Structural Re-weighting Improves Graph Domain Adaptation

Shikun Liu [1] **Tianchun Li** [2] **Yongbin Feng** [3] **Nhan Tran** [3] **Han Zhao** [4] **Qiu Qiang** [2] **Pan Li** [1]

Abstract

In many real-world applications, graph-structured data used for training and testing have differences in distribution, such as in high energy physics (HEP) where simulation data used for training may not match real experiments. Graph domain adaptation (GDA) is a method used to address these differences. However, current GDA primarily works by aligning the distributions of node representations output by a single graph neural network encoder shared across the training and testing domains, which may often yield sub-optimal solutions. This work examines different impacts of distribution shifts caused by either graph structure or node attributes and identifies a new type of shift, named conditional structure shift (CSS), which current GDA approaches are provably sub-optimal to deal with. A novel approach, called structural reweighting (StruRW), is proposed to address this issue and is tested on synthetic graphs, four benchmark datasets, and a new application in HEP. StruRW has shown significant performance improvement over the baselines in the settings with large graph structure shifts, and reasonable performance improvement when node attribute shift dominates. [1]

1. Introduction

Graph neural networks (GNNs) have recently become the de facto tool to learn the representations of graph-structured data (Scarselli et al., 2008; Kipf & Welling, 2017). Despite their exceptional performance on benchmarks (Hu et al., 2020a; 2021), GNNs have been found to struggle in high-stakes real-world applications where there is a data-distribution shift between the training and test phases (Li et al., 2022b; Hu et al., 2020b; Gaudelet et al., 2021).

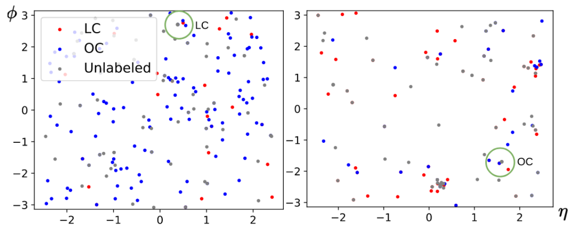

Figure 1: Examples of pileup events in two different PU levels: PU30 (Left) and PU10 (Right). Charged particles can be labeled as LC or OC, as highlighted by red or blue dots, while the labels of neutral particles are often unknown. Graph methods are often used here (Bertolini et al., 2014; Li et al., 2022c), where KNN graphs can be built and one can leverage nearby particle features and labels to make inference, highlighted by e.g. the two circles in the figure. In either of the two circles, there are two 2 labeled OC's and 1 labeled LC, while the ground-truth label of the neutral particle in the center may be different from each other.

This study is motivated by applications in high energy physics (HEP) (Shlomi et al., 2020), where GNNs are often trained on simulated data with an abundance of labels and then applied to real experiments with limited labels (Nachman & Shimmin, 2019). However, real experiments have complex, time-varying environments that may differ from simulated setups. One such example is the change in pile-up (PU) levels in Large Hadron Collider (LHC) experiments (Highfield, 2008). PU level refers to the number of collisions around the main collision of interest, which can change over time and differ from the levels used to generate simulation data. Modeling the data using graphs, the connection patterns between particles in different PU levels will significantly change, as depicted in Fig. 1. This poses a major challenge for GNNs to distinguish particles from the leading collision (class LC) from those from other collisions (class OC), which is a crucial task in HEP data analysis (Perloff et al., 2012; Li et al., 2022c). Similar shifts also occur in social and biological networks, where the interaction patterns between nodes with different labels can change over time (Wang et al., 2021a) or across different species (Cho et al., 2016), as listed in Table 1.

[1]Department of Electrical and Computer Engineering, Georgia Institute of Technology, Georgia, U.S.A [2]Department of Electrical and Computer Engineering, Purdue University, West Lafayette, U.S.A [3]Fermi National Accelerator Laboratory, Batavia, U.S.A [4]Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, U.S.A. Correspondence to: Shikun Liu <shikun.liu@gatech.edu>, Pan Li <panli@gatech.edu>.

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

[1]Our code is available at: `https://github.com/Graph-COM/StruRW`

Table 1: Conditional Structure Shift (CSS, computed according to Eq. (6)) across real datasets that are used for evaluation in this work. CSS will be explained in Sec. 4.1.

	DBLP and ACM		Cora		Arxiv		
Domains	$A \to D$	$D \to A$	Word	Degree	Time1	Time2	Degree
$\widehat{CSS}$	7.4276	7.4276	0.5583	0.9980	1.0106	1.2148	2.6131

Graph domain adaptation (GDA) has been proposed to deal with such distribution shift problems. Current GDA methods frequently utilize GNNs as a means of creating dense node representations, and then implement regularization in order to ensure these representations remain consistent across both the training (source) and test (target) domains (Wu et al., 2020; Xiao et al., 2022; Zhu et al., 2021a). However, this approach largely overlooks the distinct effects of distribution shifts caused by graph structures and node representations, and as a result, may not yield optimal solutions.

In this work, we investigate different types of distribution shifts of graph-structured data and offer significant understanding into GDA for node classification problems. First, we show that if the objective is to acquire node representations with distributions that remain invariant across domains, adding regularization to the last-layer node representations is adequate. Imposing regularization on intermediate node representations or matching node initial attributes across two domains may actually induce extra loss.

Though with the above observation, we further show that it is suboptimal in many cases to achieve such distribution invariance via a single stand-alone GNN encoder shared across domains. To illustrate the problem, we revisit the HEP example in Fig. 1: when the PU level is high (PU30), an unlabeled particle that is connected to one LC particle and two OC particles is more likely to be classified as LC. Conversely, in instances where the PU level is low (PU10), the particle with the same neighborhood may be more likely to be classified as LC due to the expectation of more OC particles in the vicinity of an OC particle. Under these scenarios, the optimal node representations with the same neighborhood should actually change to fit different domains rather than keep invariant. In this work, we formally define this new type of distribution shift as conditional structure shift (CSS). The CSS not only exists under the HEP setting but in other real applications, like social networks. For instance, different periods of time in citation networks may present different citation relations across fields due to the change of focus on interdisciplinary work or related work over time. We will discuss the detailed degree of CSS with other real datasets in Section 4.1. Current GDA methods fail to address CSS properly.

To deal with CSS, we propose a novel GDA algorithm named structural re-weighting (StruRW) as shown in Fig. 2. StruRW computes the edge probabilities between differ-

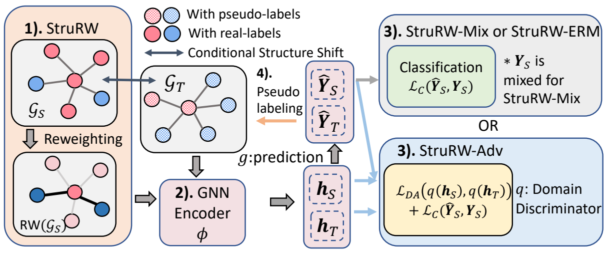
1). StruRW
$\mathcal{G}_S$
Reweighting
RW($\mathcal{G}_S$)
With pseudo-labels
With real-labels
Conditional Structure Shift
$\mathcal{G}_T$
2). GNN Encoder ϕ
4). Pseudo labeling
$\widehat{\boldsymbol{Y}}_S$
$\widehat{\boldsymbol{Y}}_T$
g:prediction
$\boldsymbol{h}_S$
$\boldsymbol{h}_T$
3). StruRW-Mix or StruRW-ERM
Classification $\mathcal{L}_C(\widehat{\boldsymbol{Y}}_S, \boldsymbol{Y}_S)$
$*\ \boldsymbol{Y}_S$ is mixed for StruRW-Mix
OR
3). StruRW-Adv
$\mathcal{L}_{DA}(q(\boldsymbol{h}_S), q(\boldsymbol{h}_T)) + \mathcal{L}_C(\widehat{\boldsymbol{Y}}_S, \boldsymbol{Y}_S)$
q: Domain Discriminator

Figure 2: This diagram demonstrates the model pipeline by combining the StruRW module, GNN encoder, then the generalized loss calculation block that supports StruRW-Adv, StruRW-Mix, and StruRW-ERM. The pseudo codes are presented in Algorithm 1.

ent classes based on the pseudo node labels estimated on the target graphs, and then uses these probabilities to guide bootstrapping of neighbors used in GNN computation on the source graphs, which eventually reduces the conditional shift of neighborhoods. The GNN composed with StruRW differentiates the encoding processes across the domains, which breaks the limitation.

We conduct extensive experiments on synthetic graphs, four real-world benchmarks and one HEP dataset to verify our theory and the effectiveness of StruRW. Across the cases, StruRW has achieved significant improvements over baselines under the settings with obvious graph structure shifts, and slight improvements for other settings dominated by node attribute shifts. Due to the page limitation, we leave the proofs of all propositions in this work in the appendix.

2. Preliminaries and Related Works

In this section, we introduce basic concepts and notations to set up the problem and review related works along the way.

Domain Adaptation (DA). This work studies unsupervised DA, where the model has access to labeled data from the source domain and unlabeled data from the target domain, and the goal is to train the model to achieve small classification error on the target domain.

To review the general idea of DA methods, denote $\mathbb{P}_X$ as the distribution of the feature $x \in \mathcal{X}$. We always use subscripts/superscripts $\mathcal{U} \in \{\mathcal{S}, \mathcal{T}\}$ to denote the source and target domains respectively. Denote $f_\mathcal{U}$ as the true labeling function that maps x to labels $y \in \mathcal{Y}$ for domain $\mathcal{U}$. For simplicity, we temporarily assume binary classification $\mathcal{Y} = \{0, 1\}$ to show theoretical insights, while the proposed algorithm can be applied to other cases. Suppose the model has a composition form $g \circ \phi$ that first maps the features to a latent space $\phi : \mathcal{X} \to \mathcal{H}$ and then performs classification $g : \mathcal{H} \to \mathcal{Y}$. Then, the classification error of the model in domain $\mathcal{U}$ can be denoted as $\epsilon_\mathcal{U}(g, \phi) = \mathbb{E}_{x \in \mathbb{P}_X^\mathcal{U}}[|g(\phi(x)) - f_\mathcal{U}(x)|]$. By adopting a derivation similar to (Wu et al., 2019; Ben-David et al., 2010), the error in the target domain can be bounded as follows. The detailed derivation is shown in Appendix A.

$$\epsilon_{\mathcal{T}}(g,\phi) \le \epsilon_{\mathcal{S}}(g,\phi) + \int_h d\mathbb{P}^{\mathcal{T}}_X(h)\left|f^{\phi}_{\mathcal{S}}(h) - f^{\phi}_{\mathcal{T}}(h)\right| + \int_h |d\mathbb{P}^{\mathcal{T}}_{\phi}(h) - d\mathbb{P}^{\mathcal{S}}_{\phi}(h) r_{\mathcal{S}}(h,\phi,g) \quad (1)$$

where $r_{\mathcal{U}}(h,\phi,g) \triangleq \int_{x:\phi(x)=h}|g(h) - f_{\mathcal{U}}(x)|d\mathbb{P}^{\mathcal{U}}_X(x)$, $f^{\phi}_{\mathcal{U}}(h) \triangleq \int_{x:\phi(x)=h} f_{\mathcal{U}}(x)d\mathbb{P}^{\mathcal{U}}_X(x)$ is the labeling function from the latent space, and $d\mathbb{P}^{\mathcal{U}}_{\phi}(h) = \int_{x:\phi(x)=h} d\mathbb{P}^{\mathcal{U}}_X(x)$.

