DARLING: Dual Hypergraph-Enhanced Curriculum-Guided Graph Structure Learning for Node Classification

Guangkai Wu[1], Gen Liu[1], Chao Li[2*], Qingtian Zeng[1*], Hui Zhou[1], Zhongying Zhao[1*]

[1]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China
wuguangkai2018@outlook.com, lg97@sdust.edu.cn, lichao@sdust.edu.cn, qtzeng@163.com, zhouhui1026@foxmail.com, zzysuin@163.com

Abstract

Graph Structure Learning (GSL) aims to simultaneously enhance the original graph and the performance of Graph Neural Networks. However, existing GSL methods for node classification fail to consider neighborhood label dependencies during training, which limits their ability to refine the graph structure in an adaptive manner. Furthermore, the training of those methods lacks a proper schedule based on graph structure quality, thereby yielding suboptimal performance. To address these challenges, we propose a novel GSL framework for node classification, termed DuAl hypeRgraph-enhanced curricuLum-guided graph structure learnING for node classification (DARLING). It first introduces a graph structure curriculum module to effectively discriminate the suboptimal graph structures by examining both the distribution of neighborhood labels and the degree of nodes. Subsequently, a self-supervised dual hypergraph similarity learning module is proposed to capture higher-order neighborhood label dependencies. This is achieved via formulating a pre-training task that involves hyperedge batch-filling within the dual hypergraph of the input graph. The experimental results on six datasets demonstrate that the proposed DARLING outperforms eleven state-of-the-art methods significantly, in terms of effectiveness and robustness.

Code —
https://github.com/ZZY-GraphMiningLab/DARLING

Introduction

The remarkable success of Graph Neural Networks (GNNs) in tasks like node classification (Kipf and Welling 2017; Zhao et al. 2023b), graph classification (Adjeisah et al. 2024), and link prediction (Liu et al. 2024) stems from their ability to learn highly informative node representations (Wu et al. 2021). This capability is built upon the message-passing framework, a core principle where nodes iteratively aggregate feature information from their local neighbors (Gilmer et al. 2017).

Despite the widespread success of GNNs, their effectiveness is fundamentally constrained by the quality of the input graph structure (Ma et al. 2022; Dong and Kluger 2023; Wang et al. 2024). Real-world graphs are often suboptimal, plagued by noise and incompleteness (Zhu et al. 2022; Zhou et al. 2023). This limitation stems from two primary factors. On the one hand, some connections may be spurious, arising from erroneous events such as accidental clicks on an e-commerce platform (Wang et al. 2023). On the other hand, the inherent nature of certain interactions makes them difficult to be captured accurately (Lechelon et al. 2022).

Graph Structure Learning (GSL) has emerged as a data-centric solution to solve this issue (Zhu et al. 2022; Zhou et al. 2023; Guo et al. 2025). Its core objective is the joint optimization of the graph topology and the GNN parameters (Chen, Wu, and Zaki 2020; Wang et al. 2021; Liu et al. 2022b; Wang et al. 2023). This process refines the graph by removing spurious edges and inferring missing ones, thereby denoising and completing the structure (Zhou et al. 2023). While existing GSL methods for node classification have achieved remarkable success, they still face two primary challenges:

(1) The over-reliance on class similarity between nodes during training, which limits their ability to adaptively refine the graph structure. Most existing approaches construct the refined graph by solely exploiting the similarity between node representations (Chen, Wu, and Zaki 2020; Liu et al. 2022b; Wang et al. 2023; Zhou et al. 2023; Han et al. 2025). Therefore, they operate under the premise that simply relying on the class similarity is a sufficient basis for GSL on node classification. However, recent research has cast doubt on this assumption, indicating that maximizing the homophily of the learned graph does not invariably yield optimal graph structures for node classification (Zhou et al. 2023). Moreover, it has been demonstrated that the node classification performance of GNNs is highly related to the reliability of the neighborhood label distribution of nodes (Ma et al. 2022; Dong and Kluger 2023; Zheng, Luan, and Chen 2024; Wang et al. 2024). This evidence strongly suggests that the information of label co-occurrence patterns within local neighborhoods, a concept we refer to as neighborhood label dependencies, should be properly integrated into the training process.

(2) The dearth of a proper schedule for graph structures during training, which leads to suboptimal performance. It has been substantiated that overfitting the low-quality graph structure leads to poor performance (Liu et al.

*Corresponding Authors.

2022b). However, most existing methods expose the model to the low-quality graph structures without a proper schedule and discrimination during training, exerting a negative impact on the refinement of the original graph structure. Although a metric has been proposed to quantify graph structural noise based on neighborhood label distribution (Dong and Kluger 2023), it fails to consider the significance of node degree when assessing the quality of graph structures (He et al. 2023; Wang et al. 2024). Furthermore, it remains unclear how to refine the original graph based on this metric.

To address the aforementioned challenges, we propose a DuAl hypeRgraph-enhanced curricuLum-guided graph structure learnING method (DARLING). We first formulate a graph structure curriculum module that incorporates both the distribution of neighborhood labels and the degree of nodes. It essentially provides a training schedule based on the quality of graph structures, ensuring effective refinement under the disparity of graph structural quality. Subsequently, we design a self-supervised dual hypergraph similarity learning module. It begins by converting the pairwise graph into its dual hypergraph and then proceeds to learn higher-order neighborhood label dependencies in a self-supervised manner. Finally, we employ a multi-view graph structure enhancement module to refine the existing graph topology guided by the insights from the graph structure curriculum. The contributions of this research are summarized as follows:

- We propose a curriculum-based approach that aims to mitigate the negative effects of low-quality graph topologies for GSL. To the best of our knowledge, it is the first exploration of curriculum learning in GSL for both topology completion and denoising.
- We design an edge-centric approach to effectively capture the largely neglected dependencies among neighborhood labels based on dual hypergraph. It also ensures the reliability of the learned similarity by employing a self-supervised similarity learning loss.
- We present a multi-view graph structure enhancement method that integrates multiple aspects to adaptively refine the input graph structure.
- We conduct experiments on six datasets, and the results show that DARLING significantly outperforms the baseline methods in terms of effectiveness and robustness.

Methodology

We introduce DARLING, a novel graph structure learning framework that refines the existing graph structure for node classification. Figure 1 illustrates the overall framework of DARLING, which comprises three key modules: **(a)** the graph structure curriculum learning module, **(b)** the self-supervised dual hypergraph similarity learning module, and **(c)** the graph structure enhancement module. The following subsections elaborate on these modules, respectively.

