Domain Adaptive Unfolded Graph Neural Networks

Zepeng Zhang, Olga Fink

Intelligent Maintenance and Operations Systems (IMOS) Lab
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
zepeng.zhang@epfl.ch, olga.fink@epfl.ch

Abstract

Over the last decade, graph neural networks (GNNs) have made significant progress in numerous graph machine learning tasks. In real-world applications, where domain shifts occur and labels are often unavailable for a new target domain, graph domain adaptation (GDA) approaches have been proposed to facilitate knowledge transfer from the source domain to the target domain. Previous efforts in tackling distribution shifts across domains have mainly focused on aligning the node embedding distributions generated by the GNNs in the source and target domains. However, as the core part of GDA approaches, the impact of the underlying GNN architecture has received limited attention. In this work, we explore this orthogonal direction, i.e., how to facilitate GDA with architectural enhancement. In particular, we consider a class of GNNs that are designed explicitly based on optimization problems, namely unfolded GNNs (UGNNs), whose training process can be represented as bi-level optimization. Empirical and theoretical analyses demonstrate that when transferring from the source domain to the target domain, the lower-level objective value generated by the UGNNs significantly increases, resulting in an increase in the upper-level objective as well. Motivated by this observation, we propose a simple yet effective strategy called cascaded propagation (CP), which is guaranteed to decrease the lower-level objective value. The CP strategy is widely applicable to general UGNNs, and we evaluate its efficacy with three representative UGNN architectures. Extensive experiments on five real-world datasets demonstrate that the UGNNs integrated with CP outperform state-of-the-art GDA baselines.

Code — https://github.com/zepengzhang/DAUGNN
Extended version — https://arxiv.org/abs/2411.13137

Introduction

Over the past decade, graph neural networks (GNNs) have become increasingly prevalent for processing graph-structured data, achieving significant advancements across a wide range of applications, including social networks, engineering applications, molecular chemistry, recommender systems, and more (Wu et al. 2020; Zhou et al. 2020; Wu et al. 2022). Despite their achievements, the majority of GNNs rely on supervised training, which requires a substantial amount of labeled data. This reliance limits their appli-

cability in real-world applications where domain shifts occur and labels for a new target domain are often unavailable (Liu et al. 2023). To mitigate such issues, a variety of graph domain adaptation (GDA) approaches have been developed (Shi et al. 2024; Cai et al. 2024). GDA aims to transfer knowledge from a labeled source domain to an unlabeled target domain, which often experiences distribution shifts. The main challenge of GDA is to address various domain shifts in node features, structural patterns, and task labels, which typically hinder knowledge transfer.

Inspired by the success of domain adaptation (DA) in sequence and image data (Ganin and Lempitsky 2015; Wang and Deng 2018), some early attempts in GDA have employed DA techniques developed for computer vision tasks to learn domain-invariant and label-discriminative embeddings (Zhang et al. 2021; Shen et al. 2021; Dai et al. 2022). However, these approaches have overlooked the unique characteristics of graph-structured data. Subsequently, researchers have begun to examine the distribution shifts that are particularly prevalent in graph-structured data. For example, Guo et al. (2023) investigate the influence of node degree distribution shifts, Shi et al. (2023) study the shifts in hierarchical graph structure, Wu, He, and Ainsworth (2023) analyze the graph subtree discrepancies, and Liu et al. (2023) explore the effect of conditional structure shifts.

While these techniques have significantly advanced the field of GDA, they primarily focus on aligning the distributions of node embeddings between the source and target domains to mitigate the impact of distribution shifts (Liu et al. 2023; Zhu et al. 2023). In computer vision, recent work has begun to examine the effect of the underlying architecture, which has been shown to play a critical role in DA (Li et al. 2023). However, the effect of the underlying architecture has been largely overlooked in the field of GDA. Some recent works have started to investigate how different components in message-passing GNNs affect domain generalization (DG) (Guo et al. 2024; Liu et al. 2024). Their results show that both the decoupled structure and the attention mechanism contribute positively to DG, while the transformation layers impair the generalization capability. Based on these findings, Liu et al. (2024) propose a GNN model with a decoupled structure, while Guo et al. (2024) further adopts the attention mechanism in the model. However, in contrast to the majority of GDA methods that can be applied to var-

ious GNN architectures, Guo et al. (2024); Liu et al. (2024) only propose two particular GNNs, indicating limitations in extending their approaches to other GNNs and addressing issues with distinctive properties like heterophily and heterogeneity. This limited applicability naturally leads to an important question: *Can we facilitate GDA across a wide spectrum of GNNs through architectural enhancement?* Since there is currently no framework encompassing all GNNs, we focus on a subset of GNNs known for their relatively high transparency, namely unfolded GNNs (UGNNs) (Yang et al. 2021; Ma et al. 2021; Zhu et al. 2021). It is important to highlight that our focus on UGNNs encompasses a broad spectrum of models, including several widely adopted GNN architectures such as the approximate personalized propagation of neural predictions (APPNP) model (Gasteiger, Bojchevski, and Günnemann 2019), the Generalized PageRank GNN (GPRGNN) model (Chien et al. 2021), and the elastic GNN (ElasticGNN) model (Liu et al. 2021). Unlike conventional GNNs, UGNNs are rigorously derived from iterative optimization algorithms for solving optimization problems. Thus, the training of a UGNN can be represented as solving a bi-level optimization problem (Zheng et al. 2024), where the optimal solution of a lower-level problem serves as input features for an upper-level loss minimization problem. Benefiting from the transparent nature of UGNNs, we can integrate desired inductive biases into the model by modifying the lower-level problem associated with UGNNs. We can also better understand the behavior of UGNNs by analyzing their corresponding bi-level optimization problems.

