Finding the Missing-half: Graph Complementary Learning for Homophily-prone and Heterophily-prone Graphs

Yizhen Zheng [1] **He Zhang** [1] **Vincent CS Lee** [1] **Yu Zheng** [2] **Xiao Wang** [3] **Shirui Pan** [4]

Abstract

Real-world graphs generally have only one kind of tendency in their connections. These connections are either homophily-prone or heterophily-prone. While graphs with homophily-prone edges tend to connect nodes with the same class (i.e., intra-class nodes), heterophily-prone edges tend to build relationships between nodes with different classes (i.e., inter-class nodes). Existing GNNs only take the original graph during training. The problem with this approach is that it forgets to take into consideration the "missing-half" structural information, that is, heterophily-prone topology for homophily-prone graphs and homophily-prone topology for heterophily-prone graphs. In our paper, we introduce ***G**raph c**O**mplement**A**ry **L**earning*, namely GOAL, which consists of two components: graph complementation and complemented graph convolution. The first component finds the missing-half structural information for a given graph to complement it. The complemented graph has two sets of graphs including both homophily- and heterophily-prone topology. In the latter component, to handle complemented graphs, we design a new graph convolution from the perspective of optimisation. The experiment results show that GOAL consistently outperforms all baselines in eight real-world datasets.

1. Introduction

Graph Neural Networks (GNNs) have gained much attention in recent years due to their promising performance on various downstream tasks, such as node clustering (Liu et al., 2022a;b), link prediction (Luo et al., 2023a), and node classification (Xiong et al., 2022; Liu et al., 2023b; Zhu et al., 2022; Zheng et al., 2022b;c). In addition, they can be applied to many real-world applications, e.g., anomaly detection (Liu et al., 2023a; Zheng et al., 2021), neural architecture search (Zheng et al., 2023; 2022a), knowledge graphs (Luo et al., 2023b), time series forecasting (Jin et al., 2022b;a) and system security (Zhang et al., 2022b; 2021; 2023; 2022a; Tan et al., 2023). Despite the success of existing GNNs, they overlook the "missing-half" structural information of the given graph, that is, heterophily-prone edges for homophily-prone graphs and homophily-prone edges for heterophily-prone graphs. This is because they only use the given real-world graph during training, which normally has only one type of connection: homophily-prone or heterophily-prone. Figure 1(a) and (b) show examples of a homophily-prone graph and a heterophily-prone graph, respectively. In particular, edges in a homophily-prone graph tend to connect intra-class nodes, whereas edges in a heterophily-prone graph tend to connect inter-class nodes.

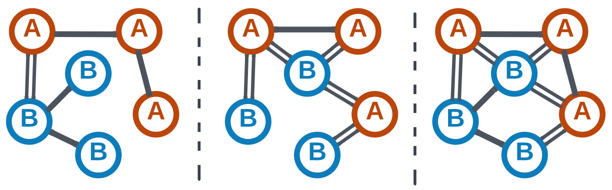

Figure 1. Example illustration of a homophily-prone graph, a heterophily-prone graph and a complemented graph. The single lines represent intra-class edges, while the doubled lines represent inter-class edges. There are two classes of nodes in these graphs, which are A class (red nodes) and B class (blue nodes).

Adding the missing-half topology to a graph can enrich its structural information abundantly. With this information, we can intuitively know for each node both its intra-class and inter-class counterparts. We can then assign each node to be closer to their intra-class peers during training, and the reverse for their inter-class peers. Thus, a natural question is "Can we find the missing-half structural information to complement real-world graphs and utilise them?"

To overcome this problem, we introduce ***Graph***

[1]Monash University [2]La Trobe University [3]Beihang University [4]Griffith University. Correspondence to: Shirui Pan <s.pan@griffith.edu.au>, Yizhen Zheng <yizhen.zheng1@monash.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

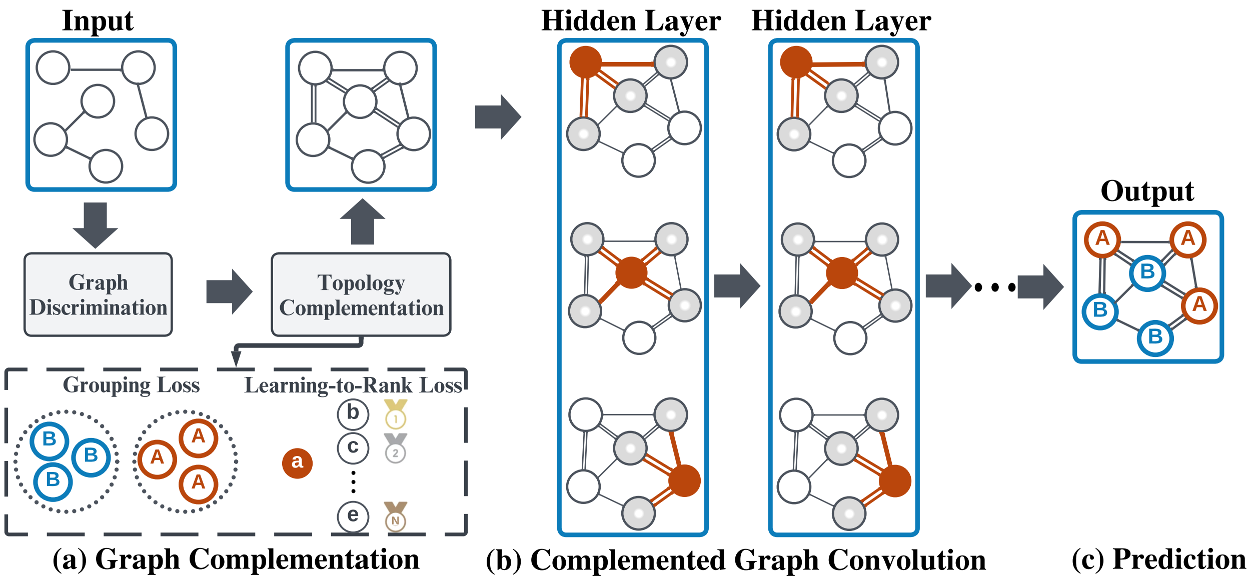

Figure 2. The overall architecture of `GOAL`. In graph complementation (a), we first perform graph discrimination to determine which part of structural information is missing from the input graph. Then, we conduct topology complementation to find and complement the input graph with another set of graph topology. This process utilises two losses including grouping loss and learning-to-rank loss. While the first loss aims to cluster intra-class nodes, the latter one attempts to predict the correct ranking of nodes in a ranking list for a target node (e.g., node a in the figure). In complemented graph convolution, we conduct a newly-designed graph convolution to utilise both heterophily- and homophily-prone connections. During convolution, each node interacts with both its homophily- and heterophily-connected neighbours. The edges involved in the convolution are red-coloured and bold. Here, the single and the doubled lines indicate two different types of connections. Grey neighbouring nodes with a white centre point are involved in the convolution process of the red-coloured target node. Finally, we obtain the output prediction for each node, e.g., A or B class.

*c**O**mplement**A**ry **L**earning*, namely `GOAL`, which can complement the given graph with the "missing-half" topology and utilise the complemented graph for model training. Figure 1(c) presents an example of the complemented graph that have two set of edges including both homophily- and heterophily-prone linkages. In order to conduct Graph Complementary Learning, there are two main challenges: **(1)** *How to design a model to complement the given graph?* and **(2)** *How to design a new graph convolution to handle complemented graphs with both homophily- and heterophily-prone topology?*

We address the first challenge by proposing a two-step pipeline, which includes graph discrimination and topology complementation. Graph discrimination uses Kolmogorov-Smirnov statistics (Lilliefors, 1967) to determine which kind of connection is missing from a graph. Then, we complement a graph using a model incorporating both grouping and learning-to-rank losses. For the second challenge, we design a proper graph convolution for complemented graphs from the perspective of optimisation. The newly-designed graph convolution can extract valuable information from both homophily- and heterophily-prone topology.

The overall architecture of Graph Complementary Learning is presented in Figure 2. It consists of two main components: graph complementation and complemented graph convolution. While the first component finds the missing-half topology for a graph, the latter one handles the complemented graph having two sets of topology. These two components are detailed in Sections 4 and 5.

The contributions of our work are threefold:

- To the best of our knowledge, we are the first to introduce Graph Complementary Learning, which finds the "missing-half" topology to complement graphs and utilise them.
- We propose a graph convolution strategy that can handle graphs with both homophily- and heterophily-prone topology.
- Our proposed `GOAL` outperforms baselines on eight real-world datasets, which validates its superiority.

