GRAPH HOMOPHILY BOOSTER: REIMAGINING THE ROLE OF DISCRETE FEATURES IN HETEROPHILIC GRAPH LEARNING

Ruizhong Qiu[*], **Ting-Wei Li**[*], **Gaotang Li, Hanghang Tong**
University of Illinois Urbana–Champaign, IL, USA
{rq5, twli, gaotang3, htong} @ illinois.edu

ABSTRACT

Graph neural networks (GNNs) have emerged as a powerful approach to modeling graph-structured data and demonstrated remarkable success in many real-world applications such as complex biological network analysis, neuroscientific analysis, and social network analysis. However, existing GNNs often struggle with heterophilic graphs, where connected nodes tend to have dissimilar features or labels. While numerous methods have been proposed to address this challenge, they primarily focus on architectural designs without directly targeting the root cause of the heterophily problem. These approaches still perform even worse than the simplest multi-layer perceptrons (MLPs) on challenging heterophilic datasets. For instance, our experiments show that 23 latest GNNs still fall behind the MLP on the ACTOR dataset. This critical challenge calls for an innovative approach to addressing graph heterophily beyond architectural designs. To bridge this gap, we propose and study a new and unexplored paradigm: *directly* increasing the graph homophily via a carefully designed graph transformation. In this work, we present a simple yet effective framework called *GRAph homoPHIly boosTEr* (GRAPHITE) to address graph heterophily. To the best of our knowledge, this work is the first method that explicitly transforms the graph to directly improve the graph homophily. Stemmed from the exact definition of homophily, our proposed GRAPHITE creates *feature nodes* to facilitate homophilic message passing between nodes that share similar features. Furthermore, we both theoretically and empirically show that our proposed GRAPHITE significantly increases the homophily of originally heterophilic graphs, with only a slight increase in the graph size. Extensive experiments on challenging datasets demonstrate that our proposed GRAPHITE significantly outperforms state-of-the-art methods on heterophilic graphs while achieving comparable accuracy with state-of-the-art methods on homophilic graphs. Furthermore, our proposed graph transformation alone can already enhance the performance of homophilic GNNs on heterophilic graphs, even though they were not originally designed for heterophilic graphs. Our code is publicly available at https://github.com/q-rz/ICLR26-GRAPHITE.

1 INTRODUCTION

Graph neural networks (GNNs) have emerged as a powerful class of models for learning on topologically structured data. Their ability to incorporate graph topology and node-level attributes has enabled them to achieve state-of-the-art performance in a wide range of applications. These include protein function prediction, where GNNs model complex biological networks (You et al., 2021; Réau et al., 2023); neuroscientific analysis, where they are used to model brain networks (Li et al., 2023a); and social network analysis, where they help uncover patterns among users (Li et al., 2023c).

A critical challenge that many GNNs are faced with is that real-world networks can exhibit heterophily, where connected nodes tend to have dissimilar features or labels. Examples include protein–protein interaction networks where different types of proteins interact (Zhu et al., 2020),

[*]Equal contribution.

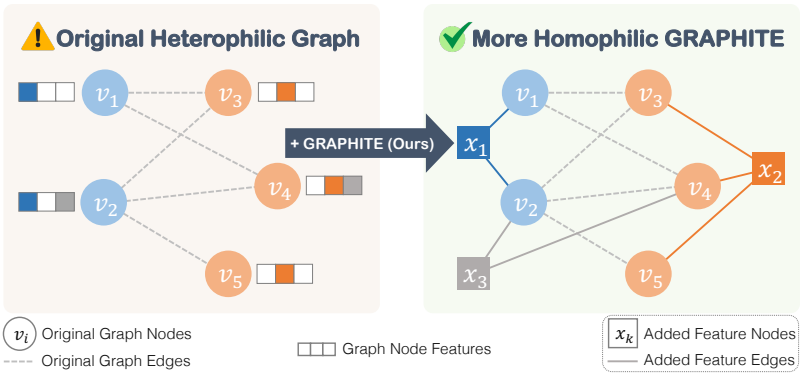

Figure 1: Illustration of our proposed GRAPHITE. Feature nodes/edges facilitate more homophilic message passing. For instance, feature node x_1 facilitates homophilic message passing between nodes v_1, v_2, and feature node x_2 facilitates homophilic message passing among nodes v_3, v_4, v_5.

or online marketplace networks where buyers connect with sellers rather than other buyers (Pandit et al., 2007). Standard GNN architectures (Kipf & Welling, 2016; Wu et al., 2019; Veličković et al., 2017; Hamilton et al., 2017; Chen et al., 2020; Abu-El-Haija et al., 2019), with their heavy reliance on neighborhood aggregation, often struggle with heterophilous graphs since aggregating features from dissimilar neighbors can dilute or distort node representations. Existing methods for heterophilic graphs mainly focus on designing new GNN architectures as workarounds for heterophilic graphs, such as separating ego and neighbor embeddings (Zhu et al., 2020; Zhang et al., 2023), incorporating multi-hop information via learnable weights (Chien et al., 2020; Dong et al., 2024), and adaptive self-gating to leverage both low- and high-frequency signals (Bo et al., 2021). More recent solutions introduce frequency-based filtering to handle both homophily and heterophily or leverage adaptive residual connections to further enhance flexibility (Xu et al., 2023; 2024a; Yan et al., 2024).

Despite plenty of architectural advances, many GNNs still perform even worse than the simplest multi-layer perceptrons (MLPs) on challenging heterophilic graphs. For instance, Table 2 shows that 23 latest GNNs still fall behind the MLP on the ACTOR dataset. This critical challenge calls for an innovative approach to addressing graph heterophily beyond architectural designs.

To bridge this gap, we propose and study a new and unexplored paradigm: *directly* increasing the graph homophily via a carefully designed graph transformation. In this work, we present a simple yet effective framework called *GRAph homoPHIly boosTEr* (GRAPHITE) to address graph heterophily. To the best of our knowledge, this work is the first method that explicitly transforms the graph to directly improve the graph homophily.

Our key idea is rooted in the exact definition of homophily/heterophily. In a homophilic/heterophilic graph, nodes that share similar features are more/less likely to be adjacent, respectively. Therefore, a natural idea to increase the graph homophily is to create "shortcut" connections between nodes with similar features so as to facilitate homophilic message passing. However, naïvely adding mutual connections between such node pairs can drastically increase the number of edges. To reduce the number of "shortcut" edges, we propose to connect such node pairs *indirectly* instead. In particular, we introduce *feature nodes* as "hubs" and connect graph nodes to their corresponding feature nodes. We further theoretically show that our proposed method can provably enhance the homophily of originally heterophilic graphs without increasing the graph size much.

Our main contributions are summarized as follows:

- **New paradigm.** We propose and study a new and unexplored paradigm: *provably* increasing the graph homophily directly via non-learning-based graph transformation. This paper is the first work on this paradigm to the best of our knowledge.
- **Proposed method.** We propose a simple yet effective method called GRAPHITE, which creates feature nodes as "shortcuts" to facilitate homophilic message passing between nodes with similar features.

- **Theoretical guarantees.** We theoretically show that GRAPHITE can *provably* enhance the homophily of originally heterophilic graphs with only a *slight* increase in size.
- **Empirical performance.** Extensive experiments on challenging datasets demonstrate the effectiveness of our proposed GRAPHITE. GRAPHITE *significantly* outperforms state-of-the-art methods on heterophilic graphs while achieving *comparable* accuracy with state-of-the-art methods on homophilic graphs. Furthermore, our proposed graph transformation alone can already enhance the performance of homophilic GNNs on heterophilic graphs.

2 PRELIMINARIES

2.1 NOTATION

An undirected graph with discrete node features can be represented as a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}$ denotes the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edge set, $\boldsymbol{X} \in \{0,1\}^{\mathcal{V} \times \mathcal{X}}$ is a binary node feature matrix representing discrete node features, and $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$ is the feature set containing all the discrete node features. In addition to that, each graph node $v_i \in \mathcal{V}$ has a node label $y_{v_i} \in \mathcal{Y}$, where $\mathcal{Y}$ is the label set with $C = |\mathcal{Y}|$ classes.

2.2 PROBLEM DEFINITION

In this paper, we study two key problems: (i) how to transform a graph to increase its homophily and (ii) how to perform node classification on a heterophilic graph datasets. Formally, we introduce the problem definitions as follows.

Problem 1 (boosting graph homophily)**.** *Given a highly heterophilic graph, transform the graph to increase its homophily.* ***Input:*** *a heterophilic graph $\mathcal{G}$.* ***Output:*** *a transformed graph $\mathcal{G}^*$ with higher homophily.*

Problem 2 (semi-supervised node classification on a heterophilic graph)**.** *Given a heterophilic graph and a set of labelled nodes, train a model to predict the labels of unlabelled nodes.* ***Input:*** *(i) a heterophilic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$; (ii) a labelled node set $\mathcal{V}_{\text{L}} \subset \mathcal{V}$ whose node labels $[y_{v_i}]_{v_i \in \mathcal{V}_{\text{L}}}$ are available.* ***Output:*** *the predicted labels of unlabeled nodes $\mathcal{V} \setminus \mathcal{V}_{\text{L}}$.*

3 PROPOSED METHOD: GRAPHITE

In this section, we propose a simple yet effective graph transformation method called ***GRAph homoPHIly boosTEr*** (GRAPHITE) that can efficiently increase the homophily of a graph. In Section 3.1, we will introduce the motivation of our proposed GRAPHITE. First, we will present the design of our proposed method GRAPHITE. Then, we will describe the neural architecture of our proposed method. Due to the page limit, proofs of theoretical results are deferred to the appendix.

3.1 MOTIVATION: NAÏVE HOMOPHILY BOOSTER

Graph heterophily is a ubiquitous challenge in graph-based machine learning. On a highly heterophilic graph, many neighboring nodes exhibit dissimilar features or belong to different classes. As a result, graph heterophily limits the effectiveness of GNN message passing, as standard aggregation schemes might fail to capture meaningful patterns in heterophilic neighbors.

Existing methods for heterophilic graphs mainly focus on designing workarounds such as new architectures or learning paradigms for heterophilic graphs, including adaptive message passing, higher-order neighborhoods, or alternative propagation mechanisms that leverage both local and global graph structures.

In contrast to existing workaround methods, we propose a new method that aims to directly increase the homophily of the graph via a specially designed graph transformation. To the best of our knowledge, this work is the first method that explicitly transforms the graph to improve the homophily of the graph.

Our idea is rooted in the exact definition of homophily and heterophily. In a heterophilic graph, nodes that share similar features are more likely to be non-adjacent. However, in a homophilic

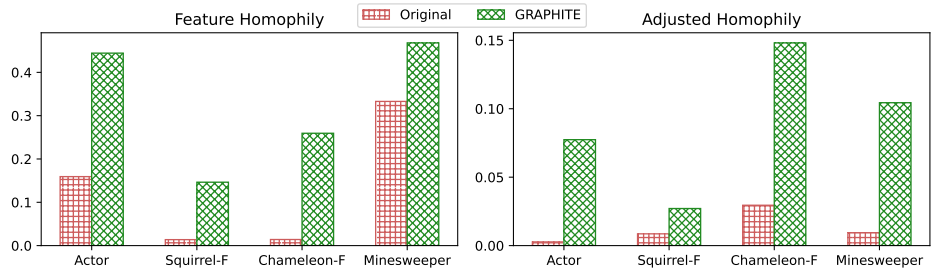

Figure 2: Our proposed GRAPHITE significantly increases the homophily of originally heterophilic graphs. We report two latest homophily metrics: *feature homophily* (Jin et al., 2022) and *adjusted homophily* (Platonov et al., 2024).

graph, nodes that share similar features should be more likely to be neighbors. Therefore, a natural idea to increase the homophily of the graph is to create "shortcut" connections between nodes with similar features, which will facilitate homophilic message passing between them.

