Pairwise Alignment Improves Graph Domain Adaptation

Shikun Liu [1] **Deyu Zou** [2] **Han Zhao** [3] **Pan Li** [1]

Abstract

Graph-based methods, pivotal for label inference over interconnected objects in many real-world applications, often encounter generalization challenges, if the graph used for model training differs significantly from the graph used for testing. This work delves into Graph Domain Adaptation (GDA) to address the unique complexities of distribution shifts over graph data, where interconnected data points experience shifts in features, labels, and in particular, connecting patterns. We propose a novel, theoretically principled method, Pairwise Alignment (Pair-Align) to counter graph structure shift by mitigating conditional structure shift (CSS) and label shift (LS). Pair-Align uses edge weights to recalibrate the influence among neighboring nodes to handle CSS and adjusts the classification loss with label weights to handle LS. Our method demonstrates superior performance in real-world applications, including node classification with region shift in social networks, and the pileup mitigation task in particle colliding experiments. For the first application, we also curate the largest dataset by far for GDA studies. Our method shows strong performance in synthetic and other existing benchmark datasets. [1]

1. Introduction

Graph-based methods are commonly used to enhance label inference for interconnected objects by utilizing their connection patterns in many real-world applications (Jackson et al., 2008; Szklarczyk et al., 2019; Shlomi et al., 2020). Nonetheless, these methods often encounter generalization challenges, as the objects that lack labels and require inference may originate from domains that differ significantly from those with abundant labeled data, thereby exhibiting distinct interconnecting patterns. For instance, in fraud detection within financial networks, label acquisition may be constrained to specific network regions due to varying international legal frameworks and diverse data collection periods (Wang et al., 2019; Dou et al., 2020). Another example is particle filtering for Large Hadron Collider (LHC) experiments (Highfield, 2008), where reliance on simulation-derived labeled data poses a challenge. These simulations may not accurately capture the nuances of real-world experimental conditions, potentially leading to discrepancies in label inference performance when applied to actual experiment scenarios (Li et al., 2022b; Komiske et al., 2017).

Graph Neural Networks (GNNs) have recently demonstrated remarkable effectiveness in utilizing object interconnections for label inference tasks (Kipf & Welling, 2016; Hamilton et al., 2017; Veličković et al., 2018). However, their effectiveness is often hampered by the vulnerability to variations in data distribution (Ji et al., 2023; Ding et al., 2021; Koh et al., 2021). This has sparked significant interest in developing GNNs capable of generalization from one domain (source domain $\mathcal{S}$) to another, potentially different domain (target domain $\mathcal{T}$). This field of study, known as graph domain adaptation (GDA), is gaining increasing attention. GDA distinguishes itself from the traditional domain adaptation setting, primarily because the data points in GDA are interlinked rather than independent. This non-IID nature of graph data renders traditional domain adaptation techniques suboptimal when applied to graphs. The distribution shifts in features, labels, and connecting patterns between objects may significantly impact the adaptation/generalization accuracy. Despite the recent progress made in GDA (Wu et al., 2020; You et al., 2023; Zhu et al., 2021; Liu et al., 2023), current solutions still struggle to tackle the various shifts prevalent in real-world graph data. We provide a detailed discussion of the limitations of existing GDA methods in Section 2.2.

This work conducts a systematic study of the distinct challenges present in GDA and proposes a novel method, named Pairwise Alignment (Pair-Align) to tackle graph structure shift for node prediction tasks. Combined with feature align-

[1]Department of Electrical and Computer Engineering, Georgia Institute of Technology, Georgia, USA [2]School of Data Science, University of Science and Technology of China, Hefei, China [3]Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, USA. Correspondence to: Shikun Liu <shikun.liu@gatech.edu>, Pan Li <panli@gatech.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]Our code and data are available at: https://github.com/Graph-COM/Pair-Align

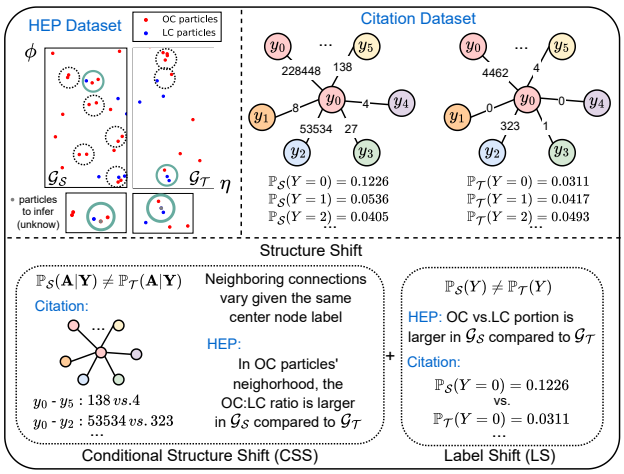

Figure 1. We illustrate structure shifts in real-world datasets: a) The HEP dataset in pileup mitigation tasks (Bertolini et al., 2014) has a shift in PU levels (change in the number of other collisions (OC) around the leading collision (LC) for proton-proton collision events), where $\mathcal{G}_{\mathcal{S}}$ is in PU30 and $\mathcal{G}_{\mathcal{T}}$ is in PU10; Here, in the green circles, the center nodes in grey are the particles whose labels are to be inferred. They have different ground-truth labels but the same neighborhood that includes one OC and one LC particle.b) The citation MAG dataset shifts in regions, where the source graph contains papers in the US and the target graph contains papers in German. More statistics on graph distribution shift from real-world examples can be found in Appendix E.5.

ment methods offered by traditional non-graph DA techniques (Ganin et al., 2016; Tachet des Combes et al., 2020), Pair-Align can in principle address a wide range of distribution shifts in graph data.

Our analysis begins with examining a graph with its adjacency matrix $\mathbf{A}$ and node labels $\mathbf{Y}$. We observe that graph structure shift ($\mathbb{P}_{\mathcal{S}}(\mathbf{A},\mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{A},\mathbf{Y})$) typically manifests as either conditional structure shift (CSS) or label shift (LS), or a combination of both. CSS refers to the change in neighboring connections among nodes within the same class ($\mathbb{P}_{\mathcal{S}}(\mathbf{A}|\mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{A}|\mathbf{Y})$) whereas LS denotes changes in the class distribution of nodes ($\mathbb{P}_{\mathcal{S}}(\mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{Y})$). These shifts are illustrated in Fig. 1 via examples in HEP and social networks, and are justified by statistics from several real-world applications.

In light of the two types of shifts, the Pair-Align method aims to estimate and subsequently mitigate the distribution shift in the neighboring nodes' representations for any given node class c. To achieve this, Pair-Align employs a bootstrapping technique to recalibrate the influence of neighboring nodes in the message aggregation phase of GNNs. This strategic reweighting is key to effectively countering CSS. Concurrently, Pair-Align calculates label weights to alleviate disparities in the label distribution between source and target domains (addressing LS) by adjusting the classification loss. Pair-Align is depicted in Figure 2.

To demonstrate the effectiveness of our pipeline, we curate the regional MAG data that partitions large citation networks according to the regions where papers got published (Hu et al., 2020; Wang et al., 2020) to simulate the region shift. To the best of our knowledge, this is the largest dataset (of $\approx$ 380k nodes, 1.35M edges) to study GDA with data retrieved from the real-world database. We also include other graph data with shifts, like the pileup mitigation task studied in Liu et al. (2023). Our method shows strong performance in these two applications. Moreover, our method also outperforms baselines significantly in synthetic datasets and other real-world benchmark datasets.

2. Preliminaries and Related Works

2.1. Notation and The Problem Setup

We use capital letters, e.g., Y to denote scalar random variables, and lower-case letters, e.g., y to denote their realizations. The bold counterparts are used for their vector-valued correspondences, e.g., $\mathbf{Y}, \mathbf{y}$, and the calligraphic letters, e.g. $\mathcal{Y}$, are for the value spaces. We always use capital letters to denote matrices. Let $\mathbb{P}$ denote a distribution, whose subscript $\mathcal{U} \in \{\mathcal{S}, \mathcal{T}\}$ indicates the domain it depicts, e.g. $\mathbb{P}_{\mathcal{S}}(Y)$. The probability of a realization, e.g. $Y = y$, can then be denoted as $\mathbb{P}_{\mathcal{S}}(Y = y)$.

Graph Neural Networks (GNNs). We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{x})$ to denote a graph with the node set $\mathcal{V}$, the edge set $\mathcal{E}$ and node features $\mathbf{x} = [\cdots x_u \cdots]_{u\in\mathcal{V}}$. We focus on undirected graphs where the graph structure can also be represented as a symmetric adjacency matrix $\mathbf{A}$ where the entries $A_{uv} = A_{vu} = 1$ when nodes u, v form an edge and otherwise 0. GNNs take $\mathbf{A}$ and $\mathbf{x}$ as input and output node representations $\{h_u, \forall u \in \mathcal{V}\}$. The standard GNNs (Hamilton et al., 2017) has a message-passing procedure. Specifically, with $h_u^{(1)} = x_u$, for each node v and each layer $k \in [L] := \{1, \ldots, L\}$,

$$h_u^{(k+1)} = \text{UPT}\left(h_u^{(k)}, \text{AGG}\left(\{\!\{h_v^{(k)} : v \in \mathcal{N}_u\}\!\}\right)\right), \quad (1)$$

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. The last-layer node representation $h_u^{(L)}$ is used to predict the label $y_u \in \mathcal{Y}$ in node classification tasks.

Domain Adaptation (DA). In DA, each domain $\mathcal{U} \in \{S, T\}$ has its own joint feature and label distribution $\mathbb{P}_{\mathcal{U}}(X, Y)$. In the unsupervised setting, we have access to labeled source data $\{(x_i, y_i)\}_{i=1}^N$ and unlabeled target data $\{(x_i)\}_{i=1}^M$ IID sampled from the source and target domain respectively. The model comprises a feature encoder $\phi : \mathcal{X} \to \mathcal{H}$ and a classifier $g : \mathcal{H} \to \mathcal{Y}$, with classification error in domain $\mathcal{U}$ denoted as $\varepsilon_{\mathcal{U}}(g \circ \phi) = \mathbb{P}_{\mathcal{U}}(g(\phi(X)) \neq Y)$. The objective is to train the model with available data to

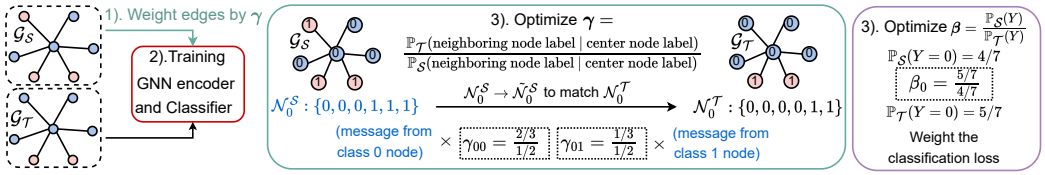

Figure 2. The pipeline contains modules in handling CSS with edge weights $\boldsymbol{\gamma}$ and handling LS with label weights $\boldsymbol{\beta}$

minimize target error $\varepsilon_{\mathcal{T}}(g \circ \phi)$ when predicting target labels. A popular DA strategy is to learn domain-invariant representation, ensuring similar $\mathbb{P}_{\mathcal{S}}(H)$ and $\mathbb{P}_{\mathcal{T}}(H)$ and minimizing the source error $\varepsilon_{\mathcal{S}}(g \circ \phi)$ to retain classification capability simultaneously (Zhao et al., 2019). This is achieved through regularization of distance measures (Long et al., 2015; Zellinger et al., 2016) or adversarial training (Ganin et al., 2016; Tzeng et al., 2017; Zhao et al., 2018).

Graph Domain Adaptation (GDA). When extending unsupervised DA to the graph-structured data, we are given a source graph $\mathcal{G}_{\mathcal{S}} = (\mathcal{V}_{\mathcal{S}}, \mathcal{E}_{\mathcal{S}}, \mathbf{x}_{\mathcal{S}})$ with node labels $\mathbf{y}_{\mathcal{S}}$ and a target graph $\mathcal{G}_{\mathcal{T}} = (\mathcal{V}_{\mathcal{T}}, \mathcal{E}_{\mathcal{T}}, \mathbf{x}_{\mathcal{T}})$. The specific distribution and shifts in graph-structured data will be defined in Sec.3. The objective is similar to DA as to minimize the target error, but with the encoder ϕ switched to a GNN to predict node labels $\mathbf{y}_{\mathcal{T}}$ in the target graph.

2.2. Related Works and Existing Gaps

GDA research falls into two main categories, aiming at addressing domain adaptation for node and graph classification tasks respectively. Often, graph-level GDA problems can view each graph as an independent sample, allowing extension of previous non-graph DA techniques to graphs, such as causal inference (Rojas-Carulla et al., 2018; Peters et al., 2017) (more are reviewed in Appendix D). Conversely, node-level GDA presents challenges due to the interconnected nodes. Previous works mainly leveraged node representations as intermediaries to address these challenges.

The dominant idea of existing work on node-level GDA focused on aligning the marginal distributions of node representations, mostly over the last layer $\mathbf{h}^{(L)}$, across two graphs inspired by the domain invariant learning in DA (Liao et al., 2021). Some of them adopted adversarial training, such as (Dai et al., 2022; Zhang et al., 2019; Shen et al., 2020a). UDAGCN (Wu et al., 2020) calculated the point-wise mutual information and inter-graph attention to exploit local and global consistency on top of the adversarial training. Other works were motivated by regularizing different distance measures. Zhu et al. (2021) regularized over the central moment discrepancy (Zellinger et al., 2016). You et al. (2023) minimized the Wasserstein-1 distance between the distributions of node representations and controlled GNN Lipschitz via regularizing graph spectral properties. Wu et al. (2023) introduced graph subtree discrepancy inspired by the WL subtree kernel (Shervashidze et al., 2011) and suggested regularizing node representations after each layer of GNNs. Furthermore, Zhu et al. (2022; 2023) recognized that there could also be a shift in the label distribution, so they proposed to align the distribution of label/pseudo-label in addition to the marginal node representation.

Nonetheless, the marginal alignment methods above are inadequate when dealing with the structure shift consisting of CSS and LS. Firstly, these methods are flawed under LS. Based on $\mathbb{P}_{\mathcal{U}}(H^{(L)}) = \sum_Y \mathbb{P}_{\mathcal{U}}(H^{(L)}|Y)\mathbb{P}_{\mathcal{U}}(Y)$, even if the marginal alignment $\mathbb{P}_{\mathcal{S}}(H^{(L)}) = \mathbb{P}_{\mathcal{T}}(H^{(L)})$ is achieved, the conditional node representations will still mismatch $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) \neq \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ under the LS, which induces more prediction error (Zhao et al., 2019; Tachet des Combes et al., 2020). Secondly, they are suboptimal under CSS. In particular, consider the HEP example in Fig. 1 (the particles in the two green circles) where CSS may yield the case that the label of the center particle (node) shifts, albeit with an unchanged neighborhood distribution. In this case, methods using a shared GNN encoder for marginal alignment definitely fail to make the correct prediction.

Liu et al. (2023) have recently analyzed this issue by using an example based on contextual stochastic block model (CSBM) (Deshpande et al., 2018) (defined in Appendix A).

Proposition 2.1. *(Liu et al., 2023) Suppose the source and target graphs are generated from the CSBM model of n nodes with the same label distributions and node feature distributions. The edge connection probabilities are set to present a conditional structure shift $\mathbb{P}_{\mathcal{S}}(\mathbf{A}|\mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{A}|\mathbf{Y})$ and showcase the example that the ground truth label of the center node changes given the same neighborhood distribution. Then, suppose a GNN encoder ϕ is shared across two domains, the target classification error $\varepsilon_{\mathcal{T}}(g \circ \phi)$ can be lower bounded by 0.25, where g is the classifier. However, the GNN encoder ϕ, if allowed to be adjusted according to the domains, can achieve $\varepsilon_{\mathcal{T}}(g \circ \phi) \to 0$ as $n \to \infty$.*

To tackle this issue, Liu et al. (2023) proposed the StruRW method to reweight edges in the source graph based on weights derived from the CSBM model. However, StruRW still suffers from many issues. We will provide a more detailed comparison with StruRW in Sec. 3.6. To the best of our knowledge, our method is the first effort to address both CSS and LS in a principled way.