2. Related Work

Graph neural networks (GNNs) can be categorised into spectral-based GNNs and spatial-based GNNs. In the spectral domain, spectral GNNs design spectral graph filters based on spectral graph theory. Bruna et al. (2014) first defines graph convolution filters as functions of eigenvalues of the graph Laplacian matrix. In ChebyNet (Defferrard et al., 2016), a spectral filter function is approximated by the Chebyshev polynomial. GPR-GNN (Chien et al., 2021) approximates graph convolutions using the Monomial basis, which can derive either low or high-pass filters. Bridging spectral- and spatial-based GNNs, GCN (Kipf & Welling, 2017) simplifies ChebyNet (Defferrard et al., 2016) with the first-order approximation and can be interpreted from the spatial perspective.

In addition, spatial-based methods have gained popularity in recent years due to their efficiency, generality, and flexibility (Wu et al., 2020). Their basic idea is that graph signals can be propagated with graph structures iteratively, which broadens a node neighbourhood. For example, GCN (Kipf & Welling, 2017) can be interpreted as aggregating nearest-neighbouring graph signals to generate updated node embeddings. SGC (Wu et al., 2019) simplifies GCN by decoupling message passing and feature propagation. In addition, it removes the non-linearity in GCN. GAT (Veličković et al., 2017) assigns learnable weights in the message-passing process. APPNP (Klicpera et al., 2019b) and GDC (Klicpera et al., 2019b) further enrich the signal propagation with personalised PageRank and graph diffusion.

Results from the above approaches have been promising. However, they directly use the original graph as input, neglecting the "missing-half" topology. To complement the missing type of connection for the original graph, we introduce Graph Complementary Learning, namely `GOAL`, which can complement graphs with "missing-half" topology and process these complemented graphs.

3. Preliminaries

Notations. Let $\mathcal{G} = \{\mathbf{X} \in \mathbb{R}^{N\times D}, \mathbf{A} \in \mathbb{R}^{N\times N}\}$ denote an undirected and attributed graph, where $\mathbf{A}$ is an adjacency matrix with N nodes and $\mathbf{X}$ is the feature matrix with dimension D. $\mathbf{A}$ encapsulates the structural information of $\mathcal{G}$, in which a non-zero entry indicates there is a connection between two nodes (e.g., node i and j if $\mathbf{A}_{i,j} = 1$). $(i, j) \in \mathcal{E}$ denotes a edge connecting node i and node j in the set of all edges $\mathcal{E}$. The number of edges in $\mathcal{E}$ is E. A diagonal degree matrix is defined as $\mathbf{D}$, whose diagonal has values of $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$. The normalised adjacency matrix is $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. The normalised Laplacian matrix $\hat{\mathbf{L}} = \mathbf{I} - \hat{\mathbf{A}}$ is a symmetric and positive semi-definite matrix whose eigenvalues λ are in range $[0, 2]$ (Chien et al., 2021).

Problem Formulation. The focus of this study is node classification tasks, which predict the class label for each node. We aim to learn a mapping $f_\theta(\mathbf{X}, \mathbf{A}) \to \mathbf{Y} \in \mathbb{R}^{N\times C}$, where $f_\theta(\cdot)$ is a learnable function to generate a prediction matrix $\mathbf{Y}$ and C is the number of classes. The class with the highest value in each row of $\mathbf{Y}$ will be considered as the predicted class nodes.

4. Graph Complementation

As shown in Figure 2 (a), this section presents the graph complementation module to answer "*How to design a model to complement the given graph?*" This module includes graph discrimination and topology complementation.

4.1. Graph Discrimination

Graph discrimination is used to determine what kind of structural information is missing from a graph. As an example, the homophily-prone Cora graph lacks the set of heterophily-prone edges. Thus, it needs to be complemented with this type of connection.

Experiment Setup. To distinguish homophily- and heterophily-prone datasets, we conducted an experiment to compare the cosine similarity distribution (CSD) between connected node pairs and randomly sampled node pairs. The cosine similarity between the nodes in a node pair (v_i, v_j) can be calculated as follows:

$$cos(v_i, v_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\parallel \mathbf{x}_i \parallel \parallel \mathbf{x}_j \parallel}, \tag{1}$$

where $cos(\cdot)$ takes the features of two nodes as input and outputs the cosine similarity between the two nodes. $\mathbf{x}_i \in \mathbb{R}^{1\times D}$ is the raw feature of node i, and $\parallel \mathbf{x}_i \parallel$ means the norm of $\mathbf{x}_i$.

The cosine similarity distribution (CSD) of connected node pairs is denoted as $\mathbf{d}_1$ (i.e., $\{cos(v_i, v_j)|(v_i, v_j) \in \mathcal{E}_\mathcal{G}\}$), where $\mathcal{E}_\mathcal{G}$ is the set of edges in graph $\mathcal{G}$. The CSD of randomly sampled node pairs $\mathbf{d}_2 = \{cos(v_i, v_j)|(v_i, v_j) \in \mathcal{E}_\mathcal{S}\}$, where $\mathcal{E}_\mathcal{S}$ are uniformly sampled between any two nodes and its size is the same as $\mathbf{d}_1$, i.e., $|\mathcal{E}_\mathcal{G}| = |\mathcal{E}_\mathcal{S}|$.

Observations. As shown in Figure 3, we compare the CSDs between connected and randomly sampled node pairs on eight datasets. Our key observations are as follows.

(1) *There is an obvious CSD difference between the connected and random node pairs in homophily-prone graphs.* Based on the visualisations in Figure 3 (e.g., Cora dataset), we observe that the CSDs of connected node pairs in homophily-prone graphs differ greatly from those of randomly sampled node pairs. Specifically, CSDs of connected nodes are normally right-shifted and flatter than those of sampled node pairs in homophily-prone graphs. The right-shiftiness is reasonable because homophily-prone linkages tend to connect intra-class nodes that have a higher similarity. In addition, homophily-prone graphs exclude many inter-class node pairs with low similarity, alleviating the positive skewness (i.e., right long tail) and causing the dis-

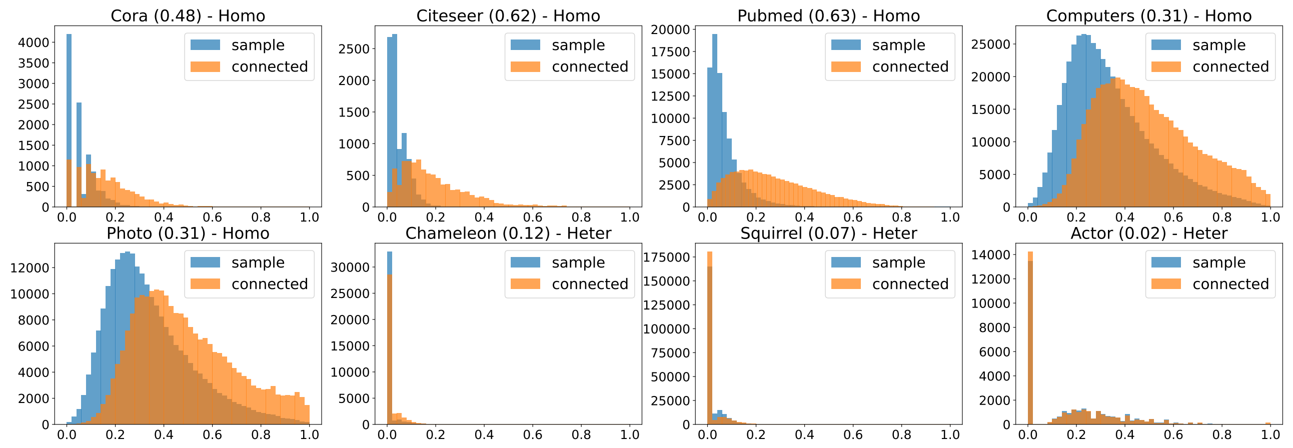

Figure 3. Here is a comparison of the CSDs between connected and randomly sampled node pairs. The number in the bracket after each dataset is the K-S statistics of the two distributions. "Homo" and "Heter" represent homophily- and heterophily-prone graphs, respectively.

tribution to be flatter.

(2) *heterophily-prone graphs have similar CSDs on connected and random node pairs.* As shown in Figure 3 (e.g., Chameleon dataset), the CSDs of the connected node pairs in heterophily-prone graphs have similar distributions to randomly sampled node pairs. The reason for this, we suggest, is that randomly sampled pairs of nodes tend to have mainly heterophilic relationships. As shown in Figure 3, the CSDs of randomly sampled pairs across all datasets are heavily positive-skewed, which means most sampled pairs have very low cosine similarity and are likely heterophilic.