In this study, we first perform empirical investigations to understand how UGNNs perform when transferring from the source to the target domain. We begin our analyses with a simple UGNN model: APPNP (Gasteiger, Bojchevski, and Günnemann 2019), whose lower-level objective corresponds to graph signal denoising (GSD) (Ma et al. 2021; Zhu et al. 2021). We empirically evaluate how the lower-level objective for APPNP changes when transferring from the source domain to the target domain and observe that the GSD objective value in the target domain is significantly larger than in the source domain. This interesting observation suggests that the increase in the lower-level objective may be related to the increase in the upper-level objective, i.e., the loss function. To better understand how general UGNNs behave when transferring from the source domain to the target domain, we theoretically analyze how the two objectives in bi-level optimization change for general UGNNs. Based on our empirical and theoretical analyses, we propose a simple yet effective strategy called **Cascaded Propagation** (CP) to enhance the DG ability of UGNNs. Specifically, CP involves reinjecting the output of the lower-level problem back as its input. By applying the CP strategy, the lower-level objective is provably decreased, which potentially leads to a reduction in the upper-level loss function as well. We then showcase how the proposed strategy works with three representative UGNN architectures, namely APPNP (Gasteiger, Bojchevski, and Günnemann 2019), GPRGNN (Chien et al. 2021), and ElasticGNN (Liu et al. 2021). It is worth highlighting that since the proposed CP strategy enhances UGNNs at the architectural level, node

distribution aligning-based methods can be employed together to further improve the effectiveness in tackling GDA tasks. Our contributions can be summarized as follows:

- We investigate both empirically and theoretically how the values of the upper- and lower-level objectives in bi-level optimization, associated with UGNNs, change when transferring from the source to the target domain. The results indicate that there is a significant increase in lower-level objectives when GDA is performed between the training and testing processes.

- Based on our empirical and theoretical analyses, we propose a simple yet effective method called Cascaded Propagation (CP). It builds upon theory-guided DG enhancement for UGNNs and is guaranteed to decrease the lower-level objective value.

- We integrate three UGNNs with the proposed CP and evaluate their performance on GDA tasks. Experiments on 8 real-world node classification GDA tasks demonstrate the effectiveness of the proposed strategy, providing substantial and consistent performance improvement across various UGNN architectures and datasets.

Preliminaries and Background

Notations. We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote an unweighted graph with $\mathcal{V}$ and $\mathcal{E}$ being the node and edge sets. The adjacency matrix is given by $\mathbf{A} \in \mathbb{R}^{N \times N}$. We denote by $\mathbf{1}$ and $\mathbf{I}$ the all-one column vector and the identity matrix, respectively. Given $\mathbf{D} = \mathrm{Diag}(\mathbf{A1}) \in \mathbb{R}^{N \times N}$ as the diagonal degree matrix, the Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. We denote by $\mathbf{A}_{\mathrm{sym}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ the symmetric normalized adjacency matrix. Subsequently, the symmetric normalized Laplacian matrix is defined as $\mathbf{L}_{\mathrm{sym}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$. In GNNs, the symmetric normalized adjacency matrix and Laplacian matrix are commonly used. Thus, in the following, we will use $\mathbf{A}$ and $\mathbf{L}$ to represent their normalized counterparts for simplicity. $\hat{\mathbf{X}} \in \mathbb{R}^{N \times M}$ is a node feature matrix or a graph signal matrix, with M being the dimension of the node feature. $\mathbf{Y} \in \mathbb{R}^{N \times C}$ is a label matrix for node classification tasks with C being the number of classes.

Graph Domain Adaptation

Unsupervised GDA (UGDA) aims to transfer the knowledge learned from a labeled source domain to an unlabeled target domain with potential domain gaps (Shi et al. 2024). We denote the source and target graph domains as $\mathcal{G}^s = \{\mathcal{V}^s, \mathcal{E}^s, \hat{\mathbf{X}}^s, \mathbf{Y}^s\}$ and $\mathcal{G}^t = \{\mathcal{V}^t, \mathcal{E}^t, \hat{\mathbf{X}}^t, \mathbf{Y}^t\}$, where $\mathbf{Y}^s$ and $\mathbf{Y}^t$ represent the labels. Note that in the setting of GDA, $\mathbf{Y}^t$ is unavailable during training. Typically, the source and target domains share the same task and node feature space. Examples include citation networks from different periods (Zhang et al. 2021), social networks from different platforms (Wang et al. 2023a), and proteins from various species (You et al. 2023). However, distribution shifts may still occur in the generated node embeddings and the topological attributes (Guo et al. 2023; You et al. 2023; Shi et al. 2023). The objective of GDA is to train a model using infor-

mation from the source graph $\mathcal{G}^s$, along with $\mathcal{V}^t, \mathcal{E}^t, \hat{\mathbf{X}}^t$, to accurately predict the labels in the target domain $\mathbf{Y}^t$.

Most existing UGDA methods focus on minimizing domain discrepancy in the node representation space (Shi et al. 2024). For example, (Shen et al. 2021) and (Wu, He, and Ainsworth 2023) train the model to learn domain-invariant representations by minimizing pre-defined domain discrepancy metrics such as maximum mean discrepancy (MMD) (Gretton et al. 2012) and graph subtree discrepancy (Wu, He, and Ainsworth 2023). Besides explicitly minimizing domain discrepancies, some approaches use adversarial training techniques that employ a domain classifier to predict the domain from which the representation is generated (Wu and Pan 2020; Shen et al. 2020; Dai et al. 2022). However, the influence of the GNN architecture on GDA has been significantly underexplored. Recently, efforts have been made to investigate the impact of different components in message passing GNNs on DG (Guo et al. 2024) and DA capabilities (Liu et al. 2024). However, these studies have primarily focused on developing specific GNN architectures. A broadly applicable enhancement that can be integrated across a wider range of GNNs remains lacking.

Unfolded Graph Neural Networks

In recent years, research has shown that the message passing layers in various GNNs could often be understood as gradient steps for solving specific optimization problems (Yang et al. 2021; Ma et al. 2021; Zhu et al. 2021; Zhang and Zhao 2022). This interpretation applies to several models, such as GCN (Kipf and Welling 2017), APPNP (Gasteiger, Bojchevski, and Günnemann 2019), and GPRGNN (Chien et al. 2021), among others. For example, in the APPNP model (Gasteiger, Bojchevski, and Günnemann 2019), the message passing scheme can be described as follows:

$$\mathbf{H}^{(0)} = \mathbf{X} = \mathsf{MLP}(\hat{\mathbf{X}}),$$
$$\mathbf{H}^{(k+1)} = (1-\alpha)\,\mathbf{A}\mathbf{H}^{(k)} + \alpha\mathbf{X}, \text{ for } k = 0, \ldots, K-1, \tag{1}$$

where $\mathbf{H}^{(k)}$ represents the learned feature after the k-th layer and α is the teleport probability. From an optimization perspective, the process defined in Eq. (1) can be seen as executing K steps of gradient descent to solve the following problem with initialization $\mathbf{H}^{(0)} = \mathbf{X}$ and step size 0.5 (Zhu et al. 2021; Ma et al. 2021; Zhang and Zhao 2022):

$$\underset{\mathbf{H} \in \mathbb{R}^{N \times M}}{\text{minimize}} \quad \alpha \|\mathbf{H} - \mathbf{X}\|_{\mathrm{F}}^2 + (1-\alpha)\,\mathrm{Tr}\big(\mathbf{H}^\top \mathbf{L}\mathbf{H}\big), \tag{2}$$

where $\mathbf{X}$ and α share the same meaning as in Eq. (1). Problem (2) carries the meaning of GSD, where the first term is a fidelity term representing the distance between the recovered graph signal $\mathbf{H}$ and the noisy graph signal $\mathbf{X}$, and the second term is the Laplacian smoothing term measuring the variation of the recovered graph signal $\mathbf{H}$.