Graph Structure Curriculum Learning

This module is designed to discriminate low-quality graph structures, thus guiding the graph structure learning process. It contains two essential parts: the structure-aware difficulty measurer and the predefined training scheduler. The difficulty measurer first calculates the degree-based difficulty scores $\boldsymbol{d}_{deg}$, as shown in Eqn. (1) and Eqn. (2).

$$\boldsymbol{q}_{deg}^{(c)} = pct(uni(\mathcal{D}^{(c)}), \mathcal{D}^{(c)}), \tag{1}$$

$$\boldsymbol{d}_{deg} = 1 - \boldsymbol{q}_{deg}, \tag{2}$$

where uni leverages the set of node degree of each class $\mathcal{D}^{(c)} = \{\tilde{\boldsymbol{D}}_{ii} \mid i \in \mathcal{V}^{(c)}\}$ to calculate the unique value within $\mathcal{D}^{(c)}$. In this way, it balances the learning of different classes of nodes. pct denotes the function of calculating the percentile within a set. It also normalizes all the difficulty scores within the range of $[0, 1]$. In order to obtain $\mathcal{D}^{(c)}$, we generate the pseudo labels of nodes following Eqn. (3) and Eqn. (4).

$$\boldsymbol{Y}^{(pl)} = f_{GCN}^{(pl)}(\tilde{\boldsymbol{A}}, \boldsymbol{X}), \tag{3}$$

$$\tilde{\boldsymbol{Y}}_i = \begin{cases} \boldsymbol{Y}_i^{(tr)}, & i \in \mathcal{V}_T \\ \boldsymbol{Y}_i^{(pl)}, & otherwise \end{cases} \tag{4}$$

where $\mathcal{V}_T$ denotes the set of training nodes, and $\boldsymbol{Y}^{(pl)}$ represents the one-hot pseudo label matrix of all nodes.

The difficulty measurer also calculates the distribution-based difficulty scores $\boldsymbol{d}_{dist}$. It first adaptively filters out the nodes that possess a small degree within each class to better estimate the neighborhood label distribution of each prototype (He et al. 2023; Wang et al. 2024) following Eqn. (5).

$$\begin{aligned} \mathcal{V}_P^{(c)} = rank(\mathcal{V}^{(c)}, \boldsymbol{q}_{deg}^{(c)}[\operatorname*{argmin}_i(\boldsymbol{q}_{deg}^{(c)}[i] - \boldsymbol{q}_{deg}^{(c)}[i-1])]), \\ s.t. \quad \boldsymbol{q}_{deg}^{(c)}[i] - \boldsymbol{q}_{deg}^{(c)}[i-1] \geq \delta, \end{aligned} \tag{5}$$

where δ controls the intensity of the filtering process, $\boldsymbol{q}_{deg}^{(c)}$ is sorted in ascending order, and $rank$ yields the nodes with large degree. We then calculate the deviation of the neighborhood label distribution $\boldsymbol{b}$. We first estimate the neighborhood label distribution of all nodes following Eqn. (6).

$$\tilde{\boldsymbol{C}} = diag^{-1}(\boldsymbol{C}\mathbb{1})\boldsymbol{C}, \tag{6}$$

where $\mathbb{1}$ is an all-ones column vector and $\boldsymbol{C} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{Y}}$. Then, we estimate the normalized neighborhood label distribution of the class prototypes, as specified in Eqn. (7).

$$\begin{aligned} \tilde{\boldsymbol{C}}^{(p)} &= diag^{-1}(\tilde{\boldsymbol{A}}^{(p)}\tilde{\boldsymbol{Y}}^{(p)}\mathbb{1})\tilde{\boldsymbol{A}}^{(p)}\tilde{\boldsymbol{Y}}^{(p)}, \\ \tilde{\boldsymbol{P}} &= diag^{-1}(\tilde{\boldsymbol{Y}}^{(p)T}\tilde{\boldsymbol{C}}^{(p)}\mathbb{1})\tilde{\boldsymbol{Y}}^{(p)T}\tilde{\boldsymbol{C}}^{(p)}, \end{aligned} \tag{7}$$

where C is the number of classes, $\tilde{\boldsymbol{P}} \in \mathbb{R}^{C \times C}$, and $\tilde{\boldsymbol{A}}^{(p)}$ is the corresponding adjacency matrix of the filtered node set of the class prototypes $\mathcal{V}_P$.

The distribution-based difficulty scores are calculated based on the deviation scores $\boldsymbol{b}$, as specified in Eqn. (8).

$$\begin{aligned} \boldsymbol{b} &= (\tilde{\boldsymbol{C}} - \tilde{\boldsymbol{Y}}\tilde{\boldsymbol{P}}) \odot (\tilde{\boldsymbol{C}} - \tilde{\boldsymbol{Y}}\tilde{\boldsymbol{P}})\mathbb{1}, \\ \boldsymbol{d}_{dist} &= \boldsymbol{q}_{dist}^{(c)} = pct(uni(\mathcal{B}^{(c)}), \mathcal{B}^{(c)}), \end{aligned} \tag{8}$$