3. Pairwise Alignment for Structure Shift

We first define shifts in graphs as feature shift and structure shift, the latter includes both the Conditional Structure Shift (CSS) and the Label Shift (LS). Then, we analyze the objective of solving structure shift and propose our pairwise alignment algorithm that handles both CSS and LS.

3.1. Distribution Shifts in Graph-structured Data

Sec. 2.2 shows the sub-optimality of enforcing marginal node representation alignment under structure shifts. In fact, the necessity of conditional distribution alignment $\mathbb{P}_{\mathcal{S}}(H|Y) = \mathbb{P}_{\mathcal{T}}(H|Y)$ to deal with feature shift $\mathbb{P}_{\mathcal{S}}(X|Y) \neq \mathbb{P}_{\mathcal{T}}(X|Y)$ has been explored in non-graph scenarios, where X denotes a feature vector and H is the representation after X passes through the encoder, i.e., $H = \phi(X)$. Early efforts such as Zhang et al. (2013); Gong et al. (2016) assumed that the shift in conditional representations from domain $\mathcal{S}$ to domain $\mathcal{T}$ follows a linear transformation and optimized conditional alignment by introducing an extra linear transformation to the source domain encoder to enhance conditional alignment $\mathbb{P}_{\mathcal{S}}(H|Y) = \mathbb{P}_{\mathcal{T}}(H|Y)$. Subsequent works learned the representations with adversarial training to enforce conditional alignment by aligning the joint distribution over the label predictions and representations (Long et al., 2018; Cicek & Soatto, 2019). Later, some works additionally considered label shift (Tachet des Combes et al., 2020; Liu et al., 2021) and proposed to match the label weighted $\mathbb{P}^{\text{lw}}_{\mathcal{S}}(H)$ with $\mathbb{P}_{\mathcal{T}}(H)$ with label weights estimated following Lipton et al. (2018).

In light of the limitations of existing works and the effort in non-graph DA research, it becomes clear that marginal alignment of node representations is insufficient for GDA, which underscores the importance of achieving conditional node representation alignment.

To address various distribution shifts for GDA in principle, we first decouple the potential distribution shifts in graph data by defining feature shift and structure shift in terms of conditional distributions and label distributions. Our data generation process can be characterized by the following model: $\mathbf{X} \leftarrow \mathbf{Y} \rightarrow \mathbf{A}$, where labels are drawn at each node first, and then edges as well as features at each node are generated. Under this model, we define the following feature shift, which denotes the change of the conditional feature generation process given the labels.

Definition 3.1 (Feature Shift)**.** Assume the node features $x_u, u \in \mathcal{V}$ are IID sampled from $\mathbb{P}(X|Y)$ given node labels y_u. Therefore, the conditional distribution of $\mathbf{x}|\mathbf{y}$, $\mathbb{P}(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}) = \prod_{u\in\mathcal{V}} \mathbb{P}(X = x_u|Y = y_u)$. The feature shift is then defined as $\mathbb{P}_{\mathcal{S}}(X|Y) \neq \mathbb{P}_{\mathcal{T}}(X|Y)$.

Definition 3.2 (Structure Shift)**.** Given the joint distribution of the adjacency matrix and node labels $\mathbb{P}(\mathbf{A}, \mathbf{Y})$. The Structure Shift is defined as $\mathbb{P}_{\mathcal{S}}(\mathbf{A}, \mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{A}, \mathbf{Y})$. With decomposition as $\mathbb{P}_{\mathcal{U}}(\mathbf{A}, \mathbf{Y}) = \mathbb{P}_{\mathcal{U}}(\mathbf{A}|\mathbf{Y})\mathbb{P}_{\mathcal{U}}(\mathbf{Y})$, it results in Conditional Structure Shift (CSS) and Label Shift (LS):

- CSS: $\mathbb{P}_{\mathcal{S}}(\mathbf{A}|\mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{A}|\mathbf{Y})$
- LS: $\mathbb{P}_{\mathcal{S}}(\mathbf{Y}) \neq \mathbb{P}_{\mathcal{T}}(\mathbf{Y})$

As shown in Fig. 1, structure shift consisting of CSS and LS widely exists in real-world applications. Feature shift here, which is equivalent to the conditional feature shift in non-graph literature, can be addressed by adapting conventional conditional shift methods. So, later, we assume that feature shift has been addressed, i.e., $\mathbb{P}_{\mathcal{S}}(X|Y) = \mathbb{P}_{\mathcal{T}}(X|Y)$.

In contrast, structure shift is unique to graph data due to the non-IID nature caused by node interconnections. Moreover, the learning of node representations is intrinsically linked to the graph structure as the GNN encoder takes $\mathbf{A}$ as input. Therefore, even if after one layer of GNN, $\mathbb{P}_{\mathcal{S}}(H^{(k)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k)}|Y)$ is achieved, CSS could still lead to misalignment of conditional node representation distributions in the next layer $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) \neq \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$. Accordingly, a tailored algorithm is needed to remove this effect of CSS, which, when combined with techniques for LS, can effectively resolve the structure shift.

3.2. Addressing Conditional Structure Shift

To remove the effect of CSS under GNN, the objective is to guarantee $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$ given $\mathbb{P}_{\mathcal{S}}(H^{(k)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k)}|Y)$. Considering one layer of GNN encoding in Eq. (1): given $\mathbb{P}_{\mathcal{S}}(H^{(k)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k)}|Y)$, the mismatch in $k+1$ layer may arise from the distribution shift of the neighboring multiset $\{\!\{h_v^{(k)} : v \in \mathcal{N}_u\}\!\}$ given the center node label y_u. Therefore, the key is to transform the neighboring multisets in the source graph to achieve conditional alignment with the target domain regarding the distributions of such neighboring multisets. Our approach first starts with a sufficient condition for such conditional alignment.

Theorem 3.3 (Sufficient conditions for addressing CSS)**.** *Given the following assumptions*

- *(Conditional Alignment in the previous layer k) $\mathbb{P}_{\mathcal{S}}(H^{(k)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k)}|Y)$ and $\forall u \in \mathcal{V}_{\mathcal{U}}$, given $Y = y_u$, $h_u^{(k)}$ is independently sampled from $\mathbb{P}_{\mathcal{U}}(H^{(k)}|Y)$.*
- *(Edge Conditional Independence) Given node labels $\mathbf{y}$, edges mutually independently exist in the graph.*

if there exists a transformation that modifies the neighborhood of node u: $\mathcal{N}_u \rightarrow \tilde{\mathcal{N}}_u, \forall u \in \mathcal{V}_{\mathcal{S}}$, such that $\mathbb{P}_{\mathcal{S}}(|\tilde{\mathcal{N}}_u||Y_u = i) = \mathbb{P}_{\mathcal{T}}(|\mathcal{N}_u||Y_u = i)$ and $\mathbb{P}_{\mathcal{S}}(Y_v|Y_u = i, v \in \tilde{\mathcal{N}}_u) = \mathbb{P}_{\mathcal{T}}(Y_v|Y_u = i, v \in \mathcal{N}_u)$, $\forall i \in \mathcal{Y}$, then $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$ is satisfied.

Remark 3.4. The assumption edge conditional independence essentially assumes an SBM model for the graph

structure, which is widely adopted for graph learning algorithm analysis (Liu et al., 2023; Wei et al., 2022).

This theorem reveals that it suffices to align two distributions with the multiset transformation on the source graph: 1) the distribution of the degree/cardinality of the neighbors $\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u||Y_u)$ and 2) the node label distribution in the neighborhood $\mathbb{P}_{\mathcal{U}}(Y_v|Y_u, v \in \mathcal{N}_u)$, both conditioned on the center node label Y_u.

Multiset Alignment. Bootstrapping the elements in the multisets can be used to align the two distributions. In the context of GNNs, which typically employ sum/mean pooling functions to aggregate the multisets, such a bootstrapping process can be translated into assigning weights to different neighboring nodes given their labels and the center node's label. Moreover, practically, mean pooling is often the preferred choice due to its superior empirical performance, which is also observed in our experiments. Aligning the distributions of the node degrees $\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u||Y_u)$ yields negligible impact with mean pooling (Xu et al., 2018). Therefore, our method focuses on aligning the distribution $\mathbb{P}_{\mathcal{U}}(Y_v|Y_u, v \in \mathcal{N}_u)$, in which the edge weights are the ratios of such probabilities across two domains:

Definition 3.5. Assume $\mathbb{P}_{\mathcal{S}}(Y_v = j|Y_u = i, v \in \mathcal{N}_u) > 0, \forall i, j \in \mathcal{Y}$, we define $\boldsymbol{\gamma} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ as:

$$[\boldsymbol{\gamma}]_{i,j} = \frac{\mathbb{P}_{\mathcal{T}}(Y_v = j|Y_u = i, v \in \mathcal{N}_u)}{\mathbb{P}_{\mathcal{S}}(Y_v = j|Y_u = i, v \in \mathcal{N}_u)}, \forall i, j \in \mathcal{Y}$$

where $[\boldsymbol{\gamma}]_{i,j}$ is the density ratio between the target and source graphs from class-i nodes to class-j nodes. Note that $[\boldsymbol{\gamma}]_{i,j} \neq [\boldsymbol{\gamma}]_{j,i}$. To differentiate the encoding with and without the adjusted edge weights for the source and target graphs, we denote the operation that first adjusts the edge weights $\boldsymbol{\gamma}$ and then apply GNN encoding as $\phi_{\boldsymbol{\gamma}}$ while the one that directly applies GNN encoding as ϕ. By assuming the conditions made in Thm 3.3 and applying them in an iterative manner for each layer of GNN, the last-layer alignment $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ can be achieved with $\mathbf{h}_{\mathcal{S}}^{(L)} = \phi_{\boldsymbol{\gamma}}(\mathbf{x}_{\mathcal{S}}, \mathbf{A}_{\mathcal{S}})$ and $\mathbf{h}_{\mathcal{T}}^{(L)} = \phi(\mathbf{x}_{\mathcal{T}}, \mathbf{A}_{\mathcal{T}})$. Note that based on conditional alignment in the distribution of randomly sampled node representations $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ and under the conditions in Thm 3.3, $\mathbb{P}_{\mathcal{S}}(\mathbf{H}^{(L)}|\mathbf{Y}) = \mathbb{P}_{\mathcal{T}}(\mathbf{H}^{(L)}|\mathbf{Y})$ can also be achieved in the matrix form.

$\boldsymbol{\gamma}$ Estimation. Till now we explain why edge reweighting using $\boldsymbol{\gamma}$ can address CSS for GNN encoding, we will detail our pairwise alignment method to obtain $\boldsymbol{\gamma}$ next. By definition, $\boldsymbol{\gamma}$ can be decomposed into another two weights.

Definition 3.6. Assume $\mathbb{P}_{\mathcal{S}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{S}}) > 0, \forall i, j \in \mathcal{Y}$, we define $\mathbf{w} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ and $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{Y}|\times 1}$ as:

$$[\mathbf{w}]_{i,j} = \frac{\mathbb{P}_{\mathcal{T}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{T}})}{\mathbb{P}_{\mathcal{S}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{S}})},$$
$$[\boldsymbol{\alpha}]_i = \frac{\mathbb{P}_{\mathcal{T}}(Y_u = i|e_{uv} \in \mathcal{E}_{\mathcal{T}})}{\mathbb{P}_{\mathcal{S}}(Y_u = i|e_{uv} \in \mathcal{E}_{\mathcal{S}})}, \forall i, j \in \mathcal{Y}$$

and $\boldsymbol{\gamma}$ can be estimated via

$$\boldsymbol{\gamma} = \text{diag}(\boldsymbol{\alpha})^{-1}\mathbf{w} \tag{2}$$

For domain $\mathcal{U}$, $\mathbb{P}_{\mathcal{U}}(Y_u, Y_v|e_{uv} \in \mathcal{E}_{\mathcal{U}})$ is the joint distribution of the label pairs of two nodes that form an edge, which can be computed for domain $\mathcal{S}$ but not for domain $\mathcal{T}$. $\mathbb{P}_{\mathcal{U}}(Y_u|e_{uv} \in \mathcal{E}_{\mathcal{U}})$ can be obtained by marginalizing $\mathbb{P}_{\mathcal{U}}(Y_u, Y_v|e_{uv} \in \mathcal{E}_{\mathcal{U}})$ over Y_v, as $\mathbb{P}_{\mathcal{U}}(Y_u = i|e_{uv} \in \mathcal{E}_{\mathcal{U}}) = \sum_{j\in\mathcal{Y}} \mathbb{P}_{\mathcal{U}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{U}})$. Also, it is crucial to differentiate $\mathbb{P}_{\mathcal{U}}(Y_u|e_{uv} \in \mathcal{E}_{\mathcal{U}})$ from $\mathbb{P}_{\mathcal{U}}(Y)$: the former is the label distribution of the end node conditioned on an edge, while the latter is the label distribution of nodes without conditions. Given $\mathbf{w}$ and two distributions computed over the source graph, $\boldsymbol{\alpha}$ can be derived via

$$[\boldsymbol{\alpha}]_i = \frac{\sum_{j\in\mathcal{Y}}([\mathbf{w}]_{i,j}\mathbb{P}_{\mathcal{S}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{S}}))}{\mathbb{P}_{\mathcal{S}}(Y_u = i|e_{uv} \in \mathcal{E}_{\mathcal{S}})}, \tag{3}$$

so next, we proceed to estimate $\mathbf{w}$ to complete $\boldsymbol{\gamma}$ calculation.

Pair-wise Alignment. Note that if (Y_u, Y_v) is viewed as a type for edge e_{uv}, $\mathbb{P}_{\mathcal{U}}(Y_u, Y_v|e_{uv} \in \mathcal{E}_{\mathcal{U}})$ essentially represents an edge-type distribution. In practice, we use *pair-wise* pseudo-label distribution alignment to estimate $\mathbf{w}$.

Definition 3.7. Let $\boldsymbol{\Sigma} \in \mathbb{R}^{|\mathcal{Y}|^2\times|\mathcal{Y}|^2}$ denote the matrix that stands for the joint distribution of the predicted types of edges and the true types of edges, and $\boldsymbol{\nu} \in \mathbb{R}^{|\mathcal{Y}|^2\times 1}$ denote the distribution of the predicted types of edges for the target domain, $\forall i, j, i', j' \in \mathcal{Y}$

$$[\boldsymbol{\Sigma}]_{ij,i'j'} = \mathbb{P}_{\mathcal{S}}(\hat{Y}_u = i, \hat{Y}_v = j, Y_u = i', Y_v = j'|e_{uv} \in \mathcal{E}_{\mathcal{S}})$$
$$[\boldsymbol{\nu}]_{ij} = \mathbb{P}_{\mathcal{T}}(\hat{Y}_u = i, \hat{Y}_v = j|e_{uv} \in \mathcal{E}_{\mathcal{T}})$$

Specifically, similar to Tachet des Combes et al. (2020, Lemma 3.2), Lemma 3.8 shows that $\mathbf{w}$ can be obtained by solving the linear system $\boldsymbol{\nu} = \boldsymbol{\Sigma}\mathbf{w}$ if $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied.