Based on the optimization interpretation of UGNNs, their training process can be further represented as bi-level optimization (Zhang and Zhao 2022; Zheng et al. 2024):

$$\min_{\mathbf{\Theta}_{\mathsf{pre}}, \mathbf{\Theta}_{\mathsf{pos}}} \quad f_{\mathsf{up}}\left(\mathbf{Y}, p_{\mathsf{pos}}(\bar{\mathbf{X}}, \mathbf{\Theta}_{\mathsf{pos}})\right)$$
$$\text{s.t.} \quad \bar{\mathbf{X}} \in \arg\min_{\mathbf{H}} f_{\mathsf{low}}\left(\mathbf{H}, \mathbf{L}, p_{\mathsf{pre}}(\hat{\mathbf{X}}, \mathbf{\Theta}_{\mathsf{pre}})\right), \tag{3}$$

where f_{up} is the upper-level loss function, f_{low} is the lower-level objective that induces the UGNN, and p_{pos} and p_{pre} are the pre-processing and post-processing functions with parameters $\mathbf{\Theta}_{\mathsf{pre}}$ and $\mathbf{\Theta}_{\mathsf{pos}}$, respectively. In the bi-level optimization problem (3), the optimal solution of the lower-level problem $\bar{\mathbf{X}} \in \mathbb{R}^{N \times M'}$ serves as input features to the upper-level objective with M' being the dimension of the node embedding. These optimal features depend on learnable parameters of the lower-level problem in such a way that the entire bi-level pipeline can be trained end-to-end.

Example 1 (Node classification with APPNP). *In an illustrative example, we use APPNP to solve a node classification problem. The objective is to train an APPNP model on a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \hat{\mathbf{X}}, \mathbf{Y}\}$ to predict $\mathbf{Y}$ based on $\{\mathcal{V}, \mathcal{E}, \hat{\mathbf{X}}\}$. In this scenario, we can choose f_{up} as the cross-entropy loss function, p_{pos} as the* softmax *function, f_{low} as the GSD objective in Problem* (2)*, and p_{pre} as a multi-layer perceptron.*

Based on the optimization perspective of GNNs, a new design philosophy of GNNs has emerged, namely the UGNNs. Unlike conventional GNNs that are often designed using heuristics, the forward layers of UGNNs are rigorously derived from the optimization steps for specific lower-level problems f_{low}. The transparent nature of UGNNs allows us to develop GNNs in a more interpretable and controllable way. The presence of an explicit optimization problem allows us to incorporate the desired inductive bias into UGNNs, for example, promoting the fairness of the model (Jiang et al. 2024), handling heterophily and heterogeneous graphs Ahn et al. (2022); Fu, Zhao, and Bian (2022), and addressing potential attacks or noise in the node features or graph structure (Liu et al. 2021; Zhang et al. 2022; Feng et al. 2023), etc. Furthermore, analyzing the obtained node embeddings based on their role in the f_{low} also gives us more explanatory details of the model (Zheng et al. 2024).

Proposed Methodology

Design Motivation

Through the lens of bi-level optimization, UGNNs inherently solve certain lower-level problems specified by $f_{\mathsf{low}}\left(\mathbf{H}, \mathbf{L}, p_{\mathsf{pre}}(\hat{\mathbf{X}}, \mathbf{\Theta}_{\mathsf{pre}})\right)$. Since $\mathbf{H}$ and $\mathbf{\Theta}_{\mathsf{pre}}$ are parameters of the lower-level problem, the lower-level objective f_{low} is specified by $\mathbf{L}$ and $\mathbf{X}$. Thus, if there is a distribution shift, i.e., transferring the model from the source domain to the target domain, the lower-level objective will change.

To investigate how f_{low} changes during GDA, we design semi-supervised node classification experiments on two social networks, i.e., Twitch gamer networks collected in Germany (DE) and England (EN) (Rozemberczki, Allen, and Sarkar 2021). The experiments are performed with APPNP, whose corresponding f_{low} is defined as in Eq. (2) and carries the meaning of GSD. We examine how the value of f_{low} changes in different domains. Specifically, we train an APPNP model on the source graph domain and then evaluate its corresponding f_{low} value on both the source domain and the target domains. We report the average performance over 10 times in Table 1, where we use the min-max normalization to rescale the results to [0, 1].

Source → Target	DE → DE	EN → DE	EN → EN	DE → EN
GSD objective	0.1308	0.7687	0.3129	0.6812

Table 1: GSD objective value corresponding to APPNP

From the results, we observe that the GSD objective value f_{low} evaluated on the domain transfer setting, i.e., EN → DE and DE → EN are much larger than f_{low} evaluated on the in-domain setting, i.e, DE → DE and EN → EN. Intuitively, as we transfer from the source to the target domain, the lower-level objective changes. Thus, the model trained in the source domain cannot approximate the solution of the lower-level problem specified by the target domain $\mathcal{G}^t$ well, resulting in a large GSD objective value in the target domain. We then naturally ask a question: *Can we develop a strategy to maintain the level of noise of UGNNs measured by f_{low} while transferring across domains?*

Domain Adaptive Unfolded Graph Neural Networks

In this subsection, we analyze how f_{up} and f_{low} change during GDA, from the perspective of bi-level optimization. We focus on UGNNs, whose training process can be represented as in Eq. 3. In the source domain, given $\mathbf{L}^s$, $\mathbf{X}^s$, and $\mathbf{Y}^s$, the UGNN model after training will have parameters

$$\boldsymbol{\Theta}^s_{\text{pre}}, \boldsymbol{\Theta}^s_{\text{pos}} \in \arg\min_{\boldsymbol{\Theta}_{\text{pre}}, \boldsymbol{\Theta}_{\text{pos}}} f_{\text{up}}\Big(\mathbf{Y}^s, p_{\text{pos}}\Big(\arg\min_{\mathbf{H}} \\ f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}^s, p_{\text{pre}}(\hat{\mathbf{X}}^s, \boldsymbol{\Theta}_{\text{pre}})\Big), \boldsymbol{\Theta}_{\text{pos}}\Big)\Big). \quad (4)$$