Before we introduce the proposed method, let's consider the following naïve approach to implementing the aforementioned idea: For each pair of nodes $v_i, v_j \in \mathcal{V}$, if they share at least a feature (i.e., $\|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty > 0$), we add a "shortcut" edge (v_i, v_j) between them. Let's call this approach the *naïve homophily booster* (NHB). The following Theorem 1 shows that NHB can indeed increase the homophily of the graph under mild and realistic assumptions.

Theorem 1 (naïve homophily booster)**.** *Given a heterophilic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, let $\mathcal{E}^\dagger$ denote the set of edges after adding the NHB "shortcut" edges, and let $\mathcal{G}^\dagger := (\mathcal{V}, \mathcal{E}^\dagger, \boldsymbol{X})$ denote the graph transformed by NHB. Under mild and realistic assumptions in Appendix D.1, we have*

$$\hom(\mathcal{G}^\dagger) > \hom(\mathcal{G}), \tag{1}$$

$$|\mathcal{E}^\dagger| - |\mathcal{E}| \leq O(|\mathcal{V}|^2). \tag{2}$$

However, Equation (2) also shows that NHB is extremely inefficient despite its effectiveness in increasing homophily. For instance, even if the graph has only 2,000 nodes, NHB can add as many as 1,999,000 "shortcut" edges. The plenty of "shortcut" edges can drastically slow down the training and the inference process of GNNs. Hence, this naïve approach is computationally impractical for GNNs. To address this computational challenge, we will instead propose an efficient homophily booster via a more careful design of "shortcut" edges.

3.2 Efficient Graph Homophily Booster

To address the computational inefficiency of the motivating naïve approach above, we propose an efficient, simple yet effective graph transformation method called *GRAph homoPHIly boosTEr* (GRAPHITE) in this subsection.

Note that the large number of NHB "shortcut" edges is because NHB *directly* connects nodes with similar features. Since there are $O(|\mathcal{V}|^2)$ node pairs in a graph, then the total number of added NHB "shortcut" edges can be as large as $O(|\mathcal{V}|^2)$.

To reduce the number of "shortcut" edges, we propose to connect such node pairs *indirectly* instead. In particular, if we can create a few auxiliary "hub" nodes so that all such node pairs are *indirectly* connected through the "hub" nodes, then we will be able to significantly reduce the number of "shortcut" edges at only a small price of adding a few "hub" nodes. Therefore, we need to develop an appropriate design of the "hub" nodes.

Graph transformation. Following the aforementioned motivation, we propose to create a *feature node* x_k for each feature k to serve as the "hub" nodes. Let $\mathcal{V}_\mathcal{X}$ denote the set of feature nodes:

$$\mathcal{V}_\mathcal{X} := \{x_k : k \in \mathcal{X}\}. \tag{3}$$

To distinguish feature nodes $\mathcal{V}_\mathcal{X}$ from nodes $\mathcal{V}$ in the original graph, we call $\mathcal{V}$ *graph nodes* from now on. For each graph node $v_i \in \mathcal{V}$, if graph node v_i has feature k (i.e., $\boldsymbol{X}[v_i, k] = 1$), we add an

edge (v_i, x_k) to connect the graph node v_i and the feature node $x_k \in \mathcal{V}_\mathcal{X}$, and we call it a *feature edge*. Let $\mathcal{E}_\mathcal{X}$ denote the set of feature edges:

$$\mathcal{E}_\mathcal{X} := \{(v_i, x_k) : v_i \in \mathcal{V},\, x_k \in \mathcal{V}_\mathcal{X},\, \boldsymbol{X}[v_i, k] = 1\} \subseteq \mathcal{V} \times \mathcal{V}_\mathcal{X}.$$

To distinguish feature edges $\mathcal{E}_\mathcal{X}$ from the original edges $\mathcal{E}$, we call $\mathcal{E}$ *graph edges* from now on.

Finally, we define the transformed graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*, \boldsymbol{X}^*)$ as follows. The nodes $\mathcal{V}^*$ of the transformed graph $\mathcal{G}^*$ are the original graph nodes $\mathcal{V}$ and the added feature nodes $\mathcal{V}_\mathcal{X}$: $\mathcal{V}^* := \mathcal{V} \cup \mathcal{V}_\mathcal{X}$. The edges $\mathcal{E}^*$ of the transformed graph $\mathcal{G}^*$ are the original graph edges $\mathcal{E}$ and the added feature edges $\mathcal{E}_\mathcal{X}$: $\mathcal{E}^* := \mathcal{E} \cup \mathcal{E}_\mathcal{X}$. We can also equivalently define the edges of the transformed graph $\mathcal{G}^*$ by its adjacency matrix. Let $\boldsymbol{A}$ denote the adjacency matrix of the original graph $\mathcal{G}$. Then, the adjacency matrix $\boldsymbol{A}^*$ of the transformed graph $\mathcal{G}^*$ can be expressed in a block matrix form:

$$\boldsymbol{A}^* = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{X} \\ \boldsymbol{X}^\top & \boldsymbol{0} \end{bmatrix}. \tag{4}$$

It remains to define node features $\boldsymbol{X}^* \in \mathbb{R}^{\mathcal{V}^* \times \mathcal{X}}$ of the transformed graph. For each graph node $v_i \in \mathcal{V}$, we use its original features as its node features:$\boldsymbol{X}^*[v_i, :] := \boldsymbol{X}[v_i, :]$. For each feature node $x_k \in \mathcal{V}_\mathcal{X}$, we define its node feature as the average feature vector among the graph nodes v_i that are connected to feature node x_k:

$$\boldsymbol{X}^*[x_k, :] := \frac{1}{|\mathcal{E}_\mathcal{X} \cap (\mathcal{V} \times \{x_k\})|} \sum_{v_i : (v_i, x_k) \in \mathcal{E}_\mathcal{X}} \boldsymbol{X}[v_i, :]. \tag{5}$$

Our proposed graph transformation GRAPHITE is illustrated in Figure 1. In this example, $\{v_1, v_2, v_3, v_4, v_5\}$ are the graph nodes, where v_1, v_2 belong to one class, and v_3, v_4, v_5 belong to the other class. Our proposed GRAPHITE adds feature nodes x_1, x_2, x_3 to the graph. We can see that feature node x_1 facilitates homophilic message passing between v_1, v_2, and that feature node x_2 facilitates homophilic message passing among v_3, v_4, v_5.

Theoretical guarantees. The transformed graph $\mathcal{G}^*$ enjoys a few theoretical guarantees. First, an important property of the feature edges is that every pair of nodes that share features can be connected through feature edges within two hops, as formally stated in Observation 2. This ensures that nodes with similar features are close to each other on the transformed graph $\mathcal{G}^*$, facilitating homophilic message passing.

Observation 2 (two-hop indirect connection)**.** *For each pair of nodes $u, v \in \mathcal{V}$, if they share at least a feature (i.e., $\|\boldsymbol{X}[v_i, :] \wedge \boldsymbol{X}[v_j, :]\|_\infty > 0$), then v_i and v_j are two-hop neighbors on the transformed graph $\mathcal{G}^*$.*

Furthermore, we theoretically show that our proposed graph transformation GRAPHITE can increase the homophily of the graph without increasing the size of the graph much, as formally stated in Theorem 3.

Theorem 3 (efficient homophily booster)**.** *Given a heterophilic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, let $\mathcal{G}^* := (\mathcal{V}^*, \mathcal{E}^*, \boldsymbol{X}^*)$ denote the graph transformed by our proposed GRAPHITE. Under mild and realistic assumptions in Appendix D.1, we have*

$$\hom(\mathcal{G}^*) > \hom(\mathcal{G}), \tag{6}$$

$$|\mathcal{V}^*| \leq O(|\mathcal{V}|), \quad |\mathcal{E}^*| \leq O(|\mathcal{E}|). \tag{7}$$

The effectiveness of our proposed GRAPHITE is also empirically validated in Section 4.3. As shown in Figure 2, our proposed GRAPHITE significantly increases the homophily of originally heterophilic graph.

3.3 Neural Architecture

The transformed graph $\mathcal{G}^*$ can be readily fed into existing GNNs to boost their performance, even when the GNNs were originally designed for homophilic graphs, as demonstrated in Table 4. Meanwhile, to maximize the GNN performance on the transformed graph $\mathcal{G}^*$, we introduce a GNN architecture specially designed for the transformed graph in this subsection.

Table 1: Summary of dataset statistics. We use four heterophilic graphs and two homophilic graphs.

Statistic	Heterophilic Graphs				Homophilic Graphs	
	ACTOR	SQUIRREL-F	CHAMELEON-F	MINESWEEPER	CORA	CITESEER
# Nodes	7600	2223	890	10000	2708	3327
# Edges	33544	46998	8854	39402	5429	4732
# Features	931	2089	2325	7	1433	3703
# Classes	5	5	5	2	7	6
Homophily	0.0028	0.0086	0.0295	0.0094	0.7711	0.6707

To help the GNN distinguish graph nodes $\mathcal{V}$ from feature nodes $\mathcal{V}_{\mathcal{X}}$, we use different edge weights for different edges. As a reference weight, suppose that graph edges $\mathcal{E}$ have weight $w_{\mathcal{E}} := 1$. Let $w_{\mathcal{X}} > 0$ denote the weight of feature edges $\mathcal{E}_{\mathcal{X}}$. Following GCN (Kipf & Welling, 2016), we also use self-loops in GNN message passing; let $w_0 > 0$ denote the weight of self-loops.

Let d_u denote the weighted degree of each node $u \in \mathcal{V}^*$. Specifically, for each graph node $v_i \in \mathcal{V}$,

$$d_{v_i} := w_0 + \sum_{(v_i,v_j)\in\mathcal{E}} w_{\mathcal{E}} + \sum_{(v_i,x_k)\in\mathcal{E}_{\mathcal{X}}} w_{\mathcal{X}}; \tag{8}$$

and for each feature node $x_k \in \mathcal{V}_{\mathcal{X}}$,

$$d_{x_k} := w_0 + \sum_{(v_i,x_k)\in\mathcal{E}_{\mathcal{X}}} w_{\mathcal{X}}. \tag{9}$$

Inspired by FAGCN (Bo et al., 2021), we use a self-gating mechanism in GNN aggregation. For each node $u \in \mathcal{V}^*$, let $\boldsymbol{h}_u \in \mathbb{R}^m$ denote the embedding of node u before GNN aggregation, where m is the embedding dimensionality. Then, the self-gating score $\alpha_{u,u'}$ between two nodes $u, u' \in \mathcal{V}^*$ is defined as

$$\alpha_{u,u'} := \tanh\Big(\frac{\boldsymbol{a}^{\mathsf{T}}(\boldsymbol{h}_u \,\|\, \boldsymbol{h}_{u'}) + b}{\tau}\Big). \tag{10}$$

where $\|$ denotes the concatenation operation, $\boldsymbol{a} \in \mathbb{R}^{2m}$ and $b \in \mathbb{R}$ are learnable parameters, and $\tau > 0$ is a temperature hyperparameter.