To minimize the target error, one common way in DA is to *push the encoder ϕ to output representations with the distribution invariant across domains* by minimizing the third term while minimizing the source error, i.e., the first term. The second term is often overlooked as it is hard to control.

Previous methods to learn invariant representations adopt some regularization methods, including adversarial training with domain discriminator (Ganin et al., 2016; Zellinger et al., 2017), or minimizing some distribution-distance measures (Long et al., 2015) such as Maximum Mean Discrepancy (MMD) (Saito et al., 2018) between the source and target latent representations.

Graph Neural Networks (GNNs). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{x})$ denote an undirected graph with a node set $\mathcal{V}$, an edge set $\mathcal{E}$ and node attributes $\mathbf{x} = [\cdots x_v \cdots]_{v\in\mathcal{V}}$. The graph structure can also be denoted as the adjacency matrix $\mathbf{A}$ where its entry $A_{uv} = 1$ if edge $uv \in \mathcal{E}$ and otherwise $A_{uv} = 0$. GNNs encode $\mathbf{A}$ and $\mathbf{x}$ into node representations $\{h_v|v \in \mathcal{V}\}$. Initialize $h^{(0)}_v = x_v$ and standard GNNs (Hamilton et al., 2017; Gilmer et al., 2017) follow a message passing procedure. Specifically, for each node v and for $l = 0, 1, ..., L-1$,

$$h^{(l+1)}_v = \text{UDT}\,(h^{(l)}_v, \text{AGG}\,(\{\!\!\{h^{(l)}_u : u \in \mathcal{N}_v\}\!\!\})), \quad (2)$$

where $\mathcal{N}_v$ denotes the set of neighbors of node v and $\{\!\!\{\cdot\}\!\!\}$ denotes a multiset. The AGG function aggregates messages from the neighbors, and the UPT function updates the node representations. In *node classification* tasks, the last-layer node representation $h^{(L)}_v$ is used to predict the label $y_v \in \mathcal{Y}$.

Graph Domain Adaptation (GDA). GDA extends DA to the setting with graph-structured data. Specifically, we have one or several graphs $\mathcal{G}_{\mathcal{S}} = (\mathcal{V}_{\mathcal{S}}, \mathcal{E}_{\mathcal{S}}, \mathbf{x}_{\mathcal{S}})$ from the source domain with node labels $\mathbf{y}_{\mathcal{S}}$ and one or several graphs $\mathcal{G}_{\mathcal{T}} = (\mathcal{V}_{\mathcal{T}}, \mathcal{E}_{\mathcal{T}}, \mathbf{x}_{\mathcal{T}})$ from the target domain. The goal is to predict node labels $\mathbf{y}_{\mathcal{T}}$ in the target domain. Different from traditional DA with independent data points, features and labels are coupled due to the graph structure. Existing graph methods address the problem by first adopting a GNN to encode the graph into node representations $\mathbf{h}^{(L)} = [\cdots h^{(L)}_v \cdots]_{v\in\mathcal{V}}$, and then enforcing invariance on the representations in $\mathbf{h}^{(L)}$ across domains.

Related Works. For the related works with specific implementations of above GDA idea, DANE (Zhang et al., 2019) introduces adversarial training of domain classifier based on those node representations. UDAGCN (Wu et al., 2020) further imposes some inter-graph attention mechanism on top of the adversarial training. SR-GNN (Zhu et al., 2021a) aims to minimize the moment distance between the node-representation distributions across domains. DGDA (Cai et al., 2021) aims to disentangle semantic, domain, and noise variables and uses semantic variables that are better aligned with target graphs for prediction. All these works did not analyze the potential distribution shifts for node classification tasks and may therefore suffer from the CSS problem. A very recent work (You et al., 2023) proposes to use graph spectral regularization to address GDA problems. Although this work extends the generalization bound in (Zhao et al., 2019) for the case with the conditional shift in the scenario of GDA, their algorithm is not designed to address the issue of conditional shift.

In addition to GDA, many works aim to train GNNs for out-of-distribution (OOD) generalization. Different from GDA, they do not assume the availability of unlabeled test data and expect to train a GNN that learns representations invariant to generic domain change. Hence, they cannot address the problem in Fig. 1 as well. For node classification tasks, EERM (Wu et al., 2022b) minimizes the variance of representations across different generated environments. Ma et al. (2019) and Liu et al. (2020) extract invariant features by disentangling the entries of node representations. Verma et al. (2021); Wang et al. (2021b) mixup node representations across different classes for training to flatten the decision boundary (Zhang et al., 2018). Qiu et al. (2020); Wu et al. (2022a); Park et al. (2021); Liu et al. (2022); You et al. (2020) adopt data augmentation to achieve betteer generalization. Other works study OOD graph classification tasks and can be categorized similarly as above (Zhu et al., 2021b; Miao et al., 2022; Chen et al.; Li et al., 2022a; Han et al., 2022; Yang et al., 2022; Suresh et al., 2021).

Other Notations In the following, we use capital letters e.g., $\mathbf{X}, X$ to denote random variables (r.v.) and the lower-case letters, e.g., $\mathbf{x}, x$ to denote specific values, except the adjacency matrix $\mathbf{A}$ that will be used to denote both. Use $\mathbf{\Pi}$ to denote a permutation matrix with a proper dimension.

3. Optimality of Last-layer Domain Invariance

In this section, we disentangle the types of distribution shifts in graph-structured data and look into the question of whether regularizing only the last-layer node representations, as commonly adopted, is optimal to learn node representations invariant across domains under various types of shifts.

3.1. Distribution Shifts in Graph-structured Data

We categorize different types of distribution shifts in graph-structured data for node classification problems.

Structure shift. Consider the joint distribution of the adjacency matrix and node labels $\mathbb{P}_{\mathbf{A}\times\mathbf{Y}}$. Structure distribution has internal symmetry where $\mathbb{P}_{\mathbf{A}\times\mathbf{Y}}(\mathbf{A},\mathbf{y}) = \mathbb{P}_{\mathbf{A}\times\mathbf{Y}}(\mathbf{\Pi}\mathbf{A}\mathbf{\Pi}^\top,\mathbf{y})$ for any $\mathbf{\Pi}$ s.t. $\mathbf{y}=\mathbf{\Pi}\mathbf{y}$. Structure shift is defined for the case when $\mathbb{P}^{\mathcal{S}}_{\mathbf{A}\times\mathbf{Y}} \neq \mathbb{P}^{\mathcal{T}}_{\mathbf{A}\times\mathbf{Y}}$.

Attribute shift. We assume that without the graph structure, the attributes x_v, $v\in\mathcal{V}$ are IID sampled from $\mathbb{P}_{X|Y}$ given node labels y_v. Therefore, the conditional distribution of $\mathbf{x}|\mathbf{y}$ satisfies $\mathbb{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \prod_{v\in\mathcal{V}}\mathbb{P}_{X|Y}(x_v|y_v)$, which satisfies $\mathbb{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \mathbb{P}_{\mathbf{X}|\mathbf{Y}}(\mathbf{\Pi}\mathbf{x}|\mathbf{y})$ for any $\mathbf{\Pi}$ such that $\mathbf{\Pi}\mathbf{y}=\mathbf{y}$. Then, Attribute shift refers to $\mathbb{P}^{\mathcal{S}}_{X|Y} \neq \mathbb{P}^{\mathcal{T}}_{X|Y}$.

We use the joint distribution to define structure shift while the conditional distribution to define attribute shift because it better aligns with practice: Graph structure captures the correlation between nodes including their labels while node attributes are often independent given their labels.

3.2. Analysis for GDA with Different Types of Shifts

Our analysis is built upon the error bound in Eq. (1) that reveals the goal of learning domain-invariant node representations while minimizing the error in the source domain $\epsilon_{\mathcal{S}}(g,\phi)$. For GDA, the GNN is denoted as ϕ to transform the graph into node representations $\mathbf{h}^{(L)} = \phi(\mathbf{x},\mathbf{A})$ and the downstream node classifier is g. Note that in GDA, the entries of $\mathbf{h}^{(L)}$ are not independent of each other. The common practice to deal with this issue is to use a sampling procedure to marginalize the joint distribution:

Definition 3.1 (Marginalization)**.** For domain $\mathcal{U}$, given node representations $\mathbf{h}^{(l)}$, marginalization is to uniformly sample one of them $h_v^{(l)}$. Denote the distribution of $h_v^{(k)}$ as $\mathbb{P}^{\mathcal{U}}_{\phi}$.

With marginalization, the goal of learning domain-invariant node representations for GDA can be reduced to

$$\min_{g,\phi}\ \epsilon_{\mathcal{S}}(g,\phi) \quad \text{s.t. } \mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}. \tag{3}$$

We break the GNN into two parts $\phi = \phi_{>l}\circ\phi_{\le l}$ where $\phi_{\le l}$ denotes the encoder of the first $l(<L)$ layers $\mathbf{h}^{(l)} = \phi_{\le l}(\mathbf{x},\mathbf{A})$ and $\mathbf{h}^{(L)} = \phi_{>l}(\mathbf{h}^{(l)},\mathbf{A})$. With some abuse of notation, let $\phi_{\le 0}$ denote the first-layer transformation of node attributes before passing them to the neighbors. We use $\mathbb{P}^{\mathcal{S}}_{\phi_{\le l}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le l}}$ to indicate that the distributions of the marginalization of $\mathbf{h}^{(l)}$ are invariant across domains.

Given these notations, our question reduces to whether imposing $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$ is optimal for Eq. (3) and whether imposing $\mathbb{P}^{\mathcal{S}}_{\phi_{\le l}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le l}}$ for some $l < L-1$ can be better. We consider two cases with or without structure shift by assuming there always exists of attribute shift because otherwise structure shift can be transformed into a shift of node representations (similar to attribute shift).

Case I: Without structure shift. As we only have attribute shift in this case, an interesting question is whether aligning the distributions of node attributes can do better since the structure has no shift.

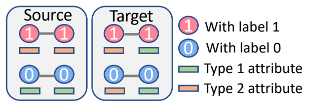

Figure 3: An example for $\mathbb{P}^{\mathcal{S}}_{\phi_{\le 0}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le 0}} \not\Rightarrow \mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$.

First, we argue that just aligning the distributions of node attributes $\mathbb{P}^{\mathcal{S}}_{\phi_{\le 0}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le 0}}$ is insufficient to achieve final invariance $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$ even without structure shift. This can be illustrated with an example shown in Fig. 3: The marginal distribution of node attributes are the same across the domains $\mathbb{P}^{\mathcal{S}}_{\phi_{\le 0}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le 0}}$ and there is no structure shift. However, after one layer of GNN, there will be a distribution shift in node representations.

