound on $tr(\mathbf{L})$. Therefore, we obtain lower bounds on $tr(\mathbf{L})$ that are increasingly more meaningful and, as a result, that ensure rank preservation in the sampled graph. This idea is formalized in Algorithm **??**.

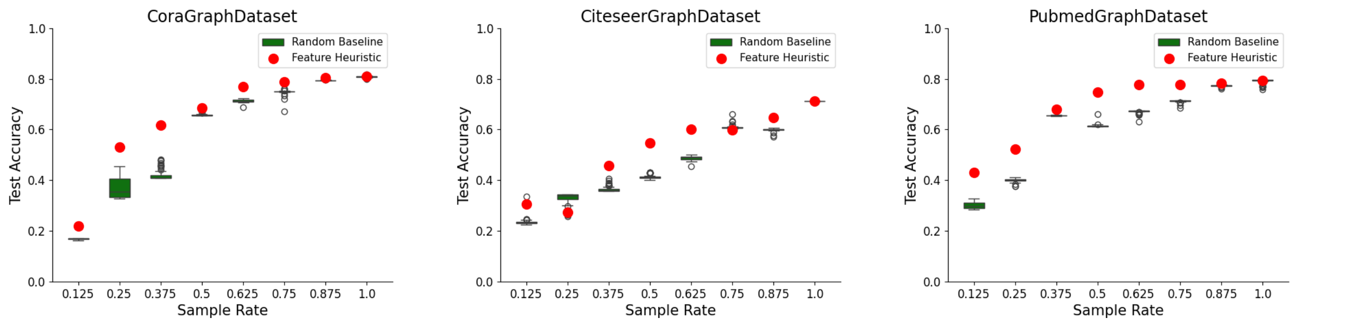

Fig. 2. The red dots are the test accuracy of models tuned on each dataset at specific sample rates, with boxplots indicating the distribution of random baseline test accuracy. Our heuristic yields better results only except for Citeseer at 25% of sample rate. Specifically, the differences of test accuracy are mostly recognizable at sample rates ranging from 37.5% to 75% across all three datasets. On average, accuracy grows slower as sample rate increases, intuitively because the marginal information is less significant with larger number of nodes already included in the subgraphs.

Complexity. When $d < n$, as is often the case in practice, Algorithm **??** offers lower computational complexity than graph sampling algorithms with rank preservation objectives, as demonstrated by Proposition **??**.

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 **??**, 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 appendices of the extended version, available here. □

Importantly, the complexity of Algorithm **??** 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 **??** is cheaper than direct maximization of $tr(\mathbf{L})$, 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 **??** is also notably cheaper than spectral algorithms such as [**?**], [**?**] 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 **??** is that it does not require sequential execution, unlike maximization of $tr(\mathbf{L})$ and [**?**], [**?**], which cannot be parallelized.

IV. Experimental Results

In this section, we present a further analysis on our sampling algorithm. Also, by comparing the test accuracy between models trained on subgraphs sampled by our algorithm and random baselines, we conclude the effectiveness and robustness of our proposed method.

Trace preservation. Among many ways to evaluate a sampling algorithm over a graph, apart from test accuracy benchmarks which we will show in Figure **??**, one of the most frequently used, and probably most informative one is the rank of $\mathbf{L}$. However, because the estimation on the rank of $\mathbf{L}$ is almost infeasible due to the high dimensionality, trace is often used as a mediate. With the lower bound we proposed in Proposition **??**, empirically from Figure **??** we see that across all three datasets, our sampling method results in subgraphs with larger adjusted $\tilde{\text{tr}}(\mathbf{L}) = \text{tr}(\mathbf{L})/n$ at almost all sample rates compared with the average of random baselines. This pattern is especially obvious over PubMed dataset.

GNN training. As for GNN experiments, for each dataset at a given sample rate, we firstly tune the best configuration for our sampling algorithm as shown in the following, then test the same configuration on random baselines with 50 different runs. Specifically, the valid/test accuracy obtained in the training and evaluation process is based on original full-size graph, which means we only limit the training process to the sampled subgraphs. This pipeline is ideal because it simulates the process of training and model selection, which makes the experiments more indicative for general purposes.

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 all ReLU activation between all layers, and used Adam as our optimizer. All of the graphs are configured to be undirected by adding edges of the opposite direction before being fed into the networks.

It is worth mentioning that the GNN of our choice is rather simple. This is because the power of a sampling method can be more apparent and easier to identify if model complexity is relatively restrained. In Figure **??**, where the box plots represent the distribution of random baseline accuracy, the test accuracy acquired by our model(the red dots) are all better by a distinctive margin except for Citeseer at sample rate 25%. In other cases, for instance, all three datasets at a sample rate of 62.5%, our heuristic won by around 10% of accuracy.

V. Conclusions

In this paper, we introduced a novel graph sampling algorithm that leverages feature homophily to efficiently preserve the structural properties of large graphs. Compared with random sampling, our proposed sampling heuristic is not only more effective in preserving the trace/rank of the graph, but also achieves superior performance when it is used to sample graphs for GNNs trained via transferability. These empirical results indicate the strong potential of our heuristic. Future work will focus on larger graph datasets such as ogbn-mag, where efficient training on subgraphs is even more pressing.

Appendix

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

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

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 $x \in \mathbb{R}^n$, rank$(\{S^k x : k = 1, \ldots, K\}) \leq$ 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

$$-h_G = tr(\mathbf{L}XX^T) = \langle \mathbf{L}, XX^T \rangle_F \leq \|\mathbf{L}\|_F^2 \cdot \|XX^T\|_F^2. \quad (6)$$

Now we rewrite it in terms of trace

$$\|\mathbf{L}\|_F^2 \cdot \|XX^T\|_F^2 = tr(\mathbf{L}^2)tr((XX^T)^2). \quad (7)$$

Because both $\mathbf{L}^2$ and $(XX^T)^2$ are positive semi-definite, we have

$$tr(\mathbf{L}^2)tr((XX^T)^2) \leq tr(\mathbf{L})^2 tr(XX^T)^2. \quad (8)$$

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

$$-h_G \leq \text{tr}(\mathbf{L}^2)\text{tr}((\hat{X}\hat{X}^T)^2) \leq \text{tr}(\mathbf{L})^2\text{tr}(\hat{X}\hat{X}^T)^2. \quad (9)$$

$\square$

Proof of Prop. III.3. Algorithm **??** involves two main steps: computation of the homophily $tr(-\mathbf{L}XX^t)$, and of the node scores diag(XX^T). Computation of $tr(-\mathbf{L}XX^t)$ 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 diag(XX^T) requires $O(dn)$ as we only calculate the diagonal elements of XX^T. $\square$