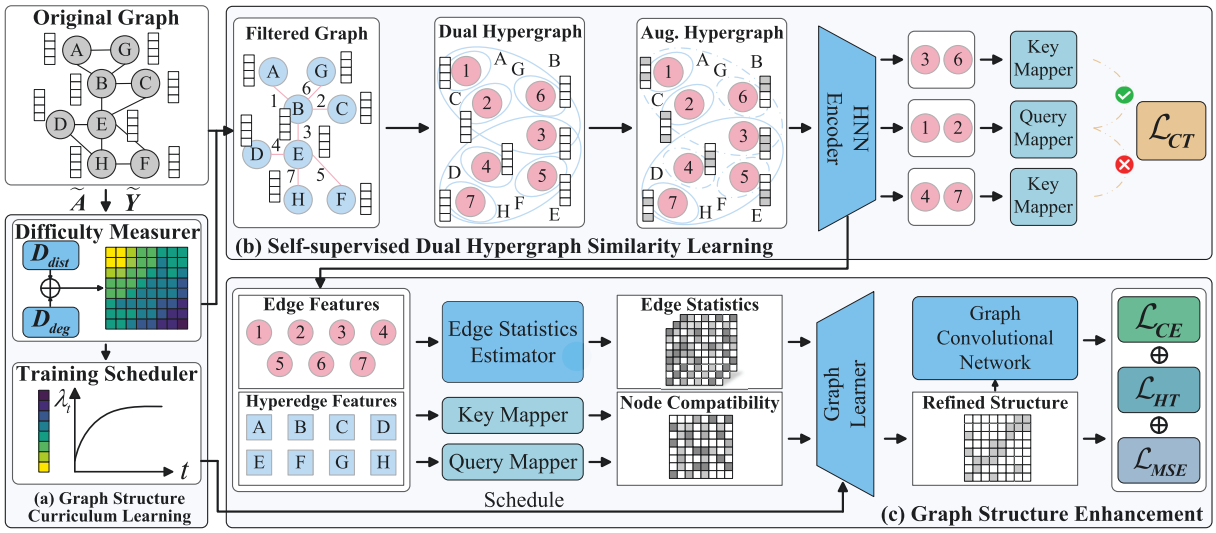

Figure 1: The overall framework of DARLING. It consists of three modules: **(a)** the graph structure curriculum learning module, **(b)** the self-supervised dual hypergraph similarity learning module, and **(c)** the multi-view graph structure enhancement module. DARLING first integrates both the node degree and the neighborhood label distribution to guide the training of the framework. Subsequently, it learns the neighborhood label dependencies based on the filtered graph and edge representations. Finally, it utilizes the hypergraph features and other aspects to refine the existing graph structure guided by the graph structure curriculum.

where $\mathcal{B}^{(c)} = \{\boldsymbol{b}_i \mid i \in \mathcal{V}^{(c)}\}$. We fuse these two aspects to obtain the final difficulty scores $\boldsymbol{d}$ following Eqn. (9).

$$\begin{aligned} \boldsymbol{d} &= \omega \boldsymbol{d}_{dist} + (1-\omega)\boldsymbol{d}_{deg}, \\ \boldsymbol{S}^{(d)} &= (1-\boldsymbol{d})(1-\boldsymbol{d})^T, \end{aligned} \tag{9}$$

where ω is a hyperparameter that balances the two aspects, and $\boldsymbol{S}^{(d)}$ denotes the sample score of node pairs.

We deploy the root pacing function as the training scheduler (Wang, Chen, and Zhu 2022; Li, Wang, and Zhu 2023), which generates the curriculum in an easy-to-difficult manner, as specified in Eqn. (10).

$$\lambda(t) = \min(1, \sqrt{\lambda_0^2 + (1-\lambda_0^2)\frac{t}{T}}), \tag{10}$$

where λ_0 is the initial ratio of the training data, and $\lambda(t)$ denotes the proportion of training data at epoch t.

Self-supervised Dual Hypergraph Similarity Learning

This module first transforms the original pairwise graph into the corresponding dual hypergraph. Then, it models the higher-order neighborhood label dependencies by completing the hyperedge batch-filling task.

Hypergraph Transformation & Augmentation. To reduce the influence of topological noise, we first filter out some edges based on the sample scores $\boldsymbol{S}^{(d)}$ before constructing the hypergraph, as shown in Eqn. (11).

$$\tilde{\boldsymbol{A}}^{(f)} = rank(\tilde{\boldsymbol{A}}, \boldsymbol{S}^{(d)}, q_f), \tag{11}$$

where q_f denotes the proportion of the kept edges, and $rank$ filters the adjacency matrix $\tilde{\boldsymbol{A}}$ by the sampling scores $\boldsymbol{S}^{(d)}$.

Subsequently, we adopt the dual hypergraph transformation originally introduced by (Jo et al. 2021) to convert the filtered adjacency matrix $\tilde{\boldsymbol{A}}^{(f)}$ into the incidence matrix $\boldsymbol{M}^{(f)}$ of the hypergraph. It perfectly matches the objective of capturing higher-order neighborhood label dependencies. We take the average of linked nodes' features in the pairwise graph to construct the node feature matrix of the hypergraph.

We perform node feature random masking $\mathcal{T}_{fm}$ and hyperedge random dropping $\mathcal{T}_{hd}$ after the transformation for self-supervised learning on the dual hypergraph (Kim et al. 2024a,b), as specified in Eqn. (12).

$$\begin{aligned} \mathcal{T}_{fm} &: \bar{\boldsymbol{E}} = \boldsymbol{E} \odot \boldsymbol{F}^{(fm)}, \boldsymbol{F}_{ij}^{(fm)} \sim B(1, 1-p_f), \\ \mathcal{T}_{hd} &: \bar{\boldsymbol{M}}^{(f)} = \boldsymbol{M}^{(f)} diag(\boldsymbol{f}^{(hd)}), \boldsymbol{f}_i^{(hd)} \sim B(1, 1-p_h), \end{aligned} \tag{12}$$

where B represents the Bernoulli distribution, p_f and p_h denote the probability of masking an entry of the feature matrix $\boldsymbol{E} \in \mathbb{R}^{E \times d}$ and dropping a hyperedge, respectively.

Hyperedge Batch-Filling Task. We first redesign the original key mapper proposed in (Kim et al. 2024a) and formulate the *hyperedge batch-filling task*. Specifically, we first encode all the nodes in the augmented dual hypergraph by a hypergraph neural network (Huang and Yang 2021; Lee and Shin 2023), as described in Eqn. (13).

$$\boldsymbol{E}' = f_{HNN}(\bar{\boldsymbol{M}}^{(f)}, \bar{\boldsymbol{E}}). \tag{13}$$

Afterwards, for a given node v_i in the original graph, we randomly split its corresponding edge set into two types of

equal-sized edge set: the key set $\mathcal{E}_{ij}^{(k)}$ and the query set $\mathcal{E}_{ij}^{(q)}$, where $j \in [\rho]$ and ρ represents the maximum number of key-query pairs. We then encode these edge sets by the redesigned mappers following Eqn. (14) and Eqn. (15).

$$\boldsymbol{h}_{ij}^{(k)} = g_{\varphi}^{(k)}(\frac{1}{|\mathcal{E}_{ij}^{(k)}|}\sum_{e_t \in \mathcal{E}_{ij}^{(k)}} \boldsymbol{E}_t'), \tag{14}$$

$$\boldsymbol{h}_{ij}^{(q)} = g_{\varphi}^{(q)}(\frac{1}{|\mathcal{E}_{ij}^{(q)}|}\sum_{e_t \in \mathcal{E}_{ij}^{(q)}} \boldsymbol{E}_t'), \tag{15}$$

where $g_{\varphi}^{(k)}$ denotes the key mapper and $g_{\varphi}^{(q)}$ represents the query mapper, and they do not share model weights. In this way, the key mapper possesses a broader receptive field and the combinations of node representations are well handled, thus better capturing the higher-order neighborhood label dependencies. We further define the similarity between a key-query pair that can reconstruct the original neighborhood label dependencies as the positive similarity s_p, otherwise the negative similarity s_n.