Second, as shown in Proposition 3.2, aligning the conditional distributions of node attributes $\mathbb{P}^{\mathcal{S}}_{\phi_{\le 0}|Y} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le 0}|Y}$ may be sufficient under some independence assumption. This seems to give a chance to outperform previous methods that impose $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$ in the last layer.

Proposition 3.2. *Suppose the node attributes and the graph structures are independent given the node labels in the two domains* $\mathbb{P}^{\mathcal{U}}_{(\mathbf{X},\mathbf{A})|\mathbf{Y}}(\mathbf{X},\mathbf{A}|\mathbf{y}) = \mathbb{P}^{\mathcal{U}}_{\mathbf{X}|\mathbf{Y}}(\mathbf{X}|\mathbf{y})\mathbb{P}^{\mathcal{U}}_{\mathbf{A}|\mathbf{Y}}(\mathbf{A}|\mathbf{y})$*. If there is no structure shift* $\mathbb{P}^{\mathcal{S}}_{\mathbf{A},\mathbf{Y}}(\mathbf{A},\mathbf{y}) = \mathbb{P}^{\mathcal{T}}_{\mathbf{A},\mathbf{Y}}(\mathbf{A},\mathbf{y})$*, a transformation* $\phi_{\le 0}$ *of the node attributes that can achieve* $\mathbb{P}^{\mathcal{S}}_{\phi_{\le 0}|Y} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le 0}|Y}$ *is sufficient to make the distributions of last-layer node representations invariant across domains, i.e.,* $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$ *without the need of further regularization.*

However, we hardly see such improvement in practice because it is challenging to align such conditional distributions since the target labels $\mathbf{Y}^{\mathcal{T}}$ are unknown. More advanced approaches are often needed, which we will review in Sec. 4.1. Given such, keeping regularization in the last layer is often needed in practice to (approximately) achieve $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$.

Case II: With structure shift. With structure shift $\mathbb{P}^{\mathcal{S}}_{\mathbf{A}\times\mathbf{Y}} \neq \mathbb{P}^{\mathcal{T}}_{\mathbf{A}\times\mathbf{Y}}$, each layer of the GNN will induce distribution shift in node representations even if the distributions in the previous layer get aligned across domain, so regularization on the last-layer node representations is generally needed to achieve $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$. Then, the question in this case is that if extra regularizations for $\mathbb{P}^{\mathcal{S}}_{\phi_{\le l}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le l}}$, for $l < L-1$ are further helpful. Unfortunately, with a simple proof, as Prop. 3.3 shows, adding such regularizations will not improve objective Eq. (3), which thus cannot improve the bound of the error in the target domain (Eq. (1)).

Proposition 3.3. *Suppose regularization on the last-layer* *node representations is always adopted to achieve $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$. Then, adding regularization to the intermediate node representations $\mathbb{P}^{\mathcal{S}}_{\phi_{\leq l}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\leq l}}$, for $l < L-1$ cannot further reduce the optimal error indicated by the objective of Eq.* (3).

Combining Case I and Case II, we claim that optimizing the error bound Eq. (1) for the target domain by solving Eq. (3) is necessary and typically optimal to regularize only the last-layer node representations to make their distributions invariant across domains.

Although the above analysis justifies some rationale of previous GDA approaches, we observe its big limitation, that is we entirely ignore the second term in Eq. (1). As shown in Fig. 1, the ground-truth labeling functions in many real-world applications with graph-structured data may shift across domains. Ignoring such a shift yields suboptimal solutions. Our next section is to formalize the above issue and propose a principled algorithm to address it.

4. The Structural Re-weighting Algorithm

In this section, we first introduce the issue of conditional structure shift (CSS). Then, we propose our structural re-weighting algorithm StruRW to remove this shift for GDA. As a generic approach to align graph distributions for node classification tasks, StruRW can also improve the vanilla training of GNNs and approaches for OOD generalization such as Mixup (Wang et al., 2021b; Verma et al., 2021).

4.1. The Issue of Conditional Structure Shift

The conditional shift has been recently investigated in the setting without graph structure (Zhang et al., 2013; Zhao et al., 2019; Tachet des Combes et al., 2020). It describes the label-conditional distribution of features shifts across domains, which corresponds to Attribute Shift $\mathbb{P}^{\mathcal{S}}_{X|Y} \neq \mathbb{P}^{\mathcal{T}}_{X|Y}$ in our context as defined in Sec. 3.1. This problem can be addressed in principle only with some proper assumptions, e.g., the features in the target domain can be written as a location-scale transformation of the features in the source domain (Zhang et al., 2013; Gong et al., 2016). Recent works have also adopted adversarial training to align the estimated conditional distributions based on pseudo labels $\hat{\mathbf{Y}}^{\mathcal{T}}$ in the target domain (Long et al., 2018) or combined with instance-re-weight approaches (Tachet des Combes et al., 2020) to address both of the issues of conditional shift and label shift (i.e., $\mathbb{P}^{\mathcal{S}}_{Y} \neq \mathbb{P}^{\mathcal{T}}_{Y}$ by using our notation).

However, none of the previous works have considered conditional structural shift (CSS) for graph-structured data:

Definition 4.1 (Conditional Structure Shift). $\mathbb{P}^{\mathcal{S}}_{\mathbf{A}|\mathbf{Y}} \neq \mathbb{P}^{\mathcal{T}}_{\mathbf{A}|\mathbf{Y}}$, where $\mathbb{P}^{\mathcal{U}}_{\mathbf{A}|\mathbf{Y}}$ is a conditional distribution induced from $\mathbb{P}^{\mathcal{U}}_{\mathbf{A}\times\mathbf{Y}} = \mathbb{P}^{\mathcal{U}}_{\mathbf{A}|\mathbf{Y}}\mathbb{P}^{\mathcal{U}}_{\mathbf{Y}}$.

According to the definition, the structure shift defined in Sec. 3.1 may be caused by either CSS or label shift. Here, we study CSS, as it happens a lot in real-world graph data but cannot be addressed by simply extending previous methods. We leave its combination with label shift $\mathbb{P}^{\mathcal{S}}_{Y} \neq \mathbb{P}^{\mathcal{T}}_{Y}$ and attribute shift $\mathbb{P}^{\mathcal{S}}_{X|Y} \neq \mathbb{P}^{\mathcal{T}}_{X|Y}$ for the future studies.

We first use an example to show the sub-optimality of previous GDA methods as their goal of pursuing domain-invariant distributions of node representations. We are inspired by the observation in Fig. 1 and propose the following example with CSS based on the Contextual Stochastic Block Model (CSBM) (Deshpande et al., 2018).

Definition 4.2 (Contextual Stochastic Block Model)**.** CSBM is the model that combines the stochastic block model and node attributes for the random graph generation. CSBM with nodes from k classes is defined with parameters $(n, \mathbf{B}, \mathbb{P}_0, \dots, \mathbb{P}_{k-1})$. Here, n is the number of nodes. $\mathbf{B}$ is a $k \times k$ edge connection probability matrix. $\mathbb{P}_i$, $0 \leq i < k$, characterizes the distribution of node attributes of a node from class i. For any node u from class i and any node v from class j in a graph generated from the model, the probability of an edge connecting them is denoted by B_{ij}, an entry of $\mathbf{B}$. $\mathbf{B} = \mathbf{B}^\top$ for undirected graphs. For the CSBM, all node attributes and edges are generated independently given node labels.

Example 4.3. Suppose graphs in the source and target domains are generated from CSBM$(n, \mathbf{B}^{\mathcal{S}}, \mathbb{P}_0, \mathbb{P}_1)$ and CSBM$(n, \mathbf{B}^{\mathcal{T}}, \mathbb{P}_0, \mathbb{P}_1)$, respectively. Suppose either class in either model contains $n/2$ nodes. With some constants $p, r \in (0, 1/2)$ and $\delta \in [-p, p]/\{0\}$, for $i \in \{0, 1\}$, let $\mathbb{P}_i(X) = r$ if $X = i$ and $\mathbb{P}_i(X) = 1 - r$ if X is $\mathtt{M.V.}$ (denoting a default value other than 1 or 0), and

$$\mathbf{B}^{\mathcal{S}} = \begin{bmatrix} p & p \\ p & p-\delta \end{bmatrix}, \mathbf{B}^{\mathcal{T}} = \begin{bmatrix} p+\delta & p \\ p & p \end{bmatrix}, \quad (4)$$

So, there is no label shift or attribute shift but contains CSS. The nodes with attribute $\mathtt{M.V.}$ on the graphs generated from the above two CSBMs are used to formulate the training and test datasets, respectively.

Given this example, we can quantitatively show the suboptimality of using a single shared encoder ϕ to learn domain-invariant node representations in the following proposition.

Proposition 4.4. *One-layer GNNs are adopted to solve the GDA task in Example 4.3. By imposing $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$ through a GNN encoder ϕ shared across the two domains, the classification error in the target domain $\epsilon_{\mathcal{T}}(g, \phi) \geq$* 0.25*, while if without such a constraint, there exists a GNN encoder ϕ such that $\epsilon_{\mathcal{T}}(g, \phi) \to 0$ as $n \to \infty$.*

4.2. StruRW to Reduce Conditional Structure Shift

The previous example inspires our algorithm StruRW to address CSS for node classification tasks. Note that one layer

Algorithm 1 StruRW with different training pipelines

```
1: **Input** One or several source graphs $\mathcal{G}_\mathcal{S}$ with node labels $\mathbf{Y}_\mathcal{S}$; One or several target graphs $\mathcal{G}_\mathcal{T}$; A GNN $\phi$, a domain discriminator $q$, and a classifier $g$; The total epoch number $n$, the epoch index $m$ to start StruRW, the epoch period $t$ for weight update and $\lambda$.
2: **while** epoch $< n$ or not converged **do**
3:     **if** epoch $\geq m$ **then**
4:         When epoch $\equiv m \,(\mathrm{mod}\, t)$, get target node representations $\mathbf{h}_\mathcal{T} = \phi(\mathcal{G}_\mathcal{T})$, and update estimation $\hat{\mathbf{B}}^\mathcal{T}$ with $\hat{\mathbf{Y}}_\mathcal{T} = g(\mathbf{h}_\mathcal{T})$ (Eq. (5))
5:         Add edge weights to $\mathcal{G}_\mathcal{S}$ according to $(1-\lambda)\mathbf{1}\mathbf{1}^\top + \lambda\hat{\mathbf{B}}^\mathcal{T}./\mathbf{B}^\mathcal{S}$
6:     **end if**
7:     Get $\mathbf{h}_\mathcal{S} = \phi(\mathcal{G}_\mathcal{S})$, $\hat{\mathbf{Y}}_\mathcal{S} = g(\mathbf{h}_\mathcal{S})$ in the source domain
8:     **Case 1: StruRW-Adv**
9:         Update $\phi, q$ via $\min_q \max_\phi \mathcal{L}_{\text{ADV}}(q(\mathbf{h}_\mathcal{S}), q(\mathbf{h}_\mathcal{T}))$
10:    **Case 2: StruRW-Mix**
11:        Get mixed-up predictions $\hat{\mathbf{Y}}_\mathcal{S}$ and labels $\mathbf{Y}_\mathcal{S}$
12:    **Case 3: StruRW-ERM**
13:        Nothing to do
14:    Update $\phi$ and $g$ as $\min_{\phi,g} \mathcal{L}_{\text{ERM}}(\hat{\mathbf{Y}}_\mathcal{S}, \mathbf{Y}_\mathcal{S})$,
15: **end while**
```

of message passing in a GNN (Eq. (2)) encodes the information of a tuple $(h_v^{(l)}, \Xi_{\mathcal{N}_v}^{(l)})$, where $\Xi_{\mathcal{N}_v}^{(l)} = \{\!\{h_u^{(l)} | u \in \mathcal{N}_v\}\!\}$ denotes the multiset of the representations of the neighbors. The graph structure here determines the cardinality of the multiset $\Xi_{\mathcal{N}_v}^{(l)}$ and the distribution of the elements in $\Xi_{\mathcal{N}_v}^{(l)}$. Our key idea is to down-sample or re-sample the elements in such multisets (i.e., bootstrapping) from the source domain so that the distribution of such multi-sets can (approximately) match that in the target domain.