Next, we describe our aggregation mechanism. For each node $u \in \mathcal{V}^*$, let $\boldsymbol{h}'_u \in \mathbb{R}^m$ denote the embedding of node u after GNN aggregation. For each graph node $v_i \in \mathcal{V}$, we define

$$\boldsymbol{h}'_{v_i} := \frac{w_0\alpha_{v_i,v_i}}{\sqrt{d_{v_i}}\sqrt{d_{v_i}}}\boldsymbol{h}_{v_i} + \sum_{(v_i,v_j)\in\mathcal{E}} \frac{\alpha_{v_i,v_j}}{\sqrt{d_{v_i}}\sqrt{d_{v_j}}}\boldsymbol{h}_{v_j} + \sum_{(v_i,x_j)\in\mathcal{E}_{\mathcal{X}}} \frac{w_{\mathcal{X}}\alpha_{v_i,x_k}}{\sqrt{d_{v_i}}\sqrt{d_{x_k}}}\boldsymbol{h}_{x_k}; \tag{11}$$

and for each feature node $x_k \in \mathcal{V}_{\mathcal{X}}$, we define

$$\boldsymbol{h}'_{x_k} := \frac{w_0\alpha_{x_k,x_k}}{\sqrt{d_{x_k}}\sqrt{d_{x_k}}}\boldsymbol{h}_{x_k} + \sum_{(v_i,x_k)\in\mathcal{E}_{\mathcal{X}}} \frac{w_{\mathcal{X}}\alpha_{v_i,x_k}}{\sqrt{d_{v_i}}\sqrt{d_{x_k}}}\boldsymbol{h}_{v_i}. \tag{12}$$

Furthermore, we add a multi-layer perceptron (MLP) with residual connections after each GNN aggregation. We use the GELU activation function (Hendrycks & Gimpel, 2016).

4 EXPERIMENTS

We conduct extensive experiments on both heterophilic and homophilic datasets to answer the following research questions:

- **RQ1:** How does our proposed GRAPHITE compare with state-of-the-art methods?
- **RQ2:** How much improvement can our GRAPHITE achieve in terms of the graph homophily?
- **RQ3:** Can our graph transformation alone enhance the accuracy of homophilic GNNs?
- **RQ4:** How should we design the features of feature nodes in our GRAPHITE?
- **RQ5:** How efficient is our GRAPHITE?
- **RQ6:** How effective is our GRAPHITE under various hyperparameters?

Table 2: Comparison with existing methods. GRAPHITE *significantly* outperforms state-of-the-art methods on heterophilic graphs while achieving *comparable* accuracy with state-of-the-art methods on homophilic graphs. Best results are marked in **bold**, and second best results are underlined.

Method	Heterophilic Graphs				Homophilic Graphs	
	ACTOR	SQUIRREL-F	CHAMELEON-F	MINESWEEPER	CORA	CITESEER
MLP	35.04 ± 1.53	33.91 ± 1.55	38.44 ± 5.14	50.99 ± 1.47	75.45 ± 1.88	71.53 ± 0.70
ChebNet	34.40 ± 1.18	31.75 ± 3.42	34.30 ± 4.33	$\underline{91.60} \pm 0.44$	81.58 ± 5.09	65.18 ± 8.37
GCN	30.21 ± 0.86	35.57 ± 1.86	40.06 ± 4.38	72.32 ± 0.93	87.50 ± 1.68	75.77 ± 0.96
SGC	29.26 ± 1.41	38.27 ± 2.16	41.40 ± 4.91	72.11 ± 0.95	88.05 ± 2.08	75.80 ± 1.75
GAT	28.86 ± 0.99	32.74 ± 3.02	40.11 ± 2.80	87.59 ± 1.35	87.11 ± 1.48	76.43 ± 1.31
GraphSAGE	34.95 ± 1.06	34.43 ± 2.68	39.33 ± 4.53	90.54 ± 0.66	87.90 ± 1.73	76.43 ± 1.19
GIN	28.29 ± 1.45	39.51 ± 2.83	40.17 ± 4.76	75.89 ± 2.09	85.65 ± 2.26	72.55 ± 1.78
APPNP	33.68 ± 1.26	33.75 ± 2.31	37.93 ± 4.33	67.36 ± 1.08	87.59 ± 1.68	75.90 ± 0.91
GCNII	34.78 ± 1.50	35.93 ± 2.87	41.56 ± 2.74	88.42 ± 0.85	87.20 ± 1.56	73.84 ± 0.91
GATv2	28.87 ± 1.39	32.49 ± 2.51	39.72 ± 6.60	88.85 ± 1.16	87.66 ± 1.52	76.59 ± 1.19
MixHop	35.40 ± 1.34	30.43 ± 2.33	37.93 ± 3.87	89.68 ± 0.57	84.53 ± 1.53	76.11 ± 0.83
TAGCN	34.92 ± 1.19	33.33 ± 2.37	41.01 ± 3.77	91.54 ± 0.56	88.38 ± 1.95	76.49 ± 1.41
DAGNN	33.15 ± 1.14	34.72 ± 2.55	38.94 ± 3.53	67.87 ± 1.26	88.27 ± 1.53	75.81 ± 0.90
JKNet	28.63 ± 0.94	$\underline{40.81} \pm 2.60$	40.39 ± 4.85	81.00 ± 0.92	86.24 ± 0.85	73.11 ± 1.82
Virtual Node	30.71 ± 0.82	38.00 ± 2.28	41.45 ± 5.46	72.36 ± 0.98	87.24 ± 2.00	69.80 ± 6.89
H2GCN	34.20 ± 1.47	34.02 ± 3.15	40.89 ± 3.13	87.08 ± 0.82	76.89 ± 2.25	75.87 ± 1.02
FSGNN	35.60 ± 1.34	37.28 ± 2.63	$\underline{43.30} \pm 3.62$	50.00 ± 0.00	87.81 ± 1.96	76.77 ± 1.13
ACM-GNN	34.04 ± 1.25	33.91 ± 2.28	39.78 ± 4.58	86.35 ± 0.99	88.58 ± 1.90	76.47 ± 0.99
FAGCN	$\underline{36.18} \pm 1.52$	36.52 ± 1.72	39.83 ± 3.93	84.69 ± 2.05	88.66 ± 2.11	$\underline{76.82} \pm 1.48$
OrderedGNN	35.64 ± 0.98	32.70 ± 2.42	38.38 ± 3.65	91.01 ± 0.50	84.81 ± 1.67	74.10 ± 1.62
GloGNN	19.80 ± 2.61	28.72 ± 2.63	40.17 ± 4.66	53.42 ± 1.47	73.02 ± 2.98	72.46 ± 2.09
GGCN	32.76 ± 1.39	35.06 ± 5.65	34.08 ± 3.44	84.76 ± 1.84	86.39 ± 1.93	75.36 ± 1.99
GPRGNN	35.42 ± 1.33	34.97 ± 2.83	40.50 ± 4.55	83.94 ± 0.98	$\mathbf{88.86} \pm 1.42$	76.49 ± 1.00
ALT	33.10 ± 1.38	37.28 ± 1.49	39.61 ± 3.36	89.06 ± 0.64	$\underline{88.82} \pm 2.02$	$\mathbf{76.88} \pm 1.20$
NodeFormer	29.26 ± 2.31	24.29 ± 2.60	34.92 ± 4.08	77.71 ± 3.50	87.44 ± 1.37	75.20 ± 1.27
SGFormer	25.89 ± 0.80	34.54 ± 2.96	42.79 ± 4.06	52.06 ± 0.50	86.24 ± 1.58	70.74 ± 1.25
DIFFormer	26.31 ± 1.19	33.17 ± 2.84	39.16 ± 4.10	69.25 ± 0.93	86.61 ± 3.04	76.65 ± 1.52
GRAPHITE (Ours)	$\mathbf{37.69} \pm 1.57$	$\mathbf{43.06} \pm 2.89$	$\mathbf{45.08} \pm 4.04$	$\mathbf{94.78} \pm 0.41$	88.23 ± 1.65	76.41 ± 1.57

4.1 EXPERIMENTAL SETTINGS

Datasets. We evaluate GRAPHITE and various baseline methods across six real-world datasets. The dataset statistics are summarized in Table 1. The reported homophily is the *adjusted homophily* introduced in Platonov et al. (2024), which exhibits more desirable properties compared to traditional edge/node homophily. We leverage *adjusted homophily* to categorize the datasets into two groups: *heterophilic* and *homophilic*. Please see Appendix A.1 for dataset descriptions.

Training and evaluation. To benchmark GRAPHITE and compare it with the baseline methods, we use *node classification* tasks with performance measured by classification accuracy on ACTOR, CHAMELEON-F, SQUIRREL-F, CORA, and CITESEER and by ROC-AUC on MINESWEEPER following Platonov et al. (2023). For all baseline methods, we use the hyperparameters provided by the authors. For the evaluation on ACTOR, CHAMELEON-F, and SQUIRREL-F, we generate 10 random splits with a ratio of $48\%/32\%/20\%$ as the training/validation/test set, respectively, following Gu et al. (2024). For the evaluation on MINESWEEPER, we directly utilize the 10 random splits provided by the original paper (Platonov et al., 2023). For the evaluation on CORA and CITESEER, we follow Luan et al. (2021) and Chien et al. (2020) to randomly generate 10 random splits with a ratio of $60\%/20\%/20\%$ as the training/validation/test set, respectively. For each experiment, we report the mean and the standard deviation of the performance metric across the corresponding 10 random splits. Please see Appendix A for additional experimental settings.

4.2 MAIN RESULTS

To answer RQ1, we compare the proposed method GRAPHITE with 27 state-of-the-art methods on six heterophilic and homophilic graphs. The results are shown in Table 2.

As shown in Table 2, our GRAPHITE achieves significant performance gains (p-value$<$0.1) over prior state-of-the-art GNN methods on heterophilic graphs while maintaining competitive accuracy on homophilic graphs. Specifically, GRAPHITE outperforms the best baseline methods by $4.17\%, 5.23\%, 5.35\%$ and 3.47% on ACTOR, SQUIRREL-F, CHEMELEON-F and MINESWEEPER, respectively. While some existing models perform well on individual datasets, they often struggle on others, highlighting their insufficient consistency. In contrast, GRAPHITE demonstrates the best results across all four heterophilic benchmarks. Another interesting observation is that while GRAPHITE is built upon FAGCN (Bo et al., 2021), it significantly surpasses FAGCN, demonstrating the beneficial effect of our graph transformation and feature edges.

Discussion. It is worth noting that most of the baseline methods cannot achieve better results compared to MLP on ACTOR, which can be explained by the fact that these methods typically treat node features and graph structure as joint input without explicitly decoupling them. The weak structural homophily exhibited by ACTOR makes typical GNNs fail to capture important feature signals, reinforcing the importance of our graph transformation strategy that boosts *feature homophily* significantly. For SQUIRREL-F, we find that JKNet is the best among baselines. This observation reveals that structure information is very important within SQUIRREL-F since JKNet aggregates feature knowledge from multi-hop neighbors to learn structure-aware representation. This finding also explains the success of GRAPHITE since the useful multi-hop information in SQUIRREL-F can be propagated even more efficiently through the constructed *feature edges*.

As another example, SGFormer performs the best on CHAMELEON-F among baseline methods. We argue that CHAMELEON-F needs a considerable amount of global messages and graph transformers are experts at capturing this type of information. Compared with NodeFormer and DIFFormer, SGFormer is the most advanced graph transformer utilizing simplified graph attention that strikes a good balance between global structural information and feature signal, preventing the over-globalizing issue (Xing et al., 2024). Similarly, GRAPHITE transforms the original graph into a form that facilitates global message exchange by the introduction of *feature edges*. As a final remark, although GRAPHITE is designed specifically to deal with heterophilic datasets, GRAPHITE still maintains competitive accuracy on homophilic datasets (CORA and CITESEER), achieving results that are on par with the best existing methods.

4.3 HOMOPHILY ANALYSIS

To answer RQ2, we conduct a homophily analysis across heterophilic datasets under two homophily metrics: *feature homophily* $H^{\text{feature}}(\mathcal{G})$ and *adjusted homophily* $H^{\text{adjusted}}(\mathcal{G})$ (see Appendix B for their formal definitions). Table 3 and Figure 2 show the relative improvements between the homophily metrics before and after applying GRAPHITE. We can observe a significant boost in both homophilily metrics after applying GRAPHITE across the four heterophilic datasets.

Table 3: Relative improvement ratio of *feature homophily* and *adjusted homophily* across datasets. GRAPHITE significantly boosts both homophily metrics. See Figure 2 for visualization.