Lemma 3.8. *If $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied, and node representations are conditionally independent of graph structures given node labels, then $\boldsymbol{\nu} = \boldsymbol{\Sigma}\mathbf{w}$.*

Empirically, we estimate $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\nu}}$ based on the classifier g, where $g(h_u^{(L)})$ denotes the soft label of node u. Specifically,

$$[\hat{\boldsymbol{\Sigma}}]_{ij,i'j'} = \frac{1}{|\mathcal{E}_{\mathcal{S}}|}\sum_{e_{uv}\in\mathcal{E}_{\mathcal{S}}, y_u=i', y_v=j'}[g(h_u^{(L)})]_i \times [g(h_v^{(L)})]_j$$
$$[\hat{\boldsymbol{\nu}}]_{ij} = \frac{1}{|\mathcal{E}_{\mathcal{T}}|}\sum_{e_{u'v'}\in\mathcal{E}_{\mathcal{T}}}[g(h_{u'}^{(L)})]_i \times [g(h_{v'}^{(L)})]_j.$$

Then, $\mathbf{w}$ can be solved via:

$$\min_{\mathbf{w}} \quad \|\hat{\boldsymbol{\Sigma}}\mathbf{w} - \hat{\boldsymbol{\nu}}\|_2, \quad \text{s.t. } \mathbf{w} \geq 0, \text{ and} \tag{4}$$
$$\sum_{i,j}[\mathbf{w}]_{i,j}\mathbb{P}_{\mathcal{S}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{S}}) = 1,$$

where the constraints guarantee a valid target edge type distribution $\mathbb{P}_{\mathcal{T}}(Y_u, Y_v | e_{uv} \in \mathcal{E}_{\mathcal{T}})$. For undirected graphs, $\mathbf{w}$ can be symmetric, so we may add an extra constraint $[\mathbf{w}]_{i,j} = [\mathbf{w}]_{j,i}$. Finally, we calculate $\boldsymbol{\alpha}$ following Eq. (3) with the obtained $\mathbf{w}$ and compute $\boldsymbol{\gamma}$ via Eq. (2). Note that in Appendix 3.5, we will discuss how to improve the robustness of the estimations of $\mathbf{w}$ and $\boldsymbol{\gamma}$.

In summary, handling CSS is an iterative process where we begin by employing an estimated $\boldsymbol{\gamma}$ as edge weights on the source graph to reduce the gap between $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)$ and $\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ due to Thm 3.3. With a reduced gap, we can estimate $\mathbf{w}$ more accurately (due to Lemma 3.8) and thus improve the estimation of $\boldsymbol{\gamma}$. Through iterative refinement, $\boldsymbol{\gamma}$ progressively enhances the conditional alignment $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ to address CSS.

3.3. Addressing Label Shift

Inspired by the techniques in Lipton et al. (2018); Azizzadenesheli et al. (2018), we estimate the ratio between the source and target label distribution by aligning the node-level pseudo-label distribution to address LS.

Definition 3.9. Assume $\mathbb{P}_{\mathcal{S}}(Y_u = i) > 0, \forall i \in \mathcal{Y}$, we define $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{Y}|\times 1}$ as the weights of the source and target label distribution: $[\boldsymbol{\beta}]_i = \frac{\mathbb{P}_{\mathcal{T}}(Y=i)}{\mathbb{P}_{\mathcal{S}}(Y=i)}, \forall i \in \mathcal{Y}$.

Definition 3.10. Let $\mathbf{C} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ denote the confusion matrix of the classifier for the source domain, and $\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{Y}|\times 1}$ denote the distribution of the label predictions for the target domain, $\forall i, i' \in \mathcal{Y}$

$$[\mathbf{C}]_{i,i'} = \mathbb{P}_{\mathcal{S}}(\hat{Y} = i, Y = i'), \quad [\boldsymbol{\mu}]_i = \mathbb{P}_{\mathcal{T}}(\hat{Y} = i)$$

The key insight is similar to the estimation of $\mathbf{w}$, when $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied, $\boldsymbol{\beta}$ can be estimated by solving a linear system $\boldsymbol{\mu} = \mathbf{C}\boldsymbol{\beta}$,

Lemma 3.11. *If $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied, and node representations are conditionally independent of each other given the node labels, then $\boldsymbol{\mu} = \mathbf{C}\boldsymbol{\beta}$.*

Empirically, with $\hat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$ can be estimated as

$$[\hat{\mathbf{C}}]_{i,i'} = \frac{1}{|\mathcal{V}_{\mathcal{S}}|} \sum_{u \in \mathcal{V}_{\mathcal{S}}, y_u = i'} [g(h_u^{(L)})]_i$$

$$[\hat{\boldsymbol{\mu}}]_i = \frac{1}{|\mathcal{V}_{\mathcal{T}}|} \sum_{u' \in \mathcal{V}_{\mathcal{T}}} [g(h_{u'}^{(L)})]_i$$

$\boldsymbol{\beta}$ can be solved with a least square problem with the constraints to guarantee a valid target label distribution $\mathbb{P}_{\mathcal{T}}(Y)$.

$$\min_{\boldsymbol{\beta}} \|\hat{\mathbf{C}}\boldsymbol{\beta} - \hat{\boldsymbol{\mu}}\|_2, \text{ s.t. } \boldsymbol{\beta} \geq 0, \sum_i [\boldsymbol{\beta}]_i \mathbb{P}_{\mathcal{S}}(Y = i) = 1 \quad (5)$$

We use $\boldsymbol{\beta}$ to weight the classification loss to handle LS. Combined with the previous module that uses $\boldsymbol{\gamma}$ to solve for CSS, our algorithm completely addresses the structure shift.

Algorithm 1 Pairwise Alignment

1: **Input** The source graph $\mathcal{G}_{\mathcal{S}}$ with node labels $\mathbf{Y}_{\mathcal{S}}$; The target graph $\mathcal{G}_{\mathcal{T}}$; A GNN ϕ and a classifier g; The total epoch number n, the epoch period t for weight update.
2: Initialize $\mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\beta} = \mathbf{1}$,
3: **while** epoch $< n$ or not converged **do**
4: Add edge weights to $\mathcal{G}_{\mathcal{S}}$ according to $\boldsymbol{\gamma}$
5: Get $\hat{\mathbf{Y}}_{\mathcal{S}} = g(\phi_{\boldsymbol{\gamma}}(\mathbf{x}_{\mathcal{S}}, \mathbf{A}_{\mathcal{S}}))$ in the source domain
6: Update ϕ and g as $\min_{\phi,g} \mathcal{L}_C^{\boldsymbol{\beta}}(\hat{\mathbf{Y}}_{\mathcal{S}}, \mathbf{Y}_{\mathcal{S}})$ Eq. (6)
7: **if** epoch $\equiv 0 \pmod t$ **then**
8: Get $\hat{\mathbf{Y}}_{\mathcal{S}}$ and $\hat{\mathbf{Y}}_{\mathcal{T}} = g(\phi(\mathbf{x}_{\mathcal{T}}, \mathbf{A}_{\mathcal{T}}))$
9: Update the estimation of $\hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{\nu}}, \hat{\mathbf{C}}, \hat{\boldsymbol{\mu}}$
10: Optimize for $\mathbf{w}$ Eq.(4) and calculate for $\boldsymbol{\gamma}$ Eq.(2)
11: Optimize for $\boldsymbol{\beta}$ following Eq.(5)
12: **end if**
13: **end while**

3.4. Algorithm Overview

Now, we are able to put everything together. The entire algorithm is shown in Alg. 1. At the start of each epoch, the estimated $\boldsymbol{\gamma}$ are used as edge weights in the source graph (line 4). Then, GNN $\phi_{\boldsymbol{\gamma}}$ paired with $\boldsymbol{\gamma}$ yields node representations that further pass through the classifier g to get soft labels $\hat{\mathbf{Y}}$ (line 5). The model is trained via the loss $\mathcal{L}_C^{\boldsymbol{\beta}}$, i.e., a $\boldsymbol{\beta}$-weighted cross-entropy loss (line 6):

$$\mathcal{L}_C^{\boldsymbol{\beta}} = \frac{1}{|\mathcal{V}_{\mathcal{S}}|} \sum_{v \in \mathcal{V}_{\mathcal{S}}} [\boldsymbol{\beta}]_{y_v} \text{cross-entropy}(y_v, \hat{y}_v) \quad (6)$$

Then, with every t epoch, update the estimations of $\mathbf{w}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}$ for the next epoch (lines 7-10).

3.5. Robust Estimation of $\boldsymbol{\gamma}, \mathbf{w}, \boldsymbol{\beta}$

To improve robustness of the estimation, we incorporate L2 regularization into the least square optimization for $\mathbf{w}$ and $\boldsymbol{\beta}$. Typically, node classification tends to have imperfect accuracy and results in similar prediction probabilities across classes. This may lead to ill-conditioned $\hat{\boldsymbol{\Sigma}}$ and $\hat{\mathbf{C}}$ in Eq.(4) and (5), respectively. Specifically, Eq.(4) and (5) can be revised as

$$\min_{\mathbf{w}} \|\hat{\boldsymbol{\Sigma}}\mathbf{w} - \hat{\boldsymbol{\nu}}\|_2 + \lambda\|\mathbf{w} - \mathbf{1}\|_2, \quad (7)$$
$$\text{s.t. } \mathbf{w} \geq 0, \sum_{i,j} [\mathbf{w}]_{i,j} \mathbb{P}_{\mathcal{S}}(Y_u = i, Y_v = j | e_{uv} \in \mathcal{E}_{\mathcal{S}}) = 1$$

$$\min_{\boldsymbol{\beta}} \|\hat{\mathbf{C}}\boldsymbol{\beta} - \hat{\boldsymbol{\mu}}\|_2 + \lambda\|\boldsymbol{\beta} - \mathbf{1}\|_2 \quad (8)$$
$$\text{s.t. } \boldsymbol{\beta} \geq 0, \sum_i [\boldsymbol{\beta}]_i \mathbb{P}_{\mathcal{S}}(Y = i) = 1.$$

where the added L2 regularization will push estimated $\mathbf{w}$ and $\boldsymbol{\beta}$ close to $\mathbf{1}$. In practice, we find this regularization

to be important in the early training stage and can guide a better weight estimation in the later stage.

We also introduce a regularization strategy to improve the robustness of $\boldsymbol{\gamma}$. This is to deal with the variance in edge formation that may affect $\mathbb{P}_{\mathcal{U}}(Y_v|Y_u, v \in \mathcal{N}_u)$ in $\boldsymbol{\gamma}$ calculation.

Take a specific example to demonstrate the idea of regularizing γ. Suppose node labels are binary and suppose we count the numbers of edges of different types in the source graph and obtain $\hat{\mathbb{P}}_{\mathcal{S}}(Y_u = 0, Y_v = 0|e_{uv} \in \mathcal{E}_{\mathcal{S}}) = 0.001$ and $\hat{\mathbb{P}}_{\mathcal{S}}(Y_u = 0, Y_v = 1|e_{uv} \in \mathcal{E}_{\mathcal{S}}) = 0.0005$. Then without any regularization, based on the estimated edge-type distributions, we obtain $\hat{\mathbb{P}}_{\mathcal{S}}(Y_v = 0|Y_u = 0, v \in \mathcal{N}_u) = 2/3$ and $\hat{\mathbb{P}}_{\mathcal{S}}(Y_v = 0|Y_u = 0, v \in \mathcal{N}_u) = 1/3$. However, the estimation $\hat{\mathbb{P}}_{\mathcal{S}}(Y_u = i, Y_v = j|e_{uv} \in \mathcal{E}_{\mathcal{S}})$ may be inaccurate when its value is close to 0. Because in this case, the number of edges of the corresponding type (i, j) is too small in the graph. These edges may be formed based on randomness. Conversely, larger observed values like $\hat{\mathbb{P}}_{\mathcal{S}}(Y_u = 0, Y_v = 0|e_{uv} \in \mathcal{E}_{\mathcal{S}}) = 0.2$ and $\hat{\mathbb{P}}_{\mathcal{S}}(Y_u = 0, Y_v = 1|e_{uv} \in \mathcal{E}_{\mathcal{S}}) = 0.1$ are often more reliable. To address the issue, we may introduce a regularization term δ when using $\mathbf{w}$ to compute $\boldsymbol{\gamma}$. We compute $\mathbf{w}' = \frac{\hat{\mathbb{P}}_{\mathcal{T}}(Y_u=i,Y_v=j|e_{uv}\in\mathcal{E}_{\mathcal{S}})+\delta}{\hat{\mathbb{P}}_{\mathcal{S}}(Y_u=i,Y_v=j|e_{uv}\in\mathcal{E}_{\mathcal{S}})+\delta} = \frac{[\mathbf{w}]_{ij}\hat{\mathbb{P}}_{\mathcal{S}}(Y_u=i,Y_v=j|e_{uv}\in\mathcal{E}_{\mathcal{S}})+\delta}{\hat{\mathbb{P}}_{\mathcal{S}}(Y_u=i,Y_v=j|e_{uv}\in\mathcal{E}_{\mathcal{S}})+\delta}$, and replace $\mathbf{w}$ with $\mathbf{w}'$ when computing $\boldsymbol{\gamma}$.

3.6. Comparison to StruRW (Liu et al., 2023)

The edge weights estimation in StruRW and Pair-Align differ in two major points. First, StruRW computes edge weights as the ratio of the source and target edge connection probabilities. This by definition, if using our notations, corresponds to $\mathbf{w}$ instead of $\boldsymbol{\gamma}$ and ignores the effect of $\boldsymbol{\alpha}$. However, Thm 3.3 shows that using $\boldsymbol{\gamma}$ is the key to reduce CSS. Second, even for the estimation of $\mathbf{w}$, StruRW suffers from inaccurate estimation. In our notation, StruRW simply assumes that $\mathbb{P}_{\mathcal{S}}(\hat{Y} = i|Y = i) = 1, \forall i \in \mathcal{Y}$, i.e., perfect training in the source domain and uses hard pseudo-labels in the target domain to estimate $\mathbf{w}$. In contrast, our optimization to obtain $\mathbf{w}$ is more stable. Moreover, StruRW ignores the effect of LS entirely. From this perspective, StruRW can be understood as a special case of Pair-Align under the assumption of no LS and perfect prediction in the target graph. Furthermore, our work is the first to rigorously formulate the idea of conditional alignment in graphs.

4. Experiments

We evaluate three variants of Pair-Align to understand how its different components deal with the distribution shift on synthetic datasets and 5 real-world datasets. These variants include PA-CSS with only $\boldsymbol{\gamma}$ as source graph edge weights to address CSS, PA-LS with only $\boldsymbol{\beta}$ as label weights to address LS, and PA-BOTH that combines both. We next briefly introduce datasets and settings while leaving more details in Appendix E.

4.1. Datasets and Experimental Settings

Synthetic Data. CSBMs (see the definition in Appendix A) are used to generate the source and target graphs with three node classes. We explore four scenarios in structure shift without feature shift, where the first three explore CSS with shifts in the conditional neighboring node's label distribution (class ratio), shifts in the conditional node's degree distribution (degree), and shifts in both. Considering these three types of shift is inspired by the argument in Thm 3.3. The fourth setting examines CSS and LS jointly. In addition, we consider two degrees of shift under each scenario with the left column being the smaller shift as shown in Table 3. The detailed configurations of the CSBM regarding edge probabilities and node features are in Appendix E.2.

MAG We extract paper nodes and their citation links from the original MAG (Hu et al., 2020; Wang et al., 2020). Papers are split into separate graphs based on their countries of publication determined by their corresponding authors. The task is to classify the publication venue of the papers. Our experiments study generation across the top 6 countries with the most number of papers (in total 377k nodes, 1.35M edges). We train models on the graphs from US/China and test them on the graphs from the rest countries.

Pileup Mitigation (Liu et al., 2023) is a dataset of a denoising task in HEP named pileup mitigation (Bertolini et al., 2014). Proton-proton collisions produce particles with leading collisions (LC) and nearby bunch crossings as other collisions (OC). The task is to identify whether a particle is from LC or OC. Nodes are particles and particles are connected if they are close in the η-ϕ space. We study two distribution shifts: the shift of pile-up (PU) conditions (mostly structure shift), where PUk indicates the averaged number of other collisions in the beam is k, and the shift in the data generating process (primarily feature shift).