Similarly, we define an oracle model that is trained directly in the target domain $\mathcal{G}^t$ as

$$\boldsymbol{\Theta}^\star_{\text{pre}}, \boldsymbol{\Theta}^\star_{\text{pos}} \in \arg\min_{\boldsymbol{\Theta}_{\text{pre}}, \boldsymbol{\Theta}_{\text{pos}}} f_{\text{up}}\Big(\mathbf{Y}^t, p_{\text{pos}}\Big(\arg\min_{\mathbf{H}} \\ f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}^t, p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}_{\text{pre}})\Big), \boldsymbol{\Theta}_{\text{pos}}\Big)\Big). \quad (5)$$

Then the goal of GDA becomes approximating $\boldsymbol{\Theta}^\star_{\text{pre}}, \boldsymbol{\Theta}^\star_{\text{pos}}$ with $\boldsymbol{\Theta}^s_{\text{pre}}, \boldsymbol{\Theta}^s_{\text{pos}}$. The p_{pos} is normally chosen as parameter-free functions, e.g., the softmax function as in Example 1, or simple linear layers. To this account, we make the following assumption.

Assumption 1. *The parameters in p_{pos} trained on the source graph domain and target graph domain are the same, i.e., $\boldsymbol{\Theta}^s_{\text{pos}} = \boldsymbol{\Theta}^\star_{\text{pos}}$.*

Note that when p_{pos} is chosen as parameter-free functions, Assumption 1 always holds true. When p_{pos} is chosen as linear layers, the reasoning ability still mainly lies in the lower-level problem part, indicating that Assumption 1 will not affect the expressivity of the model largely. Thus, a model under Assumption 1 should have similar performance compared with a model without Assumption 1. We provide an empirical validation of this aspect in Appendix.

Under Assumption 1, when we apply the UGNN model trained in the source domain as defined in Eq. (4) to the target domain, we will obtain the following loss value:

$$f_{\text{up}}\Big(\mathbf{Y}^t, p_{\text{pos}}(\arg\min_{\mathbf{H}} f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}^t, p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}^s_{\text{pre}})\Big), \boldsymbol{\Theta}^\star_{\text{pos}})\Big).$$

Compared to the loss function of the oracle model, the only difference is the embedding matrix $\bar{\mathbf{X}}$ obtained from the lower-level problem. Thus, to make the loss value obtained with the UGNN model trained in the source domain close to the loss value obtained with the oracle model, we need the embedding matrix obtained by the UGNN model trained in source domain $\mathcal{G}^s$, i.e.,

$$\bar{\mathbf{X}}^{s\to t} \in \arg\min_{\mathbf{H}} f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}^t, p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}^s_{\text{pre}})\Big) \quad (6)$$

close to the embedding matrix obtained by the oracle model, i.e.,

$$\bar{\mathbf{X}}^\star \in \arg\min_{\mathbf{H}} f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}^t, p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}^\star_{\text{pre}})\Big). \quad (7)$$

However, since we do not have access to $\mathbf{Y}^t$, we cannot obtain $\boldsymbol{\Theta}^\star_{\text{pre}}$. In other words, we do not have access to $\bar{\mathbf{X}}^\star$ during the training stage. Thus, it is infeasible to directly make $\bar{\mathbf{X}}^{s\to t}$ close to $\bar{\mathbf{X}}^\star$. To make the problem tractable, we instead try to make $f^{s\to t}_{\text{low}} = f_{\text{low}}\Big(\bar{\mathbf{X}}^{s\to t}, \mathbf{L}^t, p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}^s_{\text{pre}})\Big)$ close to $f^\star_{\text{low}} = f_{\text{low}}\Big(\bar{\mathbf{X}}^\star, \mathbf{L}^t, p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}^\star_{\text{pre}})\Big)$. However, the value of $f^\star_{\text{low}}$ is still unavailable as it still relies on $\boldsymbol{\Theta}^\star_{\text{pre}}$. According to the empirical results in Section , the value of $f^\star_{\text{low}}$ is much smaller than the value of $f^{s\to t}_{\text{low}}$ in practice. Motivated by this observation, we suggest that simply reducing the value of $f^{s\to t}_{\text{low}}$ can make it closer to $f^\star_{\text{low}}$.

To achieve such a goal, we propose a simple yet effective strategy. Specifically, we propose to reset $p_{\text{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}^s_{\text{pre}})$ as $\bar{\mathbf{X}}^{s\to t}$ and solve the lower-level problem again. By doing so, we will obtain a new embedding matrix as follows:

$$\bar{\mathbf{X}}^{cp} \in \arg\min_{\mathbf{H}} f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}^t, \bar{\mathbf{X}}^{s\to t}\Big). \quad (8)$$

In particular, a UGNN generates the solution to Eq. (8) by applying the message passing process with input node feature matrix $\bar{\mathbf{X}}^{s\to t}$ and graph Laplacian matrix $\mathbf{L}^t$. In other words, this amounts to feeding the output of the message passing process in UGNN back to the model. Thus, we name this strategy as cascaded propagation (CP). Moreover, after CP, we will arrive at a new loss function value as $f^{cp}_{\text{low}} = f_{\text{low}}\big(\bar{\mathbf{X}}^{cp}, \mathbf{L}^t, \bar{\mathbf{X}}^{s\to t}\big)$.

Assumption 2. *The lower-level objective function f_{low} can be decomposed into three parts as*

$$f_{\text{low}}\Big(\mathbf{H}, \mathbf{L}, p_{\text{pre}}(\hat{\mathbf{X}}, \boldsymbol{\Theta}_{\text{pre}})\Big) = \sum_{v\in\mathcal{V}} \kappa(\mathbf{h}_v, p_{\text{pre}}(\hat{\mathbf{x}}_v, \boldsymbol{\Theta}_{\text{pre}})) \\ + \sum_{(u,v)\in\mathcal{E}} \xi(\mathbf{h}_u, \mathbf{h}_v) + \sum_{v\in\mathcal{V}} \eta(\mathbf{h}_v),$$

where κ represents the feature fidelity term, ξ represents the feature smoothing term, and η represents node-wise constraints. Moreover, p_{pre} parameterized by $\boldsymbol{\Theta}$ only appears in function κ, which satisfies

$$\kappa(\mathbf{h}_1, \mathbf{h}_2) \geq 0, \text{ and } \kappa(\mathbf{h}_1, \mathbf{h}_1) = 0, \ \forall \mathbf{h}_1, \mathbf{h}_2 \in \mathbb{R}^{M'}. \quad (9)$$

For UGNN models satisfying Assumption 2, after the CP process as defined in (8), the value of the lower-level objective f_{low} is guaranteed to decrease, which is formally illustrated in the following theorem. It is important to highlight

that the bi-level optimization problems associated with most UGNNs satisfy Assumption 2, which also covers a wide spectrum of message-passing GNNs (Zheng et al. 2024).