Dataset	$H^{\text{feature}}(\mathcal{G})$	$H^{\text{adjusted}}(\mathcal{G})$
ACTOR	+179%	+2767%
SQUIRREL-F	+961%	+215%
CHAMELEON-F	+1739%	+402%
MINESWEEPER	+41%	+1023%

Discussion. Overall, GRAPHITE effectively boosts both homophily metrics across all heterophilic datasets. Specifically, Squirrel-F and Chameleon-F demonstrate significant boosts in terms of *feature homophily*. This is mainly because their discrete features directly correspond to specific topics and each feature edge will contribute much higher feature similarity than usual edges. On the other hand, Actor and Minesweeper showcase much higher *adjusted homophily* after applying GRAPHITE. For Actor, this favorable behavior can be attributed to the high correlation between page co-occurrences and node labels; while for Minesweeper, the sum of label-specific node degrees (defined in Equation (14)) increases much due to the transformation performed by GRAPHITE.

Baseline methods. In our experiments, we consider a wide range of GNN baselines, including MLP (structure-agnostic), homophilic GNNs, heterophilic GNNs, and Graph Transformers. The full list is shown in Appendix A.2.

Table 4: Effectiveness of the proposed graph transformation. GRAPHITE transformed graphs alone can already enhance the performance of homophilic GNNs.

Dataset	ACTOR		MINESWEEPER	
+GRAPHITE?	✗	✓	✗	✓
GCN	30.21 ± 0.86	**34.83** ± 1.28	72.32 ± 0.93	**75.38** ± 1.56
GAT	28.86 ± 0.99	**32.09** ± 1.35	87.59 ± 1.35	**88.66** ± 0.88
GraphSAGE	34.95 ± 1.06	**35.09** ± 1.06	90.54 ± 0.66	**90.85** ± 0.67
JKNet	28.63 ± 0.94	**35.96** ± 1.40	81.00 ± 0.92	**85.56** ± 2.59
GIN	28.29 ± 1.45	**33.75** ± 1.83	75.89 ± 2.09	**87.07** ± 1.71

Table 5: Comparison of aggregators for the features of feature nodes. All aggregators in fact perform similarly, so we choose averaging due to its simplicity and efficiency.

Dataset	Averaging (Ours)	Learned Embeddings	Learned Attention	Majority Voting
ACTOR	37.69	37.46	37.13	37.59
SQUIRREL-F	43.06	43.53	43.46	43.28
CHAMELEON-F	45.08	44.80	44.36	45.64
MINESWEEPER	94.78	94.47	94.56	94.75

4.4 ABLATION STUDIES

Graph transformation. To further demonstrate the effectiveness of our proposed graph transformation GRAPHITE and answer RQ3, we compare the performance of homophilic GNNs on the original graph and that on the transformed graph. In this experiment, we use two larger-scale datasets, ACTOR and MINESWEEPER, and five representative homophilic GNNs, GCN, GAT, GraphSAGE, JKNet, and GIN. The results are presented in Table 4.

From Table 4, we can see that our proposed GRAPHITE consistently improves the performance of the five representative homophilic GNNs on both datasets, even though these GNNs are not specially designed for modeling feature nodes. For example, the accuracy of GAT on ACTOR is enhanced from 30.21% to 34.83%, which is a relative improvement of 15.29%. The results demonstrate that our proposed graph transformation GRAPHITE can significantly enhance the performance of homophilic GNNs on originally heterophilic graphs, echoing the fact that our proposed graph transformation can significantly increase the graph homophily.

Features of feature nodes. GRAPHITE uses averaging aggregation to define the features of feature nodes. To elaborate on the rationale of this simple aggregator and answer RQ4, we compare with other aggregators on heterophilic datasets. The results are shown in Table 5. In fact, all aggregators have similar accuracies while averaging is more efficient than learned embeddings and attention-weighted aggregation and is simpler than majority voting. Therefore, we use averaging in our method due to its simplicity and efficiency.

Computational efficiency. To evaluate the efficiency of our method and answer RQ5, we provide a comparison of running times with and without our transformation, respectively. The results are shown in Table 6 (sorted by graph sizes). We can see that our graph transformation has only a small impact on running time while significantly improving accuracy. Notably, the computational overhead gets smaller on larger graphs (e.g., Minesweeper), justifying the scalability of our method. This is because the number of added nodes (i.e., the number of features) is typically negligibly small compared with the number of nodes on large graphs, and the number of feature edges is proportional to the space complexity of the node feature matrix $\boldsymbol{X}$.

Hyperparameters. Since our method has a few hyperparameters including τ and $w_{\mathcal{X}}$, to answer RQ6, we provide a sensitivity analysis of them on MINESWEEPER. The results are shown in Table 7. From the table, we can see that our method is not sensitive to these hyperparameters, and our method consistently outperform the best baseline under various hyperparameter values.

5 RELATED WORK

A substantial body of research has explored the challenges of heterophily in graph neural networks (GNNs). Many early approaches sought to improve information aggregation, such as MixHop (Abu-

Table 6: Running time comparison with and without GRAPHITE. Graphs are sorted in decreasing order by graph sizes. Our graph transformation has only a small impact on running time while significantly improves accuracy.

Method	MINESWEEPER	ACTOR	SQUIRREL-F	CHAMELEON-F
No transformation	1.9 min	1.5 min	0.7 min	0.5 min
With transformation	2.3 min	2.0 min	1.1 min	0.7 min

Table 7: Sensitivity analysis of hyperparameters τ and $w_{\mathcal{X}}$ on MINESWEEPER. Our method consistently outperform the best baseline under various hyperparameters.

Best Baseline	GRAPHITE (Ours)				
	$\tau = 0.1$	$\tau = 0.5$	$\tau = 1.0$	$\tau = 1.5$	$\tau = 2.0$
91.60	93.48	94.29	94.78	94.66	94.18
Best Baseline	**GRAPHITE (Ours)**				
	$w_{\mathcal{X}} = 0.1$	$w_{\mathcal{X}} = 0.25$	$w_{\mathcal{X}} = 0.5$	$w_{\mathcal{X}} = 0.75$	$w_{\mathcal{X}} = 1.0$
91.60	94.78	94.16	93.95	93.53	93.15

El-Haija et al., 2019), which mixes different-hop neighborhood features, and GPRGNN (Chien et al., 2020), which employs generalized PageRank propagation for adaptive message passing. Other methods focus on explicit heterophilic adaptations, such as H2GCN (Zhu et al., 2020), which separates ego- and neighbor-embeddings, and FAGCN (Bo et al., 2021), which learns optimal representations via frequency-adaptive filtering. Additional works, including OrderedGNN (Song et al., 2023), GloGNN (Li et al., 2022), and GGCN (Yan et al., 2022), leverage structural ordering, global context, and edge corrections, respectively, to enhance performance on heterophilic graphs. Recent advances explore alternative formulations, such as component-wise signal decomposition (e.g. ALT, Xu et al., 2023) and adaptive residual mechanisms (Xu et al., 2024a; Yan et al., 2024) for greater flexibility. Beyond architectural innovations, rigorous benchmarking efforts (Lim et al., 2021; Zhu et al., 2024; Platonov et al., 2023) have been introduced to standardize evaluations and assess generalization across diverse graph properties. A broader synthesis of heterophilic GNN techniques can be found in recent surveys (Zheng et al., 2022; Zhu et al., 2023; Luan et al., 2024; Gong et al., 2024). Please refer to Appendix C for additional related work.

6 CONCLUSION & FUTURE WORK

In this paper, we propose GRAPHITE, a simple yet efficient framework to address the heterophily issue in node classification. By introducing feature nodes that connect to graph nodes with corresponding discrete features, we can solve the heterophily issue by increasing the graph homophily ratio. Through theoretical analysis and empirical study, we validate that GRAPHITE can indeed effectively increase the graph homophily. Our extensive experiments demonstrate that GRAPHITE consistently outperforms state-of-the-art methods on heterophilic graph datasets and achieves comparable performance on homophilic graph datasets. An interesting future direction would be extending the proposed graph transformation to general graphs with continuous node features; possible approaches include clustering the continuous features into a few clusters and binning the continuous features into discrete buckets. Other future directions include (i) studying how our graph transformation affects graph properties, (ii) connecting to the node distinguishability principle (Luan et al., 2023), and (iii) identifying an optimal subset of features (Zheng et al., 2025).

ACKNOWLEDGEMENTS

This work is supported by NSF (2416070). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

ETHICS STATEMENT

Our study is based entirely on publicly available graph datasets commonly used in the GNN literature and does not involve private or sensitive information. We develop a graph transformation framework that explicitly increases graph homophily to enable more effective message passing. To ensure methodological soundness and reproducibility, we provide both theoretical analyses and extensive empirical evaluations across heterophilic datasets. The release of code and data splits is intended solely for academic research to advance the understanding of graph machine learning and to support future work on graph neural networks, and are not designed for sensitive or high-stakes applications.

REPRODUCIBILITY STATEMENT

We include the conceptual framework, transformation steps, method details and evaluation setup in the paper and appendix. To facilitate reproducibility, we also publicly release our code at `https://github.com/q-rz/ICLR26-GRAPHITE`.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 21–29. PMLR, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint*, 2020.

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. DiffWire: Inductive graph rewiring via the lovász bound. *arXiv preprint*, 2022.

Wenxuan Bao, Ruxi Deng, Ruizhong Qiu, Tianxin Wei, Hanghang Tong, and Jingrui He. Latte: Collaborative test-time adaptation of vision-language models in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Federico Barbero, Ameya Velingker, Amin Saberi, Michael Bronstein, and Francesco Di Giovanni. Locality-aware graph-rewiring in gnns. *arXiv preprint*, 2023.

Burak Bartan, Ruizhong Qiu, Rafael Esteves, Yuwei Ren, Weiliang Will Zeng, and An Chen. FineAMP: optimization-based automatic mixed precision quantization for efficient diffusion model inference. *The 17th International OPT Workshop on Optimization for Machine Learning*, 2025.

Ali Behrouz and Farnoosh Hashemi. Graph Mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in GNNs through the lens of effective resistance. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 2528–2547. PMLR, 2023.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3950–3957, 2021.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint*, 2021.

Eunice Chan, Zhining Liu, Ruizhong Qiu, Yuheng Zhang, Ross Maciejewski, and Hanghang Tong. Group fairness via group consensus. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1788–1808, 2024.

Lingjie Chen, Ruizhong Qiu, Siyu Yuan, Zhining Liu, Tianxin Wei, Hyunsik Yoo, Zhichen Zeng, and Deqing Yang. WAPITI: a watermark for finetuned open-source LLMs. *arXiv preprint*, 2024.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Sirui Chen, Yunzhe Qi, Mengting Ai, Yifan Sun, Ruizhong Qiu, Jiaru Zou, and Jingrui He. Influence-preserving proxies for gradient-based data selection in LLM finetuning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. *arXiv preprint*, 2020.

Chengming Cui, Tianxin Wei, Ziyi Chen, Ruizhong Qiu, Zhichen Zeng, Zhining Liu, Xuying Ning, Duo Zhou, and Jingrui He. AdaFuse: adaptive ensemble decoding with test-time scaling for LLMs. *arXiv preprint*, 2026.

Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *Learning on Graphs Conference*, pp. 38–1. PMLR, 2022.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 7865–7885. PMLR, 2023.

Yanfei Dong, Mohammed Haroon Dupty, Lambert Deng, Zhuanghua Liu, Yong Liang Goh, and Wee Sun Lee. Differentiable cluster graph neural network. *arXiv preprint*, 2024.

Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint*, 2017.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *The Seventh International Conference on Learning Representations*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

Chenghua Gong, Yao Cheng, Jianxiang Yu, Can Xu, Caihua Shan, Siqiang Luo, and Xiang Li. A survey on learning from graphs with heterophily: Recent advances and future directions. *arXiv preprint*, 2024.