Arxiv (Hu et al., 2020) is a citation network of Arxiv papers to classify papers' subject areas. We study the shift in time by using papers published in earlier periods to train and test on papers published later. Specifically, we traine on papers published from 1950 to 2007/ 2009/ 2011 and test on paper published between 2014 to 2016 and 2016 to 2018.

DBLP and ACM (Tang et al., 2008; Wu et al., 2020) are two paper citation networks obtained from DBLP and ACM. Nodes are papers and edges represent citations between papers. The goal is to predict the research topic of a paper. We train the GNN on one network and test it on the other.

Baselines DANN (Ganin et al., 2016) and IWDAN (Ta-

Table 1. Performance on MAG datasets (accuracy scores). The **bold** font and underline indicate the best model and baseline respectively

DOMAINS	$US \to CN$	$US \to DE$	$US \to JP$	$US \to RU$	$US \to FR$	$CN \to US$	$CN \to DE$	$CN \to JP$	$CN \to RU$	$CN \to FR$
ERM	26.92 ± 1.08	26.37 ± 1.16	37.63 ± 0.36	21.71 ± 0.38	20.11 ± 0.34	31.47 ± 1.25	13.29 ± 0.36	22.15 ± 0.89	10.92 ± 0.82	10.86 ± 1.04
DANN	24.20 ± 1.19	26.29 ± 1.44	37.92 ± 0.25	21.76 ± 1.58	20.71 ± 0.29	30.23 ± 0.99	13.46 ± 0.40	21.48 ± 1.26	11.94 ± 1.90	10.65 ± 0.53
IWDAN	23.39 ± 0.93	25.97 ± 0.41	34.98 ± 0.68	22.80 ± 3.03	21.75 ± 0.81	31.72 ± 1.24	13.39 ± 1.06	19.86 ± 1.21	10.93 ± 1.33	11.64 ± 4.56
UDAGCN	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
STRURW	31.58 ± 3.10	30.03 ± 2.23	37.20 ± 0.27	28.97 ± 2.98	22.73 ± 1.73	37.08 ± 1.09	19.93 ± 1.82	29.76 ± 2.56	17.94 ± 9.82	15.81 ± 3.76
SPECREG	23.74 ± 1.32	26.68 ± 1.44	37.68 ± 0.25	21.47 ± 0.84	20.91 ± 0.53	26.52 ± 1.75	13.76 ± 0.65	20.50 ± 0.08	10.50 ± 0.53	10.45 ± 1.16
PA-CSS	37.93 ± 1.65	38.49 ± 2.66	47.38 ± 0.61	35.07 ± 10.2	$\mathbf{28.64} \pm 0.08$	43.28 ± 0.16	25.91 ± 2.70	37.42 ± 5.64	32.05 ± 0.81	22.83 ± 2.46
PA-LS	27.00 ± 0.50	26.89 ± 0.90	38.96 ± 0.94	21.42 ± 0.91	20.63 ± 0.45	31.21 ± 1.45	15.02 ± 1.04	23.22 ± 0.57	11.44 ± 0.57	11.16 ± 0.56
PA-BOTH	$\mathbf{40.06} \pm 0.99$	$\mathbf{38.85} \pm 4.71$	$\mathbf{47.43} \pm 1.82$	$\mathbf{37.07} \pm 5.28$	25.21 ± 3.79	$\mathbf{45.16} \pm 0.50$	$\mathbf{26.19} \pm 1.01$	$\mathbf{38.26} \pm 2.27$	$\mathbf{33.34} \pm 1.94$	$\mathbf{24.16} \pm 1.13$

Table 2. Performance on Pileup datasets (f1 scores). The **bold** font and underline indicate the best model and baseline respectively

	PILEUP CONDITIONS						PHYSICAL PROCESSES	
DOMAINS	PU10 → 30	PU30 → 10	PU10 → 50	PU50 → 10	PU30 → 140	PU140 → 30	$gg \to qq$	$qq \to gg$
ERM	48.17 ± 3.87	64.17 ± 1.50	48.73 ± 0.45	70.11 ± 1.12	18.76 ± 1.50	33.02 ± 28.77	67.70 ± 0.31	72.63 ± 0.54
DANN	49.99 ± 2.07	64.62 ± 0.70	48.44 ± 0.78	68.70 ± 1.42	28.20 ± 1.20	21.95 ± 20.37	66.48 ± 0.67	71.78 ± 0.87
IWDAN	35.85 ± 1.73	62.24 ± 0.15	26.49 ± 0.40	67.82 ± 0.62	8.91 ± 3.17	40.02 ± 1.93	66.85 ± 0.69	73.10 ± 0.29
UDAGCN	45.39 ± 2.07	62.27 ± 1.23	44.75 ± 1.76	68.93 ± 0.55	19.95 ± 0.84	29.66 ± 5.57	65.99 ± 1.06	71.99 ± 0.61
STRURW	52.41 ± 1.74	67.72 ± 0.22	47.25 ± 1.96	70.93 ± 0.66	37.81 ± 0.64	37.84 ± 2.82	67.66 ± 0.55	72.72 ± 0.68
SPECREG	52.61 ± 1.06	65.34 ± 0.62	48.85 ± 0.94	67.95 ± 2.23	28.86 ± 1.58	28.79 ± 25.83	66.66 ± 0.40	72.73 ± 0.42
PA-CSS	$\mathbf{56.00} \pm 0.14$	58.44 ± 3.19	50.77 ± 0.70	60.95 ± 6.09	40.31 ± 0.31	37.24 ± 7.69	67.75 ± 0.27	73.24 ± 0.38
PA-LS	46.84 ± 0.45	67.12 ± 0.65	48.51 ± 1.46	71.17 ± 0.70	36.29 ± 0.92	46.38 ± 0.96	67.63 ± 0.38	$\mathbf{73.40} \pm 0.13$
PA-BOTH	55.45 ± 0.21	$\mathbf{68.29} \pm 0.41$	$\mathbf{51.43} \pm 0.42$	$\mathbf{71.23} \pm 0.63$	$\mathbf{40.53} \pm 0.25$	$\mathbf{51.21} \pm 2.88$	$\mathbf{67.77} \pm 0.70$	73.36 ± 0.12

chet des Combes et al., 2020) are non-graph methods, we adapt them to the graph setting with GNN as the encoder. UDAGCN (Wu et al., 2020), StruRW (Liu et al., 2023) and SpecReg (You et al., 2023) are chosen as GDA baselines. We use GraphSAGE (Hamilton et al., 2017) as backbones and the same model architecture for all baselines.

Evaluation and Metric The source graph is used for training, 20 percent of the node labels in the target graph are used for validation and the rest 80 percent are held out for testing. We select the best model based on the target validation scores and report its scores on the target testing nodes in tables. We use accuracy for MAG, Arxiv, DBLP, ACM, and synthetic datasets. For the MAG datasets, we evaluate the top 19 classes as we group the remaining classes as a dummy class. The Pileup dataset uses the binary f1 score.

Hyperparameter Study Our hyperparameter tuning is mainly for the robustness estimation for γ and β detailed in section 3.5. We will discuss them in Appendix E.3.

4.2. Result Analysis

In the MAG dataset, Pair-Align methods markedly outperform baselines, as detailed in Table 1. Most baselines generally match the performance of ERM suggesting their limited effectiveness in addressing CSS and LS. StruRW, however, stands out, emphasizing the need for CSS mitigation in MAG. When compared to StruRW, Pair-Align not only demonstrates superior handling of CSS but also offers advantages in LS mitigation, resulting in over 25% relative improvements. Also, IWDAN has not shown improvements due to the suboptimality of performing only conditional feature alignment yet ignoring the structure, highlighting the importance of tailored solutions for GDA like Pair-Align.

HEP results are in Table 2. Considering the shift in pileup (PU) conditions, baselines with graph structure regularization, like StruRW and SpecReg, achieve better performance. This matches our expectations that PU condition shifts introduce mostly structure shifts as shown in Fig 1 and our methods further significantly outperform these baselines in addressing such shifts. Specifically, we observe PA-CSS excels in transitioning from low PU to high PU, while PA-LS is more effective in the opposite direction. This difference stems from the varying dominant impacts of LS and CSS. High PU datasets have more imbalanced label distribution with a large OC: LC ratio, where LS induces more negative effects over CSS, necessitating the LS mitigation. Conversely, the cases from low PU to high PU, mainly influenced by CSS, can be addressed better by PA-CSS. Regarding shifts in physical processes, Pair-Align methods still rank the best, but all models have close performance since structure shift now becomes minor as shown in Table 9.

The synthetic dataset results in Table 3 well justify our theory. We observe minimal performance decay with ERM in scenarios with only degree shifts, indicating that node degree impacts are minimal under mean pooling in GNNs. Additionally, while CSS with both shifts results in lower ERM performance compared to shift only in class ratio, our Pair-Align method achieves similar performance, highlighting the adequacy of focusing on shifts in the conditional neighborhood node label distribution for CSS. Pair-Align notably outperforms baselines in CSS scenarios, especially where class ratio shifts are more pronounced (as in the second case of each scenario). With joint shifts in CSS and LS, Pair-Align methods perform the best and IWDAN is the best baseline as it is designed to address conditional shifts and LS in non-graph tasks.

For the Arxiv and DBLP/ACM datasets in Table 4, the Pair-Align methods demonstrate reasonable improvements over baselines. Regarding the Arxiv dataset, Pair-Align is particularly effective when the training on pre-2007 papers, which

Table 3. Synthetic CSBM results (accuracy). The **bold** font and the underline indicate the best model and baseline respectively

	CSS (ONLY CLASS RATIO SHIFT)		CSS (ONLY DEGREE SHIFT)		CSS (SHIFT IN BOTH)		CSS + LS	
ERM	94.22 ± 0.97	57.04 ± 3.83	99.01 ± 0.28	96.21 ± 0.27	88.90 ± 0.22	58.01 ± 1.91	61.35 ± 4.64	61.65 ± 0.80
IWDAN	95.85 ± 0.70	76.75 ± 1.32	98.97 ± 0.05	$\mathbf{97.15} \pm 0.33$	$\underline{93.65} \pm 0.70$	79.53 ± 3.57	$\underline{92.42} \pm 0.72$	$\underline{87.01} \pm 2.14$
UDAGCN	96.82 ± 0.70	69.93 ± 5.17	$\mathbf{99.52} \pm 0.05$	$\underline{97.04} \pm 0.28$	93.17 ± 1.02	67.44 ± 4.95	87.67 ± 3.21	83.69 ± 2.35
STRURW	$\underline{96.83} \pm 0.33$	$\underline{86.65} \pm 5.62$	98.87 ± 0.19	95.93 ± 0.55	92.09 ± 0.55	$\underline{80.00} \pm 7.49$	75.38 ± 12.11	75.96 ± 2.96
SPECREG	93.46 ± 1.21	62.97 ± 1.01	98.94 ± 0.03	96.69 ± 0.23	89.58 ± 1.58	61.28 ± 1.19	76.73 ± 3.18	83.40 ± 1.38
PA-CSS	96.65 ± 1.21	91.79 ± 1.68	98.92 ± 0.52	96.24 ± 0.23	94.99 ± 0.49	91.20 ± 0.95	94.95 ± 0.69	$\mathbf{95.66} \pm 0.45$
PA-LS	94.22 ± 0.95	57.14 ± 3.73	$\underline{99.02} \pm 0.29$	96.17 ± 0.26	88.85 ± 0.22	57.96 ± 1.84	61.39 ± 4.59	67.91 ± 9.98
PA-BOTH	$\mathbf{97.24} \pm 0.33$	$\mathbf{91.97} \pm 1.49$	98.20 ± 1.04	96.25 ± 0.33	$\mathbf{95.44} \pm 0.51$	$\mathbf{91.67} \pm 0.38$	$\mathbf{95.24} \pm 0.11$	95.55 ± 0.65

Table 4. Performance on Arxiv and DBLP/ACM datasets (accuracy). The **bold** and underline indicate the best model and baseline

	1950-2007		1950-2009		1950-2011		DBLP AND ACM	
DOMAINS	$2014-2016$	$2016-2018$	$2014-2016$	$2016-2018$	$2014-2016$	$2016-2018$	$A \to D$	$D \to A$
ERM	37.91 ± 0.31	35.22 ± 0.71	43.50 ± 0.35	40.19 ± 3.62	51.76 ± 0.93	52.56 ± 1.06	57.26 ± 1.90	47.77 ± 6.61
DANN	37.31 ± 1.54	36.84 ± 1.40	$\underline{43.57} \pm 0.47$	42.04 ± 2.70	53.02 ± 0.67	52.69 ± 1.26	65.34 ± 5.91	54.36 ± 6.20
IWDAN	36.16 ± 2.91	25.48 ± 9.77	41.26 ± 2.08	35.91 ± 4.28	46.73 ± 0.62	42.70 ± 3.21	$\underline{66.96} \pm 7.38$	56.13 ± 6.48
UDAGCN	38.10 ± 1.62	OOM	42.85 ± 2.09	OOM	53.13 ± 0.31	OOM	57.05 ± 5.43	$\underline{58.42} \pm 6.65$
STRURW	$\underline{38.56} \pm 0.77$	$\underline{37.17} \pm 2.75$	43.55 ± 2.37	$\underline{43.55} \pm 2.37$	$\underline{53.19} \pm 0.45$	$\mathbf{53.64} \pm 0.65$	60.03 ± 2.18	52.13 ± 1.25
SPECREG	37.09 ± 0.62	33.46 ± 0.83	43.14 ± 2.16	43.06 ± 1.09	52.63 ± 1.29	52.46 ± 0.83	31.03 ± 2.45	53.04 ± 2.21
PA-CSS	39.75 ± 0.96	40.54 ± 2.44	44.04 ± 0.83	44.32 ± 1.61	$\mathbf{53.75} \pm 0.48$	51.10 ± 1.30	65.20 ± 3.69	60.60 ± 3.86
PA-LS	39.47 ± 0.88	$\mathbf{41.14} \pm 2.07$	43.40 ± 1.97	43.44 ± 1.65	52.48 ± 0.53	$\underline{52.83} \pm 0.98$	$\mathbf{72.41} \pm 1.29$	61.40 ± 1.92
PA-BOTH	$\mathbf{39.98} \pm 0.77$	40.23 ± 0.30	$\mathbf{44.60} \pm 0.42$	$\mathbf{44.43} \pm 0.34$	53.56 ± 0.98	51.60 ± 0.24	70.97 ± 3.87	$\mathbf{63.36} \pm 2.90$

possess larger shifts as shown in Table 10. Also, all baselines perform similarly with no significant gap between the GDA methods and the non-graph methods, suggesting that addressing structure shift has limited benefits in this dataset. Likewise, regarding the DBLP and ACM datasets, we observe the performance gain of methods that align marginal node feature distribution, like DANN and UDAGCN, indicating this dataset contains mostly feature shifts. While in the cases where LS is large ($A \to D$ or Arxiv training on pre-2007, testing on 2016-2018 as shown in Table 10), PA-LS achieves the best performance.

Ablation Study

Among the three variants of Pair-Align, PA-BOTH performs the best in most cases. PA-CSS contributes more compared to PA-LS when CSS dominates (MAG datasets, Arxiv, and HEP from low PU to high PU). PA-LS alone offers slight improvements except with highly imbalanced training labels (from high PU to low PU in HEP datasets). But when combined with PA-CSS, it will yield additional benefits.

5. Conclusion

This work studies the distribution shifts in graph-structured data. We analyze distribution shifts in real-world graph data and decompose structure shifts into two components: conditional structure shift (CSS) and label shift (LS). Our novel approach, Pairwise Alignment (Pair-Align), well tackles both CSS and LS in both theory and practice. Importantly, this work also curates a new, by far the largest dataset MAG which reflects the actual need for region-based generalization of graph learning models. We believe this large dataset can incentivize more in-depth studies on GDA.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

We greatly thank Yongbin Feng for discussing relevant HEP applications and Mufei Li for discussing relevant MAG dataset curation. S. Liu, D. Zou, and P. Li are partially supported by NSF award PHY-2117997 and IIS-2239565. The work of HZ was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement Number: HR00112320012 and a research grant from the IBM-Illinois Discovery Accelerator Institute (IIDAI).