To ensure the quality of graph structure learning and similarity learning (Sun et al. 2020; Wen et al. 2023), we adopt a variant of the Circle loss (Sun et al. 2020; Gao et al. 2022) instead of the softmax-based node classification loss for the hyperedge batch-filling task, as specified in Eqn. (16).

$$\begin{aligned}\mathcal{L}_{CT} = \log\big[1 + \sum_{j=1}^{L}\exp(\gamma(\alpha_n^j)^{\tau_{sl}}(s_n^j - \Delta_n)) \\ \sum_{i=1}^{K}\exp(-\gamma(\alpha_p^i)^{\tau_{sl}}(s_p^i - \Delta_p))\big],\end{aligned} \tag{16}$$

where $\alpha_p^i = [1 + m - s_p^i]_+$, $\alpha_n^j = [s_n^j + m]_+$, $\Delta_p = 1 - m$, $\Delta_n = m$. m and γ are essential for similarity learning. $[\cdot]_+$ is the "cut-off at zero" operation, and τ_{sl} is the temperature parameter to balance the weighting factors. L and K are the number of negative pairs and positive pairs, respectively.

Graph Structure Enhancement

In this subsection, we present a multi-view graph structure enhancement module that integrates various features of node pairs to improve the quality of the original graph structure.

Node Compatibility. To enrich the input of the graph learner with higher-order neighborhood label dependencies, we use the edge representations to calculate the node compatibility scores $\boldsymbol{S}^{(a)}$, as shown in Eqn. (17) to Eqn. (19). It essentially enhances the adaptivity of the graph learner.

$$\boldsymbol{Q}_u = \frac{1}{|\mathcal{E}_u|}\sum_{i \in \mathcal{E}_u} \boldsymbol{E}_i', \tag{17}$$

$$\boldsymbol{S}_{vu}^{(a)} = \cos(g_{\varphi}^{(k)}(\boldsymbol{Q}_u), g_{\varphi}^{(q)}(\boldsymbol{Q}_v)), \tag{18}$$

$$\boldsymbol{S}_{uv}^{(a)} = \cos(g_{\varphi}^{(k)}(\boldsymbol{Q}_v), g_{\varphi}^{(q)}(\boldsymbol{Q}_u)), \tag{19}$$

where $\mathcal{E}_u$ denotes the set of edges that connect to node u. We fine-tune the mappers to further bridge the gap between the pretraining task and GSL. A high value of $\boldsymbol{S}_{uv}^{(a)}$ indicates that the higher-order neighborhood label dependencies of node v are more likely to complete the one of node u.

Edge Statistics Estimator. It computes several features to further enrich the structural information passed to the graph learner. We enumerate these features as follows.

Inner similarity indicates the complexity of the neighborhood label dependencies, as specified in Eqn. (20).

$$\boldsymbol{S}_u^{(i)} = \frac{1}{|\mathcal{E}_u|(|\mathcal{E}_u| - 1)}\sum_{i \in \mathcal{E}_u}\sum_{j \neq i, j \in \mathcal{E}_u}\cos(\boldsymbol{E}_i', \boldsymbol{E}_j'). \tag{20}$$

Cross similarity holds the same spirit as node compatibility, but it is obtained at the edge level rather than the hyperedge level, as described in Eqn. (21) and Eqn. (22).

$$\boldsymbol{S}_{vu}^{(c)} = \max_{i \in \mathcal{E}_u}\frac{1}{|\mathcal{E}_v|}\sum_{j \in \mathcal{E}_v}\cos(\boldsymbol{E}_i', \boldsymbol{E}_j'), \tag{21}$$

$$\boldsymbol{S}_{uv}^{(c)} = \max_{i \in \mathcal{E}_v}\frac{1}{|\mathcal{E}_u|}\sum_{j \in \mathcal{E}_u}\cos(\boldsymbol{E}_i', \boldsymbol{E}_j'). \tag{22}$$

Label information directly introduces the class information of a given pair of nodes (u, v), as described in Eqn. (23).

$$\boldsymbol{l}_{uv} = \frac{\tilde{\boldsymbol{Y}}_u + \tilde{\boldsymbol{Y}}_v}{2}. \tag{23}$$

Multi-view Graph Learner. It combines multiple features to refine the graph structure, as explained in Eqn. (24).

$$\begin{aligned}\boldsymbol{S}_{uv} = [\boldsymbol{S}_{uv}^{(m)}, \boldsymbol{S}_{uv}^{(a)}, \boldsymbol{S}_{uv}^{(g)}, \boldsymbol{S}_u^{(i)}, \boldsymbol{S}_v^{(i)}, \boldsymbol{S}_{uv}^{(c)}, \boldsymbol{S}_{vu}^{(c)}, \boldsymbol{h}_u, \boldsymbol{h}_v]^T, \\ \boldsymbol{e}_{uv} = \boldsymbol{S}_{uv}||\boldsymbol{l}_{uv}, \\ \hat{\boldsymbol{A}}_{uv}^{(o)} = \frac{1}{\eta}\sum_{i=1}^{\eta}\Big[1 + \tanh\big(f_{MLP}^{(i)}(\boldsymbol{e}_{uv})\big)\Big],\end{aligned} \tag{24}$$

where $||$ represents the concatenation operation, η denotes the number of models used to make predictions. $\boldsymbol{S}^{(m)}$ and $\boldsymbol{S}^{(g)}$ are the class similarity scores produced by a MLP model and the GCN model, respectively (Liu et al. 2022b). $\boldsymbol{h}_u$ is the homophily for node u (Wei et al. 2023). The graph learner is trained based on the downstream task to adaptively balance the relative weighting of multiple views.