Ming Gu, Zhuonan Zheng, Sheng Zhou, Meihan Liu, Jiawei Chen, Tanyu Qiao, Liangcheng Li, and Jiajun Bu. Universal inceptive GNNs by eliminating the smoothness-generalization dilemma. *arXiv preprint*, 2024.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Xinyu He, Jian Kang, Ruizhong Qiu, Fei Wang, Jose Sepulveda, and Hanghang Tong. On the sensitivity of individual fairness: Measures and robust algorithms. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 829–838, 2024.

Xinyu He, Chenhan Xiao, Haoran Li, Ruizhong Qiu, Zhe Xu, Yang Weng, Jingrui He, and Hanghang Tong. PowerGrow: feasible co-growth of structures and dynamics for power grid synthesis. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units. *arXiv preprint*, 2016.

Di Jin, Rui Wang, Meng Ge, Dongxiao He, Xiang Li, Wei Lin, and Weixiong Zhang. RAW-GNN: Random walk aggregation based graph neural network. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pp. 2108–2114, 2022.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Qi Yu, Dongqi Fu, Jingrui He, and Hanghang Tong. TRQA: time series reasoning question and answering benchmark. *OpenReview*, 2025.

Kedar Karhadkar, Pradeep Kr Banerjee, and Guido Montúfar. FoSR: First-order spectral rewiring for addressing oversquashing in gnns. *arXiv preprint*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Gaotang Li, Marlena Duda, Xiang Zhang, Danai Koutra, and Yujun Yan. Interpretable sparsification of brain graphs: Better practices and effective designs for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1223–1234, 2023a.

Gaotang Li, Ruizhong Qiu, Xiusi Chen, Heng Ji, and Hanghang Tong. Beyond log likelihood: Probability-based objectives for supervised fine-tuning across the model capability continuum. *arXiv preprint*, 2025a.

Mufei Li, Dongqi Fu, Limei Wang, Si Zhang, Hanqing Zeng, Kaan Sancak, Ruizhong Qiu, Haoyu Peter Wang, Xiaoxin He, Xavier Bresson, Yinglong Xia, Chonglin Sun, and Pan Li. Haystack engineering: Context engineering meets the long-context challenge in large language models. *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025b.

Ting Wei Li, Qiaozhu Mei, and Jiaqi Ma. A metadata-driven approach to understand graph neural networks. *Advances in Neural Information Processing Systems*, 36:15320–15340, 2023b.

Ting-Wei Li, Ruizhong Qiu, and Hanghang Tong. Graph data selection for domain adaptation: A model-free approach. In *Advances in Neural Information Processing Systems 38*, 2025c.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 13242–13256. PMLR, 2022.

Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549:126441, 2023c.

Zihao Li, Zhichen Zeng, Xiao Lin, Feihao Fang, Yanru Qu, Zhe Xu, Zhining Liu, Xuying Ning, Tianxin Wei, Ge Liu, Hanghang Tong, and Jingrui He. Flow matching meets biology and life science: a survey. *arXiv preprint*, 2025d.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20887–20902, 2021.

Xiao Lin, Zhining Liu, Dongqi Fu, Ruizhong Qiu, and Hanghang Tong. BackTime: Backdoor attacks on multivariate time series forecasting. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xiao Lin, Zhining Liu, Ze Yang, Gaotang Li, Ruizhong Qiu, Shuke Wang, Hui Liu, Haotian Li, Sumit Keswani, Vishwa Pardeshi, Huijun Zhao, Wei Fan, and Hanghang Tong. MORALISE: a structured benchmark for moral alignment in visual language models. *arXiv preprint*, 2025.

Xiao Lin, Zhicheng Tang, Weilin Cong, Mengyue Hang, Kai Wang, Yajuan Wang, Zhichen Zeng, Ting-Wei Li, Hyunsik Yoo, Zhining Liu, Xuying Ning, Ruizhong Qiu, Wen-Yen Chen, Shuo Chang, Rong Jin, Huayu Li, and Hanghang Tong. Mixture of sequence: Theme-aware mixture-of-experts for long-sequence recommendation. In *Proceedings of the ACM Web Conference 2026*, 2026.

Lihui Liu, Zihao Wang, Ruizhong Qiu, Yikun Ban, Eunice Chan, Yangqiu Song, Jingrui He, and Hanghang Tong. Logic query of thoughts: Guiding large language models to answer complex logic queries with knowledge graphs. *arXiv preprint*, 2024a.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020.

Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. Topological augmentation for class-imbalanced node classification, 2023.

Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. Class-imbalanced graph learning without class rebalancing. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b.

Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Yada Zhu, Hendrik Hamann, and Hanghang Tong. AIM: Attributing, interpreting, mitigating data unfairness. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2014–2025, 2024c.

Zhining Liu, Ziyi Chen, Hui Liu, Chen Luo, Xianfeng Tang, Suhang Wang, Joy Zeng, Zhenwei Dai, Zhan Shi, Tianxin Wei, Benoit Dumoulin, and Hanghang Tong. Seeing but not believing: Probing the disconnect between visual attention and answer correctness in vlms. *arXiv preprint*, 2025a.

Zhining Liu, Ze Yang, Xiao Lin, Ruizhong Qiu, Tianxin Wei, Yada Zhu, Hendrik Hamann, Jingrui He, and Hanghang Tong. Breaking silos: Adaptive model fusion unlocks better time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025b.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint*, 2021.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1362–1375, 2022.

Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28748–28760, 2023.

Sitao Luan, Chenqing Hua, Qincheng Lu, Liheng Ma, Lirong Wu, Xinyu Wang, Minkai Xu, Xiao-Wen Chang, Doina Precup, Rex Ying, Stan Z. Li, Jian Tang, Guy Wolf, and Stefanie Jegelka. The heterophilic graph learning handbook: Benchmarks, models, theoretical analysis, applications and challenges. *arXiv preprint*, 2024.

Jiaqi Ma, Xingjian Zhang, Hezheng Fan, Jin Huang, Tianyue Li, Ting-Wei Li, Yiwen Tu, Chenshu Zhu, and Qiaozhu Mei. Graph learning indexer: A contributor-friendly and metadata-rich platform for graph learning benchmarks. In *Learning on Graphs Conference*, pp. 7–1. PMLR, 2022.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design. *arXiv preprint*, 2021.

Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 25956–25979. PMLR, 2023.

Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 201–210, 2007.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *The Eighth International Conference on Learning Representations*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

Chendi Qian, Andrei Manolache, Kareem Ahmed, Zhe Zeng, Guy Van den Broeck, Mathias Niepert, and Christopher Morris. Probabilistically rewired message-passing neural networks. *arXiv preprint*, 2023.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 35, pp. 25531–25546, 2022.

Ruizhong Qiu, Dingsu Wang, Lei Ying, H Vincent Poor, Yifang Zhang, and Hanghang Tong. Reconstructing graph diffusion history from a single snapshot. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1978–1988, 2023.

Ruizhong Qiu, Jun-Gi Jang, Xiao Lin, Lihui Liu, and Hanghang Tong. TUCKET: A tensor time series data structure for efficient and accurate factor analysis over time ranges. *Proceedings of the VLDB Endowment*, 17(13), 2024.

Ruizhong Qiu, Gaotang Li, Ting-Wei Li, Tianxin Wei, Jingrui He, and Hanghang Tong. Efficient inference scaling for safety assurance. *NeurIPS 2025 Workshop on Vision–Language Model Real-World Deployment*, 2025a.

Ruizhong Qiu, Zhe Xu, Wenxuan Bao, and Hanghang Tong. Ask, and it shall be given: On the Turing completeness of prompting. In *13th International Conference on Learning Representations*, 2025b.

Ruizhong Qiu, Weiliang Will Zeng, Hanghang Tong, James Ezick, and Christopher Lott. How efficient is LLM-generated code? A rigorous & high-standard benchmark. In *13th International Conference on Learning Representations*, 2025c.

Ruizhong Qiu, Hanqing Zeng, Yinglong Xia, Yiwen Meng, Ren Chen, Jiarui Feng, Dongqi Fu, Qifan Wang, Jiayi Liu, Jun Xiao, Xiangjun Fan, Benyu Zhang, Hong Li, Zhining Liu, Hyunsik Yoo, Zhichen Zeng, Tianxin Wei, and Hanghang Tong. ReMix: reinforcement routing for mixtures of LoRAs in LLM finetuning. *arXiv preprint*, 2026.

Manon Réau, Nicolas Renaud, Li C. Xue, and Alexandre M.J.J. Bonvin. DeepRank-GNN: A graph neural network framework to learn patterns in protein–protein interfaces. *Bioinformatics*, 39(1), 2023.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2), 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

Dai Shi, Yi Guo, Zhiqi Shao, and Junbin Gao. How curvature enhance the adaptation power of framelet GCNs. *arXiv preprint*, 2023a.

Dai Shi, Andi Han, Lequan Lin, Yi Guo, and Junbin Gao. Exposition on over-squashing problem on GNNs: Current methods, benchmarks and challenges. *arXiv preprint*, 2023b.

Yunchong Song, Chenghu Zhou, Xinbing Wang, and Zhouhan Lin. Ordered GNN: Ordering message passing to deal with heterophily and over-smoothing. In *The Eleventh International Conference on Learning Representations*, 2023.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 807–816, 2009.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *The Sixth International Conference on Learning Representations*, 2018.

Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. Networked time series imputation via position-aware graph enhanced variational autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2256–2268, 2023.

Tianxin Wei, Yuning You, Tianlong Chen, Yang Shen, Jingrui He, and Zhangyang Wang. Augmentations in hypergraph contrastive learning: Fabricated and generative. *Advances in neural information processing systems*, 35:1909–1922, 2022.

Tianxin Wei, Ruizhong Qiu, Yifan Chen, Yunzhe Qi, Jiacheng Lin, Wenju Xu, Sreyashi Nag, Ruirui Li, Hanqing Lu, Zhengyang Wang, Chen Luo, Hui Liu, Suhang Wang, Jingrui He, Qi He, and Xianfeng Tang. Robust watermarking for diffusion models: A unified multi-dimensional recipe, 2024.

Tianxin Wei, Xuying Ning, Xuxing Chen, Ruizhong Qiu, Yupeng Hou, Yan Xie, Shuang Yang, Zhigang Hua, and Jingrui He. CoFiRec: coarse-to-fine tokenization for generative recommendation. *arXiv preprint*, 2025a.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-Memory: Benchmarking LLM agent test-time learning with self-evolving memory. *arXiv preprint*, 2025b.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, Zihao Li, Mengting Ai, Duo Zhou, Wenxuan Bao, Yunzhe Li, Gaotang Li, Cheng Qian, Yu Wang, Xiangru Tang, Yin Xiao, Liri Fang, Hui Liu, Xianfeng Tang, Yuji Zhang, Chi Wang, Jiaxuan You, Heng Ji, Hanghang Tong, and Jingrui He. Agentic reasoning for large language models: A survey. *arXiv preprint*, 2026a.

Tianxin Wei, Ruizhong Qiu, Yifan Chen, Yunzhe Qi, Jiacheng Lin, Wenxuan Bao, Wenju Xu, Sreyashi Nag, Ruirui Li, Hanqing Lu, Zhengyang Wang, Chen Luo, Hui Liu, Suhang Wang, Jingrui He, Qi He, and Xianfeng Tang. DiffKGW: stealthy and robust diffusion model watermarking. *Transactions on Machine Learning Research*, 2026b.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. NodeFormer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27387–27401, 2022.

Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. *arXiv preprint*, 2023.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems*, volume 36, 2024a.

Ziwei Wu, Lecheng Zheng, Yuancheng Yu, Ruizhong Qiu, John Birge, and Jingrui He. Fair anomaly detection for imbalanced groups. *arXiv preprint*, 2024b.

Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. Less is more: On the over-globalizing problem in graph transformers. *arXiv preprint*, 2024.

Haobo Xu, Yuchen Yan, Dingsu Wang, Zhe Xu, Zhichen Zeng, Tarek F. Abdelzaher, Jiawei Han, and Hanghang Tong. SLOG: An inductive spectral graph neural network beyond polynomial filter. In *Proceedings of the 41st International Conference on Machine Learning*, 2024a.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *The Sixth International Conference on Learning Representations*, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018b.

Zhe Xu, Yuzhong Chen, Qinghai Zhou, Yuhang Wu, Menghai Pan, Hao Yang, and Hanghang Tong. Node classification beyond homophily: Towards a general solution. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2862–2873, 2023.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024b.

Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Minghua Xu, Mahashweta Das, Hao Yang, and Hanghang Tong. From trainable negative depth to edge heterophily in graphs. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining*, pp. 1287–1292. IEEE, 2022.

Hyunsik Yoo, Zhichen Zeng, Jian Kang, Ruizhong Qiu, David Zhou, Zhining Liu, Fei Wang, Charlie Xu, Eunice Chan, and Hanghang Tong. Ensuring user-side fairness in dynamic recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pp. 3667–3678, 2024.

Hyunsik Yoo, SeongKu Kang, Ruizhong Qiu, Charlie Xu, Fei Wang, and Hanghang Tong. Embracing plasticity: Balancing stability and plasticity in continual recommender systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025a.

Hyunsik Yoo, Ruizhong Qiu, Charlie Xu, Fei Wang, and Hanghang Tong. Generalizable recommender system during temporal popularity distribution shifts. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025b.

Ronghui You, Shuwei Yao, Hiroshi Mamitsuka, and Shanfeng Zhu. DeepGraphGO: Graph neural network for large-scale, multispecies protein function prediction. *Bioinformatics*, 37 (Supplement_1):i262–i271, 2021.

Qi Yu, Zhichen Zeng, Yuchen Yan, Lei Ying, R. Srikant, and Hanghang Tong. Joint optimal transport and embedding for network alignment. In *Proceedings of the ACM on Web Conference 2025*, pp. 2064–2075, 2025.

Qi Yu, Zhichen Zeng, Yuchen Yan, Zhining Liu, Baoyu Jing, Ruizhong Qiu, Ariful Azad, and Hanghang Tong. PlanetAlign: a comprehensive Python library for benchmarking network alignment. In *The Fourteenth International Conference on Learning Representations*, 2026.

Zhichen Zeng, Si Zhang, Yinglong Xia, and Hanghang Tong. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM web conference 2023*, pp. 372–382, 2023a.

Zhichen Zeng, Ruike Zhu, Yinglong Xia, Hanqing Zeng, and Hanghang Tong. Generative graph dictionary learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 40749–40769, 2023b.

Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16660–16668, 2024a.

Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. Graph mixup on approximate Gromov–Wasserstein geodesics. In *Proceedings of the 41st International Conference on Machine Learning*, 2024b.

Zhichen Zeng, Mengyue Hang, Xiaolong Liu, Xiaoyi Liu, Xiao Lin, Ruizhong Qiu, Tianxin Wei, Zhining Liu, Siyang Yuan, Chaofei Yang, Yiqun Liu, Hang Yin, Jiyan Yang, and Hanghang Tong. Hierarchical LoRA MoE for efficient CTR model scaling. *arXiv preprint*, 2025a.

Zhichen Zeng, Xiaolong Liu, Mengyue Hang, Xiaoyi Liu, Qinghai Zhou, Chaofei Yang, Yiqun Liu, Yichen Ruan, Laming Chen, Yuxin Chen, Yujia Hao, Jiaqi Xu, Jade Nie, Xi Liu, Buyun Zhang, Wei Wen, Siyang Yuan, Hang Yin, Xin Zhang, Kai Wang, Wen-Yen Chen, Yiping Han, Huayu Li, Chunzhi Yang, Bo Long, Philip S. Yu, and Jiyan Yang Hanghang Tong. InterFormer: Effective heterogeneous interaction learning for click-through rate prediction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6225–6233, 2025b.

Zhichen Zeng, Wenxuan Bao, Xiao Lin, Ruizhong Qiu, Tianxin Wei, Xuying Ning, Yuchen Yan, Chen Luo, Monica Xiao Cheng, Jingrui He, and Hanghang Tong. Subspace alignment for vision-language model test-time adaptation. *arXiv preprint*, 2026a.

Zhichen Zeng, Ruizhong Qiu, Wenxuan Bao, Tianxin Wei, Xiao Lin, Yuchen Yan, Tarek F. Abdelzaher, Jiawei Han, and Hanghang Tong. Pave your own path: Graph gradual domain adaptation on fused Gromov–Wasserstein geodesics. *Transactions on Machine Learning Research*, 2026b.

Zhichen Zeng, Qi Yu, Xiao Lin, Ruizhong Qiu, Xuying Ning, Tianxin Wei, Yuchen Yan, Jingrui He, and Hanghang Tong. Harnessing consistency for robust test-time LLM ensemble. *Findings of EACL 2026*, 2026c.

Acong Zhang, Ping Li, and Guanrong Chen. Steering graph neural networks with pinning control. *arXiv preprint*, 2023.

Yunkai Zhang, Qiang Zhang, Diji Yang, Ryan Lin, Ruizhong Qiu, Benyu Zhang, Hanchao Yu, Jason Liu, Yinglong Xia, Zhuokai Zhao, Lizhu Zhang, Xiangjun Fan, Zhuoran Yu, Abhishek Kumar, and Zeyu Zheng. Guiding generative recommender systems with structured human priors via multi-head decoding. In *Proceedings of the ACM Web Conference 2026*, 2026.

Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S. Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint*, 2022.

Yilun Zheng, Xiang Li, Sitao Luan, Xiaojiang Peng, and Lihui Chen. Let your features tell the differences: Understanding graph convolution by feature splitting. In *The Thirteenth International Conference on Learning Representations*, 2025.

Duo Zhou, Yuji Zhang, Tianxin Wei, Ruizhong Qiu, Ke Yang, Xiao Lin, Cheng Qian, Jingrui He, Hanghang Tong, Heng Ji, and Huan Zhang. Geometric disentangelment unlearning. *arXiv preprint*, 2025.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7793–7804, 2020.

Jiong Zhu, Yujun Yan, Mark Heimann, Lingxiao Zhao, Leman Akoglu, and Danai Koutra. Heterophily and graph neural networks: Past, present and future. *IEEE Data Engineering Bulletin*, 2023.

Jiong Zhu, Gaotang Li, Yao-An Yang, Jing Zhu, Xuehao Cui, and Danai Koutra. On the impact of feature heterophily on link prediction with graph neural networks. *arXiv preprint*, 2024.

Jiaru Zou, Yikun Ban, Zihao Li, Yunzhe Qi, Ruizhong Qiu, Ling Yang, and Jingrui He. Transformer copilot: Learning from the mistake log in LLM fine-tuning. In *Advances in Neural Information Processing Systems 38*, 2025a.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. *arXiv preprint*, 2025b.

A EXPERIMENTAL SETTINGS (CONT'D)

A.1 DATASETS (CONT'D)

For heterophilic group, we consider the following datasets, which are widely used as benchmarks for studying graph learning methods under heterophilic settings.

- ACTOR (Pei et al., 2020): ACTOR dataset is an actor-only induced subgraph of the film dataset introduced by (Tang et al., 2009). The nodes are actors and the edges denote co-occurrence on the same Wikipedia page. The node features are keywords on the pages and we classify nodes into five categories.
- Squirrel-Filtered (SQUIRREL-F, Platonov et al., 2023): SQUIRREL-F is a page-page dataset. It is a subset of the Wiki dataset (Rozemberczki et al., 2021) that focus on the topic related to squirrel. Nodes are web pages and edges are mutual links between pages. The node features are important keywords in the pages and we classify nodes into five categories in terms of traffic of the webpage.
- Chameleon-Filtered (CHAMELEON-F, Platonov et al., 2023): CHAMELEON-F is a page-page dataset. It is a subset of the Wiki dataset (Rozemberczki et al., 2021) that focus on the topic related to chameleon. Nodes are web pages and edges are mutual links between pages. The node features are important keywords in the pages and we classify nodes into five categories in terms of traffic of the webpage.
- MINESWEEPER (Platonov et al., 2023): MINESWEEPER is a synthetic dataset that simulates a Minesweeper game with 100×100 grid. Each node is connected to its neighboring nodes where 20% nodes are selected as mines at random. Node features are numbers of neighboring mines and the goal is to predict whether each test node is mine. These datasets are widely used as benchmarks for studying graph learning methods under heterophilic settings.

For the homophilic group, we consider the following datasets, which are standard homophilic network benchmarks.

- CORA (Sen et al., 2008) : Cora dataset is a citation network, where nodes represent scientific papers in the machine learning field, and edges correspond to citation relationships between these papers. Each node is associated with a set of features that describe the paper, represented as a bag-of-words model. The task for this dataset is to classify each paper into one of seven categories, reflecting the area of research the paper belongs to.
- CITESEER (Sen et al., 2008): CiteSeer dataset is a citation network of scientific papers. It consists of research papers as nodes, with citation links forming the edges between them. Each node is associated with a set of features derived from the paper's content, which is a bag-of-words representation of the paper's text. The task for this dataset is to classify each paper into one of six categories, each representing a specific field of study.

A.2 BASELINE METHODS (CONT'D)

We briefly introduce GNN-based baseline methods as follows.

The first category is *homophilic GNNs*, which are originally designed under the homophily assumption.

- ChebNet (Defferrard et al., 2016): Uses Chebyshev polynomials to approximate graph convolutions.
- GCN (Kipf & Welling, 2016): Employs a first-order Chebyshev approximation for spectral graph convolutions.
- SGC (Wu et al., 2019): Simplifies GCN by removing non-linearities and collapsing weight matrices for efficiency.
- GAT (Veličković et al., 2018): Introduces attention mechanisms to assign adaptive importance to edges.

- GraphSAGE (Hamilton et al., 2017): Uses several aggregators for inductive graph learning.
- GIN (Xu et al., 2018a): Employs sum-based aggregation to maximize graph structure expressiveness.
- APPNP (Gasteiger et al., 2019): Combines personalized PageRank with neural propagation.
- GCNII (Chen et al., 2020): Extends GCN with residual connections and identity mapping for deep GNN training.
- GATv2 (Brody et al., 2021): Enhances GAT with dynamic attention coefficients for flexible neighbor weighting.
- MixHop (Abu-El-Haija et al., 2019): Aggregates multi-hop neighborhood features by mixing different powers of adjacency matrices.
- TAGCN (Du et al., 2017): Introduces trainable polynomial filters for adaptive, multi-scale feature extraction.
- DAGNN (Liu et al., 2020): Uses dual attention to decouple message aggregation and transformation, improving depth scalability.
- JKNet (Xu et al., 2018b): Uses a jumping knowledge mechanism to combine features from different layers adaptively. We default the backbone GNN model to GCN.
- Virtual Node (Gilmer et al., 2017): Introduces an auxiliary global node to facilitate message passing. We default the backbone GNN model to GCN.

The second category is *heterophilic GNN*s, which are designed for graphs where connected nodes often have different labels.