References

Azizzadenesheli, K., Liu, A., Yang, F., and Anandkumar, A. Regularized learning for domain adaptation under label shifts. *International Conference on Learning Representations*, 2018.

Bertolini, D., Harris, P., Low, M., and Tran, N. Pileup per particle identification. *Journal of High Energy Physics*, 2014.

Bevilacqua, B., Zhou, Y., and Ribeiro, B. Size-invariant graph representations for graph classification extrapolations. *International Conference on Machine Learning*, 2021.

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. *Advances in Neural Information Processing Systems*, 2022.

Chen, Y., Bian, Y., Zhou, K., Xie, B., Han, B., and Cheng, J. Does invariant graph learning via environment augmentation learn invariance? *Advances in Neural Information Processing Systems*, 2023.

Chuang, C.-Y. and Jegelka, S. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. *Advances in Neural Information Processing Systems*, 2022.

Cicek, S. and Soatto, S. Unsupervised domain adaptation via regularized conditional alignment. *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

Dai, Q., Wu, X.-M., Xiao, J., Shen, X., and Wang, D. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

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

Ding, M., Kong, K., Chen, J., Kirchenbauer, J., Goldblum, M., Wipf, D., Huang, F., and Goldstein, T. A closer look at distribution shifts and out-of-distribution generalization on graphs. *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., and Yu, P. S. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020.

Fan, S., Wang, X., Mo, Y., Shi, C., and Tang, J. Debiasing graph neural networks via learning disentangled causal substructure. *Advances in Neural Information Processing Systems*, 2022.

Fan, S., Wang, X., Shi, C., Cui, P., and Wang, B. Generalizing graph neural networks on out-of-distribution graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

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*, 2016.

Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., and Schölkopf, B. Domain adaptation with conditional transferable components. *International Conference on Machine Learning*, 2016.

Gui, S., Liu, M., Li, X., Luo, Y., and Ji, S. Joint learning of label and environment causal independence for graph out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 2017.

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

Highfield, R. Large hadron collider: Thirteen ways to change the world. *The Daily Telegraph. London. Retrieved*, 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*, 2020.

Jackson, M. O. et al. *Social and economic networks*, volume 3. Princeton university press Princeton, 2008.

Ji, Y., Zhang, L., Wu, J., Wu, B., Li, L., Huang, L.-K., Xu, T., Rong, Y., Ren, J., Xue, D., et al. Drugood: Out-of-distribution dataset curator and benchmark for ai-aided drug discovery–a focus on affinity prediction problems with noise annotations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Jia, T., Li, H., Yang, C., Tao, T., and Shi, C. Graph invariant learning with subgraph co-mixup for out-of-distribution generalization. *arXiv preprint arXiv:2312.10988*, 2023.

Jin, W., Zhao, T., Ding, J., Liu, Y., Tang, J., and Shah, N. Empowering graph representation learning with test-time graph transformation. *International Conference on Learning Representations*, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2016.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., et al. Wilds: A benchmark of in-the-wild distribution shifts. *International Conference on Machine Learning*, 2021.

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

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

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*, 2022b.

Liao, P., Zhao, H., Xu, K., Jaakkola, T., Gordon, G. J., Jegelka, S., and Salakhutdinov, R. Information obfuscation of graph neural networks. *International Conference on Machine Learning*, 2021.

Ling, H., Jiang, Z., Liu, M., Ji, S., and Zou, N. Graph mixup with soft alignments. *International Conference on Machine Learning*, 2023.

Lipton, Z., Wang, Y.-X., and Smola, A. Detecting and correcting for label shift with black box predictors. *International Conference on Machine Learning*, 2018.

Liu, M., Fang, Z., Zhang, Z., Gu, M., Zhou, S., Wang, X., and Bu, J. Rethinking propagation for unsupervised graph domain adaptation. *arXiv preprint arXiv:2402.05660*, 2024.

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

Liu, X., Guo, Z., Li, S., Xing, F., You, J., Kuo, C.-C. J., El Fakhri, G., and Woo, J. Adversarial unsupervised domain adaptation with conditional and label shift: Infer, align and iterate. *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

Long, M., Cao, Y., Wang, J., and Jordan, M. Learning transferable features with deep adaptation networks. *International Conference on Machine Learning*, 2015.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. Conditional adversarial domain adaptation. *Advances in Neural Information Processing Systems*, 2018.

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

Pang, J., Wang, Z., Tang, J., Xiao, M., and Yin, N. Sagda: Spectral augmentation for graph domain adaptation. *Proceedings of the 31st ACM International Conference on Multimedia*, 2023.

Peters, J., Janzing, D., and Schölkopf, B. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Rojas-Carulla, M., Schölkopf, B., Turner, R., and Peters, J. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 2018.

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

Shen, X., Dai, Q., Mao, S., Chung, F.-l., and Choi, K.-S. Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 2020b.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

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

Sui, Y., Wu, Q., Wu, J., Cui, Q., Li, L., Zhou, J., Wang, X., and He, X. Unleashing the power of graph data augmentation on covariate distribution shift. *Advances in Neural Information Processing Systems*, 2023.

Szklarczyk, D., Gable, A. L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N. T., Morris, J. H., Bork, P., et al. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 2019.

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*, 2020.

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

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations*, 2018.

Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., Yu, Q., Zhou, J., Yang, S., and Qi, Y. A semi-supervised graph attentive network for financial fraud detection. *IEEE International Conference on Data Mining*, 2019.

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., and Kanakia, A. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020.

Wang, Q., Wang, Y., and Ying, X. Improved invariant learning for node-level out-of-distribution generalization on graphs. *Submitted to The Twelfth International Conference on Learning Representations*, 2023.

Wang, Y., Wang, W., Liang, Y., Cai, Y., and Hooi, B. Mixup for node and graph classification. *Proceedings of the Web Conference*, 2021.

Wei, R., Yin, H., Jia, J., Benson, A. R., and Li, P. Understanding non-linearity in graph neural networks from the bayesian-inference perspective. *Advances in Neural Information Processing Systems*, 2022.

Wu, J., He, J., and Ainsworth, E. Non-iid transfer learning on graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

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

Wu, Q., Zhang, H., Yan, J., and Wipf, D. Handling distribution shifts on graphs: An invariance perspective. *International Conference on Learning Representations*, 2022.

Wu, Y., Wang, X., Zhang, A., He, X., and Chua, T.-S. Discovering invariant rationales for graph neural networks. *International Conference on Learning Representations*, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *International Conference on Learning Representations*, 2018.

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

Yehudai, G., Fetaya, E., Meirom, E., Chechik, G., and Maron, H. From local structures to size generalization in graph neural networks. *International Conference on Machine Learning*, 2021.

Yin, N., Shen, L., Li, B., Wang, M., Luo, X., Chen, C., Luo, Z., and Hua, X.-S. Deal: An unsupervised domain adaptive framework for graph-level classification. *Proceedings of the 30th ACM International Conference on Multimedia*, 2022.

Yin, N., Shen, L., Wang, M., Lan, L., Ma, Z., Chen, C., Hua, X.-S., and Luo, X. Coco: A coupled contrastive framework for unsupervised domain adaptive graph classification. *Internationl Conference on Machine Learning*, 2023.

You, Y., Chen, T., Wang, Z., and Shen, Y. Graph domain adaptation via theory-grounded spectral regularization. *International Conference on Learning Representations*, 2023.

Yu, J., Xu, T., Rong, Y., Bian, Y., Huang, J., and He, R. Graph information bottleneck for subgraph recognition. *International Conference on Learning Representations*, 2020.

Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., and Saminger-Platz, S. Central moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2016.

Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. Domain adaptation under target and conditional shift. *International Conference on Machine Learning*, 2013.

Zhang, X., Du, Y., Xie, R., and Wang, C. Adversarial separation network for cross-network node classification. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

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

Zhao, H., Zhang, S., Wu, G., Moura, J. M., Costeira, J. P., and Gordon, G. J. Adversarial multiple source domain adaptation. *Advances in Neural Information Processing Systems*, 2018.

Zhao, H., Des Combes, R. T., Zhang, K., and Gordon, G. On learning invariant representations for domain adaptation. *International Conference on Machine Learning*, 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*, 2021.

Zhu, Q., Zhang, C., Park, C., Yang, C., and Han, J. Shift-robust node classification via graph adversarial clustering. *arXiv preprint arXiv:2203.15802*, 2022.

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

A. Some Definitions

Definition A.1 (Contextual Stochastic Block Model)**.** (Deshpande et al., 2018)

The Contextual Stochastic Block Model (CSBM) is a framework combining the stochastic block model with node features for random graph generation. A CSBM with nodes belonging to k classes is defined by parameters $(n, \mathbf{B}, \mathbb{P}_0, \ldots, \mathbb{P}_{k-1})$, where n represents the total number of nodes. The matrix $\mathbf{B}$, a $k \times k$ matrix, denotes the edge connection probability between nodes of different classes. Each $\mathbb{P}i$ (for $0 \leq i < k$) characterizes the feature distribution of nodes from class i. In a graph generated from CSBM, the probability that an edge exists between a node u from class i and a node v from class j is specified by B_{ij}, an element of $\mathbf{B}$. For undirected graphs, $\mathbf{B}$ is symmetric, i.e., $\mathbf{B} = \mathbf{B}^\top$. In CSBM, node features and edges are generated independently, conditioned on node labels.

B. Omitted Proofs

B.1. Proof for Theorem 3.3

Theorem 3.3 (Sufficient conditions for addressing CSS)**.** *Given the following assumptions*

- *(Conditional Alignment in the previous layer k)* $\mathbb{P}_{\mathcal{S}}(H^{(k)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k)}|Y)$ *and* $\forall u \in \mathcal{V}_{\mathcal{U}}$*, given* $Y = y_u$*,* $h_u^{(k)}$ *is independently sampled from* $\mathbb{P}_{\mathcal{U}}(H^{(k)}|Y)$*.*
- *(Edge Conditional Independence) Given node labels* $\mathbf{y}$*, edges mutually independently exist in the graph.*

if there exists a transformation that modifies the neighborhood of node u: $\mathcal{N}_u \to \tilde{\mathcal{N}}_u, \forall u \in \mathcal{V}_{\mathcal{S}}$, such that $\mathbb{P}_{\mathcal{S}}(|\tilde{\mathcal{N}}_u||Y_u = i) = \mathbb{P}_{\mathcal{T}}(|\mathcal{N}_u||Y_u = i)$ and $\mathbb{P}_{\mathcal{S}}(Y_v|Y_u = i, v \in \tilde{\mathcal{N}}_u) = \mathbb{P}_{\mathcal{T}}(Y_v|Y_u = i, v \in \mathcal{N}_u)$, $\forall i \in \mathcal{Y}$, then $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$ is satisfied.

Proof. We analyze the distribution $\mathbb{P}_{\mathcal{U}}(H^{(k+1)}|Y)$ to see which distributions should be aligned to achieve $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$. Since $h_u^{(k+1)} = \text{UPT}\left(h_u^{(k)}, \text{AGG}\left(\{\!\{h_v^{(k)} : v \in \mathcal{N}_u\}\!\}\right)\right)$, $\mathbb{P}_{\mathcal{U}}(H^{(k+1)}|Y)$ can be expanded as follows:

$$
\begin{aligned}
&\mathbb{P}_{\mathcal{U}}(h_u^{(k)}, \{\!\{h_v^{(k)} : v \in \mathcal{N}_u\}\!\}|Y_u = i) \\
&\overset{(a)}{=} \mathbb{P}_{\mathcal{U}}(h_u^{(k)}|Y_u = i)\mathbb{P}_{\mathcal{U}}(\{\!\{h_v^{(k)} : v \in \mathcal{N}_u\}\!\}|Y_u = i) \\
&= \mathbb{P}_{\mathcal{U}}(h_u^{(k)}|Y_u = i)\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u| = d|Y_u = i)\mathbb{P}_{\mathcal{U}}(\{\!\{h_v^{(k)}\}\!\}|Y_u = i, v \in \mathcal{N}_u, |\mathcal{N}_u| = d) \\
&= \mathbb{P}_{\mathcal{U}}(h_u^{(k)}|Y_u = i)\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u| = d|Y_u = i)\mathbb{P}_{\mathcal{U}}(\{\!\{h_{v_1}^{(k)}, \cdots h_{v_d}^{(k)}\}\!\}|Y_u = i, v_t \in \mathcal{N}_u, \text{for } t \in [1, d]) \\
&\overset{(b)}{=} \mathbb{P}_{\mathcal{U}}(h_u^{(k)}|Y_u = i)\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u| = d|Y_u = i)(d!)\prod_{t=1}^{d}\mathbb{P}_{\mathcal{U}}(h_{v_t}^{(k)}|h_{v_{1:t-1}}^{(k)}, Y_u = i, v_t \in \mathcal{N}_u) \\
&= \mathbb{P}_{\mathcal{U}}(h_u^{(k)}|Y_u = i)\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u| = d|Y_u = i)(d!)\prod_{t=1}^{d}(\sum_{j\in\mathcal{Y}}\mathbb{P}_{\mathcal{U}}(h_{v_t}^{(k)}|Y_{v_t} = j, h_{v_{1:t-1}}^{(k)}, Y_u = i, v_t \in \mathcal{N}_u) \\
&\quad \mathbb{P}_{\mathcal{U}}(Y_{v_t} = j|h_{v_{1:t-1}}^{(k)}, Y_u = i, v_t \in \mathcal{N}_u)) \\
&\overset{(c)}{=} \mathbb{P}_{\mathcal{U}}(h_u^{(k)}|Y_u = i)\mathbb{P}_{\mathcal{U}}(|\mathcal{N}_u| = d|Y_u = i)(d!)\prod_{t=1}^{d}(\sum_{j\in\mathcal{Y}}\mathbb{P}_{\mathcal{U}}(h_{v_t}^{(k)}|Y_{v_t} = j)\mathbb{P}_{\mathcal{U}}(Y_{v_t} = j|Y_u = i, v_t \in \mathcal{N}_u))
\end{aligned} \tag{9}
$$

(a) is based on the assumption that node attributes and edges are conditionally independent of others given the node labels. (b), here we suppose that the observed messages h_v are different $\forall v \in \mathcal{N}_u$, and this assumption does not affect the result of the theorem. If some of them are identical, we modify the coefficient $d!$ as $\frac{d!}{\Pi_{t=1}^{d} m_i!}$, where m_t denotes the repeated messages. For simplicity, we assume that $m_t = 1, \forall t \in [1, d]$. (c) is based on the assumption that given $Y = y_u$, $h_u^{(k)}$ is independently sampled from $\mathbb{P}_{\mathcal{U}}(H^{(k)}|Y)$

With the goal to achieve $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$, it suffices to achieve by making the input distribution equal across the source and the target

$$
\mathbb{P}_{\mathcal{S}}(h_u^{(k)}, \{\!\{h_u^{(k)} : v \in \mathcal{N}_u\}\!\}|Y_u = i) = \mathbb{P}_{\mathcal{T}}(h_u^{(k)}, \{\!\{h_u^{(k)} : v \in \mathcal{N}_u\}\!\}|Y_u = i)
$$

since the source and target graphs undergo the same set of functions. Based on Eq. (9),this means it suffices to let $\mathbb{P}_{\mathcal{S}}(h_u^{(k)}|Y_u=i)=\mathbb{P}_{\mathcal{T}}(h_u^{(k)}|Y_u=i)$ and $\mathbb{P}_{\mathcal{S}}(h_{v_t}^{(k)}|Y_{v_t}=j)=\mathbb{P}_{\mathcal{T}}(h_{v_t}^{(k)}|Y_{v_t}=j)$ since $\mathbb{P}_{\mathcal{S}}(H^{(k)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(k)}|Y)$ is assumed to be true. Therefore, as long as there exists a transformation that modifies the $\mathcal{N}_u \to \tilde{\mathcal{N}}_u$ such that

$$\mathbb{P}_{\mathcal{S}}(|\tilde{\mathcal{N}}_u|=d|Y_u=i)=\mathbb{P}_{\mathcal{T}}(|\mathcal{N}_u|=d|Y_u=i); \quad \mathbb{P}_{\mathcal{S}}(Y_v=j|Y_u=i,v\in\tilde{\mathcal{N}}_u))=\mathbb{P}_{\mathcal{T}}(Y_v=j|Y_u=i,v\in\mathcal{N}_u))$$