Specifically, consider the first layer of a GNN ϕ that runs on graphs sampled from k-class CSBM$(n, \mathbf{B}^\mathcal{U}, \mathbb{P}_0, ..., \mathbb{P}_{k-1})$ for domain $\mathcal{U} \in \{\mathcal{S}, \mathcal{T}\}$. Here, $\mathbf{B}^\mathcal{S} \neq \mathbf{B}^\mathcal{T}$, which indicates that there exists a CSS comparing a class-i node v in the target domain and a class-i node v' in the source domain. In the multiset $\Xi_{\mathcal{N}_v}^{(0)}$ (or $\Xi_{\mathcal{N}_{v'}}^{(0)}$), there will be in expectation $nB_{ij}^\mathcal{T}$ (or $nB_{ij}^\mathcal{S}$ resp.) many node attributes sampled from $\mathbb{P}_j$ for $j \in [k]$. Therefore, to align the cardinality and the distribution of elements of the multiset $\Xi_{\mathcal{N}_{v'}}^{(0)}$ with those of $\Xi_{\mathcal{N}_v}^{(0)}$, we propose to resample (if $B_{ij}^\mathcal{T} > B_{ij}^\mathcal{S}$) or downsample ($B_{ij}^\mathcal{T} < B_{ij}^\mathcal{S}$) the elements of the class-j neighbors of v' to $nB_{ij}^\mathcal{T}$ many. The following-up layers adopt the same sampling strategy.

In practice, GNNs often adopt sum/mean pooling (also in our experiments) to aggregate these multisets. Then, the above sampling strategy reduces to adding a weight for each element in the source domain during message aggregation. The weight is $B_{ij}^\mathcal{T}/B_{ij}^\mathcal{S}$ for the element passed from a class-j node to a class-i node. For other aggregation methods, a similar type of analysis can be adopted to determine the weights. To compute such weights, $\mathbf{B}^\mathcal{S}$ can be estimated based on Eq. (5) by using the node labels in the source domain. To estimate $\mathbf{B}^\mathcal{T}$, we propose to use the pseudo labels estimated by the model during the training process, i.e., using $(\hat{y}_u, \hat{y}_v)$ instead of (y_u, y_v) in Eq. (5).

$$B_{ij} = \frac{|\{e_{uv} \in \mathcal{E} | y_u = i, y_v = j\}|}{|\{v \in \mathcal{V} | y_v = i\}| \times |\{v \in \mathcal{V} | y_v = j\}|}. \tag{5}$$

As the edge weights are based on the estimation of pseudo labels in practice that may have errors, we introduce a hyperparameter λ to control the degree of reliance on this weight, i.e., the weight to be used in practice follows $(1-\lambda) + \lambda * B_{ij}^\mathcal{T}/B_{ij}^\mathcal{S}$.

Furthermore, to better understand model performance in practice, we would like to quantify the degree of CSS in each real dataset to help better understand the model performance. The metric we developed is as follows:

$$\widehat{\text{CSS}} = \frac{1}{k*k}\sum_{i,j} \Delta B_{ij}, \text{ where} \tag{6}$$

$$\Delta B_{ij} = \frac{1}{2}\left(\frac{|B_{ij}^\mathcal{S} - B_{ij}^\mathcal{T}|}{B_{ij}^\mathcal{S}} + \frac{|B_{ij}^\mathcal{S} - B_{ij}^\mathcal{T}|}{B_{ij}^\mathcal{T}}\right). \tag{7}$$

where k is the number of classes. This metric measures the relative level of difference between the edge connection probability matrix, which reflects the degree of CSS. There is no CSS when the metric is equal to 0. We calculate the degree of CSS for each real dataset we use for experiments in Table 1.

Lastly, we should note that the above analysis has limitations. First, we did not consider attribute shift. Attribute shift, if exists, can often be (approximately) addressed by traditional DA approaches to handle conditional shift for non-graph data (Long et al., 2018; Tachet des Combes et al., 2020). In our experiments, we have not tried these more advanced approaches but our methods have already outperformed the baselines. Second, the above analysis is based on CSBM, so the derived weights are shared across the edges when the pairs of the labels of the two end nodes are the same. We believe this constraint can be further relaxed and improved.

4.3. StruRW Combined with Different Approaches

StruRW is a generic approach to reduce CSS and should be widely applicable. Therefore, we combine StruRW with three different GNN training pipelines, including StruRW-Adv with adversarial-based training (Ganin et al., 2016), StruRW-Mix with mixup training on graphs (Wang et al., 2021b) and StruRW-ERM with vanilla GNN training. These different combinations can be viewed as options that handle the attribute shift and CSS at different levels that vary across applications. For instance, StruRW-ERM or StruRW-Mix often performs well if there is no or only small attribute shift, respectively, while StruRW-Adv will perform better with larger attribute shifts.

The algorithm is summarized in Algorithm 1, where StruRW is a separate module before the GNN encodes the data, which is compatible with different training pipelines. After m training epochs, StruRW calculates the edge weights for the source graphs to reduce CSS (lines 3-6). Different training pipelines may have different training losses. Besides the traditional empirical risk minimization (ERM) loss (via $\min_{\phi,g}\mathcal{L}_{\text{ERM}}$ in Eq. (8)) in line 14, StruRW-Adv follows DANN (Ganin et al., 2016) that trains the GNN ϕ and a domain discriminator q (via $\max_\phi \min_q \mathcal{L}_{\text{ADV}}$ in Eq. (9)) in line 9. Adversarial training comes into play where q tries to correctly identify the source and target samples, while ϕ seeks to align the distributions of the source and target samples to confuse q.

$$\mathcal{L}_{\text{ERM}} \triangleq \sum_{u\in\mathcal{V}_\mathcal{S}} \text{cross-entropy}(y_v, g(h_v)) \tag{8}$$

$$\mathcal{L}_{\text{ADV}} \triangleq -(\sum_{u\in\mathcal{V}_\mathcal{S}} \log[q(h_u)] + \sum_{u\in\mathcal{V}_\mathcal{T}} \log[1-q(h_v)]) \tag{9}$$

where h_u, h_v are node from $\mathbf{h}_\mathcal{S}$ and $\mathbf{h}_\mathcal{T}$. StruRW-Mix also adopts the loss $\min_{\phi,g}\mathcal{L}_{\text{ERM}}$ while the output $\mathbf{h}_\mathcal{S}$ and label $\mathbf{Y}$ for loss calculation are the post-mixup features and labels. The details can be found in (Wang et al., 2021b).

5. Experiments

We evaluate StruRW with the combination with the three training pipelines introduced in Sec. 4.3 and compare them with existing GDA and Graph OOD baselines. The experiments are done on one synthetic dataset, one real dataset from the HEP scientific application, and four real-world benchmark networks under various types of distribution shifts. We will briefly introduce the datasets, baselines, and experiment settings. More details such as the statistics of the datasets and hyperparameter tuning can be found in Appendix E.

5.1. Datasets

CSBM is the synthetic dataset we use that consists of graphs generated from 3-class CSBMs. Each class in each graph contains 1000 nodes. We do not consider attribute shift but only structure shift to directly demonstrate the effectiveness of StruRW. The node attributes in three classes in both domains satisfy Gaussains $\mathbb{P}_0 = \mathcal{N}([-1,0], I), \mathbb{P}_1 = \mathcal{N}([1,0], I), \mathbb{P}_2 = \mathcal{N}([3,2], I)$. The intra-class edge probabilities are both 0.02 for the two domains. The inter-class edge probability (q in table 2) in the target domain is 0.002 while that in the source domain varies from 0.001 to 0.016.

DBLP and ACM are two paper citation networks obtained from DBLP and ACM respectively. Each node represents a paper, and each edge indicates a citation between two papers. The goal is to predict the research topic of a paper. Here, we train the GNN on one network and test it on the other, which is denoted by $D \to A$ or $A \to D$. The original networks are provided by ArnetMiner (Tang et al., 2008). We use the processed versions from (Wu et al., 2020).

Arxiv introduced in (Hu et al., 2020a) is another citation network between all Computer Science (CS) Arxiv papers from 40 classes on different subject areas. Attributes are the embeddings of words in titles and abstracts. The domain can be split based on either publication times or node degrees. For evaluation with different levels of publication time shift, we use papers published between 2018 to 2020 to test while using papers published in other time periods for training: **Time 1** is from 2005 to 2007 and **Time 2** is from 2011 to 2014. We follow (Gui et al., 2022) to partition the network into two domains based on node degrees.

Cora is the fourth citation network with 70 classes (Bojchevski & Günnemann, 2018). Two domain splits are considered, named **Word** and **Degree**. The **Word** split is based on the diversity of words of a paper and the **Degree** split is based on node degrees, where we follow (Gui et al., 2022).

Pileup Mitigation is a dataset to evaluate the approaches for a critical data processing step in HEP named pileup mitigation (Bertolini et al., 2014). Particles are generated by the proton-proton collisions in the Large Hadron Collider with primary collisions (LC) and nearby bunch crossings (OC). There are multiple graphs used for training and testing. Each graph corresponds to a beam of proton-proton collisions. The particles generated from the collisions give the nodes in the graph. We connect the particles with edges if they are close in the $\eta-\phi$ space as shown in Fig. 1. As mentioned in the introduction, the task is to identify whether a neutral particle is from LC or OC. The labels of charged particles are often known. In this application, the distribution shifts may come from two sources, the shift of the types of particle decay between $pp \to Z(\nu\nu)+$ and $pp \to gg$ (Martínez et al., 2019) generated from LC (mostly attribute shift with slightly structural shift), and the shift of pile-up (PU) conditions (mostly structural shift). PUk means the number of collisions in the beam other than LC is k, where our dataset includes the cases $k \in \{10, 30, 50, 140\}$.