- H2GCN (Zhu et al., 2020): Enhances GNNs by ego-/neighbor-embedding seperation, higher-order neighbors and intermediate representation combinations.
- FSGNN (Maurya et al., 2021): Employs soft feature selection and hop normalization over GNN layers to form a simple, shallow GNN. We use their default 3-hop variant.
- ACM-GNN (Luan et al., 2022): Introduces adaptive channel mixing to diversify local information. We use their default ACM-GCN+ variant.
- FAGCN (Bo et al., 2021): Uses frequency adaptive filtering to learn optimal graph representations.
- OrderedGNN (Song et al., 2023): Aligns the order to encode neighborhood information and avoids feature mixing.
- GloGNN (Li et al., 2022): Incorporates global structural information to enhance graph learning beyond local neighborhoods.
- GGCN (Yan et al., 2022): Utilizes structure/feature-based edge correction to combat oversmoothing and heterophily.
- GPRGNN (Chien et al., 2020): Introduces generalized PageRank propagation to capture the graph structure.
- ALT (Xu et al., 2023): Decomposes graph into components, extracts signals from these components, and adaptively integrate these signals.

The last category is *graph transformers*, which adapt transformer architectures to graph data and look beyond local neighborhood aggregation.

- NodeFormer (Wu et al., 2022): Introduces all-pair message passing on layer-specific adaptive latent graphs, enabling global feature propagation with linear complexity.
- SGFormer (Wu et al., 2024a): Develops a graph encoder backbone that efficiently computes all-pair interactions with one-layer attentive propagation.
- DIFFormer (Wu et al., 2023): Proposes an energy-constrained diffusion model, leading to variants that are efficient and capable of capturing complex structures.

A.3 TRAINING & EVALUATION (CONT'D)

For our method, we use $w_{\mathcal{X}} \in \{0.01, 0.1, 0.6, 8\}$, $w_0 \in \{0.1, 0.2, 0.3, 0.5, 1, 8\}$, $\tau \in \{0.01, 0.1, 1\}$, and dropout rate 0.2. We use the GNN architecture described in the method section with 8 GNN layers with hidden dimensionality 512 and add a two-layer MLP after each GNN layer for heterophilic graphs and use FAGCN for homophilic graphs. We use original node features as described in Section 3.2, except that we use zeros as the features of graph nodes on Squirrel-F and that we normalize the features of graph nodes on Cora and CiteSeer after computing the features of feature nodes. We train the GNN with learning rate 0.00003 for 1000 steps using the Adam optimizer (Kingma & Ba, 2014). Our method was implemented in PyTorch 2.7.0 and Deep Graph Library (DGL) 2.4.0, and experiments were run on Intel Xeon CPU @ 2.20GHz with 96GB memory and NVIDIA Tesla V100 32GB GPU.

B DEFINITION OF HOMOPHILY METRICS

To measure to what extent GRAPHITE can boost graph homophily on heterophlic datasets, we consider two popular homophily metrics: *feature homophily* (Jin et al., 2022) and *adjusted homophily* (Platonov et al., 2024). Formally, given a graph $\mathcal{G}$, *feature homophily* H^{feature} is defined as follows:

$$H^{\text{feature}}(\mathcal{G}) := \frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} \text{sim}(v_i, v_j), \tag{13}$$

where $\text{sim}(v_i, v_j) := \cos(\boldsymbol{X}[v_i, :], \boldsymbol{X}[v_j, :])$ is the cosine-similarity computed between features of nodes v_i, v_j. This metric is a variant of the *generalized edge homophily ratio* H^{edge} proposed by (Jin et al., 2022), which measures the feature similarity between each of the connected node pairs in the graph dataset. Then, the *adjusted homophily* H^{adjusted} is defined as follows:

$$H^{\text{adjusted}}(\mathcal{G}) := \frac{H^{\text{edge}}(\mathcal{G}) - \sum_{c=1}^{C} D_c^2/(2|\mathcal{E}|)^2}{1 - \sum_{c=1}^{C} D_c^2/(2|\mathcal{E}|)^2}, \tag{14}$$

where C denotes the number of classes and $H^{\text{edge}}(\mathcal{G})$ is *edge homophily*, which is defined similarly as Equation (13) with the similarity function $\text{sim}(v_i, v_j) = \mathbf{1}_{\{y_{v_i} = y_{v_j}\}}$, and

$$D_c := \sum_{v \in \mathcal{V}} \deg(v) 1_{[y_v = c]} \tag{15}$$

is the sum of degrees $\deg(v)$ of nodes with label c, where y_v denotes the label of node v. Since we do not have node labels for the *feature nodes* when computing *adjusted homophily*, we assign them a "soft label," which is a uniform probability distribution over the labels of its 1-hop neighbors.

C ADDITIONAL RELATED WORK

In modern machine learning research (Ma et al., 2022; Yu et al., 2026; 2025; Zhang et al., 2026; Bao et al., 2025; Chen et al., 2024; Wei et al., 2026a;b; 2025a;b; 2024; 2022; Cui et al., 2026; Chen et al., 2026; Liu et al., 2025a;b; 2024a;b;c; 2023; Bartan et al., 2025; Zeng et al., 2026a;b;c; 2025a;b; 2024a;b; 2023a;b; Zou et al., 2025a;b; Lin et al., 2026; 2025; 2024; Zhou et al., 2025; Jing et al., 2025; Qiu et al., 2026; 2025a;b;c; 2024; 2023; 2022; Xu et al., 2024b; Li et al., 2025a;b;c;d; 2023b; Yoo et al., 2025a;b; 2024; Chan et al., 2024; Wu et al., 2024b; He et al., 2026; 2024; Wang et al., 2023), a problem related to heterophily is over-squashing. The over-squashing problem in message passing neural networks arises when long-range information is exponentially compressed, preventing effective dissemination across the graph (Alon & Yahav, 2020; Shi et al., 2023b). A primary research direction addresses this issue by identifying topological bottlenecks and modifying graph connectivity. Topping et al. (2021) established an initial framework linking oversquashing to graph Ricci curvature, demonstrating that negatively curved edges act as bottlenecks. Building on this idea, subsequent works have developed rewiring strategies inspired by curvature-based principles (Nguyen et al., 2023; Shi et al., 2023a). Beyond curvature, Black et al. (2023) introduced a perspective using effective resistance. Another line of research leverages spectral methods to counteract over-squashing, with notable approaches including spectral gaps (Arnaiz-Rodríguez et al.,

2022), expander graph constructions (Deac et al., 2022), and first-order spectral rewiring (Karhadkar et al., 2022). More recently, Di Giovanni et al. (2023) provided a comprehensive analysis of the factors contributing to oversquashing. Additional solutions explore advanced rewiring strategies and novel message-passing paradigms (Barbero et al., 2023; Qian et al., 2023; Behrouz & Hashemi, 2024).

D Theoretical Analysis

D.1 Assumptions

In this subsection, we introduce the assumptions of our theoretical analysis, which are mild and realistic.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ with $\mathcal{E} \neq \varnothing$ and $\boldsymbol{X} \in \{0,1\}^{\mathcal{V}\times\mathcal{X}}$, we define the feature similarity metric as $\mathrm{sim}(v_i, v_j) := \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty$ and use the feature homophily as the homophily metric:

$$\mathrm{hom}(\mathcal{G}) := \frac{1}{|\mathcal{E}|} \sum_{(v_i,v_j)\in\mathcal{E}} \mathrm{sim}(v_i, v_j). \tag{16}$$

Furthermore, we assume that the original graph $\mathcal{G}$ is heterophilic. That is, we have $\mathrm{hom}(\mathcal{G}) < 1$ while there exists a pair of nodes, $v_i, v_j \in \mathcal{V}$ ($v_i \neq v_j$), such that $\mathrm{sim}(v_i, v_j) > 0$ but $(v_i, v_j) \notin \mathcal{E}$.

Besides that, we assume that the given graph $\mathcal{G}$ does not have too dense features. Formally, we assume that $|\mathcal{X}| \leq O(|\mathcal{V}|)$ and that $\|\boldsymbol{X}\|_0 \leq O(|\mathcal{E}|)$. For the transformed graph $\mathcal{G}^*$, we assume that every feature is used: for any feature $k \in \mathcal{X}$, there exists a graph node $v_i \in \mathcal{V}$ such that $\boldsymbol{X}[v_i, k] = 1$.

D.2 Technical Lemma

Here, we prove a technical lemma that we will use later.

Lemma 4. *Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}$ be two nonempty, finite multisets such that*

$$\frac{1}{|\mathcal{A}|} \sum_{z\in\mathcal{A}} z < \frac{1}{|\mathcal{B}|} \sum_{z\in\mathcal{B}} z.$$

Then,

$$\frac{1}{|\mathcal{A} \sqcup \mathcal{B}|} \sum_{z\in\mathcal{A}\sqcup\mathcal{B}} z > \frac{1}{|\mathcal{A}|} \sum_{z\in\mathcal{A}} z.$$

Proof. To simplify notation, let

$$\mu_A := \frac{1}{|\mathcal{A}|} \sum_{z\in\mathcal{A}} z, \tag{17}$$

$$\mu_B := \frac{1}{|\mathcal{B}|} \sum_{z\in\mathcal{B}} z, \tag{18}$$

$$\Delta := \mu_\mathcal{B} - \mu_\mathcal{A} > 0. \tag{19}$$

Then,

$$\frac{1}{|\mathcal{A}\sqcup\mathcal{B}|}\sum_{z\in\mathcal{A}\sqcup\mathcal{B}} z - \frac{1}{|\mathcal{A}|}\sum_{z\in\mathcal{A}} z \tag{20}$$

$$= \frac{1}{|\mathcal{A}|+|\mathcal{B}|}\Big(\sum_{z\in\mathcal{A}} z + \sum_{z\in\mathcal{B}} z\Big) - \frac{1}{|\mathcal{A}|}\sum_{z\in\mathcal{A}} z \tag{21}$$

$$= \frac{1}{|\mathcal{A}|+|\mathcal{B}|}\Big(|\mathcal{A}|\cdot\frac{1}{|\mathcal{A}|}\sum_{z\in\mathcal{A}} z + \sum_{z\in\mathcal{B}} z\Big) - \frac{1}{|\mathcal{A}|}\sum_{z\in\mathcal{A}} z \tag{22}$$

$$= \frac{1}{|\mathcal{A}|+|\mathcal{B}|}\Big(|\mathcal{A}|\cdot\mu_{\mathcal{A}} + \sum_{z\in\mathcal{B}} z\Big) - \mu_{\mathcal{A}} \tag{23}$$

$$= \frac{1}{|\mathcal{A}|+|\mathcal{B}|}\big(|\mathcal{A}|\cdot\mu_{\mathcal{A}} + |\mathcal{B}|\cdot\mu_{\mathcal{B}}\big) - \mu_{\mathcal{A}} \tag{24}$$

$$= \frac{1}{|\mathcal{A}|+|\mathcal{B}|}\big(|\mathcal{A}|\cdot\mu_{\mathcal{A}} + |\mathcal{B}|\cdot\mu_{\mathcal{B}} - (|\mathcal{A}|+|\mathcal{B}|)\cdot\mu_{\mathcal{A}}\big) \tag{25}$$

$$= \frac{1}{|\mathcal{A}|+|\mathcal{B}|}\big(|\mathcal{B}|\cdot\mu_{\mathcal{B}} - |\mathcal{B}|\cdot\mu_{\mathcal{A}}\big) \tag{26}$$

$$= \frac{|\mathcal{B}|}{|\mathcal{A}|+|\mathcal{B}|}(\mu_{\mathcal{B}} - \mu_{\mathcal{A}}) \tag{27}$$

$$= \frac{|\mathcal{B}|}{|\mathcal{A}|+|\mathcal{B}|}\Delta > 0. \tag{28}$$

It follows that

$$\frac{1}{|\mathcal{A}\sqcup\mathcal{B}|}\sum_{z\in\mathcal{A}\sqcup\mathcal{B}} z > \frac{1}{|\mathcal{A}|}\sum_{z\in\mathcal{A}} z. \qquad \square$$

D.3 Proof of Theorem 1

Homophily. Since the original graph $\mathcal{G}$ is homophilic, then there exists a pair of nodes, $v_i, v_j \in \mathcal{V}$ ($v_i \neq v_j$), such that $\mathrm{sim}(v_i, v_j) = \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty > 0$ but $(v_i, v_j) \notin \mathcal{E}$. According to the definition of $\mathcal{E}^\dagger$, we know that $(v_i, v_j) \in \mathcal{E}^\dagger \setminus \mathcal{E} \neq \varnothing$, so $\mathcal{E}^\dagger \setminus \mathcal{E} \neq \varnothing$.