Then, $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$ □

Remark B.1. Iteratively, we can achieve $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ given no feature shift initially $\mathbb{P}_{\mathcal{S}}(X|Y)=\mathbb{P}_{\mathcal{T}}(X|Y)$ as $\mathbb{P}_{\mathcal{S}}(H^{(1)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(1)}|Y)$

$$\text{base case: } \mathbb{P}_{\mathcal{S}}(H^{(1)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(1)}|Y)\Rightarrow\mathbb{P}_{\mathcal{S}}(H^{(2)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(2)}|Y)$$
$$\text{inductive step: } \mathbb{P}_{\mathcal{S}}(H^{(k)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(k)}|Y),\overset{(d)}{\Rightarrow}\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$$
$$\text{Therefore, } \mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y).$$

(d) is proved above that when using a multiset transformation to align two distributions, this can be guaranteed

Under the assumption that given $Y=y_u$, $h_u^{(k)}$ is independently sampled from $\mathbb{P}_{\mathcal{U}}(H^{(k)}|Y)$, $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ can induce $\mathbb{P}_{\mathcal{S}}(\mathbf{H}^{(L)}|\mathbf{Y})=\mathbb{P}_{\mathcal{T}}(\mathbf{H}^{(L)}|\mathbf{Y})$ since $\mathbb{P}(\mathbf{H}^{(L)}=\mathbf{h}^{(L)}|\mathbf{Y}=\mathbf{y})=\Pi_{u\in\mathcal{V}}\mathbb{P}(H^{(L)}=h_u|Y=y_u)$

B.2. Proof for Lemma 3.8

Lemma B.2. *If $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied, and node representations are conditionally independent of graph structures given node labels, then $\boldsymbol{\nu}=\boldsymbol{\Sigma}\mathbf{w}$.*

Proof.

$$\begin{aligned}
\mathbb{P}_{\mathcal{T}}(\hat{Y}_u=i,\hat{Y}_v=j|A_{uv}=1) &= \sum_{i',j'\in\mathcal{Y}}\mathbb{P}_{\mathcal{T}}(\hat{Y}_u=i,\hat{Y}_v=j|Y_u=i',Y_v=j',A_{uv}=1)\mathbb{P}_{\mathcal{T}}(Y_u=i',Y_v=j'|A_{uv}=1)\\
&\overset{(a)}{=}\sum_{i',j'\in\mathcal{Y}}\mathbb{P}_{\mathcal{T}}(\hat{Y}_u=i|Y_u=i')\mathbb{P}_{\mathcal{T}}(\hat{Y}_v=j|Y_v=j')\mathbb{P}_{\mathcal{T}}(Y_u=i',Y_v=j'|A_{uv}=1)\\
&\overset{(b)}{=}\sum_{i',j'\in\mathcal{Y}}\mathbb{P}_{\mathcal{S}}(\hat{Y}_u=i|Y_u=i')\mathbb{P}_{\mathcal{S}}(\hat{Y}_v=j|Y_v=j')\mathbb{P}_{\mathcal{T}}(Y_u=i',Y_v=j'|A_{uv}=1)\\
&=\sum_{i',j'\in\mathcal{Y}}\mathbb{P}_{\mathcal{S}}(\hat{Y}_u=i,\hat{Y}_v=j|Y_u=i',Y_v=j',A_{uv}=1)\mathbb{P}_{\mathcal{T}}(Y_u=i',Y_v=j'|A_{uv}=1)\\
&=\sum_{i',j'\in\mathcal{Y}}\mathbb{P}_{\mathcal{S}}(\hat{Y}_u=i,\hat{Y}_v=j,Y_u=i',Y_v=j'|A_{uv}=1)\frac{\mathbb{P}_{\mathcal{T}}(Y_u=i',Y_v=j'|A_{uv}=1)}{\mathbb{P}_{\mathcal{S}}(Y_u=i',Y_v=j'|A_{uv}=1)}\\
&=\sum_{i',j'\in\mathcal{Y}}[\boldsymbol{\Sigma}]_{ij,i'j'}[\mathbf{w}]_{i'j'}
\end{aligned}$$

(a) is because $\hat{y}_u=g(h_u^{(L)})$ and the assumption that node representations and graph structures are conditionally independent of others given the node labels. And (b) is achieved since $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied, such that $\mathbb{P}_{\mathcal{S}}(g(h_u^{(L)})=i|Y_u=i')=\mathbb{P}_{\mathcal{T}}(g(h_u^{(L)})=i|Y_u=i'),\forall i'\in\mathcal{Y}$ □

B.3. Proof for Lemma 3.11

Lemma B.3. *If $\mathbb{P}_{\mathcal{S}}(H^{(L)}|Y)=\mathbb{P}_{\mathcal{T}}(H^{(L)}|Y)$ is satisfied, and node representations are conditionally independent of each other given the node labels, then $\boldsymbol{\mu}=\mathbf{C}\boldsymbol{\beta}$.*

Proof.

$$\begin{aligned}\mathbb{P}_{\mathcal{T}}(\hat{Y}_u = i) &= \sum_{i' \in \mathcal{Y}} \mathbb{P}_{\mathcal{T}}(\hat{Y}_u = i | Y_u = i') \mathbb{P}_{\mathcal{T}}(Y_u = i') \\ &\overset{(a)}{=} \sum_{i' \in \mathcal{Y}} \mathbb{P}_{\mathcal{S}}(\hat{Y}_u = i | Y_u = i') \mathbb{P}_{\mathcal{T}}(Y_u = i') \\ &= \sum_{i' \in \mathcal{Y}} \mathbb{P}_{\mathcal{S}}(\hat{Y}_u = i, Y_u = i') \frac{\mathbb{P}_{\mathcal{T}}(Y_u = i')}{\mathbb{P}_{\mathcal{S}}(Y_u = i')} \\ &= \sum_{i' \in \mathcal{Y}} [\mathbf{C}]_{i,i'} [\boldsymbol{\beta}]_{i'}\end{aligned}$$

(a) is because, when $\mathbb{P}_{\mathcal{S}}(H^{(k+1)}|Y) = \mathbb{P}_{\mathcal{T}}(H^{(k+1)}|Y)$ is satisfied, $\mathbb{P}_{\mathcal{S}}(g(h_u^{(L)}) = i | Y_u = i') = \mathbb{P}_{\mathcal{T}}(g(h_u^{(L)}) = i | Y_u = i'), \forall i' \in \mathcal{Y}$ □

C. Algorithm Details

C.1. Details in optimization for γ

C.1.1. Empirical estimation of $\boldsymbol{\Sigma}$ and $\boldsymbol{\nu}$ in matrix form

For the least square problem that solves for $\mathbf{w}$

$$\boldsymbol{\Sigma}\mathbf{w} = \boldsymbol{\nu}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{|\mathcal{Y}|^2 \times |\mathcal{Y}|^2}$, $\mathbf{w} \in \mathbb{R}^{|\mathcal{Y}|^2 \times 1}$, $\boldsymbol{\nu} \in \mathbb{R}^{|\mathcal{Y}|^2 \times 1}$

Empirically, we estimate the value of $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\nu}}$ as following:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{|\mathcal{E}_{\mathcal{S}}|} \mathbf{E}^{\mathcal{S}} \mathbf{M}^{\mathcal{S}}$$

$\mathbf{E}^{\mathcal{S}} \in \mathbb{R}^{|\mathcal{Y}|^2 \times |\mathcal{E}_{\mathcal{S}}|}$, where each column represents the joint distribution of the classes prediction associated with the starting and ending node of each edge in the source graph. $[\mathbf{E}^{\mathcal{S}}]_{:,uv} = g(h_u^{(L)}) \otimes g(h_v^{(L)}), \forall \text{edge } uv \in \mathcal{E}_{\mathcal{S}}$. And each entry $[\mathbf{E}^{\mathcal{S}}]_{ij,uv} = [g(h_u^{(L)})]_i \times [g(h_v^{(L)})]_j, \forall i, j \in \mathcal{Y}$. $\mathbf{M}^{\mathcal{S}} \in \mathbb{R}^{|\mathcal{E}_{\mathcal{S}}| \times |\mathcal{Y}|^2}$ encodes the ground truth of the starting and ending node of an edge, as $[\mathbf{M}^{\mathcal{S}}]_{uv,y_u y_v} = 1$ for each edge $uv \in \mathcal{E}_{\mathcal{S}}$.

$$\hat{\boldsymbol{\nu}} = \frac{1}{|\mathcal{E}_{\mathcal{T}}|} \mathbf{E}^{\mathcal{T}} \mathbf{1}$$

Similarly, $\mathbf{E}^{\mathcal{T}} \in \mathbb{R}^{|\mathcal{Y}|^2 \times |\mathcal{E}_{\mathcal{T}}|}$, where each column represents the joint distribution of the classes prediction associated with the starting and ending node of each edge in the target graph. $[\mathbf{E}^{\mathcal{T}}]_{:,uv} = g(h_u^{(L)}) \otimes g(h_v^{(L)}), \forall \text{edge } uv \in \mathcal{E}_{\mathcal{T}}$. And each entry $[\mathbf{E}^{\mathcal{T}}]_{ij,uv} = [g(h_u^{(L)})]_i \times [g(h_v^{(L)})]_j, \forall i, j \in \mathcal{Y}$. $\mathbf{1} \in \mathbb{R}^{|\mathcal{E}_{\mathcal{T}}| \times 1}$ is the all one vector.

C.1.2. Calculate for $\boldsymbol{\alpha}$ in matrix form

To finally solve for the ratio weight $\boldsymbol{\gamma}$, we need the value $\boldsymbol{\alpha}$.

$$\begin{aligned}\boldsymbol{\alpha}_i = \frac{\mathbb{P}_{\mathcal{T}}(y_u = i | A_{uv} = 1)}{\mathbb{P}_{\mathcal{S}}(y_u = i | A_{uv} = 1)} &= \frac{\sum_j \mathbb{P}_{\mathcal{T}}(y_u = i, y_v = j | A_{uv} = 1)}{\sum_j \mathbb{P}_{\mathcal{S}}(y_u = i, y_v = j | A_{uv} = 1)} \\ &= \frac{\sum_j \frac{\mathbb{P}_{\mathcal{T}}(y_u = i, y_v = j | A_{uv} = 1)}{\mathbb{P}_{\mathcal{S}}(y_u = i, y_v = j | A_{uv} = 1)} \mathbb{P}_{\mathcal{S}}(y_u = i, y_v = j | A_{uv} = 1)}{\sum_j \mathbb{P}_{\mathcal{S}}(y_u = i, y_v = j | A_{uv} = 1)} \\ &= \frac{\sum_j \frac{\mathbb{P}_{\mathcal{T}}(y_u = i, y_v = j | A_{uv} = 1)}{\mathbb{P}_{\mathcal{S}}(y_u = i, y_v = j | A_{uv} = 1)} \mathbb{P}_{\mathcal{S}}(y_u = i, y_v = j | A_{uv} = 1)}{\mathbb{P}_{\mathcal{S}}(y_u = i | A_{uv} = 1)}\end{aligned}$$

In matrix form, we construct $\mathbf{K} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|^2}$, where $[K]_{i,ij} = \frac{\mathbb{P}_{\mathcal{S}}(y_u=i,y_v=j|A_{uv}=1)}{\mathbb{P}_{\mathcal{S}}(y_u=i|A_{uv}=1)}, \forall i,j \in |\mathcal{Y}|$. Note that $[K]_{i,i'j} = 0$ for $i' \neq i, \forall j \in |\mathcal{Y}|$.

$$\boldsymbol{\alpha} = \mathbf{K}\mathbf{w}$$

C.2. Details in optimization for $\boldsymbol{\beta}$

For the least square problem that solves for $\boldsymbol{\beta}$

$$\mathbf{C}\boldsymbol{\beta} = \boldsymbol{\mu}$$

where $\mathbf{C} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$, $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{Y}|\times 1}$, $\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{Y}|\times 1}$

Empirically, we estimate the value of $\hat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$ in matrix form as following:

$$\hat{\mathbf{C}} = \frac{1}{|\mathcal{V}_{\mathcal{S}}|}\mathbf{D}^{\mathcal{S}}\mathbf{L}^{\mathcal{S}}$$

$\mathbf{D}^{\mathcal{S}} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{V}_{\mathcal{S}}|}$, where each column represents the distribution of the class prediction of each node in the source graph. $[\mathbf{D}^{\mathcal{S}}]_{:,u} = g(h_u^{(L)}), \forall u \in \mathcal{V}_{\mathcal{S}}$. And each entry $[\mathbf{D}^{\mathcal{S}}]_{i,u} = [g(h_u^{(L)})]_i, \forall i \in \mathcal{Y}$. $\mathbf{L}^{\mathcal{S}} \in \mathbb{R}^{|\mathcal{V}_{\mathcal{S}}|\times|\mathcal{Y}|}$ that encodes the ground truth class of each node, as $[\mathbf{L}^{\mathcal{S}}]_{u,y_u} = 1$ for each node $u \in \mathcal{V}_{\mathcal{S}}$.

$$\hat{\boldsymbol{\mu}} = \frac{1}{|\mathcal{V}_{\mathcal{T}}|}\mathbf{D}^{\mathcal{T}}\mathbf{1}$$

Similarly, $\mathbf{D}^{\mathcal{T}} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{V}_{\mathcal{T}}|}$, where each column represents the distribution of the class prediction of each node in the target graph. $[\mathbf{D}^{\mathcal{T}}]_{:,u} = g(h_u^{(L)}), \forall u \in \mathcal{V}_{\mathcal{T}}$. And each entry $[\mathbf{D}^{\mathcal{T}}]_{i,u} = [g(h_u^{(L)})]_i, \forall i \in \mathcal{Y}$. $\mathbf{1} \in \mathbb{R}^{|\mathcal{V}_{\mathcal{T}}|\times 1}$ is the all one vector.

D. More Related Works

Other node-level DA works Other domain invariant learning-based methods, like Shen et al. (2020b) proposed to align the class-conditioned representations with conditional MMD distance by using pseudo-label predictions for the target domain, Zhang et al. (2021) aimed to use separate networks to capture the domain-specific features in addition to a shared encoder for adversarial training and further Pang et al. (2023) transformed the node features into spectral domain through Fourier transform for alignment. Other approaches like Cai et al. (2021) disentangled semantic, domain, and noise variables and used semantic variables that are better aligned with target graphs for prediction. Liu et al. (2024) explored the role of GNN propagation layers and linear transformation layers, thus proposing to use a shared transformation layer with more propagation layers on the target graph instead of a shared encoder.

Node-level OOD works In addition to GDA, many works target the out-of-distribution (OOD) generalization without access to unlabeled target data. For the node classification task, EERM (Wu et al., 2022) and LoRe-CIA (Wang et al., 2023) both extended the idea of invariant learning to node-level tasks, where EERM minimized the variance over representations across different environments and LoRe-CIA enforced the cross-environment Intra-class Alignment of node representations to remove their reliance on spurious features. Wang et al. (2021) extended mixup to the node representation under node and graph classification tasks.