5.2. Baselines and Settings

Baselines StruRW is combined with the training pipelines of adversarial training, mixup and ERM. Therefore, we choose the corresponding baselines DANN (Ganin et al., 2016), graph Mixup (Wang et al., 2021b) and the vanilla ERM with

Table 2: Synthetic CSBM results. The **bold** font and the underline indicate the first and second best model respectively, † indicates the significant improvement, where the mean-1*std of a method > the mean of its corresponding backbone model.

	$q = 0.016$	$q = 0.014$	$q = 0.012$	$q = 0.01$	$q = 0.006$	$q = 0.001$
ERM	36.52 ± 3.76	41.62 ± 5.92	48.66 ± 6.31	57.29 ± 5.28	89.72 ± 2.62	100 ± 0
DANN	64.25 ± 5.69	72.56 ± 8.54	79.63 ± 6.84	86.29 ± 8.14	96.88 ± 1.35	100 ± 0
CDAN	67.53 ± 4.98	75.38 ± 7.46	82.51 ± 6.95	89.73 ± 7.44	97.03 ± 1.09	100 ± 0
UDAGCN	51.98 ± 1.31	57.83 ± 3.05	59.74 ± 1.52	65.97 ± 1.66	98.25 ± 0.52	100 ± 0
EERM	57.36 ± 4.52	65.88 ± 3.09	70.12 ± 10.26	72.87 ± 13.70	95.01 ± 3.88	100 ± 0
MIXUP	62.54 ± 2.77	69.21 ± 2.03	74.92 ± 1.56	82.87 ± 3.45	96.89 ± 0.38	100 ± 0
STRURW-ERM	$85.24^{\dagger} \pm 1.63$	$87.92^{\dagger} \pm 1.77$	$90.26^{\dagger} \pm 1.05$	$93.84^{\dagger} \pm 0.98$	$98.28^{\dagger} \pm 0.14$	$\mathbf{100} \pm 0$
STRURW-ADV	$\underline{86.37}^{\dagger} \pm 3.92$	$\underline{89.22}^{\dagger} \pm 1.83$	$\underline{91.53}^{\dagger} \pm 2.41$	$\underline{94.08}^{\dagger} \pm 0.98$	$\mathbf{98.40}^{\dagger} \pm 0.34$	$\mathbf{100} \pm 0$
STRURW-MIX	$\mathbf{88.48}^{\dagger} \pm 1.93$	$\mathbf{89.76}^{\dagger} \pm 1.15$	$\mathbf{92.08}^{\dagger} \pm 1.13$	$\mathbf{94.26}^{\dagger} \pm 0.99$	$\underline{98.35}^{\dagger} \pm 0.23$	$\mathbf{100} \pm 0$

Table 3: Performance on real datasets. The **bold** font and underline indicate the first and second best model respectively, † indicates the significant improvement, where the mean-1*std of a method > the mean of its corresponding backbone model.

	DBLP AND ACM		CORA			ARXIV	
DOMAINS	$A \to D$	$D \to A$	WORD	DEGREE	TIME1	TIME2	DEGREE
ERM	62.48 ± 3.58	64.70 ± 1.18	64.35 ± 0.44	53.28 ± 0.38	28.08 ± 0.24	49.52 ± 0.22	57.41 ± 0.14
DANN	59.02 ± 7.79	65.77 ± 0.46	63.92 ± 0.70	49.61 ± 0.74	24.33 ± 1.19	48.67 ± 0.37	56.13 ± 0.18
CDAN	60.56 ± 4.38	64.35 ± 0.83	62.46 ± 0.94	52.50 ± 0.96	25.85 ± 1.15	49.22 ± 0.75	56.43 ± 0.45
UDAGCN	59.62 ± 2.86	64.74 ± 2.51	64.23 ± 2.19	58.37 ± 0.72	25.64 ± 3.04	48.84 ± 1.48	55.77 ± 0.83
EERM	40.88 ± 5.10	51.71 ± 5.07	67.43 ± 2.86	$\underline{58.63} \pm 1.12$	OOM	OOM	OOM
MIXUP	49.93 ± 0.89	63.36 ± 0.66	$\mathbf{67.73} \pm 0.38$	58.18 ± 0.52	28.04 ± 0.18	49.98 ± 0.34	$\underline{59.22} \pm 0.22$
STRURW-ERM	$\mathbf{70.19}^{\dagger} \pm 2.10$	65.07 ± 1.98	64.34 ± 0.43	$55.27^{\dagger} \pm 0.48$	$\mathbf{28.46}^{\dagger} \pm 0.18$	48.78 ± 0.40	$57.45^{\dagger} \pm 0.15$
STRURW-ADV	$\underline{66.56}^{\dagger} \pm 9.44$	$\mathbf{66.57}^{\dagger} \pm 0.42$	63.92 ± 0.75	$52.69^{\dagger} \pm 0.36$	24.35 ± 1.25	$\underline{49.01} \pm 0.38$	$56.36^{\dagger} \pm 0.22$
STRURW-MIX	50.42 ± 1.13	$\underline{66.33}^{\dagger} \pm 0.91$	$\mathbf{67.73} \pm 0.39$	$\mathbf{60.37}^{\dagger} \pm 0.39$	$\underline{28.28}^{\dagger} \pm 0.52$	$\mathbf{50.34}^{\dagger} \pm 0.31$	$\mathbf{59.99}^{\dagger} \pm 0.09$

GCN (Kipf & Welling, 2017) as the backbone for direct comparisons. We also adopt UDAGCN (Wu et al., 2020), EERM (Wu et al., 2022b) and CDAN (Long et al., 2018) with the same backbone for further comparisons. CDAN was proposed to handle the conditional shift and the label shift of the distributions of last-layer node representations. We choose GCN as most baselines use this backbone in their original literature.

Settings and Metric. By the definition of GDA, the graphs in the source domain are used for training, while the graphs in the target domain are used for validation and testing. Specifically, we use 20 percent of node labels in the target domain for validation, and the rest 80 percent are held out for testing. The estimation of $\hat{\mathbf{B}}^{\mathcal{T}}$ in the target domain for StruRW uses the ground-truth labels of the target validation nodes (as assumed to be known) and the pseudo labels for the hold-out target testing nodes. The final evaluation scores included in the tables are based on the accuracy score for the node classification tasks on the hold-out target testing nodes. The selection of the best model is based on the score on the target validation nodes. All results are summarized based on 5 times independent experiments.

5.3. Result Analysis

The experiment results over the synthetic datasets are in Table 2. As the performance of ERM shows, CSS may cause significant performance decay. All baseline methods can deal with CSS to some extent while still performing significantly worse than StruRW-based approaches. Also, the improvement of StruRW increases with how much CSS the data holds. Particularly, StruRW is able to boost the performance by more than 20% over the best baseline. The results match our expectations well since the synthetic datasets are precisely aligned with the motivation of StruRW.

Table 3 includes the results for four real-world citation datasets. For all the datasets, StruRW-ERM, StruRW-Adv, and StruRW-Mix outperform their corresponding baseline models ERM, DANN, and Mixup, respectively. Moreover, across all the datasets, one of StruRW-ERM, StruRW-Adv and StruRW-Mix achieves the best performance, and over six of the seven settings, StruRW based methods have achieved significant improvement, i.e., the differences in means greater than one times the std of our models. Note that it is hard to expect a significant improvement of StruRW in the GDA setting without much CSS, e.g., the setting of **Word** (**Cora**) whose distribution shift is mostly due to attribute shift. In comparison, in the settings of **Degree** (**Cora**) and **Degree** (**Arxiv**), and **DBLP and ACM**, the improvements based on reweighting are more significant. The results match our intuition and are supported by the quantitative CSS we calculated in Table 1. Over the datasets with larger CSS scores, StruRW demonstrates more significant improvement over the baselines. The StruRW-based methods performance largely relies on the corresponding baseline performances. StruRW-Adv tends to be less stable and works better when there is a large distribution shift. StruRW-ERM and StruRW-Mix are much more stable and

Table 4: HEP dataset with different PU conditions and Physical process. The **bold** font indicate the best model, † indicates the significant improvement, where the mean-1*std of a method $>$ the mean of its corresponding backbone model.

	PU Conditions				Physical processes	
Domains	PU30 $\to$ 10	PU10 $\to$ 30	PU140 $\to$ 50	PU50 $\to$ 140	$gg \to Z(\nu\nu)$	$Z(\nu\nu) \to gg$
ERM	69.83 ± 0.43	70.73 ± 0.46	68.70 ± 0.56	68.28 ± 0.65	63.09 ± 0.48	66.53 ± 1.04
DANN	70.14 ± 0.52	71.29 ± 0.58	69.01 ± 0.42	68.98 ± 0.63	63.15 ± 0.66	66.24 ± 0.97
StruRW-ERM	$71.35^{\dagger} \pm 0.76$	$71.95^{\dagger} \pm 0.24$	$69.43^{\dagger} \pm 0.65$	69.05 ± 0.36	63.55 ± 0.40	$\mathbf{67.73} \pm 0.93$
StruRW-Adv	$\mathbf{70.77}^{\dagger} \pm 0.52$	$\mathbf{71.96} \pm 0.73$	$\mathbf{69.88}^{\dagger} \pm 0.71$	$\mathbf{70.54} \pm 0.84$	$\mathbf{64.36}^{\dagger} \pm 0.58$	66.91 ± 0.67

have close performances when the distribution shift is small.

Finally, for the **HEP** datasets, we compare StruRW-ERM and StruRW-Adv with the corresponding baselines ERM and DANN. Note that the current pipelines of StruRW-Mix and Mixup are not suitable for this dataset as these HEP datasets contain multiple graphs for either training or testing since how to properly mix up node attributes across graphs needs a non-trivial design, which is left for future study. A similar issue comes with other baselines such as UDAGCN originally proposed for single graphs used for training and testing. Under the domain shift caused by different PU conditions, we have often observed significant improvements over the case adapting from the higher PU levels to lower PU levels, while when being trained on lower PU levels and tested on higher PU levels, there are some but marginal improvements. These results match previous findings in the studies on this HEP application with ML technique(Li et al., 2021; Komiske et al., 2017). We suspect the reason is that the model learned with low PU levels tends to be more robust to the distribution shift. StruRW-based methods also help with the cases with shifts in particle types, although the improvements are not significant.

Besides the difficulty of the physics task itself that causes marginal performance in absolute accuracy scores, we suspect two additional reasons that may diminish the StruRW performance for HEP datasets. The first reason is that this pileup mitigation task is a binary classification, which is often easier than multi-class classification tasks due to the simpler decision boundary. The second reason may come from the multi-graph training and testing procedure, where the average overweight calculations in StruRW technique can limit the model performance.