Theorem 1. *Under Assumption 2, when transferring from the source domain to the target domain, the UGNNs with CP always have a smaller or equivalent lower-level objective value compared to their vanilla counterparts, i.e.,*

$$
\begin{aligned}
f_{\mathsf{low}}^{cp} &= f_{\mathsf{low}}\left(\bar{\mathbf{X}}^{cp}, \mathbf{L}^t, \bar{\mathbf{X}}^{s \to t}\right) \\
&\le f_{\mathsf{low}}\left(\bar{\mathbf{X}}^{s \to t}, \mathbf{L}^t, p_{\mathsf{pre}}(\hat{\mathbf{X}}^t, \boldsymbol{\Theta}_{\mathsf{pre}}^s)\right). \quad (10) \\
&= f_{\mathsf{low}}^{s \to t}
\end{aligned}
$$

Remark 1. *According to Theorem 1, repeatedly applying CP can further reduce the lower objective. However, UGNNs experience performance degradation when the number of layers is too high. As a result, with additional CP, even though the gap in the performance after domain transfer narrows, the absolute performance does not necessarily improve.*

In the following, we elaborate in detail on how to integrate CP into UGNNs. In particular, we employ CP with three representative UGNNs, namely APPNP (Gasteiger, Bojchevski, and Günnemann 2019), GPRGNN (Chien et al. 2021), and ElasticGNN (Liu et al. 2021). The GPRGNN has the same lower-level objective as APPNP but with a learnable and layer-wise independent parameter α. The ElasticNet is designed from the following lower-level problem:

$$
\begin{aligned}
f_{\mathsf{low}}\left(\mathbf{H}, \mathbf{L}, p_{\mathsf{pre}}(\hat{\mathbf{X}}, \boldsymbol{\Theta}_{\mathsf{pre}})\right) &= \frac{1}{2}\left\| \mathbf{H} - p_{\mathsf{pre}}(\hat{\mathbf{X}}, \boldsymbol{\Theta}_{\mathsf{pre}}) \right\|_{\mathrm{F}}^2 \\
&+ \frac{\lambda_1}{2}\mathrm{Tr}\left(\mathbf{H}^\top \mathbf{L}\mathbf{H}\right) + \lambda_2 \left\|\Delta\mathbf{F}\right\|_1,
\end{aligned}
$$
(11)

where λ_1 and λ_2 are two weighting parameters and $\boldsymbol{\Delta}$ is the incident matrix satisfying $\boldsymbol{\Delta}^\top \boldsymbol{\Delta} = \mathbf{L}$.

Example 2 (APPNP with CP). *The APPNP integrated with CP, named APPNP$_{CP}$, features the following message passing scheme:*

$$
\begin{aligned}
\mathbf{H}^{(0)} &= \mathbf{X}, \quad \mathbf{H}^{(k+1)} = (1-\alpha)\,\mathbf{A}\mathbf{H}^{(k)} + \alpha\mathbf{X}, \\
&\quad \text{for } k = 0, \ldots, K-1, \\
&\quad \mathbf{H}^{(k+1)} = (1-\alpha)\,\mathbf{A}\mathbf{H}^{(k)} + \alpha\mathbf{H}^{(K)}, \\
&\quad \text{for } k = K, \ldots, 2K,
\end{aligned}
$$
(12)

with $\mathbf{H}^{(2K)}$ being the output.

Example 3 (GPRGNN with CP). *The GPRGNN integrated with CP, named GPRGNN$_{CP}$, has the following message passing scheme:*

$$
\mathbf{H}^{(K)} = \sum_{k=0}^{K} \gamma^{(k)} \mathbf{A}^k \mathbf{X}, \quad \mathbf{H}^{(2K)} = \sum_{k=0}^{K} \gamma^{(k)} \mathbf{A}^k \mathbf{H}^{(K)}, \quad (13)
$$

with $\mathbf{H}^{(2K)}$ being the output.

The message passing scheme of ElasticGNN integrated with CP, named ElasticGNN$_{CP}$, is provided in Appendix. As shown in Example 2 and Example 3, applying CP to

APPNP and GPRGNN can also be interpreted as adding residual connections using different sources. The CP strategy is an architectural enhancement for UGNNs, independent of information from the target domain, making it readily applicable to graph DG. To further leverage the information from the target domain $\mathcal{G}^t = \{\mathcal{V}^t, \mathcal{E}^t, \mathbf{X}^t\}$ in GDA tasks, existing methods on aligning the distribution of node representations between the source and target domains can be used. In our experiments, we incorporate a classical alignment loss, specifically MMD (Gretton et al. 2012), into the loss function with a trade-off parameter ξ. The additional alignment loss helps to reduce domain discrepancies, facilitating the creation of domain-invariant representations.

The proposed CP strategy is compatible with UGNNs that can be represented in the form of Eq. (3). UGNNs can be flexibly designed to address node-level (Yang et al. 2021; Han et al. 2023; Xue et al. 2023), edge-level (Wang et al. 2023b), and graph-level (Chen et al. 2022) problems, allowing the CP strategy to improve the DG capabilities of UGNNs across various downstream tasks. Furthermore, as an architectural enhancement, CP can complement existing UGDA methods that align node embedding distributions between the source and the target domains (Shi et al. 2024).

Experiments

In this section, we evaluate the effectiveness of the proposed CP strategy for UGNNs. First, we introduce the experimental settings. Then, we assess the effectiveness of CP on node classification GDA tasks using three UGNN models: APPNP, GPRGNN, and ElasticGNN. Finally, we conduct ablation studies to investigate the contributions of different components in UGNNs with CP.

Experiment Settings

Datasets. We conduct experiments on three citation networks, namely ACMv9 (A), Citationv1 (C), and DBLPv7 (D) (Zhang et al. 2021), and two social networks, namely Germany (DE) and England (EN) (Rozemberczki and Sarkar 2021). Details on datasets are provided in Appendix.