Graph-level DA and OOD works The shifts and methods in graph-level problems are significantly different from those for node-level tasks. The shifts in graph-level tasks can be modeled as IID by considering individual graphs and often satisfy the covariate shift assumption, which makes some previous IID works applicable. Under the availability of target graphs, there are several graph-level GDA works like (Yin et al., 2023; 2022), where the former utilized contrastive learning to align the graph representations with similar semantics and the latter employed graph augmentation to match the target graphs under adversarial training. Regarding the scenarios in which we do not have access to the target graphs, it becomes the graph OOD problem. A dominant line of work in graph-level OOD is based on invariant learning originating from causality to identify a subgraph that remains invariant across graphs under distribution shifts. Among these works, Wu et al. (2021); Chen et al. (2022); Li et al. (2022a); Yang et al. (2022); Chen et al. (2023); Gui et al. (2023); Fan et al. (2022; 2023) aimed to find the invariant subgraph, and Miao et al. (2022); Yu et al. (2020) used graph information bottleneck. Furthermore, another line of

works adopted graph augmentation strategies, like (Sui et al., 2023; Jin et al., 2022) and some mixup-based methods (Han et al., 2022; Ling et al., 2023; Jia et al., 2023). Moreover, some works focused on handling the size shift (Yehudai et al., 2021; Bevilacqua et al., 2021; Chuang & Jegelka, 2022).

E. Experiments details

E.1. Dataset Details

Dataset Statistics Here we report the number of nodes, number of edges, feature dimension, and the number of labels for each dataset. The Arxiv-year means the graph with papers till that year. The edges are all undirected edges, which are counted twice in the edge list.

Table 5. real dataset statistics

	ACM	DBLP	ARXIV-2007	ARXIV-2009	ARXIV-2016	ARXIV-2018
#NODES	7410	5578	4980	9410	69499	120740
#EDGES	11135	7341	5849	13179	232419	615415
NODE FEATURE DIMENSION	7537	7537	128	128	128	128
#LABELS	6	6	40	40	40	40

Table 6. MAG dataset statistics

	US	CN	DE	JP	RU	FR
#NODES	132558	101952	43032	37498	32833	29262
#EDGES	697450	285561	126683	90944	67994	78222
NODE FEATURE DIMENSION	128	128	128	128	128	128
#LABELS	20	20	20	20	20	20

Table 7. Pileup dataset statistics

	GG-10	QQ-10	GG-30	QQ-30	GG-50	GG-140
#NODES	18611	17242	41390	38929	60054	154750
#EDGES	53725	42769	173392	150026	341930	2081229
NODE FEATURE DIMENSION	28	28	28	28	28	28
#LABELS	2	2	2	2	2	2

DBLP and ACM are two paper citation networks obtained from DBLP and ACM, originally from (Tang et al., 2008) and processed by (Wu et al., 2020). We use the processed version. Nodes are papers and undirected edges represent citations between papers. The goal is to predict the 6 research topics of a paper: "Database", "Data mining", "Artificial intelligent", "Computer vision", "Information Security" and "High Performance Computing".

Arxiv introduced in (Hu et al., 2020) is another citation network of Computer Science (CS) Arxiv papers to predict 40 classes on different subject areas. The feature vector is a 128-dimensional word2vec vector with the average embedding of the paper's title and abstract. Originally it is a directed graph with directed citations between papers, we convert it into an undirected graph.

E.1.1. MORE DETAILS MAG DATASETS

MAG is a subset of the Microsoft Academic Graph (MAG) as detailed in (Hu et al., 2020; Wang et al., 2020), originally containing entities as papers, authors, institutions, and fields of study. There are four types of directed relations in the original graph connecting two types of entities: an author "is affiliated with" an institution, an author "writes" a paper, a paper "cites" a paper, and a paper "has a topic of" a field of study. The node feature for a paper is the word2vec vector with 128 dimensions. The task is to predict the publication venue of papers, which in total has 349 classes. We curate the graph

to include only paper nodes and convert directed citation links to undirected edges. Papers are split into separate graphs based on the country of the institution the corresponding author is affiliated with. Then, we detail the process of generating a separate "paper-cites-paper" homogeneous graph for each country from the original ogbn-mag dataset.

Determine the country of origin for each paper. The rule of determining the country of the paper is based on the country of the institute the corresponding author is affiliated with. Since the original ogbn-mag dataset does not indicate the information of the corresponding author, we retrieve the metadata of the papers via OpenAlex,[2]. Specifically, there is a boolean variable on OpenAlex boolean indicating whether an author is the corresponding author for each paper. Then, we further locate the institution this corresponding author is affiliated with and retrieve that institution's country to use as the country code for the paper. All these operations can be done through OpenAlex. However, not all papers include this corresponding author information on OpenAlex. Regarding the papers that miss this information, we determine the country of this paper through a majority vote based on the institution country of all authors in this paper. Namely, we first identify all authors recorded in the original dataset via the "author—writes—paper" relation and acquire the institute information for these authors through the relation of "author—is_affiliated_with—institution". Then, with the country information retrieved from OpenAlex for these institutions, we do a majority vote to determine the final country code for the paper.

Generate country-specific graphs. Based on the country information obtained above, we generate a separate citation graph for a given country C. It will contain all papers that have a country code of C and the edges indicating the citation relationships within these papers. The edge_index set $\mathcal{E}$ is initialized as $\varnothing$. For each citation pair (v_i, v_j) in the original "paper-cites-paper" graph, it is added to $\mathcal{E}$ *iff.* both v_i and v_j have the same country affiliation C. We then obtain the node set $\mathcal{V}$ based on all unique nodes appearing in $\mathcal{E}$. In the scope of this work, we only focus on the top 19 publication venues with the most papers for classification and combine the rest of the classes into a single dummy class.

E.1.2. More details for HEP datasets

Initially, there are multiple graphs with each graph representing a collision event in the large hadron collider (LHC). Here, we collate the graphs together to form a single large graph. We use 100 graphs in each domain to create the single source and target graph respectively. In the source graph, the nodes in 60 graphs are used for training, 20 are used for validation and 20 are used for testing. In the target graph, the nodes in 20 graphs are used for validation and 80 are used for testing. The particles can be divided into charged and neutral particles, where the labels of the charged particles are known by the detector. Therefore, the classifications are only done on the neutral particles. The node features contain the particle's position in η axis, pt as energy, the pdgID one hot encoding to indicate the type of particle, and the label of the particle (label for changed, unknown for neutral) to help with classification as neighborhood information.

Pileup (PU) levels indicate the number of other collisions in the background event, it is closely related to the label distribution of LC and OC. For instance, a high PU graph will have mostly OC particles and few LC particles. Also, it will cause significant CSS as the distribution of particles easily influences the connections between them. The physical processes correspond to different types of signal decay of the particles, which mainly causes some slight feature shifts and nearly no LS or CSS under the same PU level.

E.2. Detailed experimental setting

Model architecture The backbone model is GraphSAGE with mean pooling having 3 GNN layers and 2 MLP layers for classification. The hidden dimension for GNN is 300 for Arxiv and MAG, 50 for Pileup, 128 for the DBLP/ACM dataset and 20 for synthetic datasets. The classifier dimension 300 for Arxiv and MAG, 50 for Pileup, 40 for DBLP/ACM dataset and 20 for synthetic datasets. If there is adversarial training with a domain classifier for some baselines, it has 3 layers and the hidden dimension is the same as the GNN dimension. All experiments are repeated three times.

Hardware All experiments are run on NVIDIA RTX A6000 with 48G memory and Quadro RTX 6000 with 24G memory. Specifically, for the UDAGCN baselines, we try with the 48G memory GPU but still out of memory.

Synthetic Datasets The synthetic dataset is generated under the contextual stochastic block model (CSBM), where there are in total of 6000 nodes and 3 classes. We vary the edge connection probability matrix and the node label distribution in different settings. The node features are generated from a Gaussian distribution where $\mathbb{P}_0 = \mathcal{N}([1,0,0], \sigma^2 I)$, $\mathbb{P}_1 = \mathcal{N}([0,1,0], \sigma^2 I)$ and $\mathbb{P}_2 = \mathcal{N}([0,0,1], \sigma^2 I), \sigma = 0.3$, and the distribution is the same for the source and target graph in all settings. We

[2]This is an alternative way considering the Microsoft Academic website and underlying APIs have been retired on Dec. 31, 2021.

denote the format of edge connection probability matrix as $\mathbf{B} = \begin{bmatrix} p & q & q \\ q & p & q \\ q & q & p \end{bmatrix}$, where p is the intra-class edge probability and q is the inter-class edge probability.

- The source graph has $\mathbb{P}_Y = [1/3, 1/3, 1/3]$ and $p = 0.02, q = 0.005$.
- For setting 1 and 2 with the shift in only class ratio, they have the same $\mathbb{P}_Y$, and setting 1 has $p = 0.015, q = 0.0075$ and setting 2 has $p = 0.01, q = 0.01$.
- For setting 3 and 4 with the shift in only cardinality, they have the same $\mathbb{P}_Y$, and setting 3 has $p = 0.02/2, q = 0.005/2$ and setting 4 has $p = 0.02/4, q = 0.005/4$.
- For setting 5 and 6 with the shift in both class ratio and cardinality, they have the same $\mathbb{P}_Y$, and setting 5 has $p = 0.015/2, q = 0.0075/2$ and setting 6 has $p = 0.01/2, q = 0.01/2$.
- For setting 7 and 8 with shifts in both CSS and label shift, they have the same edge connection probability as $p = 0.015/2, q = 0.0075/2$ but different label distributions. Setting 7 has $\mathbb{P}_Y = [0.5, 0.25, 0.25]$ and setting 8 has $\mathbb{P}_Y = [0.1, 0.3, 0.6]$.

Pileup Regarding the experiments studying the shift in pileup levels, the pair with PU10 and PU30 is from signal qq. The other two pairs with PU10 and PU50, PU30 and PU140 are from signal gg. The experiments that study the shift in physical processes are from the same PU level 10. Compared to the Pileup datasets used in the StruRW paper (Liu et al., 2023), we investigate the physical process shift with datasets from signal qq and signal gg instead of signal gg and signal $Z(\nu\nu)$. Also, we conduct more experiments to study the pileup shifts under the same physical process being signal qq (PU10 vs. PU30) or signal gg (PU10 vs. PU50 and PU30 vs. PU140). In addition, the StruRW paper treats each event as a single graph. They train the algorithm using multiple training graphs and adopt the edge weights as the average from each graph. In this paper, we collate the graphs for all events together for training and weight estimations.

Arxiv The graph is formed based on the ending year, meaning that the graph contains all nodes till the specified ending year. For instance, for the experiments where the source papers ended in 2007, the source graph contains all nodes and edges associated with papers that were published no later than 2007. Then, if the target years are from 2014 to 2016, then the entire target graph contains all papers published till 2016, but we only evaluate on the papers published from 2014 to 2016.

DBLP/ACM Since we observe that this dataset presents additional feature shift, so we additionally add adversarial layers to align the node representations. Basically, it is the combination of Pair-Align with label-weighted adversarial feature alignment, and the hyperparameters with additional adversarial layers are the same with DANN and will be detailed below. Also, note that to systematically control the label shift degree in this relatively small graph (< 10000 nodes), the split of nodes for training/validation/testing is done regarding each class of nodes. This is slightly different from the data in previous papers using this dataset, so the results may not be directly comparable.

E.3. Hyperparameter tuning

Hyperparameter tuning involves adjusting δ for edge probability regularization in $\boldsymbol{\gamma}$ calculation and λ for L2 regularization in the least square optimizations for $\mathbf{w}$ and $\boldsymbol{\beta}$. Selecting δ correlates to the degree of structure shift and λ is chosen based on the number of labels and classification performance. In datasets like Arxiv and MAG, where classification is challenging and labels are numerous, leading to ill-conditioned or rank-deficient confusion matrices, a larger λ is required. For simpler tasks with fewer classes, like synthetic and low PU datasets, a lower λ suffices. δ should be small for larger CSS (MAG and Pileup) and large with smaller CSS (Arxiv and physical process shift in Pileup) to counteract the spurious $\boldsymbol{\gamma}$ value that may caused by variance in edge formation. Below is the detailed range of hyperparameters.

The learning rate is 0.003 and the number of epochs is 400 for all experiments. The hyperparameters are tuned mainly for the robustness control, as the δ in regularizing edges and λ in L2 regularization for optimization of $\mathbf{w}$ and $\boldsymbol{\beta}$.

Here, for all datasets, $\lambda_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$ is chosen from $\{0.005, 0.01, 0.1, 1, 5\}$ to reweight the ERM loss to handle the LS. Additionally, we also consider reweighting the ERM loss by source label distribution together. Specifically, we found it useful in the case with imbalanced training label distribution, like both directions in DBLP/ACM datasets, transitioning from high PU to low PU, and the Arxiv training with papers pre-2007 and pre-2009. In other cases, we do not reweight the ERM loss by source label distribution.

- For the synthetic datasets, the δ is selected from $\{1\text{e}{-6}, 1\text{e}{-5}, 1\text{e}{-4}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{0.005, 0.01, 0.1\}$
- For the MAG dataset, the δ is selected from $\{1\text{e}{-5}, 1\text{e}{-4}, 1\text{e}{-3}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{0.1, 1, 5, 10\}$
- For the DBLP/ACM dataset, the δ is selected from $\{5\text{e}{-5}, 1\text{e}{-4}, 5\text{e}{-4}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{20, 25, 30\}$
- For the Pileup dataset, regarding the settings with pileup shift, δ is selected from $\{1\text{e}{-6}, 1\text{e}{-5}, 1\text{e}{-4}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{0.005, 0.01, 0.1, 1\}$. Regarding the settings with physical process shift, δ is selected from $\{1\text{e}{-5}, 1\text{e}{-4}, 5\text{e}{-4}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{1, 5, 10, 20\}$
- For the Arxiv dataset, regarding the settings with training data till 2007, the δ is selected from $\{5\text{e}{-3}, 1\text{e}{-2}, 3\text{e}{-2}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{1, 2, 5\}$. Regarding the settings with training data till 2009, the δ is selected from $\{3\text{e}{-2}, 5\text{e}{-2}, 8\text{e}{-2}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{15, 20, 25\}$. Regarding the settings with training data till 2011, the δ is selected from $\{3\text{e}{-4}, 5\text{e}{-4}, 8\text{e}{-4}\}$, $\lambda_{\mathbf{w}}$ is selected from $\{30, 50, 80\}$

E.4. Baseline Tuning

- For DANN, we tune two hyperparameters as the coefficient before the domain alignment loss and the max value of the rate added during the gradient reversal layer. The rate is calculated as $q = \min((\text{epoch} + 1)/\text{nepochs}), \text{max-rate})$. For all datasets, DA loss coefficient is selected from $\{0.2, 0.5, 1\}$ and max-rate is selected from $\{0.05, 0.2, 1\}$.
- For IWDAN, we tune three hyperparameters, the same two parameters as the coefficient before the domain alignment loss and the max value of the rate added during the gradient reversal layer. For all datasets, DA loss coefficient is selected from $\{0.5, 1\}$ and max-rate is selected from $\{0.05, 0.2, 1\}$. Also, we tune the coefficient to update the label weight calculated after each epoch as $(1 - \lambda) *$ new weight $+ \lambda *$ previous weight, where λ is selected from $\{0, 0.5\}$.
- For SpecReg, we totally tune for 5 hyperparameters and we follow the original hyperparameters for the dataset Arxiv and DBLP/ACM. For DBLP/ACM dataset, γ_{adv} is selected from $\{0.01, 0.2\}$, γ_{smooth} is selected from $\{0.01, 0.1\}$, threshold-smooth is selected from $\{0.01, -1\}$, γ_{mfr} is selected from $\{0.01, 0.1\}$, threshold-mfr is selected from $\{0.75, -1\}$. For Arxiv dataset, γ_{adv} is selected from $\{0.01\}$, γ_{smooth} is selected from $\{0, 0.1\}$, threshold-smooth is selected from $\{0, 1\}$, γ_{mfr} is selected from $\{0, 0.1\}$, threshold-mfr is selected from $\{0, 1\}$. For the other datasets, γ_{adv} is selected from $\{0.01\}$, γ_{smooth} is selected from $\{0.01, 0.1\}$, threshold-smooth is selected from $\{0.1, 1\}$, γ_{mfr} is selected from $\{0.01, 0.1\}$, threshold-mfr is selected from $\{0.1, 1\}$. Note that for the DBLP and ACM datasets, we implement their module (following their published code) on top of GNN instead of the UDAGCN model for fair comparison among baselines.
- For UDAGCN, we also tune the two hyperparameters from DANN as the coefficient before the domain alignment loss and the max value of the rate added during the gradient reversal layer. The rate is calculated as $q = \min((\text{epoch} + 1)/\text{nepochs}), \text{max-rate})$. For all datasets, DA loss coefficient is selected from $\{0.2, 0.5, 1\}$ and max-rate is selected from $\{0.05, 0.2, 1\}$.
- For StruRW, we use the StruRW-ERM baseline and we tune the λ that controls the edge weights in GNN as $(1 - \lambda) + \lambda *$ edge weight with range $\{0.1, 0.3, 0.7, 1\}$ and the epochs to start reweighting the edges from $\{100, 200, 300\}$.