Method. Based on the above observations, we propose a method to distinguish homophily- and heterophily-prone datasets based on the CSD difference between connected node pairs and randomly sampled node pairs. Specifically, we use Kolmogorov-Smirnov statistics (K-S statistics) (Lilliefors, 1967) to evaluate the difference in distribution between connected and random node pairs. A desirable property of K-S statistics is that it ranges from 0 to 1, which allows fair comparison between different datasets. Based on this measure, we set a threshold to determine which kind of connection is likely missing. K-S statistics can be calculated with the following formula:

$$KS(\mathbf{d}_1, \mathbf{d}_2) = max \parallel F(\mathbf{d}_1) - F(\mathbf{d}_2) \parallel, \qquad (2)$$

where $KS(\cdot)$ takes two distributions as input and output K-S statistics. $F(\cdot)$ is a function to calculate the cumulative distribution function (CDF) of the given distribution. $F(\mathbf{d}_1)$ is the sample CDF and $F(\mathbf{d}_2)$ is the reference CDF. In our case, $\mathbf{d}_1$ and $\mathbf{d}_2$ are the CSDs of connected nodes and randomly sampled nodes, respectively.

We choose 0.2 as the threshold for K-S statistics. If the K-S statistics of a graph exceed 0.2, it will be considered homophily-prone; otherwise, it will be heterophily-prone. We choose 0.2 as it indicates a clear/distinguishable difference between two distributions as suggested in (da Silva et al., 2021). For each dataset, we randomly sampled ten sets of random node pairs and calculate the K-S statistics between the CSDs of these sets and connected nodes. The average results are shown in Figure 3 (in brackets after each dataset name). From the figure, we conclude that this approach successfully distinguishes all graphs (i.e., Cora, Citeseer, Pubmed, Computers and Photo are homophily-prone / Chameleon, Squirrel, and Actor are heterophily-prone) in terms of their tendency in connections.

4.2. Topology Complementation

In topology complementation, we aim to find the missing-half structural information and complement the given graph with additional structural information. To achieve this, we train a model with two losses: grouping and learning-to-rank loss. The overall loss can be formulated as follows:

$$\mathcal{L} = \mathcal{L}_{grp} + \mathcal{L}_{rank}, \qquad (3)$$

where $\mathcal{L}_{grp}$ is the grouping loss and $\mathcal{L}_{rank}$ is the learning-to-rank loss.

Grouping Loss. Grouping loss can help a model group intra-class nodes together while separating inter-class nodes. To do this, we first obtain node embeddings with a backbone GNN encoder, e.g., GAT (Veličković et al., 2017). Then, node embeddings are input to a (Multi-layer Perceptron) MLP model. Afterwards, the MLP model needs to increase the similarity between intra-class node pairs, while doing the opposite for inter-class node pairs by optimising the grouping loss. Specifically, we first obtain all intra-class node pairs as the positive set $\mathbb{P} = \{(v_i, v_j) | Y_{v_i} = Y_{v_j}\}$, while all inter-class node pairs as the negative set $\mathbb{N} = \{(v_i, v_j) | Y_{v_i} \neq Y_{v_j}\}$. Here, v_i and Y_{v_i} represent node i and

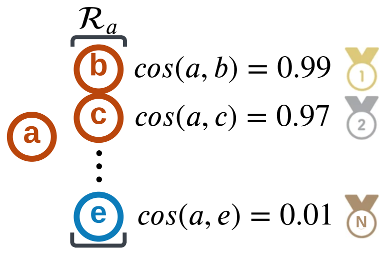

Figure 4. An example of ranking list $\mathcal{R}_a$ for node a. Red and blue coloured nodes indicate two different classes of nodes.

its label, respectively. The whole process can be formulated as follows:

$$\mathbf{Z}_{\text{gc}} = \text{MLP}(\mathbf{Z}_{\text{gnn}});\ \mathbf{Z}_{\text{gnn}} = \text{GNN}(\mathcal{G}), \tag{4}$$

$$\mathcal{L}_{pos} = -log(sig(\frac{\sum_{(i,j)\in\mathbb{P}} \mathbf{Z}_{gc}^T[v_i]\cdot \mathbf{Z}_{gc}[v_j]}{|\mathbb{P}|}) + \epsilon),$$
$$\mathcal{L}_{neg} = -log(1 - sig(\frac{\sum_{(i,j)\in\mathbb{N}} \mathbf{Z}_{gc}^T[v_i]\cdot \mathbf{Z}_{gc}[v_j]}{|\mathbb{N}|} + \epsilon)), \tag{5}$$

$$\mathcal{L}_{grp} = \mathcal{L}_{pos} + \mathcal{L}_{neg}, \tag{6}$$

where $\cdot$ represents matrix multiplication, $\mathbf{Z}_{gnn}$ is the output embedding of the GNN encoder taking $\mathcal{G}$ as input, $\mathbf{Z}_{gc}$ is the output embedding of the MLP model, $sig(\cdot)$ is the sigmoid function, ϵ is a tiny number to prevent computation error. Note that $\mathbf{Z}_{gnn}$ is obtained by the pretrained GNN encoder with cross-entropy loss. We take the embedding before the prediction layer as $\mathbf{Z}_{gnn}$.

Learning-to-Rank Loss. In this paper, Learning-to-Rank aims to predict the ranking of other nodes based on a given node and their cosine similarity. Concretely, for a target node, we consider intra-class nodes with higher cosine similarity having higher ranks, while inter-class with lower cosine similarity have lower ranks. Each node will have its corresponding ranking list and an example is illustrated in Figure 4. With this loss, the model is expected to predict the correct ranking of nodes in the ranking list for each target node.

We adopt ListNet loss (Cao et al., 2007), a listwise learning-to-rank approach, to conduct the learning-to-rank task. It can optimise the ranking of a list of items by minimising the cross entropy between the predicted and true ranking of items in a ranking list. To build the ranking list $\mathcal{R}_i$ for a target node i, we first sort its intra-class nodes in the training set based on cosine similarity and obtain $L_i^{intra} = \{v_j^1, ...v_j^Q | Y_{v_j^q} = Y_{v_i}; cos(v_i, v_j^q) > cos(v_i, v_j^{q+1})\}$. Here, $1 \leq q \leq Q$ and Q is the number of intra-class nodes for node i. $cos(\cdot)$ is the cosine similarity function. Similarly, we can obtain $L_i^{inter} = \{v_j^1, ...v_j^E | Y_{v_j^e} \neq Y_{v_i}; cos(v_i, v_j^e) > cos(v_i, v_j^{e+1})\}$, where $1 \leq e \leq E$ and E is the number of inter-class nodes. Finally, we can obtain the ranking list $\mathcal{R}_i = \{L_i^{intra}[1:K] \parallel L_i^{inter}[-K:-1]\}$, which includes top K most similar intra-class and least similar inter-class nodes in the training set for node i. $\parallel$ means concatenation. The position of each node in $\mathcal{R}_i$ determines its true ranking. The ListNet loss can be formulated as follows:

$$\hat{\mathbf{y}}_{v_i} = \mathbf{Z}_{gc}[i]^T \cdot \mathbf{Z}_{gc}[\mathcal{R}_i],$$
$$\mathcal{L}_{rank} = -\sum_{i=1}^{B}(p(\mathbf{y}_{v_i})log(p(\hat{\mathbf{y}}_{v_i}))), \tag{7}$$

where B represents the size of the training set, $p(\cdot)$ is the softmax function, $\hat{\mathbf{y}}_{v_i}$ indicates the predicted score list for the nodes in ranking list $\mathcal{R}_i$, and $\mathbf{y}_{v_i}$ denotes the ground truth obtained from $\mathcal{R}_i$ (Cao et al., 2007).

Topology Complementation. After the topology complementation module is well trained by minimising $\mathcal{L} = \mathcal{L}_{grp} + \mathcal{L}_{rank}$, it can generate an additional topology for a graph based on its graph discrimination result in Section 4.1. For graphs discriminated as homophily-prone graphs, additional heterophily-prone edges are built by finding the K_{heter} least similar nodes for each node. Specifically, we compute the dot product similarity between all node pairs via $\mathbf{Z}_{gc} \cdot \mathbf{Z}_{gc}^T$. Then, we find those top K_{heter} least similar nodes of each node and connect them to it. In contrast, for graphs discriminated as heterophily-prone graphs, we find the K_{homo} most similar nodes for each node to build the homophily-prone topology.

5. Complemented Graph Convolution

Given the complemented graphs, this section aims to answer "*how to design a new graph convolution to handle complemented graphs with both homophily- and heterophily-prone topology?*" In this section, we first present the convolution designed for complemented graphs, followed by how to derive it from our optimisation goal, which is proposed for simultaneously digging homophily- and heterophily-prone topology information.