We also apply several post-processing techniques (Zhu et al. 2022; Liu et al. 2022b) to ensure the quality of the refined graph structure, as specified in Eqn. (25) to Eqn. (27).

$$\hat{\boldsymbol{A}}_{uv}^{(sp)} = \begin{cases}\hat{\boldsymbol{A}}_{uv}^{(o)}, & \hat{\boldsymbol{A}}_{uv}^{(o)} > \epsilon \\ 0, & otherwise\end{cases} \tag{25}$$

$$\hat{\boldsymbol{A}}^{(norm)} = \hat{\boldsymbol{D}}_{in}^{-\frac{1}{2}}\hat{\boldsymbol{A}}^{(sp)}\hat{\boldsymbol{D}}_{out}^{-\frac{1}{2}}, \tag{26}$$

$$\hat{\boldsymbol{A}} = \mu\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}} + (1 - \mu)\hat{\boldsymbol{A}}^{(norm)}, \tag{27}$$

where ϵ controls the intensity of graph sparsification, and μ balances the effect of graph structure learning.

Method	Cora	Citeseer	Pubmed	Roman-empire	Amazon-ratings	Minesweeper
GCN	82.03±0.65	71.60±0.66	79.31±0.81	51.00±0.33	49.57±0.51	78.15±0.90
IDGL	84.04±0.67	73.10±0.69	82.29±0.55	47.80±0.50	46.43±0.60	51.35±0.70
GRCN	84.10±0.47	72.85±0.59	79.05±0.27	45.39±0.60	50.02±0.57	73.25±0.79
ProGNN	81.13±0.39	72.08±0.69	79.58±0.53	55.83±0.64	40.74±0.63	52.01±1.17
GEN	83.10±0.52	73.07±0.57	80.45±0.87	60.27±0.52	49.26±0.46	79.16±1.12
CoGSL	83.23±0.73	72.97±0.67	79.69±0.56	47.24±0.45	41.85±0.42	61.54±1.35
SUBLIME	83.80±0.65	72.71±0.78	80.24±0.84	63.12±0.49	48.37±0.46	50.76±1.21
GSR	83.35±0.67	72.20±0.76	81.86±0.57	60.47±0.56	49.08±0.59	51.23±0.83
PROSE	83.21±0.75	72.60±0.47	**83.04±0.79**	61.18±0.59	47.59±0.72	77.14±1.11
GCN-SA	83.67±0.54	72.70±0.63	80.78±0.65	63.47±0.62	48.49±0.55	53.74±0.48
UnGSL	84.12±0.81	73.51±0.58	81.25±0.75	64.52±0.64	49.12±0.39	51.83±1.32
DARLING (ours)	**85.18±0.69**	**74.62±0.74**	82.80±0.26	**66.03±0.28**	**51.07±0.74**	**81.27±1.38**
Relative Improv.	+1.26%	+1.51%	-0.29%	+2.34%	+2.10%	+2.67%

Table 1: The experimental results of node classification. The best results are highlighted in **bold**, and the runner-ups are underlined, respectively.

Optimization. To optimize the performance of node classification, the cross-entropy loss is first employed as part of the optimization objective, as described in Eqn. (28).

$$\mathcal{L}_{CE} = CELoss(\boldsymbol{Y}^{(cl)}, \boldsymbol{Y}^{(tr)}), \tag{28}$$

where $\boldsymbol{Y}^{(cl)}$ is the predicted softmax probability matrix.

We adopt the head-tail loss proposed in (Liu et al. 2022b) to mitigate the supervision starvation problem (Fatemi, El Asri, and Kazemi 2021) following (Wang et al. 2023), as specified in Eqn. (29) and Eqn. (30).

$$\mathcal{L}_{HT} = \frac{1}{2N}\sum_{i=1}^{N}\Big[\ell(\boldsymbol{z}_{l,i}, \boldsymbol{z}_{a,i}) + \ell(\boldsymbol{z}_{a,i}, \boldsymbol{z}_{l,i})\Big], \tag{29}$$

$$\ell(\boldsymbol{z}_{l,i}, \boldsymbol{z}_{a,i}) = -\log \frac{\exp\big(\cos(\boldsymbol{z}_{l,i}, \boldsymbol{z}_{a,i})/\tau_{ht}\big)}{\sum_{k=1}^{N}\exp\big(\cos(\boldsymbol{z}_{l,i}, \boldsymbol{z}_{a,k})/\tau_{ht}\big)}, \tag{30}$$

where l denotes the learner view, a denotes the anchor view, τ_{ht} represents the temperature parameter.

To ensure that the graph learner is directly aware of the curriculum, we design a regression loss based on the anchor view used by the head-tail loss, as specified in Eqn. (31).

$$\mathcal{L}_{MSE} = \frac{1}{|\mathcal{E}_t|}\sum_{e_{uv}\in\mathcal{E}_t}(\hat{\boldsymbol{A}}_{uv}^{(norm)} - \boldsymbol{A}_{uv}^{(a)})^2, \tag{31}$$

where $\mathcal{E}_t$ denotes the curriculum generated at epoch t.

Finally, the overall loss function for the training of the graph learner is described in Eqn. (32).

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha\mathcal{L}_{HT} + \theta\mathcal{L}_{MSE}, \tag{32}$$

where α and θ are hyperparameters that balance the contribution of each part of the loss.

We deploy an alternate optimization strategy to train the model. Specifically, we first train the classifier by $\mathcal{L}_{CE}$. The graph learner is optimized by the overall loss $\mathcal{L}$ with the weight of the classifier fixed. In each training epoch t for the graph learner, the graph learner is trained following the graph structure curriculum $\mathcal{E}_t$. Hence, it gradually learns how to refine the graph structure by only predicting the easiest node pairs that are assigned by the training scheduler defined in Eq. (10) during the training phase. In the inference phase, the trained graph learner refines all the possible node pairs for the evaluation of the classifier. This curriculum-based strategy steers the training on high-quality graph topology in the early training stages, thus mitigating the detrimental effect of low-quality graph structures.

Experiments

In this section, we conduct several experiments to evaluate the performance of DARLING, which aims to refine the existing graph structure for node classification. We aim to answer the following questions.

- **RQ1:** Can DARLING effectively refine the existing graph structure for node classification?
- **RQ2:** How do the proposed modules contribute to the performance of DARLING?
- **RQ3:** How does DARLING perform under extra structural perturbations?
- **RQ4:** What is the impact of hyperparameter settings on the performance of DARLING?