E.5. Shift statistics of datasets

We design two metrics to measure the degree of structure shift in terms of CSS and LS.

The metric of CSS is based on the node label distribution in the neighborhood of each class of nodes as $\mathbb{P}_{\mathcal{U}}(Y_v|Y_u, v \in \mathcal{N}_u)$. Specifically, we calculate the total variation distance of this conditional neighborhood node label distribution of each class $\forall i \in \mathcal{Y}$ as:

$$
\begin{aligned}
&TV(\mathbb{P}_{\mathcal{S}}(Y_v|Y_u = i, v \in \mathcal{N}_u), \mathbb{P}_{\mathcal{T}}(Y_v|Y_u = i, v \in \mathcal{N}_u)) \\
&= \frac{1}{2}\|\mathbb{P}_{\mathcal{S}}(Y_v|Y_u = i, v \in \mathcal{N}_u) - \mathbb{P}_{\mathcal{T}}(Y_v|Y_u = i, v \in \mathcal{N}_u)\|_1 \\
&= \frac{1}{2}\sum_{j \in \mathcal{Y}} |\mathbb{P}_{\mathcal{S}}(Y_v = j|Y_u = i, v \in \mathcal{N}_u) - \mathbb{P}_{\mathcal{T}}(Y_v = j|Y_u = i, v \in \mathcal{N}_u)|
\end{aligned}
$$

Then, we take a weighted average of the TV distance for each class based on the label distribution of end nodes conditioned on an edge $\mathbb{P}_{\mathcal{U}}(Y_u|e_{uv} \in \mathcal{E}_{\mathcal{U}})$ since classes that appear more often as a center node in the neighborhood may affect more in the structure shift. The CSS-src in the table indicates the weighted average by $\mathbb{P}_{\mathcal{S}}(Y_u|e_{uv} \in \mathcal{E}_{\mathcal{S}})$ and CSS-tgt in the table indicates the weighted average by $\mathbb{P}_{\mathcal{T}}(Y_u|e_{uv} \in \mathcal{E}_{\mathcal{T}})$, and CSS-both is the average of CSS-src and CSS-tgt.

The metric of LS is calculated as the total variation distance between the source and target label distribution as:

$$TV(\mathbb{P}_{\mathcal{S}}(Y), \mathbb{P}_{\mathcal{T}}(Y)) = \frac{1}{2} \sum_{i \in \mathcal{Y}} |\mathbb{P}_{\mathcal{S}}(Y = i) - \mathbb{P}_{\mathcal{T}}(Y = i)|$$

The shift metrics for each dataset are shown in the following tables.

Table 8. MAG dataset shift metrics

	$US \to CN$	$US \to DE$	$US \to JP$	$US \to RU$	$US \to FR$	$CN \to US$	$CN \to DE$	$CN \to JP$	$CN \to RU$	$CN \to FR$
CSS-SRC	0.1639	0.2299	0.1322	0.3532	0.2530	0.2062	0.1775	0.1487	0.2120	0.1540
CSS-TGT	0.2062	0.2217	0.1438	0.2866	0.2854	0.1639	0.2311	0.1323	0.2027	0.2661
CSS-BOTH	0.1850	0.2258	0.1380	0.3199	0.2692	0.1850	0.2043	0.1405	0.2073	0.2100
LS	0.2734	0.1498	0.1699	0.3856	0.1706	0.2734	0.2691	0.1522	0.2453	0.2256

Table 9. HEP pileup dataset shift metrics

	PILEUP CONDITIONS						PHYSICAL PROCESSES	
DOMAINS	PU10 $\to$ 30	PU30 $\to$ 10	PU10 $\to$ 50	PU50 $\to$ 10	PU30 $\to$ 140	PU140 $\to$ 30	$gg \to qq$	$qq \to gg$
CSS-SRC	0.1941	0.1567	0.2910	0.2111	0.1871	0.1307	0.0232	0.0222
CSS-TGT	0.1567	0.1941	0.2111	0.2910	0.1307	0.1871	0.0222	0.0232
CSS-BOTH	0.1754	0.1754	0.2510	0.2510	0.1589	0.1589	0.0227	0.0227
LS	0.2258	0.2258	0.3175	0.3175	0.1590	0.1590	0.0348	0.0348

Table 10. Real dataset shift metrics

	1950-2007		1950-2009		1950-2011		DBLP AND ACM	
DOMAINS	$2014-2016$	$2016-2018$	$2014-2016$	$2016-2018$	$2014-2016$	$2016-2018$	$A \to D$	$D \to A$
CSS-SRC	0.2070	0.2651	0.1531	0.2010	0.1023	0.1443	0.1400	0.2241
CSS-TGT	0.2404	0.3060	0.2043	0.2737	0.1504	0.2301	0.2241	0.1400
CSS-BOTH	0.2237	0.2844	0.1787	0.2374	0.1263	0.1872	0.1820	0.1820
LS	0.2938	0.4396	0.2990	0.4552	0.2853	0.4438	0.3435	0.3435

Table 11. Synthetic CSBM dataset shift metrics

	CSS (ONLY CLASS RATIO SHIFT)		CSS (ONLY DEGREE SHIFT)		CSS (SHIFT IN BOTH)		CSS + LS	
CSS-SRC	0.1655	0.3322	0.0042	0.0053	0.1673	0.3308	0.1777	0.2939
CSS-TGT	0.1655	0.3322	0.0042	0.0053	0.1673	0.3308	0.1215	0.1840
CSS-BOTH	0.1655	0.3322	0.0042	0.0053	0.1673	0.3308	0.1496	0.2389
LS	0	0	0	0	0	0	0.1650	0.2667

E.6. More results analysis

In this section, we will discuss more regarding our experimental results and provide some explanations of our Pair-Align performance and comparison over the baselines.

Synthetic Data As discussed in the main text, our major conclusion is that our Pair-Align is practical for handling alignment by focusing only on the conditional neighborhood node label distribution to address class ratio shifts. Although Pair-Align's performance is not the best among the baselines when there is a shift in node degree, we argue that in practice, ERM training alone is adequate under node degree shifts, especially when the graph size is large. Here, the graph size is only 6000—a

small size in practical terms—and the ERM performance with a node degree shift ratio of 2 already achieved 99% accuracy. It should be perfect when the graph size is larger. Also, in the second setting with a degree shift, the degree ratio shift of 4 is relatively large, but the accuracy remains at 96%. We expect that the decay should be negligible when the graph size is larger, often at least 10 times larger than 6000.

Regarding performance gains in addressing structure shifts, we observe that PA-CSS demonstrates significant improvements, particularly in the second case of each scenario with larger degree shifts. Among the baselines, StruRW consistently outperforms others in different CSS scenarios, except in node degree shifts. This is expected since StruRW is specifically designed to handle CSS. Plus, in the synthetic CSBM data used here, the instability commonly associated with using hard pseudo-labels does not significantly affect performance due to easy classification task. However, compared to our Pair-Align methods, StruRW still shows limited performance even with only CSS shifts. When both CSS and LS shifts occur, IWDAN emerges as the best baseline, as its algorithm addresses both conditional shifts and LS in non-graph problems effectively. In synthetic datasets, shifts are less complex than in real-world graph-structured data, allowing IWDAN to lead to empirical improvements. Our PA-BOTH outperforms all in scenarios involving CSS and LS shifts. By comparing PA-CSS and PA-LS, we found that when both CSS and LS occur, the impact of CSS often dominates, making PA-CSS more effective than PA-LS. However, this observation is based on our source graph's balanced label distribution and does not hold in the HEP pileup dataset when moving from highly imbalanced data (high PU conditions) to more balanced data (low PU conditions), which we will discuss later in relation to the Pileup dataset.

Another advantage of using synthetic dataset results is that they help us understand the experimental results on real datasets better. For example, by combining the shift statistics from Table 11 with the experimental results, we see that a CSS metric value around 0.16 does not significantly impact the performance, thus not clearly demonstrating the effectiveness of Pair-Align. However, Pair-Align methods show substantial benefits under larger shifts, with metric values around 0.3.

MAG Overall, our Pair-Align methods demonstrated significant advantages over the majority of baseline approaches, including the top-performing baseline, StruRW. When considering the relative improvement to ERM performance (as well as the performance of other baselines, except StruRW), there is an average relative benefit of over 45% when training on the US graph and nearly 100% when training on the CN graph. This substantial improvement corroborates our discussion regarding the existing gap, where current methods fall short in effectively addressing structure shifts. As detailed in the main text, our PA-CSS methods not only surpass StruRW in performance but also yield additional benefits from handling LS, as the LS degree indicated in Table 8. We believe the primary advantages stem from our principled approach to addressing CSS with γ, which remains unbiased by LS, and the enhanced robustness afforded by using soft label predictions and regularized least square estimations. This also elucidates the shortcomings of IWDAN, a non-graph method for addressing conditional shift and LS, which underperforms under the MAG dataset conditions as discussed in the main text.

We next explore the relationship between performance improvements and the degree of structure shift. The experimental results align closely with the CSS measurements shown in Table 8. For example, the transitions from US to JP and CN to JP involve a smaller degree of CSS compared to other scenarios, resulting in relatively modest improvements. Similarly, generalizations between the US and CN also show fewer benefits. Conversely, the impact of LS is less evident in the outcomes associated with PA-LS, as this approach alone yields only marginal improvements. However, when we evaluate the additional gains from LS mitigation provided by PA-BOTH in comparison to PA-CSS, scenarios with larger LS (such as $US \to CN$, $CN \to US$, $US \to RU$, and $CN \to DE$) demonstrate more substantial benefits.

Pileup Mitigation The most crucial discussions concerning HEP pileup datasets are detailed in the main text, particularly focusing on the distinct impacts of CSS and LS in transitions from high PU conditions to low PU conditions, and vice versa. This underscores that while the two directions have identical measures of LS, the direction of generalization is crucial. From a training perspective, it is clear that a model trained on a highly imbalanced dataset may neglect nodes in minor classes, leading to worse performance on more balanced datasets. To improve generalization, it is essential to adjust the classification loss to increase the model's focus on these minor nodes during training. This explains why PA-CSS alone does not yield benefits in scenarios transitioning from high to low PU, and why PA-LS becomes necessary. Conversely, when transitioning from low to high PU, PA-CSS suffices to address CSS, as LS has a minimal effect on performance in this direction.

We then review baseline performance under the shift in pileup (PU) conditions. As noted in the main text, methods primarily addressing feature shifts, such as DANN, UDAGCN, and IWDAN, underperform, underscoring that PU conditions predominantly affect graph structure rather than node features. This observation aligns with the physical interpretation of PU shifts described in the dataset details in E.1.2. PU shift correlates with changes in the number of other collisions (OC) during collision events, directly influencing the OC ratio and the pattern of node connections, as illustrated in Fig 1. Given

that node features are derived from particle labels (either OC or LC), the feature distribution remains largely unchanged despite variations in the OC to LC ratio. Consequently, feature shifts are minimal under PU conditions.

Consequently, the baselines like StruRW and SpecReg show some benefits over others in regularizing and adjusting graph structure to handle structure shift. Specifically, SpecReg shows enhanced benefits during the transition from low PU to high PU, possibly due to its regularization of spectral smoothness, which mitigates edge perturbations beneficially under CSS conditions. Despite these improvements in the pileup dataset, SpecReg does not perform as well in other datasets characterized by CSS, such as MAG. This may be attributed to the fact that spectral regularization is more effective in scenarios with a limited variety of node connections, akin to the binary cases in the pileup dataset. However, it appears less capable of managing more complex shifts in neighborhood distribution involving multiple classes, as seen in datasets like MAG or Arxiv.

Conversely, StruRW achieves comparable performances to PA-BOTH in scenarios transitioning from high PU to low PU, predominantly influenced by LS. This effectiveness is likely due to the fact that their edge weights incorporate $\mathbf{w}$, which includes $\boldsymbol{\alpha}$ that implicitly contains the node label ratio. While our analysis suggests that using $\mathbf{w}$ directly is not a principled approach for addressing CSS and LS, it proves beneficial in scenarios where LS significantly affects outcomes, providing a better calibration compared to approaches that do not address LS, like PA-CSS. However, while StruRW holds an advantage over PA-CSS, its performance still lags behind PA-BOTH, which offers a more systematic solution for both CSS and LS.

Arxiv Results from the Arxiv datasets align well with expectations and the shift measures detailed in Table 10. Notably, CSS is most pronounced when the source graph includes papers published before 2007, with experimental results showing the most substantial improvements under these conditions. In the scenario where papers from 2016-2018 are used for testing, both PA-CSS and PA-BOTH outperform the baselines significantly, yet PA-LS emerges as the superior variant. This aligns with the LS metrics reported, which indicate a significant LS in this context. A similar pattern is observed when training on papers from before 2011 and testing on those from 2016-2018, with PA-LS achieving the best results.

For the target comprising papers from 2014-2016, our model continues to outperform baselines, albeit with a narrower margin compared to other datasets. In this case, not only does our method perform comparably, but all baselines also show similar performance levels, suggesting limited potential for improvements in this dataset. Furthermore, insights from synthetic experiments reveal that a CSS metric value around 0.16 does not lead to substantial performance degradation, which accounts for the moderate improvements over baselines in scenarios other than those using the source graph with pre-2007 papers.

In our evaluation of baseline performances, we note that StruRW emerges as the superior baseline, effectively handling CSS. In contrast, IWDAN tends to underperform relative to other baselines, which we attribute primarily to inaccuracies and instability in its label weight estimation. Designed for computer vision tasks where accuracy is typically high, IWDAN lacks mechanisms for regularization and robustness in its estimation processes, leading to its underperformance in our experiments involving tasks with a total of 40 classes. Meanwhile, the performance of other baselines is comparable to the ERM training.

DBLP/ACM The generalization results between the DBLP and ACM datasets offer insights into the comparative effects of feature shift versus structure shift. As discussed in the main text, baselines focused on feature alignment tend to perform well in this dataset, suggesting that this dataset is predominantly influenced by feature shifts rather than structural shifts and that feature alignment can address the shift effectively. This trend also leads to non-graph methods performing comparably to, or even better than, graph-based methods due to the dominance of feature shifts.

In response to these observations, we integrated adversarial training into our method to align feature shifts and investigated whether additional benefits could be derived from mitigating structure shifts. Our analysis of the experimental results, in conjunction with the shift measures detailed in Table 10, reveals a significant LS between these two datasets. Specifically, we note that the ACM graph exhibits a more imbalanced label distribution compared to the DBLP graph. This finding aligns with the experimental outcomes, where PA-LS emerges as the most effective model and IWDAN as the best baseline when training on ACM and testing on DBLP. Both methods are adept at handling LS, supporting our earlier assertion that LS plays a crucial role when transitioning from an imbalanced dataset to a more balanced one. Conversely, in the transition from DBLP to ACM, where LS has a lesser impact, PA-BOTH proves to be the most effective.