5.1. Complemented Graph Convolution Solution

In this paper, we propose **C**omplemented **G**raph **C**onvolution (CGC) from the perspective of optimisation. Specifically, the CGC operation can be formulated as follows:

$$\mathbf{H}^{l+1} = \sigma((\alpha\mathbf{I} + \beta\hat{\mathbf{A}}_o - \gamma\hat{\mathbf{A}}_t - \delta\hat{\mathbf{A}}_{to})\mathbf{H}^l\mathbf{W}^l), \tag{8}$$

where $\mathbf{H}^l \in \mathbb{R}^{N\times D'}$ is node representations at l-th layer ($\mathbf{H}^{(0)} = \mathbf{X}$), $\mathbf{W}^l \in \mathbb{R}^{D'\times D'}$ is a weight matrix at l-th layer ($\mathbf{W}^0 \in \mathbb{R}^{D\times D'}$), $\hat{\mathbf{A}}_o$ represents the normalised homophily-half of graph, $\hat{\mathbf{A}}_t$ represents the normalised heterophily-half of graph, and $\hat{\mathbf{A}}_{to} = \hat{\mathbf{A}}_t\hat{\mathbf{A}}_o$ is the combined topology. α, β, γ, and δ are four parameters that determine the strength of self, homophily-prone, heterophily-prone and combined information, respectively. The combined information involves both homophily- and heterophily-prone topology. These parameters are ranged from $[0, 5]$.

5.2. CGC: Interpretation and Derivation from Optimisation Perspectives

To take full advantage of both homophily- and heterophily-prone structural information, we propose to *maximise the similarity between connected nodes from homophily-prone edges while doing the opposite with heterophily-prone edges*. Next, we first present the optimisation objective which provides an interpretation of the underlying mechanism and then show how we derive our CGC from it. An interpretation of our CGC from the graph signal processing perspective is also given in Appendix B.

According to a recent study on interpreting and unifying GNNs (Zhu et al., 2021), existing various GNN architectures (e.g., GCN) can be derived from a framework with the optimisation perspective. In this paper, we follow the same pipeline and propose the following optimisation objective to conduct learning on complement graphs, i.e.,

$$\mathcal{O}_{\text{CGC}} = \min_{\mathbf{H}}\{tr(\mathbf{H}^T(3\mathbf{I} - \hat{\mathbf{A}}_o + \hat{\mathbf{A}}_t - \hat{\mathbf{A}}_t\hat{\mathbf{A}}_o)\mathbf{H})\}, \tag{9}$$

where $tr(\cdot)$ indicates the trace of a matrix. As shown in the following, this objective can be regarded as a combination of three objectives $\mathcal{O}_o, \mathcal{O}_t$, and $\mathcal{O}_c$, which respectively utilise homophily- ($\hat{\mathbf{A}}_o$), heterophily-prone ($\hat{\mathbf{A}}_t$) and combined topology ($\hat{\mathbf{A}}_{to}$) to implement our intuition. Concretely,

$$\begin{aligned}
&\mathcal{O}_{\text{CGC}} = \mathcal{O}_o + \mathcal{O}_t + \mathcal{O}_c,\\
&\quad\mathcal{O}_o = \min_{\mathbf{H}}\{tr(\mathbf{H}^T(\mathbf{I} - \hat{\mathbf{A}}_o)\mathbf{H})\}\\
&\quad\quad = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}^{E} \hat{\mathbf{A}}_o[i,j] \parallel \mathbf{H}_i - \mathbf{H}_j \parallel^2,\\
&\quad\mathcal{O}_t = \min_{\mathbf{H}}\{tr(\mathbf{H}^T(\mathbf{I} + \hat{\mathbf{A}}_t)\mathbf{H})\}\\
&\quad\quad = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}^{E} \hat{\mathbf{A}}_t[i,j] \parallel \mathbf{H}_i + \mathbf{H}_j \parallel^2,\\
&\quad\mathcal{O}_c = \min_{\mathbf{H}}\{tr(\mathbf{H}^T(\mathbf{I} + \hat{\mathbf{A}}_{to})\mathbf{H})\}; \hat{\mathbf{A}}_{to} = \hat{\mathbf{A}}_t\hat{\mathbf{A}}_o\\
&\quad\quad = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}^{E} \hat{\mathbf{A}}_{to}[i,j] \parallel \mathbf{H}_i + \tilde{\mathbf{H}}_j \parallel^2.
\end{aligned} \tag{10}$$

In our optimisation objective $\mathcal{O}_{\text{CGC}}$, $\mathcal{O}_o$ aims to maximise the similarity between connected nodes from the homophilic view, while $\mathcal{O}_t$ and $\mathcal{O}_c$ are devised to minimise that from the heterophilic perspective. Next, we will show how to convert these objectives to convolution operations, respectively.

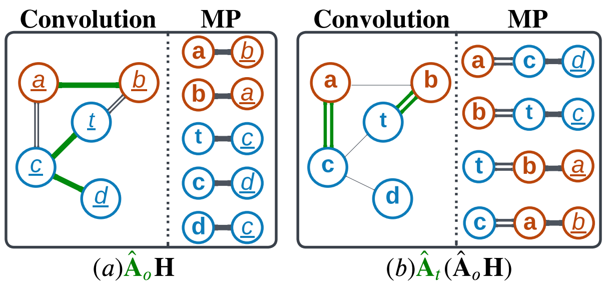
Convolution
MP
Convolution
MP
$(a)\hat{\mathbf{A}}_o\mathbf{H}$
$(b)\hat{\mathbf{A}}_t(\hat{\mathbf{A}}_o\mathbf{H})$

Figure 5. Illustration of $\hat{\mathbf{A}}_{to}\mathbf{H}$. "MP" means message passing chains. The left-hand side of each subfigure presents the convolution process, while the right-hand side shows message-passing chains. Edges involved in the convolution are coloured in green and bold. The underlined italic letter represents the initial node embedding. After convolution in (a), the convoluted nodes are represented with bold roman letters. The straight lines are homophily-prone connections, while the doubled lines are the opposite.

- $\mathcal{O}_o$ aims to minimise the Euclidean distance (ED) between target nodes and their homophily-prone neighbours, and it can be solved with $\frac{\partial tr(\mathbf{H}^T(\mathbf{I}-\hat{\mathbf{A}}_o)\mathbf{H})}{\partial \mathbf{H}} = 0$ as the objective is a convex function. This is because $\hat{\mathbf{L}}_o = \mathbf{I} - \hat{\mathbf{A}}_o$ is a positive semi-definite matrix as $\hat{\mathbf{L}}_o$ is the laplacian matrix of $\hat{\mathbf{A}}_o$. Therefore, we can regard $\mathbf{H}^T\hat{\mathbf{L}}_o\mathbf{H}$ as the energy of the system $\hat{\mathbf{L}}_o$ (Strang, 2006). As a result, we can derive:

$$\hat{\mathbf{L}}_o\mathbf{H} = 0 \rightarrow \mathbf{H} - \hat{\mathbf{A}}_o\mathbf{H} = 0 \rightarrow \mathbf{H} = \hat{\mathbf{A}}_o\mathbf{H}. \tag{11}$$

  This result is also validated in (Zhu et al., 2021).

- $\mathcal{O}_t$ is proposed to maximise the dissimilarity between nodes and their heterophily-prone neighbours. We can also solve it via $\frac{\partial tr(\mathbf{H}^T(\mathbf{I}+\hat{\mathbf{A}}_t)\mathbf{H})}{\partial \mathbf{H}} = 0$. This is because $\tilde{\mathbf{A}}_t = \mathbf{I} + \hat{\mathbf{A}}_t$ is still a positive semi-definitive matrix. The eigenvalue range of a normalised adjacency matrix $\hat{\mathbf{A}}_t$ is $[-1, 1]$ (Huang et al., 2020). Thus, $\hat{\mathbf{A}}_t\mathbf{x} = \lambda_t\mathbf{x} \rightarrow \hat{\mathbf{A}}_t\mathbf{x} + \mathbf{x} = \lambda_t\mathbf{x} + \mathbf{x} \rightarrow (\mathbf{I} + \hat{\mathbf{A}}_t)\mathbf{x} = (\lambda_t + 1)\mathbf{x}$, where $\mathbf{x}$ is a vector, and λ_t is the eigenvalue of $\hat{\mathbf{A}}_t$. Then, we know the eigenvalues of $\tilde{\mathbf{A}}_t$ are in the range $[0, 2]$ and $\tilde{\mathbf{A}}_t$ is a positive semi-definite matrix. As a result, $\mathcal{O}_t$ is still convex and we have:

$$(\mathbf{I} + \hat{\mathbf{A}}_t)\mathbf{H} = 0 \rightarrow \mathbf{H} = -\hat{\mathbf{A}}_t\mathbf{H}. \tag{12}$$