Hyperparameters. Besides the normal hyperparameter tuning including learning rate, model architecture, and epoch as some basic setups, our StruRW relies on three hyperparameters: the epoch m to start StruRW, the time period t to calculate weights, and the λ for the degree to adopt the reweighted message. A general rule to select λ is that if the original CSS is large, we may want to pay attention to the reweighted message more so as to alleviate the CSS. The hyperparameter study over λ is demonstrated in Fig. 4 under the settings with ACM $\to$ DBLP and DBLP $\to$ ACM. The specific values of these hyperparameters and some baselines hyperparameters are reported in Appendix E.

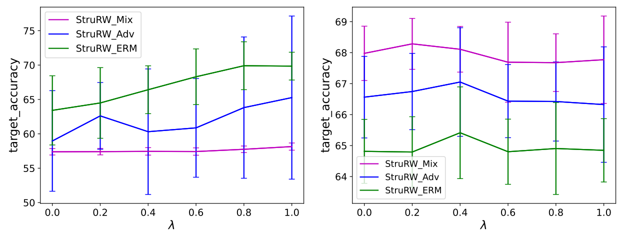

Figure 4: The hyperparameter study of λ for three StruRW models over the datasets ACM$\to$DBLP and DBLP$\to$ACM.

6. Conclusion

This work studies graph domain adaptation for node classification problems. We analyze the effects of different types of distribution shifts in graph-structured data. We have shown the advantages of the common solution to align last-layer node representations for GDA while disclosing the issues of using a shared GNN encoding pipeline to achieve so. We show that such a limitation can be caused by a newly identified type of distribution shift, named conditional structural shift, which widely shows up in practice. To reduce CSS in the data, we have proposed a new approach StruRW that asks to reweight the graphs in the source domain during GNN encoding. Extensive evaluation over synthetic graphs, real-world graphs, and the pileup-mitigation application in HEP has demonstrated the effectiveness of StruRW.

Acknowledgement

We greatly thank all the reviewers for their valuable feedback and thank Mia Liu for discussing relevant applications. S. Liu, T. Li, and P. Li are partially supported by NSF award OAC-2117997. Q.Qiu is partially supported by NIH. The work of HZ was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement Number: HR00112320012, a Facebook Research Award, and Amazon AWS Cloud Credit. YF and NT are supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the Department of Energy (DOE), Office of Science, Office of High Energy Physics and the DOE Early Career Research Program under Award No. DE-0000247070.

References

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Bertolini, D., Harris, P., Low, M., and Tran, N. Pileup per particle identification. *Journal of High Energy Physics*, 2014(10):1–22, 2014.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1ZdKJ-0W`.

Cai, R., Wu, F., Li, Z., Wei, P., Yi, L., and Zhang, K. Graph domain adaptation: A generative view. *arXiv preprint arXiv:2106.07482*, 2021.

Chen, Y., Zhang, Y., Bian, Y., Yang, H., KAILI, M., Xie, B., Liu, T., Han, B., and Cheng, J. Learning causally invariant representations for out-of-distribution generalization on graphs. In *Advances in Neural Information Processing Systems*.

Cho, H., Berger, B., and Peng, J. Compact integration of multi-network topology for functional analysis of genes. *Cell systems*, 3(6):540–548, 2016.

Deshpande, Y., Sen, S., Montanari, A., and Mossel, E. Contextual stochastic block models. *Advances in Neural Information Processing Systems*, 31, 2018.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B., Vickers, R., Roberts, C., Tang, J., et al. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6):bbab159, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., and Schölkopf, B. Domain adaptation with conditional transferable components. In *International conference on machine learning*, pp. 2839–2848. PMLR, 2016.

Gui, S., Li, X., Wang, L., and Ji, S. GOOD: A graph out-of-distribution benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=8hHg-zs_p-h`.

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

Han, X., Jiang, Z., Liu, N., and Hu, X. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, pp. 8230–8248. PMLR, 2022.

Highfield, R. Large hadron collider: Thirteen ways to change the world. *The Daily Telegraph. London. Retrieved*, pp. 10–10, 2008.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020a.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. Ogb-lsc: A large-scale challenge for machine learning on graphs. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Komiske, P. T., Metodiev, E. M., Nachman, B., and Schwartz, M. D. Pileup mitigation with machine learning (pumml). *Journal of High Energy Physics*, 2017(12): 1–20, 2017.

Li, H., Zhang, Z., Wang, X., and Zhu, W. Learning invariant graph representations for out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, 2022a.

Li, K., DeCost, B., Choudhary, K., Greenwood, M., and Hattrick-Simpers, J. A critical examination of robustness and generalizability of machine learning prediction of materials properties. *arXiv preprint arXiv:2210.13597*, 2022b.

Li, T., Liu, S., Feng, Y., Tran, N., Liu, M., and Li, P. Semi-supervised graph neural network for particle-level noise removal. In *NeurIPS 2021 AI for Science Workshop*, 2021. URL `https://openreview.net/forum?id=kTIngiqLU-X`.

Li, T., Liu, S., Feng, Y., Paspalaki, G., Tran, N., Liu, M., and Li, P. Semi-supervised graph neural networks for pileup noise removal. *The European Physics Journal C*, 2022c.

Liu, S., Ying, R., Dong, H., Li, L., Xu, T., Rong, Y., Zhao, P., Huang, J., and Wu, D. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pp. 14054–14072. PMLR, 2022.

Liu, Y., Wang, X., Wu, S., and Xiao, Z. Independence promoted graph disentangled networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4916–4923, 2020.

Long, M., Cao, Y., Wang, J., and Jordan, M. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pp. 97–105. PMLR, 2015.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31, 2018.

Ma, J., Cui, P., Kuang, K., Wang, X., and Zhu, W. Disentangled graph convolutional networks. In *International conference on machine learning*, pp. 4212–4221. PMLR, 2019.

Martínez, J. A., Cerri, O., Spiropulu, M., Vlimant, J., and Pierini, M. Pileup mitigation at the large hadron collider with graph neural networks. *The European Physical Journal Plus*, 134(7):333, 2019.

Miao, S., Liu, M., and Li, P. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pp. 15524–15543. PMLR, 2022.

Nachman, B. and Shimmin, C. Ai safety for high energy physics. *arXiv preprint arXiv:1910.08606*, 2019.

Park, H., Lee, S., Kim, S., Park, J., Jeong, J., Kim, K.-M., Ha, J.-W., and Kim, H. J. Metropolis-hastings data augmentation for graph neural networks. *Advances in Neural Information Processing Systems*, 34:19010–19020, 2021.

Perloff, A. et al. Pileup measurement and mitigation techniques in cms. In *Journal of Physics: Conference Series*, volume 404, pp. 012045. IOP Publishing, 2012.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2020.

Saito, K., Watanabe, K., Ushiku, Y., and Harada, T. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3723–3732, 2018.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Shlomi, J., Battaglia, P., and Vlimant, J.-R. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 15920–15933, 2021.

Tachet des Combes, R., Zhao, H., Wang, Y.-X., and Gordon, G. J. Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33:19276–19289, 2020.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, 2008.

Verma, V., Qu, M., Kawaguchi, K., Lamb, A., Bengio, Y., Kannala, J., and Tang, J. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10024–10032, 2021.

Wang, Y., Chang, Y.-Y., Liu, Y., Leskovec, J., and Li, P. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021a.

Wang, Y., Wang, W., Liang, Y., Cai, Y., and Hooi, B. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, pp. 3663–3674, 2021b.

Wu, L., Lin, H., Huang, Y., , and Li, S. Z. Knowledge distillation improves graph structure augmentation for graph neural networks. *Advances in Neural Information Processing Systems*, 2022a.

Wu, M., Pan, S., Zhou, C., Chang, X., and Zhu, X. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, pp. 1457–1467, 2020.

Wu, Q., Zhang, H., Yan, J., and Wipf, D. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=FQOC5u-1egI`.

Wu, Y., Winston, E., Kaushik, D., and Lipton, Z. Domain adaptation with asymmetrically-relaxed distribution alignment. In *International conference on machine learning*, pp. 6872–6881. PMLR, 2019.

Xiao, J., Dai, Q., Xie, X., Dou, Q., Kwok, K.-W., and Lam, J. Domain adaptive graph infomax via conditional adversarial networks. *IEEE Transactions on Network Science and Engineering*, 2022.

Yang, N., Zeng, K., Wu, Q., Jia, X., and Yan, J. Learning substructure invariance for out-of-distribution molecular representations. In *Advances in Neural Information Processing Systems*, 2022.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

You, Y., Chen, T., Wang, Z., and Shen, Y. Graph domain adaptation via theory-grounded spectral regularization. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=OysfLgrk8mk`.

Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., and Saminger-Platz, S. Central moment discrepancy (CMD) for domain-invariant representation learning. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SkB-_mcel`.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. Domain adaptation under target and conditional shift. In *International conference on machine learning*, pp. 819–827. PMLR, 2013.

Zhang, Y., Song, G., Du, L., Yang, S., and Jin, Y. Dane: Domain adaptive network embedding. In *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4362–4368, 2019.

Zhao, H., Des Combes, R. T., Zhang, K., and Gordon, G. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pp. 7523–7532. PMLR, 2019.

Zhu, Q., Ponomareva, N., Han, J., and Perozzi, B. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems*, 34:27965–27977, 2021a.

Zhu, Q., Yang, C., Xu, Y., Wang, H., Zhang, C., and Han, J. Transfer learning of graph neural networks with ego-graph information maximization. *Advances in Neural Information Processing Systems*, 34:1766–1779, 2021b.