Experiment Settings

Datasets. We select five real-world datasets, namely Cora, Citeseer, Pubmed (Sen et al. 2008), Roman-empire, and Amazon-ratings (Platonov et al. 2023), and one synthetic dataset, namely Minesweeper (Platonov et al. 2023). The selected datasets possess diverse properties, e.g., homophily and average node degree. Moreover, we follow the suggestions from (Platonov et al. 2023) and carefully select the three heterophilous datasets, ensuring the stability and validity of the experiments. Notably, the six datasets are from

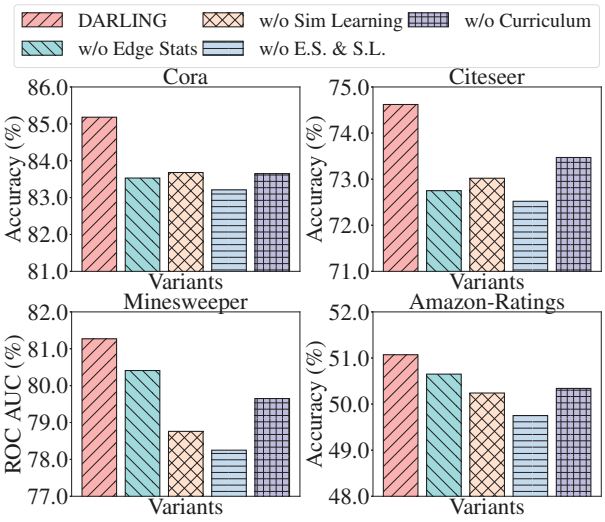

Figure 2: Results of ablation study on Cora, Citeseer, Minesweeper, and Amazon-ratings.

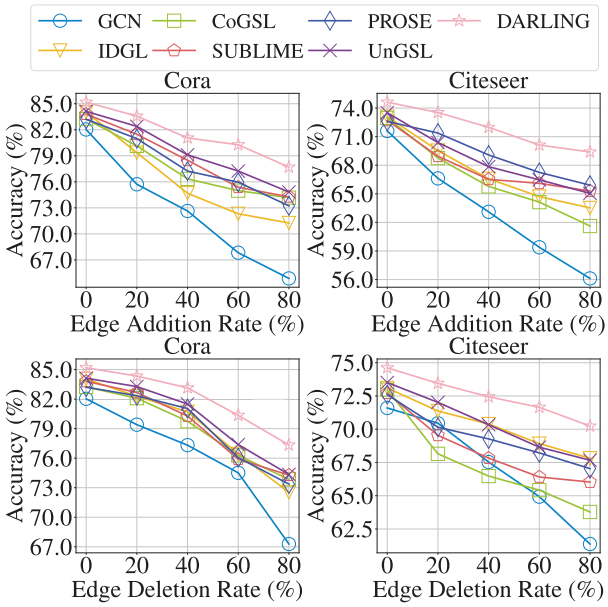

Figure 3: Results of robustness analysis under random graph structure perturbation.

various application, including academic citations, natural language processing, product co-purchasing, and computer games, which further ensures the reliability of the evaluation.

Baselines. We compare DARLING with eleven baselines, including GCN (Kipf and Welling 2017) and ten GSL methods (IDGL (Chen, Wu, and Zaki 2020), GRCN (Yu et al. 2021), ProGNN (Jin et al. 2020), GEN (Wang et al. 2021), CoGSL (Liu et al. 2022a), SUBLIME (Liu et al. 2022b), GSR (Zhao et al. 2023a), PROSE (Wang et al. 2023), GCN-SA (Jiang et al. 2024), and UnGSL (Han et al. 2025)).

Implementation Details. We use the official source codes to reproduce all baselines. For UnGSL, we choose the SUBLIME+UnGSL version as its implementation since it performs overall the best in the original paper (Han et al. 2025). To ensure a fair comparison, we consistently use a GCN backbone (Kipf and Welling 2017) without performance-enhancing tricks like LayerNorm, following the established protocol in (Zhou et al. 2023). All methods, including ours, are evaluated over five random runs. For binary classification dataset, e.g., Minesweeper, we report the ROC AUC metric. For other datasets, we report the accuracy metric. We adopt the splitting strategy of datasets applied in OpenGSL (Zhou et al. 2023). For the heterophilous datasets from (Platonov et al. 2023), we use the first five splits for evaluation. Readers are referred to the public code for more details on the implementation.

Performance Comparison (RQ1)

We perform experiments five times and report the mean value and standard deviation of the model performance on node classification. The results are shown in Table 1. It is observed that all baseline GSL methods struggle to achieve overall better results than other methods on all datasets. It suggests that most existing GSL methods for node classification cannot adaptively refine the original graph structure, which confirms the observation of OpenGSL (Zhou et al. 2023). Despite the difficulty of this task, our method, DARLING, achieves effective optimization over the backbone GCN model and outperforms the SOTA methods on 5 out of 6 datasets. These results demonstrate that DARLING can refine the original graph structure more adaptively, thus achieving better performance when facing various complex scenarios.

Ablation Study (RQ2)

We design four variants of DARLING by dropping certain parts of the whole framework. The first variant, "w/o Edge Stats", discards both the multi-view edge statistics and the multi-view graph learner, and it only uses the node compatibility scores for training. In the second variant called "w/o Sim Learning", the dual hypergraph similarity learning module is disabled, and the similarity scores related to edge representations are dropped. In the third variant, "w/o E.S. & S.L.", only the class similarity generated by GCN is used to calculate the refined graph. For the last variant dubbed "w/o Curriculum", we drop the graph structure curriculum module, and all functions related to curriculum learning are disabled. The results of our ablation study are shown in Figure 2. It is evident that the original framework, DARLING, achieves the best results on all four datasets, and all variants suffer different levels of performance degradation in the evaluation. Specifically, the drop of the similarity learning module causes a great performance decline, especially for Minesweeper. This result highlights the importance of the modeling and utilization of neighborhood label dependencies, particularly in scenarios where the target graph is heterophilous and node features carry less information on

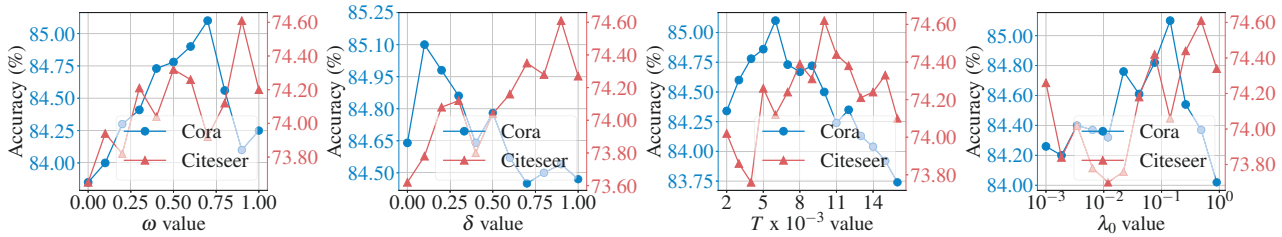

Figure 4: Sensitivity of hyperparameters related to graph structure curriculum module.