- $\mathcal{O}_c$ can maximise the ED between target nodes and their neighbours with respect to $\hat{\mathbf{A}}_{to}$, which can be regarded as the "enriched heterophily-prone neighbours". Specifically, edges in $\hat{\mathbf{A}}_{to}$ connect each node with the homophily-prone neighbours of its heterophily-prone neighbours, i.e., $\hat{\mathbf{A}}_{to}$ connects target nodes with extended heterophily-prone neighbours. As shown in Figure 5, when computing $\hat{\mathbf{A}}_{to}\mathbf{H} = \hat{\mathbf{A}}_t\hat{\mathbf{A}}_o\mathbf{H}$, all nodes first receives messages from their 1-hop homophily-prone neighbours as shown in Figure 5(a). Then, these nodes receive messages from their convoluted heterophily-prone neighbours as shown in Figure 5(b). When solving $\mathcal{O}_c$, $\tilde{\mathbf{A}}_{to} = \mathbf{I} + \hat{\mathbf{A}}_{to}$ is positive semi-definite since $\rho(\hat{\mathbf{A}}_{to}) = ||\tilde{\mathbf{A}}_{to}||_2 = ||\hat{\mathbf{A}}_t\hat{\mathbf{A}}_o||_2 \leq ||\hat{\mathbf{A}}_t||_2||\hat{\mathbf{A}}_o||_2 = \rho(\hat{\mathbf{A}}_t)\rho(\hat{\mathbf{A}}_o) \leq 1$, where $\rho(\cdot)$ indicates the spectral radius of a matrix. We also provide empirical evidence of this statement in Appendix A. Thus, we can solve $\mathcal{O}_c$ via $\frac{\partial tr(\mathbf{H}(\mathbf{I}+\hat{\mathbf{A}}_{to})\mathbf{H})}{\partial \mathbf{H}} = 0$:

$$(\mathbf{I} + \hat{\mathbf{A}}_{to})\mathbf{H} = 0 \rightarrow \mathbf{H} = -\hat{\mathbf{A}}_{to}\mathbf{H}. \tag{13}$$

When $\mathcal{O}_{\texttt{CGC}}$ is optimised, the learnt node embedding converges to $\mathbf{H} = (\hat{\mathbf{A}}_o - \hat{\mathbf{A}}_t - \hat{\mathbf{A}}_{to})\mathbf{H}$. After involving non-learning activation and affine transformation, we obtain the generalised CGC operation:

$$\mathbf{H} = \sigma((\mathbf{I} + \hat{\mathbf{A}}_o - \hat{\mathbf{A}}_t - \hat{\mathbf{A}}_{to})\mathbf{H}\mathbf{W}). \tag{14}$$

Note that Equation 14 is a special case of Equation 8 (i.e., $\alpha, \beta, \gamma, \delta = 1$), the item $\mathbf{I}$ here is used to maintain self-information of nodes.

The above analysis demonstrates our CGC can utilise both homophily- and heterophily-prone topology simultaneously, which helps GNNs take full advantage of the topology information in complemented graphs. In addition, bridging the aforementioned optimisation objectives with graph spectral theory, CGC can be interpreted from the spectral perspective. In light of this, we propose the following proposition to interpret our CGC.

Proposition 5.1. *In our CGC convolution, the message-passing among nodes (i.e., $\mathbf{H} = (\hat{\mathbf{A}}_o - \hat{\mathbf{A}}_t - \hat{\mathbf{A}}_{to})\mathbf{H}$) under the spatial perspective is equivalent to applying a linear low-pass spectral filter $\mathbf{F}_{low} = 1 - \lambda$ to the Laplacian of $\hat{\mathbf{A}}_o$, a linear high-pass spectral filter $\mathbf{F}_{high} = \lambda - 1$ to the Laplacian of $\hat{\mathbf{A}}_t$ and the same high-pass spectral filter to the Laplacian of $\hat{\mathbf{A}}_{to}$.*

The detailed analysis and proof for Proposition 5.1 is presented in Appendix B. As a result, the low-frequency band of signals in the Laplacian of $\hat{\mathbf{A}}_o$ is preserved, while the high-frequency band of signals in the Laplacian of $\hat{\mathbf{A}}_t$ and $\hat{\mathbf{A}}_{to}$ remain.

6. Experiment

For experiments, we first conduct node classification tasks to evaluate the effectiveness of GOAL on eight real-world datasets. Then we conduct ablation studies of GOAL. Then, we evaluate the quality of the found missing-half topology in terms of homophily ratio for eight datasets. After that, we provide an automatic parameter-tuning method to alleviate the burden of parameter tuning. Finally, we introduce how to extend GOAL to large-scale datasets. Our code is available at https://github.com/zyzisastudyreallyhardguy/GOAL-Graph-Complementary-Learning

6.1. Node Classification on Real-World Datasets

In this section, we compare GOAL with eight baselines on eight real-world datasets for node classification. The selected baselines are MLP, GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2017), APPNP (Klicpera et al., 2019a), GraphSage (Hamilton et al., 2017), ChebyNet (Defferrard et al., 2016), GPR-GNN (Chien et al., 2021) and JKNET (Chen et al., 2020). The adopted datasets include three citation networks (i.e., Cora, Citeseer, and Pubmed) (Yang et al., 2016), two Amazon co-purchasing networks (i.e., Computers and Photo) (Shchur et al., 2018), two Wikipedia graphs (i.e., Chameleon and Squirrel) (Rozemberczki et al., 2021) and the Actor co-occurrence graph (Pei et al., 2020). While the first five are homophily-prone graphs, the last three are heterophily-prone. The dataset statistics and the parameter settings of GOAL are presented in Appendix C.

For dataset split, we randomly split the node set of a dataset according to the ratio 60%/20%/20% for training, validation and test set respectively. We randomly generate the split 10 times and run each baseline and GOAL. Experiment results including the averaged accuracy and standard deviation are summarised in Table 6.

As shown in Table 6, our GOAL outperforms all baselines on all datasets, which validates the effectiveness of GOAL. Notably, GOAL is nearly 9.0% and 3.7% better than the most competitive baseline on Squirrel and Chameleon, respectively. The improvement can be attributed to the utilisation of the missing-half topology, which provides additional structural information during training. For example, when the given graph is homophily-prone, the found topology provides additional information about the inter-class counterparts of target nodes. Then, with GOAL, target nodes can be avoided from being similar to these inter-class peers and thus improve their embedding quality. In contrast, the missing-half topology provides additional homophily-prone structural information for heterophily-prone datasets.

6.2. Ablation Study

We conduct an ablation study to evaluate the effectiveness of each component of GOAL. Specifically, we consider seven variants including $\texttt{GOAL}_{\alpha=0}$, $\texttt{GOAL}_{\beta=0}$, $\texttt{GOAL}_{\gamma=0}$, $\texttt{GOAL}_{\delta=0}$, $\texttt{GOAL}_{\beta}$, $\texttt{GOAL}_{\gamma}$, and $\texttt{GOAL}_{\delta}$. The first four variants mean GOAL without one component, and the latter four variants mean GOAL with only one component, e.g., $\texttt{GOAL}_{\beta}$ means $\alpha, \gamma, \delta = 0$ and $\beta \neq 0$. The experiment result is

Table 1. Node classification for eight baselines on eight datasets.

	Cora	Citeseer	Pubmed	Computer	Photo	Chameleon	Squirrel	Actor
MLP	72.09 ± 0.32	71.67 ± 0.40	87.47 ± 0.14	83.59 ± 0.89	90.49 ± 0.20	46.55 ± 0.42	30.67 ± 0.52	28.75 ± 0.88
GCN	87.50 ± 1.04	75.11 ± 1.12	87.20 ± 0.52	83.55 ± 0.38	89.30 ± 0.82	62.72 ± 2.09	47.26 ± 0.34	29.98 ± 1.18
GAT	88.25 ± 1.22	75.75 ± 1.23	85.88 ± 0.38	85.36 ± 0.50	90.81 ± 0.22	62.19 ± 3.78	51.80 ± 1.04	28.17 ± 1.19
APPNP	88.36 ± 0.61	76.03 ± 1.27	86.21 ± 0.25	88.32 ± 0.36	94.44 ± 0.36	50.88 ± 1.18	33.58 ± 1.00	29.82 ± 0.82
GraphSage	88.01 ± 1.29	75.17 ± 1.35	87.39 ± 0.84	88.54 ± 0.69	94.23 ± 0.62	58.82 ± 2.29	41.19 ± 0.75	31.76 ± 0.73
ChebyNet	87.49 ± 0.90	75.50 ± 0.87	89.05 ± 0.29	89.77 ± 0.36	95.02 ± 0.41	59.98 ± 1.54	40.18 ± 0.55	35.85 ± 1.05
GPR-GNN	88.65 ± 0.75	75.70 ± 0.81	88.53 ± 0.30	87.63 ± 0.48	94.60 ± 0.30	67.96 ± 2.55	49.52 ± 5.00	30.78 ± 0.61
JKNET	86.99 ± 1.60	75.38 ± 1.30	88.64 ± 0.51	86.97 ± 0.56	92.68 ± 0.58	64.63 ± 3.08	44.91 ± 1.94	28.48 ± 1.25
GOAL	$\mathbf{88.75 \pm 0.87}$	$\mathbf{77.15 \pm 0.95}$	$\mathbf{89.25 \pm 0.55}$	$\mathbf{91.33 \pm 0.38}$	$\mathbf{95.60 \pm 0.44}$	$\mathbf{71.65 \pm 1.66}$	$\mathbf{60.53 \pm 1.60}$	$\mathbf{36.46 \pm 1.02}$

Table 2. The ablation study for comparing seven variants of GOAL and GOAL.