Baselines. We apply the proposed CP strategy to three UGNNs, named APPNP$_{CP}$, GPRGNN$_{CP}$, and ElasticGNN$_{CP}$. We use the MMD alignment loss to reduce domain discrepancy. Note that the proposed CP method is compatible with other alignment losses as well. To evaluate the effectiveness of CP, we compared with several baselines, including node embedding methods: DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and Node2vec (Grover and Leskovec 2016); source only GNNs: GCN (Kipf and Welling 2017), graph attention network (GAT) (Veličković et al. 2018), and graph isomorphism network (GIN) (Xu et al. 2019), which are only trained on the source graph without any DA techniques; and GDA methods: unsupervised domain adaptive GCNs (UDAGCN) (Wu et al. 2020), adversarial cross-network deep network embedding (ACDNE) (Shen et al. 2020), adversarial separation network (ASN) (Zhang et al. 2021), adversarial domain adaptation with GCN (AdaGCN) (Dai et al. 2022), generic graph adaptive network (GRADE) (Wu, He, and Ainsworth 2023), GNN

	D → A		C → A		A → D		C → D		A → C		D → C	
	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1
DeepWalk	19.83	26.23	19.33	21.94	19.87	25.94	17.51	22.57	17.72	21.05	22.76	29.46
Node2vec	22.05	28.61	17.99	21.76	19.50	24.54	24.98	28.95	25.84	29.89	16.22	21.16
GCN	59.42	63.35	70.39	70.58	65.29	69.05	71.37	74.53	74.78	77.38	69.79	74.17
GAT	43.95	52.93	42.14	50.37	41.36	53.80	45.25	55.85	43.64	57.13	50.04	55.52
GIN	56.50	58.98	59.48	60.46	50.49	59.10	63.48	66.27	62.49	68.61	63.21	69.25
UDAGCN	55.89	58.16	67.22	66.80	64.83	66.95	69.46	71.77	60.33	72.15	61.12	73.28
ACDNE	72.64	71.29	74.79	73.59	73.59	76.24	75.74	77.21	80.09	81.75	78.83	80.14
ASN	71.49	70.15	73.17	72.74	71.40	73.80	73.98	76.36	77.81	80.64	75.17	78.23
AdaGCN	69.47	69.67	70.77	71.67	71.39	75.04	72.34	75.59	76.51	79.32	74.22	78.20
GRADE	59.35	63.72	69.34	69.55	63.03	68.22	70.02	73.95	72.52	76.04	69.32	74.32
SpecReg	72.34	71.01	73.15	72.04	73.98	75.93	73.64	75.74	78.83	80.55	77.78	79.04
A2GNN	**75.00**	73.69	**76.53**	**75.19**	70.28	75.83	73.12	75.82	78.77	81.53	76.31	80.30
APPNP$_{CP}$	73.28	73.30	75.11	74.48	**75.71**	**77.80**	**75.91**	**77.77**	**81.06**	**82.82**	78.29	**80.53**
GPRGNN$_{CP}$	74.78	**74.50**	75.52	74.71	75.15	**77.42**	75.04	77.55	80.95	**82.83**	**79.39**	**81.02**
ElasticGNN$_{CP}$	**75.74**	**74.66**	**76.72**	**75.70**	**75.49**	77.38	**75.78**	**77.64**	**81.16**	82.56	**78.44**	80.09

Table 2: Unsupervised GDA performance on citation networks.

with spectral regularization (SpecReg) (You et al. 2023), and asymmetric adaptive GNN (A2GNN) (Liu et al. 2024). It is worth noting that our method enhances the models from an architectural perspective, which is perpendicular to most existing GDA methods. Thus, they can be applied in combination to further enhance performance.

Implementation Details. We follow the experiment setting introduced in (Wu and Pan 2020; Shen et al. 2020) and applied as well in (Liu et al. 2024). We use 80% of labeled nodes in the source domain for training, 20% of the labeled nodes in the source domain for validation, and all the nodes in the unlabeled target domain for testing. All the experiments are conducted on a Tesla V100 GPU. The node representation dimension is set to 128, and the number of layers is set to 8. We use the Adam optimizer (Kingma and Ba 2015). We apply grid search for the learning rate and the weight decay parameter in the range of {1e-4, 5e-4,1e-3, 5e-3}. The trade-off parameter ξ for the MMD loss is searched in the range of {1,2,3,4,5}. For APPNP and GPRGNN, there is an additional teleport parameter α which is searched in the range of {0.1, 0.2, 0.5}. For ElasticGNN, there are two additional parameters λ_1 and λ_2 which are searched in the range of {3,6,9}. The experiments are conducted five times using different seeds, and we report the average performance in terms of Macro-F1 (Ma-F1) and Micro-F1 (Mi-F1) scores. The Macro-F1 score represents the unweighted average of the F1 scores for each class, treating all classes equally regardless of their sizes. In contrast, the Micro-F1 score is the weighted average of the F1 scores for each class, where the weights correspond to the proportion of instances of each class in the dataset.

Graph Domain Adaptation

We evaluate the GDA performance of UGNNs with CP and compare them with other baselines. We conduct six GDA tasks with citation networks, including A → C, A → D, C → A, C → D, D → A, D → C. The results on these citation networks are presented in Table 2. The best result is

	DE → EN		EN → DE	
	Ma-F1	Mi-F1	Ma-F1	Mi-F1
DeepWalk	46.54	52.18	49.97	55.08
Node2vec	46.96	52.64	50.10	54.61
GCN	54.55	54.77	51.04	62.03
GAT	49.50	54.84	40.08	43.65
GIN	49.91	52.39	44.26	55.26
UDAGCN	**58.19**	**59.74**	56.35	58.69
ACDNE	56.31	58.08	57.92	58.79
ASN	51.21	55.45	45.90	60.45
AdaGCN	35.30	54.56	31.18	40.22
GRADE	56.38	56.40	56.83	61.18
SpecReg	50.30	56.43	46.13	61.45
A2GNN	56.57	57.26	60.32	62.44
APPNP$_{CP}$	55.98	56.74	**60.48**	**62.76**
GPRGNN$_{CP}$	54.77	56.12	59.30	61.48
ElasticNet$_{CP}$	**57.69**	**59.87**	**61.39**	**63.08**

Table 3: Unsupervised GDA on social networks.