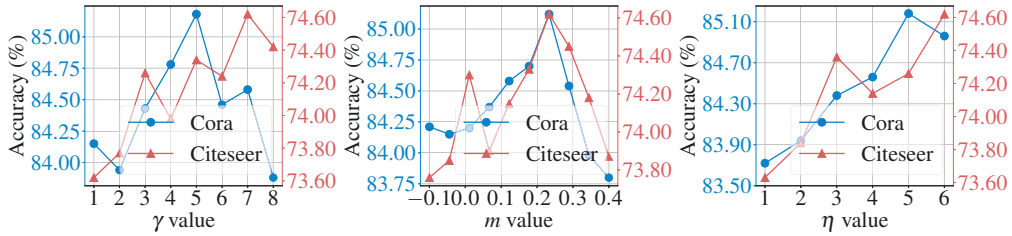

Figure 5: Sensitivity of hyperparameters related to the simiarity learning module and the graph structure enhancement module.

the downstream task. Discarding the multi-view edge statistics also results in a deterioration in node classification performance, especially for the two homophilous datasets. Evidently, the discard of both similarity learning and multi-view edge statistics results in the greatest deterioration of the node classification performance. In addition, the absence of all functions related to curriculum learning exposes the model to low-quality graph structures, thus hindering the model from achieving its best performance.

Robustness Analysis (RQ3)

Following (Chen, Wu, and Zaki 2020; Liu et al. 2022b; Wang et al. 2023; Li et al. 2023; Han et al. 2025), we conduct the experiments by randomly dropping or adding edges into the original graph structure. The ratios of the modified edges are changed from 0.0 to 0.8 to control the severity of the random perturbations. The results are shown in Figure 3. It is evident that all methods suffer a decrease in performance when encountering random structure perturbation. However, DARLING consistently achieves the best performance and receives the least decrease, demonstrating that our proposed method has stronger robustness against graph structural perturbations.

Hyperparameter Analysis (RQ4)

We conduct a hyperparameter analysis and depict the results in Figure 4 and Figure 5. The hyperparameter ω balances the contribution of the distribution-based and degree-based difficulty scores. It can be observed that both of them are crucial for effective curriculum learning. The hyperparameter δ adjusts the number of filtered nodes before calculating the distribution-based difficulty. It can be noted that the pre-filtering process improves the effectiveness of the estimated difficulty scores, thus encouraging more adaptive refinement of the original graph structure. The hyperparameters λ_0 and T control the learning speed of DARLING. The experimental results indicate that a fair amount of node pairs for the initial training phase are crucial to achieving optimal results, but the further increase hinders the curriculum learning process, causing a decrease in node classification performance. Therefore, a proper learning curve should be selected prior to curriculum learning. The hyperparameters γ and m control the sensitivity of the similarity learning process. It can be observed that a too large or too small value of both γ and m results in deteriorated performance. A very large value of both hyperparameters causes instability of the training process, while a very small value of them increases the noise in the cosine similarity. The hyperparameter η represents the number of f_{MLP} used in the multi-view graph learner. It is evident that the performance of DARLING generally increases as η increases because of the improved stability of the training process.

Conclusion

In this paper, we propose a novel framework called DARLING for graph structure learning on node classification. It first schedules the training process to mitigate the negative impact exerted by low-quality graph structures based on graph structure curriculum learning. Subsequently, the higher-order neighborhood label dependencies are crucially captured by performing similarity learning on the dual hypergraph of the input graph. Finally, the original graph is adaptively refined based on the graph structure curriculum and neighborhood label dependencies. We conduct experiments and compare the performance with several competitive baselines on multiple datasets. The experimental results demonstrate that DARLING significantly outperforms the baseline methods in terms of effectiveness and robustness. In the future, we plan to explore the application of GSL in more complicated scenarios, such as dynamic graphs and hypergraphs.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 62472263, 52374221, 52574256, 62502288), the National Key R&D Program of China (Grant No. 2022ZD0119501), the Taishan Scholar Program of Shandong Province (tstp20250506, tsqn202211154), Shandong Youth Innovation Team, the Natural Science Foundation of Shandong Province (Grant No. ZR2024MF034, ZR2025QC624), the Ministry of Education in China Foundation for Humanities and Social Sciences (Grant No. 24YJAZH058, 24YJCZH461), the Guangxi Key Laboratory of Trusted Software (KX202305).

References

Adjeisah, M.; Zhu, X.; Xu, H.; and Ayall, T. A. 2024. Graph Contrastive Multi-view Learning: A Pre-training Framework for Graph Classification. *Knowledge-Based Systems*, 301: 112112.

Chen, Y.; Wu, L.; and Zaki, M. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In *Advances in Neural Information Processing Systems*, 19314–19326.

Dong, M.; and Kluger, Y. 2023. Towards Understanding and Reducing Graph Structural Noise for GNNs. In *Proceedings of the 40th International Conference on Machine Learning*, 8202–8226.

Fatemi, B.; El Asri, L.; and Kazemi, S. M. 2021. SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, 22667–22681.

Gao, H.; Li, J.; Qiao, P.; and Zheng, C. 2022. Weight-Aware Graph Contrastive Learning. In *International Conference on Artificial Neural Networks*, 719–730.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, 1263–1272.

Guo, Y.; Bo, D.; Yang, C.; Lu, Z.; Zhang, Z.; Liu, J.; Peng, Y.; and Shi, C. 2025. Data-Centric Graph Learning: A Survey. *IEEE Transactions on Big Data*, 11(1): 1–20.

Han, S.; Zhou, Z.; Chen, J.; Hao, Z.; Zhou, S.; Wang, G.; Feng, Y.; Chen, C.; and Wang, C. 2025. Uncertainty-Aware Graph Structure Learning. *Proceedings of the ACM on Web Conference 2025*, 4863–4874.

He, L.; Bai, L.; Yang, X.; Liang, Z.; and Liang, J. 2023. Exploring the Role of Edge Distribution in Graph Convolutional Networks. *Neural Networks*, 168: 459–470.

Huang, J.; and Yang, J. 2021. UniGNN: a Unified Framework for Graph and Hypergraph Neural Networks. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 2563–2569.