	Cora	Citeseer	Pubmed	Computer	Photo	Chameleon	Squirrel	Actor
GOAL$_{\alpha=0}$	88.47 ± 0.92	76.16 ± 2.70	87.51 ± 0.42	90.51 ± 0.53	94.05 ± 0.70	$\mathbf{71.65} \pm 1.66$	$\mathbf{60.53 \pm 1.60}$	35.71 ± 0.80
GOAL$_{\beta=0}$	87.63 ± 1.41	76.28 ± 0.42	85.66 ± 0.59	60.38 ± 2.52	82.44 ± 2.83	67.27 ± 2.28	54.68 ± 0.68	35.20 ± 0.77
GOAL$_{\gamma=0}$	87.69 ± 1.58	76.92 ± 1.10	88.58 ± 0.74	$\mathbf{91.33 \pm 0.38}$	93.76 ± 0.53	69.46 ± 1.63	56.41 ± 2.31	35.49 ± 0.64
GOAL$_{\delta=0}$	88.12 ± 2.14	76.95 ± 1.01	$\mathbf{89.25 \pm 0.55}$	90.71 ± 1.51	94.67 ± 0.68	$\mathbf{71.65 \pm 1.66}$	$\mathbf{60.53 \pm 1.60}$	35.31 ± 0.68
GOAL$_{\beta}$	87.54 ± 1.31	75.54 ± 0.81	87.45 ± 0.35	89.59 ± 1.42	93.97 ± 0.66	69.08 ± 1.15	57.06 ± 1.29	35.89 ± 1.24
GOAL$_{\gamma}$	85.24 ± 1.53	74.70 ± 2.87	86.01 ± 0.83	84.25 ± 1.16	91.75 ± 0.66	66.53 ± 2.09	50.12 ± 4.83	35.37 ± 0.52
GOAL$_{\delta}$	84.83 ± 1.85	74.85 ± 2.93	83.24 ± 1.23	78.19 ± 5.23	90.54 ± 0.55	54.34 ± 2.81	25.11 ± 4.06	35.28 ± 0.50
GOAL	$\mathbf{88.75 \pm 0.87}$	$\mathbf{77.15 \pm 0.95}$	$\mathbf{89.25 \pm 0.55}$	$\mathbf{91.33 \pm 0.38}$	$\mathbf{95.60 \pm 0.44}$	$\mathbf{71.65 \pm 1.66}$	$\mathbf{60.53 \pm 1.60}$	$\mathbf{36.46 \pm 1.02}$

presented in Table 2.

Interestingly, GOAL$_\beta$ standalone already achieves SOTA results on heterophily-prone datasets (i.e., Chameleon, Squirrel and Actor) using only the learnt homophily-prone topology. On the other hand, GOAL$_\gamma$ standalone achieves competitive results with less than 5% gap apart from the best result on most homophily-prone datasets. In addition, GOAL$_{\delta=0}$ reaches the best performance of GOAL in only three out of eight datasets. This indicates the consideration of extended heterophily-prone neighbours is useful.

6.3. Evaluation on the Found Missing Half Topology

We evaluate the quality of the found missing-half topology based on the homophily ratio. Here we provide the definition of homophily ratio:

Definition 6.1. The homophily ratio of a graph, which evaluates the proportion of homophilic connections can be calculated as follows:

$$\mathcal{H}(\mathcal{G}) = \frac{\sum_{(i,j)\in\mathcal{E}_\mathcal{G}}(Y_{v_i} == Y_{v_j})}{|\mathcal{E}_\mathcal{G}|}, \tag{15}$$

where $\mathcal{E}_\mathcal{G}$ is the set of edges in $\mathcal{G}$, $|\cdot|$ indicates the size of a set, Y_{v_i} represents the label for node i, and $==$ evaluates whether two nodes have the identical label and assign 1 to intra-class edges, otherwise 0.

In particular, a learnt heterophily-prone topology should have a low homophily ratio, while a learnt homophily-prone topology should have a high homophily ratio. This indicates the learnt topology is accurate.

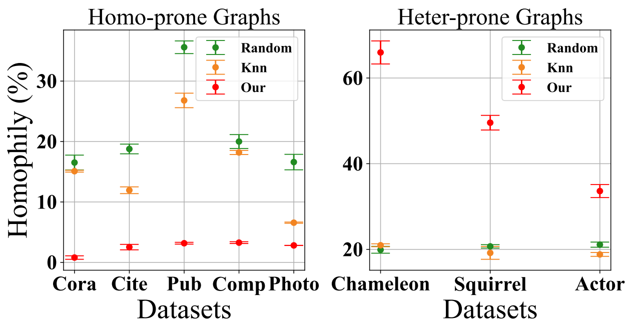

Figure 6. Generated graphs homophily ratio comparison among our method in section 4, KNN and randomly generated graph. The two boundary line of data points in the figure indicates their standard deviation. 'Cite', 'Pub' and 'Comp' respectively mean CiteSeer, Pubmed, and Amazon Computers datasets.

We compare our graph complementation method in section 4 with graphs generated with the KNN algorithm and randomly generated graphs using Erdős–Rényi model (Erdős et al., 1960) in terms of homophily ratio. Given two parameters N (i.e., number of nodes) and E (i.e., number of edges), the Erdős–Rényi model uniformly samples E edges from all possible node pairs in a graph to create random graphs. In our experiment, all generated graphs have the same number of edges. For our method and KNN, we generate complementary heterophily-prone topology for homophily-prone

Table 3. GOAL performance compared with 5 selected baselines on Ogbn-arxiv.

Model	GCN	GAT	APPNP	JKNET	SAGE	**GOAL**
Ogbn-arxiv	70.60	70.52	70.56	70.90	70.34	71.25

Table 4. GOAL$_{at}$ performance compared with GOAL, the best and the second-best baseline performance on five selected datasets.

Dataset	Pubmed	Computer	Chameleon	Squirrel	Actor
Best	89.05	89.77	67.96	51.80	35.85
Second	88.64	88.54	64.63	49.52	31.76
GOAL	89.25	91.33	71.65	60.53	36.46
GOAL$_{at}$	88.37	89.85	69.58	61.76	35.52

datasets and do the opposite for heterophily-prone datasets as introduced in section 4.2. For all methods, we generate the graphs ten times and calculate their homophily ratio. The mean and standard deviation of the homophily ratio on the generated graphs are presented in Figure 6.

As shown in the figure, on homophily-prone datasets (i.e., Cora, Citeseer, Pubmed, Computers, and Photo), our method has a significantly lower homophily ratio compared with generated graphs from the other two methods. Graphs generated with our method for heterophily-prone graphs (i.e., Chameleon, Squirrel, and Actor), on the other hand, have significantly higher homophily ratios than the other two baselines. For example, the homophily ratio for the Chameleon graph is only 23%. However, the learnt homophily-prone topology with our method for this dataset has its homophily ratio reaching 67%.

6.4. Scalability Extension

The scalability of GOAL can be enhanced by incroprating node sampling during the graph complementation step. We have successfully extend GOAL to a large-scale dataset Ogbn-arxiv, containing 169,343 nodes and 1,166,243 edges. The experiment results of GOAL and five baseline counterparts are shown in Table 3. Notably, GOAL outperforms the five baselines, achieving the best results.

During graph complementation, node pair comparisons occur when calculating both the grouping loss and learning-to-rank loss. To simplify the computation of grouping loss, we can sample a fixed number of intra-class and inter-class node pairs rather than computing the loss for all possible node pairs. For learning-to-rank loss, we first need to construct ranking lists for each node in the training set. Rather than finding similarities between all node pairs for ranking list construction, we can sample a node subset and calculate similarities solely between the training set nodes and the sampled nodes. By constructing the ranking list using the sampled node set, the learning-to-rank loss can be easily computed, thereby improving the scalability of GOAL effectively.

6.5. Automatic Prameter-Tuning

Our proposed Complemented Graph Convolution method may be hard to tune, as it has four hyper-parameters: α, β, γ, and δ to control the weight of each convolution component in Equation 8. To alleviate this problem, we also introduce an automatic parameter-tuning version of GOAL, namely, GOAL$_{at}$. GOAL$_{at}$can automatically tune these four parameters by transforming these parameters into learnable parameters. Thus, as opposed to hand-tuned hyperparameters, they can be updated automatically during model training via gradient descent. We have conducted experiments on a selected set of 5 datasets, and the results are presented in Table 4.