highlighted in red and the second best result is highlighted in blue. The standard deviations of the results are presented in the Appendix. As shown in Table 2, UGNNs with CP achieve the best results on all GDA tasks. On average, compared with the best result of the baselines on each GDA task, UGNNs with CP provide a 0.74% improvement in the Macro-F1 score and a 0.92% improvement in the Micro-F1 score. Individually, each UGNN with CP also outperforms the best baseline on average. Specifically, compared with the baseline that achieves the best average result, APPNP$_{CP}$ provides a 0.62% improvement in the Macro-F1 score and a 0.72% improvement in the Micro-F1 score; GPRGNN$_{CP}$ provides a 0.86% improvement in the Macro-F1 score and a 0.94% improvement in the Micro-F1 score; ElasticGNN$_{CP}$ provides a 1.34% improvement in the Macro-F1 score and a

Method	D → A		C → A		A → D		C → D		A → C		D → C	
	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1
APPNP	63.46	67.92	72.15	72.54	63.91	68.47	74.20	76.77	76.30	79.05	71.87	75.38
+MMD	72.20	72.07	73.96	73.92	73.92	76.90	75.41	77.53	80.36	82.19	76.61	78.68
+CP	73.28	73.30	75.11	74.48	**75.71**	**77.80**	**75.91**	**77.77**	**81.06**	**82.82**	78.29	**80.53**
GPRGNN	61.44	68.58	73.52	73.47	63.08	67.43	73.63	76.93	77.71	80.57	73.10	77.21
+MMD	70.92	72.27	74.89	74.14	74.30	76.64	74.18	77.26	80.14	82.69	77.80	79.95
+CP	**74.78**	**74.50**	**75.52**	**74.71**	75.15	**77.42**	75.04	77.55	80.95	**82.83**	**79.39**	**81.02**
ElasticGNN	65.51	68.82	71.83	72.05	62.94	66.22	72.10	74.31	75.33	77.75	74.22	76.97
+MMD	74.49	73.55	73.16	72.63	73.84	75.80	74.72	76.29	80.30	81.57	77.54	79.10
+CP	**75.74**	**74.66**	**76.72**	**75.70**	**75.49**	77.38	**75.78**	**77.64**	**81.16**	82.56	**78.44**	80.09

Table 4: Performance contribution of MMD and CP for UGNNs.

0.94% improvement in the Micro-F1 score.

We also draw similar conclusions regarding social networks. Specifically, we conduct two GDA tasks with social networks: DE → EN and EN → DE. The node classification performance under GDA is presented in Table 3. On average, compared with the best result of the baselines on each GDA task, ElasticGNN$_{CP}$ provides a 0.28% improvement in the Macro-F1 score and a 0.38% improvement in the Micro-F1 score. Notably, although UDAGCN performs well on the DE → EN task, ElasticGNN$_{CP}$ significantly outperforms it on the EN → DE task. Specifically, ElasticGNN$_{CP}$ outperforms UDAGCN on the EN → DE task by 5.04% in the Macro-F1 score and by 4.39% in the Micro-F1 score. In conclusion, UGNNs with CP can achieve better or comparable results compared with state-of-the-art GDA methods.

Ablation Study

In this section, we conduct ablation studies to investigate the role of different components in UGNNs with CP. We provide results for APPNP, GPRGNN, and ElasticGNN in Table 4. We evaluate the performance of vanilla UGNNs, UGNNs with MMD (denoted as +MMD), and UGNNs with CP and MMD (denoted as +CP). The results demonstrate that vanilla UGNNs do not perform well on the target domain. By incorporating the MMD alignment loss, the performance of UGNNs significantly improves, indicating the importance of reducing domain discrepancy. Finally, the results of UGNNs with CP demonstrate that the CP strategy can further improve model performance.

In UGNNs with CP, we apply MMD as an additional loss, controlled by the trade-off parameter ξ. In the following, we investigate the influence of ξ on the performance of UGNNs with CP. Specifically, we use the ElasticGNN$_{CP}$ as an example, as it performs the best in GDA according to the results in Table 2 and Table 3. We conduct experiments on the GDA task A → C, varying the trade-off parameter ξ from 0 to 5. The result is visualized in Figure 1. The results show that when $\xi = 0$, the model does not perform well, highlighting the importance of minimizing domain discrepancy. The model with MMD consistently outperforms the model without MMD for ξ in the range of [1,5]. Ablation study results on other tasks are provided in Appendix.
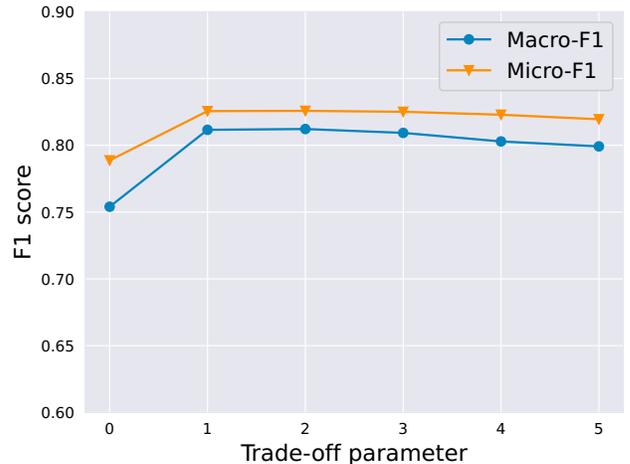

Figure 1: The influence of ξ on A → C task.

Conclusion

In this paper, we reveal through empirical and theoretical analyses that the lower-level objective function value associated with UGNNs significantly increases when transferring from the source domain to the target domain. To address this, we propose a simple yet effective cascaded propagation strategy to enhance the domain generalization ability of UGNNs, which is provably effective in decreasing the lower-level objective function value in the target domain. Experiments on five real-world datasets demonstrate the effectiveness of the cascaded propagation method.

The CP strategy is an architectural enhancement, and hence it can be used simultaneously with node representation distribution alignment methods. The MMD alignment loss used in the experiments does not consider the specific characteristics of graph-structured data. In future work, alignment methods designed especially for graph-structure data can potentially improve performance. Moreover, currently, the architectural enhancement (i.e., the proposed CP) and the alignment method (i.e., MMD) are developed independently; a more holistic design may lead to further performance improvement.

Acknowledgments

References

Ahn, H.; Yang, Y.; Gan, Q.; Wipf, D.; and Moon, T. 2022. Descent Steps of a Relation-Aware Energy Produce Heterogeneous Graph Neural Networks. *arXiv preprint arXiv:2206.11081*.

Cai, R.; Wu, F.; Li, Z.; Wei, P.; Yi, L.; and Zhang, K. 2024. Graph domain adaptation: A generative view. *ACM Transactions on Knowledge Discovery from Data*, 18(3): 1–24.