A. Derivation of The Error Bound in The Target Domain Eq. (1)

We follow the derivation in (Wu et al., 2019). Let $f_{\mathcal{U}}^{\phi}(x) = \int_{x:\phi(x)=h} f_{\mathcal{U}}(x) d\mathbb{P}_X^{\mathcal{U}}(x)$. First, we have $r_{\mathcal{U}}(h,\phi,g) = \int_{x:\phi(x)=h} |g(h) - f_{\mathcal{U}}(x)| d\mathbb{P}_X^{\mathcal{U}}(x) = |g(h) - f_{\mathcal{U}}^{\phi}(h)|$ for $\mathcal{U} \in \{\mathcal{S},\mathcal{T}\}$. And thus,

$$r_{\mathcal{S}}(h,\phi,g) - r_{\mathcal{T}}(h,\phi,g) = |g(h) - f_{\mathcal{S}}^{\phi}(h)| - |g(h) - f_{\mathcal{T}}^{\phi}(h)| \le |f_{\mathcal{S}}^{\phi}(h) - f_{\mathcal{T}}^{\phi}(h)|. \tag{10}$$

Therefore, the target error can be bounded as

$$\begin{aligned}
\epsilon_{\mathcal{T}}(g,\phi) &= \epsilon_{\mathcal{T}}(g,\phi) + \epsilon_{\mathcal{S}}(g,\phi) - \epsilon_{\mathcal{S}}(g,\phi) && (11)\\
&= \epsilon_{\mathcal{S}}(g,\phi) + \int_x |g(\phi(x)) - f_{\mathcal{T}}(x)| d\mathbb{P}_X^{\mathcal{T}}(x) - \int_x |g(\phi(x)) - f_{\mathcal{S}}(x)| d\mathbb{P}_X^{\mathcal{S}}(x) && (12)\\
&= \epsilon_{\mathcal{S}}(g,\phi) + \int_h r_{\mathcal{T}}(h,\phi,g) d\mathbb{P}_\phi^{\mathcal{T}}(h) - \int_h r_{\mathcal{S}}(h,\phi,g) d\mathbb{P}_\phi^{\mathcal{S}}(h) && (13)\\
&= \epsilon_{\mathcal{S}}(g,\phi) + \int_h d\mathbb{P}_\phi^{\mathcal{T}}(h)(r_{\mathcal{T}}(h,\phi,g) - r_{\mathcal{S}}(h,\phi,g)) + \int_h (d\mathbb{P}_\phi^{\mathcal{T}}(h) - d\mathbb{P}_\phi^{\mathcal{S}}(h)) r_{\mathcal{S}}(h,\phi,g) && (14)\\
&\le \epsilon_{\mathcal{S}}(g,\phi) + \int_h d\mathbb{P}_\phi^{\mathcal{T}}(h)|r_{\mathcal{T}}(h,\phi,g) - r_{\mathcal{S}}(h,\phi,g)| + \int_h |d\mathbb{P}_\phi^{\mathcal{T}}(h) - d\mathbb{P}_\phi^{\mathcal{S}}(h)| r_{\mathcal{S}}(h,\phi,g) && (15)\\
&\overset{a)}{\le} \epsilon_{\mathcal{S}}(g,\phi) + \int_h d\mathbb{P}_\phi^{\mathcal{T}}(h)|f_{\mathcal{S}}^{\phi}(h) - f_{\mathcal{T}}^{\phi}(h)| + \int_h |d\mathbb{P}_\phi^{\mathcal{T}}(h) - d\mathbb{P}_\phi^{\mathcal{S}}(h)| r_{\mathcal{S}}(h,\phi,g) && (16)
\end{aligned}$$

where a) uses Eq. (10).

B. Proof for Proposition 3.2

Proof. The initial node attributes $x_v, v \in \mathcal{V}$ are independently sampled from the conditional distribution $\mathbb{P}_{X|Y}$ given the node labels y_v. No matter whether attribute shift $\mathbb{P}_{X|Y}^{\mathcal{S}} \neq \mathbb{P}_{X|Y}^{\mathcal{T}}$ exists, our condition is that the transformation $\phi_{\le 0}$ maps the attributes x_v to $h_v^{(0)}$ and satisfies $\mathbb{P}_{\phi_{\le 0}|Y}^{\mathcal{S}} = \mathbb{P}_{\phi_{\le 0}|Y}^{\mathcal{T}}$. The goal is to prove that if $\mathbb{P}_{\phi_{\le 0}|Y}^{\mathcal{S}} = \mathbb{P}_{\phi_{\le 0}|Y}^{\mathcal{T}}$, then the after the GNN, node representations will reach $\mathbb{P}_{\phi}^{\mathcal{S}} = \mathbb{P}_{\phi}^{\mathcal{T}}$.

Since the message-passing process at each GNN layer relies on the same adjacency matrix A for neighborhood aggregation, we can prove this by induction. However, it is hard to prove $\mathbb{P}_{\phi_{\le l}|Y}^{\mathcal{S}} = \mathbb{P}_{\phi_{\le l}|Y}^{\mathcal{T}} \Rightarrow \mathbb{P}_{\phi_{\le l+1}|Y}^{\mathcal{S}} = \mathbb{P}_{\phi_{\le l+1}|Y}^{\mathcal{T}}$ because $h_v^{(l)}$'s are not independent. So, we are to consider the joint distribution of $\{h_v^{(l)} | v \in \mathcal{V}\}$ and graph structure $\mathbf{A}$ given node labels, i.e., $\mathbb{P}_{\mathbf{H}^{(l)}\times\mathbf{A}|Y}^{\mathcal{S}}$, and prove $\mathbb{P}_{\mathbf{H}^{(l)}\times\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(l)}\times\mathbf{A}|Y}^{\mathcal{T}} \Rightarrow \mathbb{P}_{\mathbf{H}^{(l+1)}\times\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(l+1)}\times\mathbf{A}|Y}^{\mathcal{T}}$. If this is true, we have $\mathbb{P}_{\mathbf{H}^{(L)}\times\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(L)}\times\mathbf{A}|Y}^{\mathcal{T}}$. By integrating over $\mathbb{P}_{\mathbf{A}|Y}^{\mathcal{U}}$, we achieve $\mathbb{P}_{\mathbf{H}^{(L)}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(L)}|Y}^{\mathcal{T}}$.

First, when $l = 0$, since $\mathbb{P}_{\phi_{\le 0}|Y}^{\mathcal{S}} = \mathbb{P}_{\phi_{\le 0}|Y}^{\mathcal{T}}$ and all $h_v^{(0)}$'s are mutually independent. We have $\mathbb{P}_{\mathbf{H}^{(0)}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(0)}|\mathbf{Y}}^{\mathcal{T}}$. Also, since there is no structure shift $\mathbb{P}_{\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{A}|Y}^{\mathcal{T}}$ and $\mathbf{A}$ and $\mathbf{X}$ are independent given $\mathbf{Y}$, we have $\mathbb{P}_{\mathbf{H}^{(0)}\times\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(0)}\times\mathbf{A}|\mathbf{Y}}^{\mathcal{T}}$

For $l > 0$, consider the lth layer of GNN that takes $h_v^{(l)}, v \in \mathcal{V}$ and $\mathbf{A}$ as input and follows Eq. (2) as:

$$h_v^{(l+1)} = \text{UDT}(h_v^{(l)}, \text{AGG}(\{\{h_v^{(l)} : u \in \mathcal{N}_v\}\})). \tag{17}$$

which depends on $\mathbf{H}^{(l)}$ and $\mathbf{A}$. So, we have $\mathbb{P}_{\mathbf{H}^{(l+1)}\times\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(l+1)}|Y,\mathbf{A}}^{\mathcal{S}} \mathbb{P}_{\mathbf{A}|Y}^{\mathcal{S}} \overset{a)}{=} \mathbb{P}_{\mathbf{H}^{(l+1)}|Y,\mathbf{A}}^{\mathcal{T}} \mathbb{P}_{\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(l+1)}\times\mathbf{A}|Y}^{\mathcal{T}}$. where a) is due to the induction condition $\mathbb{P}_{\mathbf{H}^{(l)}\times\mathbf{A}|Y}^{\mathcal{S}} = \mathbb{P}_{\mathbf{H}^{(l)}\times\mathbf{A}|Y}^{\mathcal{T}}$, which concludes the proof.

□

C. Proof for Proposition 3.3

Proof. Actually, this proposition is easy to obtain from the perspective of optimization. Since the goal is always with the constraint $\mathbb{P}_{\phi}^{\mathcal{S}} = \mathbb{P}_{\phi}^{\mathcal{T}}$, adding an intermediate-layer regularization, say $\mathbb{P}_{\phi_{\le l}}^{\mathcal{S}} = \mathbb{P}_{\phi_{\le l}}^{\mathcal{T}}$, which makes the optimization problem (1) as

$$\min_{\phi_{>l},\phi_{\le l}} \epsilon_{\mathcal{S}}(\phi) \quad \text{s.t. } \mathbb{P}^{\mathcal{S}}_{\phi_{\le l}} = \mathbb{P}^{\mathcal{T}}_{\phi_{\le l}}, \mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi} \tag{18}$$

Comparing the objective function and constraints from Eq. (3) and Eq. (18), we find the same objective but with additional invariant representation constraints in the intermediate layer of GNN. As for both the constraints on the final layer of representations are imposed, additional constraints will only restrict the feasible region for GNN parameters to further reduce the source error. Therefore, Eq. (3) has an optimal solution no worse than Eq. (18) in terms of a lower source classification error, which ultimately determines the bound in Eq. (3). □

D. Proof for Proposition 4.4

Recall that node attributes in both domains follow:

$$\mathbb{P}_0(X) = \begin{cases} r & \text{if } X = 0 \\ 1-r & \text{if } X = \text{Missing Value (M.V.)} \end{cases}, \quad \mathbb{P}_1(X) = \begin{cases} r & \text{if } X = 1 \\ 1-r & \text{if } X = \text{Missing Value (M.V.)} \end{cases} \tag{19}$$

To classify a node v with M.V. as its attribute, if we use one-layer GNN, the classification essentially reduces to classify the multi-set Ξ_v of attributes from its neighbors. Let us analyze this multi-set.

This multi-set Ξ_v has the following equivalent representation: It contains at most $n-1$ elements that have values chosen from $\{0, 1, \text{M.V.}\}$. Therefore, Ξ_v can be represented as a 3-dim vector (c_0, c_1, c_2) where c_0, c_1, c_2 represent the multiplicity of each type of element $0, 1, \text{M.V.}$ in the multiset and satisfy $c_1 + c_2 + c_3 \le n-1$. Our analysis is based on analyzing $\mathbb{P}^{\mathcal{U}}(\Xi_v = (c_0, c_1, c_2)|Y_v)$ for $\mathcal{U} \in \{\mathcal{S}, \mathcal{T}\}$ and $Y_v \in \{0, 1\}$.

Case 1: Let us first prove that when we use a shared GNN encoder ϕ to impose $\mathbb{P}^{\mathcal{S}}_{\phi} = \mathbb{P}^{\mathcal{T}}_{\phi}$, $\epsilon_{\mathcal{T}}(g, \phi) \ge 0.25$.

Given a GNN model ϕ and the classifier g, we partition the feature space into the 0-space $\boldsymbol{\Xi}_0(g,\phi) = \{(c_1, c_2, c_3) : g \circ \phi(c_1, c_2, c_3) = 0, c_1 + c_2 + c_3 \le n-1, c_i \in \mathbb{Z}_{\ge 0}\}$ and the 1-space $\boldsymbol{\Xi}_1(g,\phi) = \{(c_1, c_2, c_3) : g \circ \phi(c_1, c_2, c_3) = 1, c_1 + c_2 + c_3 \le n-1, c_i \in \mathbb{Z}_{\ge 0}\}$.