The table demonstrates that GOAL$_{at}$ significantly outperforms the second-best baseline on all datasets except for PubMed. Furthermore, GOAL$_{at}$ achieves state-of-the-art performance on three out of the five datasets. Remarkably, GOAL$_{at}$ even surpasses the original model on the Squirrel dataset by a margin of 1.2%.

7. Future Work

In this paper, we introduce Graph Complementary Learning to address the "missing-half" topology problem. Though we have achieved promising results, we can still consider the following aspects as future directions. For example, could we improve the graph complementation by using a more advanced encoder instead of MLP or adding more losses? Could we design a better graph convolution with other objectives? More importantly, graph complementary learning can be employed in various real-world graphs, e.g., social networks, to enrich their structural information.

Acknowledgements

This research was supported by an Australian Research Council (ARC) Future Fellowship (FT210100097).

References

Balcilar, M., Renton, G., Héroux, P., Gauzere, B., Adam, S., and Honeine, P. Bridging the gap between spectral and spatial domains in graph neural networks. *arXiv preprint arXiv:2003.11702*, 2020.

Bo, D., Wang, X., Shi, C., and Shen, H. Beyond low-frequency information in graph convolutional networks. In *AAAI*. AAAI Press, 2021.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs.

International Conference on Learning Representations, 2014.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. *International Conference on Learning Representations*, 2021.

da Silva, C. L., Salazar, S. D., Brum, C. G., and Terra, P. Survey of electron density changes in the daytime ionosphere over the arecibo observatory due to lightning and solar flares. *Scientific Reports*, 11(1):1–12, 2021.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Erdős, P., Rényi, A., et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

He, M., Wei, Z., Huang, Z., and Xu, H. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *NeurIPS*, 2021.

Huang, W., Rong, Y., Xu, T., Sun, F., and Huang, J. Tackling over-smoothing for general graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Jin, M., Li, Y.-F., and Pan, S. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *Advances in Neural Information Processing Systems*, 2022a.

Jin, M., Zheng, Y., Li, Y.-F., Chen, S., Yang, B., and Pan, S. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering*, 2022b.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *International Conference on Learning Representations*, 2019a.

Klicpera, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019b.

Lilliefors, H. W. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402, 1967.

Liu, Y., Tu, W., Zhou, S., Liu, X., Song, L., Yang, X., and Zhu, E. Deep graph clustering via dual correlation reduction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7603–7611, 2022a.

Liu, Y., Xia, J., Zhou, S., Wang, S., Guo, X., Yang, X., Liang, K., Tu, W., Li, Z. S., and Liu, X. A survey of deep graph clustering: Taxonomy, challenge, and application. *arXiv preprint arXiv:2211.12875*, 2022b.

Liu, Y., Ding, K., Liu, H., and Pan, S. Good-d: On unsupervised graph out-of-distribution detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 339–347, 2023a.

Liu, Y., Zheng, Y., Zhang, D., Lee, V., and Pan, S. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. *Proceedings of the AAAI conference on artificial intelligence*, 2023b.

Luo, L., Haffari, G., and Pan, S. Graph sequential neural ode process for link prediction on dynamic and sparse graphs. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 778–786, 2023a.

Luo, L., Li, Y.-F., Haffari, G., and Pan, S. Normalizing flow-based neural process for few-shot knowledge graph completion. In *The 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023b.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Strang, G. *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 2006.

Tan, Y., Liu, Y., Long, G., Jiang, J., Lu, Q., and Zhang, C. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations*, 2017.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Beyond low-pass filtering: Graph convolutional networks with automatic filtering. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Xiong, B., Zhu, S., Potyka, N., Pan, S., Zhou, C., and Staab, S. Pseudo-riemannian graph convolutional networks. In *NeurIPS*, 2022.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Zhang, H., Wu, B., Yang, X., Zhou, C., Wang, S., Yuan, X., and Pan, S. Projective ranking: A transferable evasion attack method on graph neural networks. In *CIKM*, pp. 3617–3621. ACM, 2021.

Zhang, H., Wu, B., Yuan, X., Pan, S., Tong, H., and Pei, J. Trustworthy graph neural networks: Aspects, methods and trends. *CoRR*, abs/2205.07424, 2022a.

Zhang, H., Yuan, X., Zhou, C., and Pan, S. Projective ranking-based gnn evasion attacks. *IEEE TKDE*, 2022b.

Zhang, H., Yuan, X., Nguyen, Q. V. H., and Pan, S. On the interaction between node fairness and edge privacy in graph neural networks. *CoRR*, abs/2301.12951, 2023.

Zheng, X., Zhang, M., Chen, C., Li, C., Zhou, C., and Pan, S. Multi-relational graph neural architecture search with fine-grained message passing. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 783–792. IEEE, 2022a.

Zheng, X., Zhang, M., Chen, C., Zhang, Q., Zhou, C., and Pan, S. Auto-heg: Automated graph neural network on heterophilic graphs. In *Proceedings of the ACM Web Conference (WWW)*, pp. 611–620, 2023.

Zheng, Y., Lee, V. C., Wu, Z., and Pan, S. Heterogeneous graph attention network for small and medium-sized enterprises bankruptcy prediction. In *Advances in Knowledge Discovery and Data Mining: 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11–14, 2021, Proceedings, Part I*, pp. 140–151. Springer, 2021.

Zheng, Y., Pan, S., Lee, V., Zheng, Y., and Yu, P. S. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. *Advances in Neural Information Processing Systems*, 35: 10809–10820, 2022b.

Zheng, Y., Zheng, Y., Zhou, X., Gong, C., Lee, V. C., and Pan, S. Unifying graph contrastive learning with flexible contextual scopes. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 793–802. IEEE, 2022c.

Zhu, M., Wang, X., Shi, C., Ji, H., and Cui, P. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pp. 1215–1226, 2021.

Zhu, Z., Peng, J., Li, J., Chen, L., Yu, Q., and Luo, S. Spiking graph convolutional networks. In *IJCAI*, pp. 2434–2440. ijcai.org, 2022.

A. Eigenvalues of $(\mathbf{I} + \hat{\mathbf{A}}_t\hat{\mathbf{A}}_o)$

Table 5. The range of eigenvalues for $\mathbf{I} + \hat{\mathbf{A}}_t\hat{\mathbf{A}}_o$ on eight datasets

Dataset	**Cora**	**Citeseer**	**Pubmed**	**Computers**	**Photo**	**Chameleon**	**Squirrel**	**Actor**
Range	[0.2151, 2]	[0.2018, 2]	[0.043, 2]	[0.3193, 2]	[0.2860, 1.9353]	[0.6252, 2]	[0.7637, 2]	[0.8435, 2]

In addition to the theoretical evidence of $\mathbf{I} + \hat{\mathbf{A}}_{to}$ is positive semi-definite when solving $\mathcal{O}_c$, we also conduct empirical evaluation on the eigenvalues of $\mathbf{I} + \hat{\mathbf{A}}_{to}$. Specifically, we calculate the eigenvalues range for $\mathbf{I} + \hat{\mathbf{A}}_{to}$ on eight datasets and the results are shown in Table 5. From the table, we can see all eigenvalues in these datasets are positive, and thus $\mathbf{I} + \hat{\mathbf{A}}_{to}$ are positive semi-definitive matrices in all datasets.

B. Interpreting CGC from the Perspective of Graph Spectral Theory

To illustrate CGC from the perspective of graph spectral theory, we first introduce some background knowledge of bridging spectral GNNs and graph convolution forms, followed by transferring two spectral filters to convolutions in CGC. After demonstrating the functionality of these graph spectral filters, we present the proof for Proposition 5.1.