Jiang, M.; Liu, G.; Su, Y.; and Wu, X. 2024. Self-Attention Empowered Graph Convolutional Network for Structure Learning and Node Embedding. *Pattern Recognition*, 153: 110537.

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph Structure Learning for Robust Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 66–74.

Jo, J.; Baek, J.; Lee, S.; Kim, D.; Kang, M.; and Hwang, S. J. 2021. Edge Representation Learning with Hypergraphs. In *Advances in Neural Information Processing Systems*, 7534–7546.

Kim, S.; Kang, S.; Bu, F.; Lee, S. Y.; Yoo, J.; and Shin, K. 2024a. HypeBoy: Generative Self-Supervised Representation Learning on Hypergraphs. In *the 12th International Conference on Learning Representations*, 1–32.

Kim, S.; Lee, S. Y.; Gao, Y.; Antelmi, A.; Polato, M.; and Shin, K. 2024b. A Survey on Hypergraph Neural Networks: An in-Depth and Step-by-Step Guide. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6534–6544.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *the 5th International Conference on Learning Representations*, 1–14.

Lechelon, M.; Meriguet, Y.; Gori, M.; Ruffenach, S.; Nardecchia, I.; Floriani, E.; Coquillat, D.; Teppe, F.; Mailfert, S.; Marguet, D.; Ferrier, P.; Varani, L.; Sturgis, J.; Torres, J.; and Pettini, M. 2022. Experimental Evidence for Long-distance Electrodynamic Intermolecular Forces. *Science Advances*, 8(7): eabl5855.

Lee, D.; and Shin, K. 2023. I'm Me, We're Us, and I'm Us: Tri-directional Contrastive Learning on Hypergraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 8456–8464.

Li, H.; Wang, X.; and Zhu, W. 2023. Curriculum Graph Machine Learning: A Survey. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, 6674–6682.

Li, Z.; Wang, L.; Sun, X.; Luo, Y.; Zhu, Y.; Chen, D.; Luo, Y.; Zhou, X.; Liu, Q.; Wu, S.; Wang, L.; and Yu, J. 2023. GSLB: The Graph Structure Learning Benchmark. In *Advances in Neural Information Processing Systems*, 30306–30318.

Liu, L.; Xie, Q.; Wen, W.; Zhu, J.; and Peng, M. 2024. Edge Contrastive Learning for Link Prediction. *Information Processing & Management*, 61(6): 103847.

Liu, N.; Wang, X.; Wu, L.; Chen, Y.; Guo, X.; and Shi, C. 2022a. Compact Graph Structure Learning via Mutual Information Compression. In *Proceedings of the ACM Web Conference 2022*, 1601–1610.

Liu, Y.; Zheng, Y.; Zhang, D.; Chen, H.; Peng, H.; and Pan, S. 2022b. Towards Unsupervised Deep Graph Structure Learning. In *Proceedings of the ACM Web Conference 2022*, 1392–1403.

Ma, Y.; Liu, X.; Shah, N.; and Tang, J. 2022. Is Homophily a Necessity for Graph Neural Networks? In *the 10th International Conference on Learning Representations*, 1–28.

Platonov, O.; Kuznedelev, D.; Diskin, M.; Babenko, A.; and Prokhorenkova, L. 2023. A Critical Look at the Evaluation of GNNs Under Heterophily: Are We Really Making Progress? In *the 11th International Conference on Learning Representations*, 1–15.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI Magazine*, 29(3): 93–93.

Sun, Y.; Cheng, C.; Zhang, Y.; Zhang, C.; Zheng, L.; Wang, Z.; and Wei, Y. 2020. Circle Loss: A Unified Perspective of Pair Similarity Optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1–10.

Wang, H.; Fu, Y.; Yu, T.; Hu, L.; Jiang, W.; and Pu, S. 2023. PROSE: Graph Structure Learning via Progressive Strategy. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2337–2348.

Wang, J.; Guo, Y.; Yang, L.; and Wang, Y. 2024. Understanding Heterophily for Graph Neural Networks. In *Proceedings of the 41st International Conference on Machine Learning*, 50489–50529.

Wang, R.; Mou, S.; Wang, X.; Xiao, W.; Ju, Q.; Shi, C.; and Xie, X. 2021. Graph Structure Estimation Neural Networks. In *Proceedings of the Web Conference 2021*, 342–353.

Wang, X.; Chen, Y.; and Zhu, W. 2022. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9): 4555–4576.

Wei, X.; Gong, X.; Zhan, Y.; Du, B.; Luo, Y.; and Hu, W. 2023. CLNode: Curriculum Learning for Node Classification. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, 670–678.

Wen, Y.; Liu, W.; Feng, Y.; Raj, B.; Singh, R.; Weller, A.; Black, M. J.; and Schölkopf, B. 2023. Pairwise Similarity Learning is SimPLE. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5308–5318.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Yu, D.; Zhang, R.; Jiang, Z.; Wu, Y.; and Yang, Y. 2021. Graph-Revised Convolutional Network. In Hutter, F.; Kersting, K.; Lijffijt, J.; and Valera, I., eds., *Machine Learning and Knowledge Discovery in Databases*, 378–393.

Zhao, J.; Wen, Q.; Ju, M.; Zhang, C.; and Ye, Y. 2023a. Self-Supervised Graph Structure Refinement for Graph Neural Networks. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, 159–167.

Zhao, Z.; Yang, Z.; Li, C.; Zeng, Q.; Guan, W.; and Zhou, M. 2023b. Dual Feature Interaction-Based Graph Convolutional Network. *IEEE Transactions on Knowledge and Data Engineering*, 35(9): 9019–9030.

Zheng, Y.; Luan, S.; and Chen, L. 2024. What is Missing For Graph Homophily? Disentangling Graph Homophily For Graph Neural Networks. In *Advances in Neural Information Processing Systems*, 68406–68452.

Zhou, Z.; Zhou, S.; Mao, B.; Zhou, X.; Chen, J.; Tan, Q.; Zha, D.; Feng, Y.; Chen, C.; and Wang, C. 2023. OpenGSL: A Comprehensive Benchmark for Graph Structure Learning. In *Advances in Neural Information Processing Systems*, 17904–17928.

Zhu, Y.; Xu, W.; Zhang, J.; Du, Y.; Zhang, J.; Liu, Q.; Yang, C.; and Wu, S. 2022. A Survey on Graph Structure Learning: Progress and Opportunities. arXiv:2103.03036.