Recall the CSBM models for source and target domains have structures:

$$\mathbf{B}^{\mathcal{S}} = \begin{bmatrix} p & p \\ p & p-\delta \end{bmatrix}, \mathbf{B}^{\mathcal{T}} = \begin{bmatrix} p+\delta & p \\ p & p \end{bmatrix}, \tag{20}$$

We know $\mathbb{P}^{\mathcal{S}}(\Xi_v|Y_v = 0) = \mathbb{P}^{\mathcal{T}}(\Xi_v|Y_v = 1)$ because in the source domain, if for v with $Y_v = 0$, the edge probability between v and any node with label 0 is p, and the edge probability between v and any node with label 1 is also p. In the target domain, if for v with $Y_v = 1$, the edge probability between v and any node with label 0 is p, and the edge probability between v and any node with label 1 is also p. Therefore, no matter what ϕ, g are chosen, $\mathbb{P}^{\mathcal{S}}[\boldsymbol{\Xi}_i(g,\phi)|Y = 0] = \mathbb{P}^{\mathcal{T}}[\boldsymbol{\Xi}_i(g,\phi)|Y = 1]$. Therefore,

$$1 = \mathbb{P}^{\mathcal{S}}[\boldsymbol{\Xi}_0(g,\phi)|Y=0] + \mathbb{P}^{\mathcal{S}}[\boldsymbol{\Xi}_1(g,\phi)|Y=0] = \mathbb{P}^{\mathcal{T}}[\boldsymbol{\Xi}_0(g,\phi)|Y=1] + \mathbb{P}^{\mathcal{S}}[\boldsymbol{\Xi}_1(g,\phi)|Y=0] \le 2(\epsilon_{\mathcal{T}}(g,\phi) + \epsilon_{\mathcal{S}}(g,\phi)).$$

The last inequality is because

$$\begin{aligned} \epsilon_{\mathcal{U}}(g,\phi) &= \mathbb{P}^{\mathcal{U}}[\boldsymbol{\Xi}_0(g,\phi)|Y=1]\mathbb{P}^{\mathcal{U}}[Y=1] + \mathbb{P}^{\mathcal{U}}[\boldsymbol{\Xi}_1(g,\phi)|Y=0]\mathbb{P}^{\mathcal{U}}[Y=0] \\ &\ge \frac{1}{2}\max\{\mathbb{P}^{\mathcal{U}}[\boldsymbol{\Xi}_0(g,\phi)|Y=1], \mathbb{P}^{\mathcal{U}}[\boldsymbol{\Xi}_1(g,\phi)|Y=0]\}. \end{aligned}$$

So, $\epsilon_{\mathcal{T}}(g,\phi) + \epsilon_{\mathcal{S}}(g,\phi) \ge 0.5$. It is also a reasonable assumption that $\epsilon_{\mathcal{T}}(g,\phi) \ge \epsilon_{\mathcal{S}}(g,\phi)$ in practice. So, we have $\epsilon_{\mathcal{T}}(g,\phi) \ge 0.25$.

Case 2: Case 1 implies that we should not impose domain invariant distributions via the GNN encoding process shared across domains. We may prove that if the GNN encoding process ϕ for the target domain can be chosen differently from that for the source domain, then there is a ϕ $\epsilon_{\mathcal{T}}(g,\phi) \to 0$ as $n \to \infty$. Here, we assume n is large enough and ignore the difference between n and $n-1$.

Given a node v with the multiset feature $\Xi_v = (c_0, c_1, c_2)$, suppose the GNN encoder ϕ follows $\phi(\Xi_v) = (c_0 - c_1)/n$.

Recall that we have the following two cases

- If v is from class 0 in the target domain, $c_1 \sim \text{Bin}(n/2, pr)$, $c_0 \sim \text{Bin}(n/2, (p+\delta)r)$
- If v is from class 1 in the target domain, $c_1 \sim \text{Bin}(n/2, pr)$, $c_0 \sim \text{Bin}(n/2, pr)$.

As c_1 and c_0 are always independent, if v is from class 0 in the target domain, $\phi(\Xi_v) = \frac{1}{n}(\sum_{i=1}^{n/2} Z_i - \sum_{i=1}^{n/2} Z_i')$, where $Z_i \sim \text{Bern}((p+\delta)r)$ and $Z_i' \sim \text{Bern}(pr)$, and all Z_i's and Z_i''s are independent. Here, $\text{Bern}(\cdot)$ is the Bernoulli distribution. Therefore, using Hoeffding's inequality, we have

$$\mathbb{P}\left(\phi(\Xi_v) - \mathbb{E}[\phi(\Xi_v)] < t\right) \le \exp(-\frac{nt^2}{2}) \tag{21}$$

If pick $t = \frac{\delta r}{4}$, $\mathbb{P}\left(\phi(\Xi_v) < pr + \frac{\delta r}{4}\right) \le \exp(-\frac{n\delta^2 r^2}{32})$. Similarly, if v is from class 0 in the target domain, we have $\mathbb{P}\left(\phi(\Xi_v) > pr + \frac{\delta r}{4}\right) \le \exp(-\frac{n\delta^2 r^2}{32})$.

Therefore, by setting the classifier as $g(h) = 0$ if $h > pr + \frac{\delta r}{4}$ or 1 if $h < pr + \frac{\delta r}{4}$. Then, the error rate in the target domain will be less than $2\exp(-\frac{n\delta^2 r^2}{32})$, which goes to 0 as n goes to ∞.

E. Supplement for Experiments

E.1. Datasets

E.1.1. Dataset Statistics for ACM, DBLP, Cora, Arxiv

Below is the summary of our real datasets with the number of nodes, number of edges, node feature dimension and number of class labels.

Table 5: real dataset statistics

	ACM	DBLP	CORA	ARXIV
#NODES	7410	5578	19793	169343
#EDGES	22270	14682	126842	2315598
NODE FEATURE DIMENSION	7537	7537	8710	128
#LABELS	6	6	70	40

E.1.2. Details for HEP datasets

Next, we detail some statistics and setup for the HEP datasets For our studies, simulated datasets have been generated of different physical processes under different pileup conditions. In this study, we select four pileup conditions where the numbers of other interactions (nPU) are 10, 30, 50, 140 respectively, and two hard scattering signal processes, $pp \to Z_{\nu\nu}+$ jets and $pp \to gg$ jets. Later on, we will shorten as $Z(\nu\nu)$ and gg for the two signals.

These HEP datasets for pileup mitigation tasks are node classification tasks but with multiple graphs. Each node represents a particle and we construct the graph based on a threshold of the relative distance between two particles in the η and ϕ space as demonstrated in fig. 1. The number of graphs we used for training is 70 and the rest of 30 are left for testing. The number of labels is 2 for all the datasets and the node feature dimension is 28. Besides, the particles can be split into charged and neutral where neutral particles do not encode ground truth label information. Under our setting, we choose to encode the ground truth of charged particles into node features so as to help with classification. The node features then contain the η, pt, pdgID one hot encoding (feature to indicate the type of particle, like Hadron and Photon), and charged label encoding. The table below includes some detailed statistics associated with this HEP dataset, which is averaged over a total of 100 graphs.

E.2. Hyperparameter Analysis

In this section, we will introduce our hyperparameter analysis. As mentioned in the experiment section, our StruRW mainly depends on three hyperparameters to calculate and apply the edge reweighting on the source graphs. The epoch m we plan

Table 6: HEP dataset statistics

	$PU10\_gg$	$PU30\_gg$	$PU50\_gg$	$PU50\_Z(\nu\nu)$	$PU140\_gg$	$PU140\_Z(\nu\nu)$
#NODES	185.17	417.84	619	570.90	1569.04	1602.14
#EDGES	1085.17	3518.43	7169.51	5894.8	42321.71	44070.80
LC/OC RATIO	2.8600	0.2796	0.1650	0.0927	0.0575	0.0347

Table 7: Optimal λ value for each real dataset

DOMAIN SPLITS	$A \rightarrow D$	$D \rightarrow A$	CORA WORD	CORA DEGREE	ARXIV TIME1	ARXIV TIME2	ARXIV DEGREE
STRURW-ERM	0.8	1	0.8	0.8	0.1	0.1	0.2
STRURW-ADV	1	0.6	0.1	1	0.1	0.1	0.2
STRURW-MIX	1	0.6	0.6	1	0.1	0.1	0.2

to start calculating the reweighting, the frequency we update the edge weights from the last calculation t, and the degree we integrate the reweighted message λ with the original message. Based on our hyperparameter tuning process, we found λ and starting epoch m tend to be important factors that impact our reweighting performance. We may want to start the reweighting early and with low lambda to rely more on the reweighted information when the CSS shift is large and has more room for improvements. Regarding the case with small CSS, we set larger λ and update with low frequency.

Other important hyperparameters are associated with the coefficient α for the gradient reversal layer in the adversarial training pipeline StruRW-Adv. The two hyperparameters we can tune are the scale added in front of α and the max value that α can take when propagating the reverse gradients. It generally helps with the stability of adversarial training.

Model Architecture Our backbone model is based on GCN (Kipf & Welling, 2017) and for all the baselines. For the DBLP and ACM datasets, we follow the hidden dimension used in the original (Wu et al., 2020) paper two layers of GNN with hidden dimension 128, encode the embeddings into 16 and followed by the classifier with hidden dimension 40. Both the Arxiv and Cora datasets use 300 hidden dimensions with 2 GNN layers. The HEP datasets use hidden dimension 50 and CSBM adopts hidden dimension 20.

learning rate and epochs We select some space to tune the learning rate, where the models mostly take the learning rate as 0.007, 0.004, and 0.001. The adversarial-based model will prefer a learning rate of 0.007 and mixup-based models will prefer a learning rate of 0.001 and 0.004. For the adversarial-based training model DANN and StruRW-Adv, we set the epochs to be 300 and for the mixup model, we will take the epochs to be 200.

GRL coefficient α This value will scale the gradient when we propagate the gradient back. The original calculation is based on the epochs where α is equal to the current epoch divided by the total epochs. Also, it can follow the calculation implemented in DANN (Ganin et al., 2016). Here we add two additional hyperparameters to tune this α for more stable performance. One is a constant that is multiplied in front of this alpha The search space we set for this parameter is mainly $\{1, 1.5, 2\}$. The other is the max value this α can take, with search space $\{0.1, 0.5, 1\}$.

starting epoch m **and the StruRW time period** t The starting epoch means that we will start imposing edge weights on the source graph after epoch m and the StruRW freq means we update the edge weights calculated every t epochs. However, note that in the middle of the t epochs, we will still keep the edge weights calculated from the last time until a new update. The search space for m is $\{100, 150, 200, 250\}$ for experiments with 300 epochs and $\{50, 100, 150\}$ for epoch 200 trainings. The search space for t is $\{1, 5, 10, 15\}$, we found this parameter does not affect the performance as much as the starting epoch. For the experiment that already has good ERM results with smaller shift like Cora, we tend to start later. For the cases where the effect of StruRW is significant, it generally starts early at epoch 50.

λ **in StruRW** This is a ratio to guide message aggregation, 0 stands for the case that completely adopts the reweighted message and 1 corresponds to the GNN original message. It is discussed in the paper's main text and in Fig.4 that we choose large λ when CSS is large and small λ when CSS is small. The specific λ for each different dataset is shown in Table 7.

Baseline hyperparameters For the baseline models, we use the same GNN backbone and the same model architecture as discussed above. The baseline DANN, ERM and Mixup share the same set of hyperparameters as StruRW-Adv, StruRW-ERM and StruRW-Mix respectively. For the UDAGCN baseline, we keep the original set of hyperparameters published in their work. For EERM baselines, I kept the original setting suggested in their paper.