B.1. Bridging Spectral GNNs with Graph Convolution

Spectral GNNs use spectral graph theory as the foundation. In this framework, signals on graphs are filtered using the eigendecomposition of graph Laplacian $\hat{\mathbf{L}}$. Since $\hat{\mathbf{L}}$ is positive semidefinite, it can be decomposed to $\hat{\mathbf{L}} = \mathbf{U}^T\Lambda\mathbf{U}$. Here $\Lambda = diag(\lambda)$ (λ represents the vector of eigenvalues) and $\mathbf{U}$ indicates the matrix of eigenvectors. Then, we define the graph Fourier transform of a signal on a graph as $\tilde{\mathbf{X}} = \mathbf{U}^T\mathbf{X} \in \mathbf{R}^{N\times D}$, and the inverse transform is $\mathbf{X} = \mathbf{U}\tilde{\mathbf{X}}$. By transposing the convolution theorem to graphs, the graph convolution kernel with spectral filtering can be defined as:

$$\mathcal{A} = \mathbf{U}diag(\mathbf{F}(\lambda))\mathbf{U}^T, \tag{16}$$

where $\mathbf{F}(\lambda)$ is the filter function for filtering eigenvalues λ. Then, a uniform formula of graph convolution bridging the spectral-based approaches and spatial-based approaches can be defined below (Balcilar et al., 2020):

$$\mathbf{H}^{l+1} = \sigma(\sum_{k=1}^{K}\mathcal{A}_k\mathbf{H}^l\mathbf{W}^{(l,k)}), \tag{17}$$

where K is the number of filter functions, $\sigma(\cdot)$ is the non-linear activate function, $\mathbf{W}^{(l,k)} \in \mathbb{R}^{D'\times D'}$ ($\mathbf{W}^{(0,k)} \in \mathbb{R}^{D\times D'}$) is a learnable weight matrix at layer l with k-th graph convolution kernel, $\mathbf{H}^l \in \mathbb{R}^{N\times D'}$ is node representations at l-th layer ($\mathbf{H}^{(0)} = \mathbf{X}$). With Equation 16 and 17, we can design various spectral-based methods using different filter functions and obtain their graph convolution. Taking ChebyNet (Defferrard et al., 2016) as an example, it uses Chebyshev polynomial to form an orthogonal basis and takes them as a series of filter functions: $\mathbf{F}_1(\lambda) = 1, \mathbf{F}_2(\lambda) = \frac{2\lambda}{\lambda_{max}} - 1, \cdots,$ $\mathbf{F}_k(\lambda) = 2\mathbf{F}_2(\lambda)\mathbf{F}_{k-1}(\lambda) - \mathbf{F}_{k-2}(\lambda)$. In this case, $\mathcal{A}_k$ in Equation 17 corresponds to the convolution kernel of each $\mathbf{F}_k(\lambda)$. Another example is GCN (Kipf & Welling, 2017), which can be considered as the simplification of ChebNet with first-order approximation with $K = 2$ and $\lambda_{max} = 2$. The filter function of GCN can be approximated by $\mathbf{F}(\lambda) = 1 - \lambda\frac{\overline{d}}{(1+\overline{d})}$, where $\overline{d}$ is the average degree of nodes (Wu et al., 2022). Thus, we can observe that GCN only consists of a low-pass filter.

B.2. Interpreting CGC with Spectral Filters

This section first demonstrates how specific spectral filters are consistent with operations in CGC followed by pointing out the desired graph topology of specific spectral filters.

B.2.1. Low-pass Filter

Effect. Existing GNN approaches primarily use low-pass filters, e.g., GCN, SGC, and APPNP (He et al., 2021). Here, we analyse a typical linear low-pass filter:

$$\mathbf{F}_{low}(\lambda) = 1 - \lambda, \tag{18}$$

Based on Equation 16 and Equation 17, we can infer its graph convolution form is:

$$\begin{aligned} \mathcal{A} &= \hat{\mathbf{A}}, \\ \mathbf{H}^{l+1} &= \sigma(\hat{\mathbf{A}}\mathbf{H}^{l}\mathbf{W}^{l}). \end{aligned} \tag{19}$$

Desired Topology. From Equation 19, we can see it is consistent with the graph convolution (i.e., Equation 11) derived from objective $\mathcal{O}_o$ in section 5.2. Accordingly, we can see that the linear low-pass filter minimises the Euclidean distance between two connected nodes. This indicates the filtering process relies heavily on the homophily-prone edges of a given graph. When homophily-prone edges are dominant in a graph, same-class nodes can be clustered using low-pass filters. In contrast, if a graph is highly heterophily-prone, these low-pass filters tend to maximise the similarity between inter-class nodes and lead to poor performance. This explains why low-pass-based GNNs such as GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2017) work poorly on heterophily-prone graphs. Thus, the desired topology for low-pass filters is homophily-prone graphs.

B.2.2. High-pass Filter

Effect. Some existing works incorporate linear high-pass filters to capture information for the high-frequency band of graph signals (Bo et al., 2021; Wu et al., 2022). Here, we consider the following linear high-pass filter:

$$\mathbf{F}_{high}(\lambda) = \lambda - 1, \tag{20}$$

According to Equations 16 and 17, the graph convolution form of Equation 20 can be defined as:

$$\begin{aligned} \mathcal{A} &= -\hat{\mathbf{A}}, \\ \mathbf{H}^{l+1} &= \sigma(-\hat{\mathbf{A}}\mathbf{H}^{l}\mathbf{W}^{l}), \end{aligned} \tag{21}$$

Desired Topology. Equation 21 is identical to the graph convolution derived from $\mathcal{O}_t$ (i.e.,Equation 12). $\mathcal{A} = -\hat{\mathbf{A}}$ indicates that the desired topology for linear high-pass filters should be heterophily-prone graphs because $\mathcal{A}$ can maximise the dissimilarity between two connected nodes. If the input topology is homophily-prone, two intra-class nodes would be separated, which is undesirable. Conversely, a heterophily-prone input topology can improve target node embeddings by setting them apart from inter-class nodes. Thus, the desired input topology for linear high-pass filters should be heterophily-prone graphs.

B.3. Overall Analysis

To validate that the induced desired topology is consistent with corresponding filters, we conduct experiments using synthetic datasets, and the result is presented in the figure below:

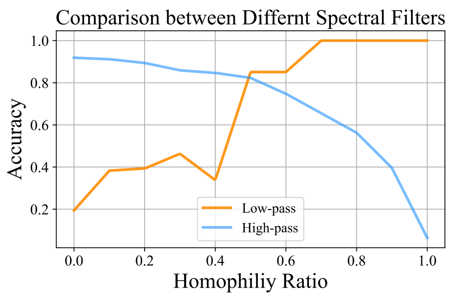

Figure 7. Comparison between Low- and High-pass Spectral Filters on a synthetic dataset.

The synthetic datasets are built based on the Cora dataset. Specifically, we delete all the edges in the Cora dataset and add synthetic connections to it. These connections have a homophily ratio ranging from 0 to 1. The definition of homophily ratio is provided in Definition 6.1. From Figure 7, we can observe that the experiment result is consistent with our findings.

When increasing the homophily ratio, the performance of the high-pass filter is on the decline, whereas the low-pass filter is on the rise. Accordingly, a high-pass filter performs well when a graph has a low homophily ratio. A Low-pass filter, on the other hand, prefers graphs with a high homophily ratio. Thus, it is reasonable to apply the low-pass filter to $\hat{\mathbf{A}}_o$ and the high-pass filter to $\hat{\mathbf{A}}_t$ and $\hat{\mathbf{A}}_{to}$. This is because $\hat{\mathbf{A}}_o$ is homophily-prone, while the other two are heterophily-prone.

B.4. Proof of Proposition 5.1

Proof. `CGC` can be considered as the combination of a linear low-pass filter and two high-pass filters. (1) As analysed in section B.2, the graph convolution derived from optimising $\mathcal{O}_o$ is identical to that from the defined low-pass filter $\mathbf{F}_{low}(\lambda) = 1 - \lambda$ with respect to $\hat{\mathbf{A}}_o$. (2) In addition, the graph convolution obtained from the defined high-pass filter $\mathbf{F}_{high}(\lambda) = \lambda - 1$ concerning $\hat{\mathbf{A}}_t$ is the same with that from optimising $\mathcal{O}_t$. Similarly, the same high-pass filter concerning $\hat{\mathbf{A}}_{to}$ and optimising $\mathcal{O}_c$ have the same graph convolution form. Thus, combining $\mathcal{O}_o$, $\mathcal{O}_t$ and $\mathcal{O}_c$ together is equivalent to applying the low-pass filter $\mathbf{F}_{low}(\lambda) = 1 - \lambda$ to the Laplacian of $\hat{\mathbf{A}}_o$ and applying the high-pass filter $\mathbf{F}_{high}(\lambda) = \lambda - 1$ to the Laplacian of $\hat{\mathbf{A}}_o$ and $\hat{\mathbf{A}}_{to}$, respectively. $\square$

C. Dataset Statistics

This section presents the dataset statistics. Specifically, we adopt eight datasets in the experiment, which are Cora, Citeseer, Pubmed, Computers, Photo, Chameleon, and Squirrel. Their number of node N, number of edges E, number of classes C, number of feature dimension D are summarised in the table below:

Table 6. Dataset statistics

	Cora	**Citeseer**	**Pubmed**	**Computers**	**Photo**	**Chameleon**	**Squirrel**	**Actor**
Nodes N	2,708	3,327	19,717	13,752	7,650	2,277	5,201	7,600
Edges E	5,278	4,552	44,324	245,861	119,081	31,371	198,353	26,659
Classes C	7	6	5	10	8	5	5	5
Feature D	1,433	3,703	500	767	745	2,325	2,089	932