Furthermore, for any $(v_i, v_j) \in \mathcal{E}^\dagger \setminus \mathcal{E}$, since $\mathrm{sim}(v_i, v_j) = \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty > 0$, then there exists a feature $k \in \mathcal{X}$ such that $\boldsymbol{X}[v_i, k] \wedge \boldsymbol{X}[v_j, k] > 0$. Since the feature matrix $\boldsymbol{X}$ is binary, then we must have

$$\boldsymbol{X}[v_i, k] = 1, \qquad \boldsymbol{X}[v_j, k] = 1. \tag{29}$$

It follows that

$$\mathrm{sim}(v_i, v_j) = \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty \tag{30}$$

$$= \max_{k'\in\mathcal{X}} |\boldsymbol{X}[v_i, k'] \wedge \boldsymbol{X}[v_j, k']| \tag{31}$$

$$\geq |\boldsymbol{X}[v_i, k] \wedge \boldsymbol{X}[v_j, k]| \tag{32}$$

$$= |1 \wedge 1| = 1. \tag{33}$$

Since $\mathrm{hom}(\mathcal{G}) < 1$, then

$$\mathrm{sim}(v_i, v_j) \geq 1 > \mathrm{hom}(\mathcal{G}). \tag{34}$$

Therefore, by Lemma 4 with

$$\mathcal{A} := \{\!\{\mathrm{sim}(v_i, v_j) : (v_i, v_j) \in \mathcal{E}\}\!\}, \tag{35}$$

$$\mathcal{B} := \{\!\{\mathrm{sim}(v_i, v_j) : (v_i, v_j) \in \mathcal{E}^\dagger \setminus \mathcal{E}\}\!\}, \tag{36}$$

we have

$$\hom(\mathcal{G}^\dagger) = \frac{1}{|\mathcal{E}^\dagger|} \sum_{(v_i,v_j)\in\mathcal{E}^\dagger} \mathrm{sim}(v_i, v_j) \tag{37}$$

$$= \frac{1}{|\mathcal{E} \sqcup (\mathcal{E}^\dagger \setminus \mathcal{E})|} \sum_{(v_i,v_j)\in\mathcal{E}\sqcup(\mathcal{E}^\dagger\setminus\mathcal{E})} \mathrm{sim}(v_i, v_j) \tag{38}$$

$$= \frac{1}{|\mathcal{A} \sqcup \mathcal{B}|} \sum_{z\in\mathcal{A}\sqcup\mathcal{B}} z \tag{39}$$

$$> \frac{1}{|\mathcal{A}|} \sum_{z\in\mathcal{A}} z \tag{40}$$

$$= \frac{1}{|\mathcal{E}|} \sum_{(v_i,v_j)\in\mathcal{E}} \mathrm{sim}(v_i, v_j) \tag{41}$$

$$= \hom(\mathcal{G}). \tag{42}$$

Number of edges. Since there are $|\mathcal{V}|$ nodes in total, then the total number of node pairs is $\binom{|\mathcal{V}|}{2}$. Recall that $\mathcal{E}^\dagger \setminus \mathcal{E}$ is the set of added edges. It follows that

$$|\mathcal{E}^\dagger| - |\mathcal{E}| = |\mathcal{E}^\dagger \setminus \mathcal{E}| \leq \binom{|\mathcal{V}|}{2} \tag{43}$$

$$= \frac{|\mathcal{V}|(|\mathcal{V}|-1)}{2} = O(|\mathcal{V}|^2). \tag{44}$$

D.4 Proof of Observation 2

Since $\mathrm{sim}(v_i, v_j) = \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty > 0$, then there exists a feature $k \in \mathcal{X}$ such that $\boldsymbol{X}[v_i,k] \wedge \boldsymbol{X}[v_j,k] > 0$. Since the feature matrix $\boldsymbol{X}$ is binary, then we must have

$$\boldsymbol{X}[v_i,k] = 1, \qquad \boldsymbol{X}[v_j,k] = 1. \tag{45}$$

This implies that $(v_i, x_k) \in \mathcal{E}^*$ and that $(v_j, x_k) \in \mathcal{E}^*$. Hence, there exists a length-2 path $v_i \to x_k \to v_j$ connecting graph nodes v_i and v_j. Therefore, v_i and v_j are two-hop neighbors of each other.

D.5 Proof of Theorem 3

Homophily. Since the original graph $\mathcal{G}$ is homophilic, then there exists a pair of nodes, $v_i, v_j \in \mathcal{V}$ ($v_i \neq v_j$), such that $\mathrm{sim}(v_i, v_j) = \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty > 0$ but $(v_i, v_j) \notin \mathcal{E}$. Since $\mathrm{sim}(v_i, v_j) = \|\boldsymbol{X}[v_i,:] \wedge \boldsymbol{X}[v_j,:]\|_\infty > 0$, then there exists a feature $k \in \mathcal{X}$ such that $\boldsymbol{X}[v_i,k] \wedge \boldsymbol{X}[v_j,k] > 0$. Since the feature matrix $\boldsymbol{X}$ is binary, then we must have

$$\boldsymbol{X}[v_i,k] = 1, \qquad \boldsymbol{X}[v_j,k] = 1. \tag{46}$$

This implies that $(v_i, x_k) \in \mathcal{E}^* \setminus \mathcal{E}$ and that $(v_j, x_k) \in \mathcal{E}^* \setminus \mathcal{E}$. Thus, $\mathcal{E}^* \setminus \mathcal{E}$ is nonempty.

Furthermore, for any feature node $x_k \in \mathcal{V}_\mathcal{X}$, since any feature edge $(v_i, x_k) \in \mathcal{E}_\mathcal{X}$ ensures $\boldsymbol{X}[v_i,k] = 1$, then we have

$$\boldsymbol{X}^*[x_k,k] = \frac{1}{|\mathcal{E}_\mathcal{X} \cap (\mathcal{V} \times \{x_k\})|} \sum_{v_i:(v_i,x_k)\in\mathcal{E}_\mathcal{X}} \boldsymbol{X}[v_i,k] \tag{47}$$

$$= \frac{1}{|\mathcal{E}_\mathcal{X} \cap (\mathcal{V} \times \{x_k\})|} \sum_{v_i:(v_i,x_k)\in\mathcal{E}} 1 \tag{48}$$

$$= \frac{1}{|\mathcal{E}_\mathcal{X} \cap (\mathcal{V} \times \{x_k\})|} \sum_{v_i:(v_i,x_k)\in\mathcal{E}\cap(\mathcal{V}\times\{x_k\})} 1 \tag{49}$$

$$= 1. \tag{50}$$

Finally, for any added feature edge $(v_i, x_k) \in \mathcal{E}^* \setminus \mathcal{E} = \mathcal{E}_{\mathcal{X}}$,

$$\begin{aligned}
\mathrm{sim}(v_i, x_k) &= \|\boldsymbol{X}[v_i, :] \wedge \boldsymbol{X}[x_k, :]\|_\infty && (51)\\
&= \max_{k' \in \mathcal{X}} |\boldsymbol{X}[v_i, k'] \wedge \boldsymbol{X}[x_k, k']| && (52)\\
&\geq |\boldsymbol{X}[v_i, k] \wedge \boldsymbol{X}[x_k, k]| && (53)\\
&= |1 \wedge 1| = 1. && (54)
\end{aligned}$$

Since $\hom(\mathcal{G}) < 1$, then

$$\mathrm{sim}(v_i, x_k) \geq 1 > \hom(\mathcal{G}). \tag{55}$$

Therefore, by Lemma 4 with

$$\begin{aligned}
\mathcal{A} &:= \{\!\{\mathrm{sim}(v_i, v_j) : (v_i, v_j) \in \mathcal{E}\}\!\}, && (56)\\
\mathcal{B} &:= \{\!\{\mathrm{sim}(v_i, x_k) : (v_i, x_k) \in \mathcal{E}_{\mathcal{X}}\}\!\}, && (57)
\end{aligned}$$

we have

$$\begin{aligned}
\hom(\mathcal{G}^*) &= \frac{1}{|\mathcal{E}^*|} \sum_{(u,u') \in \mathcal{E}^*} \mathrm{sim}(u, u') && (58)\\
&= \frac{1}{|\mathcal{E} \sqcup \mathcal{E}_{\mathcal{X}}|} \sum_{(u,u') \in \mathcal{E} \sqcup \mathcal{E}_{\mathcal{X}}} \mathrm{sim}(u, u') && (59)\\
&= \frac{1}{|\mathcal{A} \sqcup \mathcal{B}|} \sum_{z \in \mathcal{A} \sqcup \mathcal{B}} z && (60)\\
&> \frac{1}{|\mathcal{A}|} \sum_{z \in \mathcal{A}} z && (61)\\
&= \frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} \mathrm{sim}(v_i, v_j) && (62)\\
&= \hom(\mathcal{G}). && (63)
\end{aligned}$$

Number of nodes. Since $|\mathcal{X}| \leq O(|\mathcal{V}|)$, then

$$|\mathcal{V}_{\mathcal{X}}| = |\mathcal{X}| \leq O(\mathcal{V}). \tag{64}$$

It follows that

$$\begin{aligned}
|\mathcal{V}^*| &= |\mathcal{V}| + |\mathcal{V}_{\mathcal{X}}| && (65)\\
&\leq |\mathcal{V}| + O(|\mathcal{V}|) && (66)\\
&= O(|\mathcal{V}|). && (67)
\end{aligned}$$

Number of edges. Since $\boldsymbol{X}$ is a binary matrix, then $\|\boldsymbol{X}\|_1 = \|\boldsymbol{X}\|_0 \leq O(|\mathcal{E}|)$. Hence,

$$\begin{aligned}
|\mathcal{E}_{\mathcal{X}}| &= \sum_{v_i \in \mathcal{V}} \sum_{x_k \in \mathcal{V}_{\mathcal{X}}} 1_{[(v_i, x_k) \in \mathcal{E}_{\mathcal{X}}]} && (68)\\
&= \sum_{v_i \in \mathcal{V}} \sum_{k \in \mathcal{X}} 1_{[(v_i, x_k) \in \mathcal{E}_{\mathcal{X}}]} && (69)\\
&= \sum_{v_i \in \mathcal{V}} \sum_{k \in \mathcal{X}} 1_{[\boldsymbol{X}[v_i, k] = 1]} && (70)\\
&= \sum_{v_i \in \mathcal{V}} \sum_{k \in \mathcal{X}} \boldsymbol{X}[v_i, k] && (71)\\
&= \sum_{v_i \in \mathcal{V}} \sum_{k \in \mathcal{X}} |\boldsymbol{X}[v_i, k]| && (72)\\
&= \|\boldsymbol{X}\|_1 = \|\boldsymbol{X}\|_0 \leq O(|\mathcal{E}|). && (73)
\end{aligned}$$

It follows that

$$\begin{aligned}
|\mathcal{E}^*| &= |\mathcal{E}| + |\mathcal{E}_{\mathcal{X}}| && (74)\\
&\leq |\mathcal{E}| + O(|\mathcal{E}|) && (75)\\
&= O(|\mathcal{E}|). && (76)
\end{aligned}$$

E USE OF LARGE LANGUAGE MODELS

We made limited and controlled use of large language models (LLMs) solely for stylistic refinement and improving readability of the text. All scientific content, methodology, experiments, and conclusions were fully conceived and validated by the authors. The role of LLMs was purely editorial and does not constitute co-authorship.