Chen, Q.; Wang, Y.; Wang, Y.; Yang, J.; and Lin, Z. 2022. Optimization-induced graph implicit nonlinear diffusion. In *International Conference on Machine Learning*, 3648–3661. PMLR.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

Dai, Q.; Wu, X.-M.; Xiao, J.; Shen, X.; and Wang, D. 2022. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(5): 4908–4922.

Feng, R.; Hou, Z.; Derr, T.; and Liu, X. 2023. Robust Graph Neural Networks via Unbiased Aggregation. *arXiv preprint arXiv:2311.14934*.

Fu, G.; Zhao, P.; and Bian, Y. 2022. p-Laplacian Based Graph Neural Networks. In *Proceedings of the International Conference on Machine Learning*, 6878–6917. PMLR.

Ganin, Y.; and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, 1180–1189. PMLR.

Gasteiger, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1): 723–773.

Grover, A.; and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Guo, G.; Wang, C.; Yan, B.; Lou, Y.; Feng, H.; Zhu, J.; Chen, J.; He, F.; and Philip, S. Y. 2023. Learning adaptive node embeddings across graphs. *IEEE Transactions on Knowledge and Data Engineering*, 35(6): 6028–6042.

Guo, K.; Wen, H.; Jin, W.; Guo, Y.; Tang, J.; and Chang, Y. 2024. Investigating Out-of-Distribution Generalization of GNNs: An Architecture Perspective. *arXiv preprint arXiv:2402.08228*.

Han, H.; Liu, X.; Mao, H.; Torkamani, M.; Shi, F.; Lee, V.; and Tang, J. 2023. Alternately optimized graph neural networks. In *International Conference on Machine Learning*, 12411–12429. PMLR.

Jiang, Z.; Han, X.; Fan, C.; Liu, Z.; Zou, N.; Mostafavi, A.; and Hu, X. 2024. Chasing Fairness in Graphs: A GNN Architecture Perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 21214–21222.

Kingma, D.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

Li, B.; Shen, Y.; Yang, J.; Wang, Y.; Ren, J.; Che, T.; Zhang, J.; and Liu, Z. 2023. Sparse Mixture-of-Experts are Domain Generalizable Learners. In *Conference on Parsimony and Learning (Recent Spotlight Track)*.

Liu, M.; Fang, Z.; Zhang, Z.; Gu, M.; Zhou, S.; Wang, X.; and Bu, J. 2024. Rethinking Propagation for Unsupervised Graph Domain Adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 13963–13971.

Liu, S.; Li, T.; Feng, Y.; Tran, N.; Zhao, H.; Qiu, Q.; and Li, P. 2023. Structural re-weighting improves graph domain adaptation. In *International Conference on Machine Learning*, 21778–21793. PMLR.

Liu, X.; Jin, W.; Ma, Y.; Li, Y.; Liu, H.; Wang, Y.; Yan, M.; and Tang, J. 2021. Elastic graph neural networks. In *Proceedings of the International Conference on Machine Learning*, 6837–6849. PMLR.

Ma, Y.; Liu, X.; Zhao, T.; Liu, Y.; Tang, J.; and Shah, N. 2021. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 1202–1211.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

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

Rozemberczki, B.; and Sarkar, R. 2021. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *arXiv preprint arXiv:2101.03091*.

Shen, X.; Dai, Q.; Chung, F.-l.; Lu, W.; and Choi, K.-S. 2020. Adversarial deep network embedding for cross-network node classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2991–2999.

Shen, X.; Dai, Q.; Mao, S.; Chung, F.-l.; and Choi, K.-S. 2021. Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5): 1935–1948.

Shi, B.; Wang, Y.; Guo, F.; Shao, J.; Shen, H.; and Cheng, X. 2023. Improving graph domain adaptation with network hierarchy. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2249–2258.

Shi, B.; Wang, Y.; Guo, F.; Xu, B.; Shen, H.; and Cheng, X. 2024. Graph Domain Adaptation: Challenges, Progress and Prospects. *arXiv preprint arXiv:2402.00904*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, M.; and Deng, W. 2018. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153.

Wang, Q.; Pang, G.; Salehi, M.; Buntine, W.; and Leckie, C. 2023a. Cross-domain graph anomaly detection via anomaly-aware contrastive alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 4676–4684.

Wang, Y.; Hu, X.; Gan, Q.; Huang, X.; Qiu, X.; and Wipf, D. 2023b. Efficient Link Prediction via GNN Layers Induced by Negative Sampling. *arXiv preprint arXiv:2310.09516*.

Wu, J.; He, J.; and Ainsworth, E. 2023. Non-IID transfer learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 10342–10350.

Wu, L.; Cui, P.; Pei, J.; and Zhao, L. 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore.

Wu, M.; and Pan, S. 2020. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, 1457–1467.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Xue, R.; Han, H.; Torkamani, M.; Pei, J.; and Liu, X. 2023. Lazygnn: Large-scale graph neural networks via lazy propagation. In *International Conference on Machine Learning*, 38926–38937. PMLR.

Yang, Y.; Liu, T.; Wang, Y.; Zhou, J.; Gan, Q.; Wei, Z.; Zhang, Z.; Huang, Z.; and Wipf, D. 2021. Graph neural networks inspired by classical iterative algorithms. In *Proceedings of the International Conference on Machine Learning*, 11773–11783. PMLR.

You, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2023. Graph domain adaptation via theory-grounded spectral regularization. In *The eleventh international conference on learning representations*.

Zhang, X.; Du, Y.; Xie, R.; and Wang, C. 2021. Adversarial separation network for cross-network node classification. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 2618–2626.

Zhang, Z.; Lu, S.; Huang, Z.; and Zhao, Z. 2022. ASGNN: Graph Neural Networks with Adaptive Structure. *arXiv preprint arXiv:2210.01002*.

Zhang, Z.; and Zhao, Z. 2022. Towards Understanding Graph Neural Networks: An Algorithm Unrolling Perspective. *arXiv preprint arXiv:2206.04471*.

Zheng, A. Y.; He, T.; Qiu, Y.; Wang, M.; and Wipf, D. 2024. BloomGML: Graph Machine Learning through the Lens of Bilevel Optimization. *arXiv preprint arXiv:2403.04763*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81.

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

Zhu, Q.; Jiao, Y.; Ponomareva, N.; Han, J.; and Perozzi, B. 2023. Explaining and adapting graph conditional shift. *arXiv preprint arXiv:2306.03256*.