Improving Graph Domain Adaptation with Network Hierarchy

Boshen Shi
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
shiboshen19s@ict.ac.cn

Yongqing Wang*
Data Intelligence System Research Center, Institute of Computing Technology, CAS, Beijing, China
wangyongqing@ict.ac.cn

Fangda Guo*
Data Intelligence System Research Center, Institute of Computing Technology, CAS, Beijing, China
guofangda@ict.ac.cn

Jiangli Shao
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
shaojiangli19z@ict.ac.cn

Huawei Shen
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
shenhuawei@ict.ac.cn

Xueqi Cheng
Data Intelligence System Research Center, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, Beijing, China
cxq@ict.ac.cn

ABSTRACT

Graph domain adaptation models have become instrumental in addressing cross-network learning problems due to their ability to transfer abundant label and structural knowledge from source graphs to target graphs. A crucial step in transfer involves measuring domain discrepancy, which refers to distribution shifts between graphs from source and target domains. While conventional models simply provide a node-level measurement, exploiting information from different levels of network hierarchy is intuitive. As each hierarchical level characterizes distinct and meaningful properties or functionalities of the original graph, integrating domain discrepancy based on such hierarchies should contribute to a more precise domain discrepancy measurement. Moreover, class conditional distribution shift is often overlooked in node classification tasks, which could potentially lead to sub-optimal performance. To address the above limitations, we propose a new graph domain adaptation model and apply it to cross-network node classification tasks. Specifically, a hierarchical pooling model to extract meaningful and adaptive hierarchical structures is designed, where both marginal and class conditional distribution shifts on each hierarchical level are jointly minimized. The effectiveness is demonstrated through theoretical analysis and experimental studies across various datasets.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

Graph domain adaptation, Cross-network learning, Graph neural network, Transfer learning

ACM Reference Format:
Boshen Shi, Yongqing Wang, Fangda Guo, Jiangli Shao, Huawei Shen, and Xueqi Cheng. 2023. Improving Graph Domain Adaptation with Network Hierarchy. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614928

*Corresponding authors.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614928

1 INTRODUCTION

Inspired by domain adaptation (DA) technique, which transfers knowledge from source domain to target domain in the presence of distribution shifts, graph domain adaptation models (GDA) adapt DA technique to graphs and aim to tackle practical cross-network learning problems, such as cross-network node classification. In particular, they leverage abundant label information from source graphs (source domains) to classify nodes in unlabeled target graphs (target domains), thus transferring valuable knowledge to the target network and reducing labeling costs associated with complex graphs. Such a task has wide-ranging real-world applications, including predicting user interests across online social networks [33, 42], inferring nodal properties in urban systems across different regions [17, 25], and annotating protein functions in various protein-protein interaction (PPI) networks [26, 43].

In the realm of cross-network learning problems, GDA models face significant shifts in both node attribute distribution and graph structures across source and target graphs. Following general DA framework, the majority of GDA models adopt Graph Neural Networks (GNNs) to integrate these domain shifts holistically in the embedding space. Following this, they interpret node embedding shift as domain discrepancy and apply common discrepancy measurements, such as Jensen–Shannon divergence [11, 34, 40, 47] or Wasserstein distance [6, 46]. Consequently, these models provide a node-level computation of domain discrepancy. However, unlike

non-structural data, such as images, graphs typically exhibit intrinsic hierarchical structures [5, 41], each level of which embodies meaningful and distinct properties of the original graphs. Moreover, the differences in functionality across graphs tend to become more evident when examining their high-level hierarchies. Therefore, it is intuitive to exploit information from hierarchical levels for measuring domain discrepancy more accurately. Figure 1 presents an example from online social networks. Users are naturally grouped by similar interests, and such user groups could be further assembled into larger communities. Beyond the individual user-level, distribution shifts at both group-level and community-level may highlight the subtle differences that are not immediately apparent at the user-level, thereby leading to a more precise discrepancy computation.

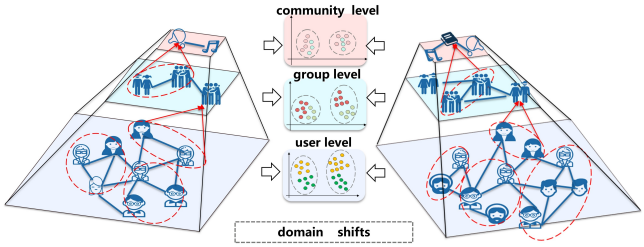

Figure 1: Domain shifts at different hierarchical levels carry meaningful and distinct information for computing domain discrepancy.

In addition to employing hierarchical discrepancy computation, GDA models, particularly when applied to cross-network node classification tasks, could benefit from considering the shifts in class-conditional distribution between node embeddings. Given source domain s, target domain t, node embedding z, and node label y, GDA models typically measure the marginal distribution shift between $P(z^s)$ and $P(z^t)$. However, the difference between $P(z^s|y^s)$ and $P(z^t|y^t)$ also carries critical information for accurately measuring domain discrepancy in graphs. The absence of y^t further underscores the necessity for an effective pseudo-labeling strategy.

To mitigate the limitations of existing GDA models, we propose a novel model *JHGDA*, which jointly minimizes both marginal and class conditional distribution shifts based on hierarchical structures on graphs. We mainly apply *JHGDA* to solve a significant task in cross-network learning: cross-network node classification. To this aim, we employ a differentiable hierarchical pooling model to extract network hierarchies for both source and target graphs. These hierarchical structures carry meaningful semantic information, such as user communities or functional molecule units, while remaining adaptive to downstream tasks. At each hierarchical level, we jointly compute both marginal and class conditional distribution shifts and aggregate the results across all levels to obtain a comprehensive domain discrepancy. Specifically, we adopt Maximum Mean Discrepancy (MMD) as a basic discrepancy measurement and apply it to source and target node embeddings. We also propose a corresponding pseudo-label strategy for obtaining class conditional information on unlabeled target graphs, which is formulated as a linear sum assignment problem. To enhance the understanding of *JHGDA*, we provide a theoretical proof demonstrating that such hierarchical domain discrepancy equals the combination of graph kernels. As graph kernels are widely adopted to assess graph similarity [16, 28], such proof strengthens the interpretability of *JHGDA*. We also establish a generalization bound for *JHGDA* to guarantee its efficacy. Finally, we evaluate the superiority of *JHGDA* through various real-world tasks, and we propose solutions to optimize both time and space complexity accordingly. To sum up, the major contributions of this paper are summarized as follows.

- We investigate GDA models which are applicable in a wide range of cross-network learning tasks, and we discover that their computation of domain discrepancy at the node-level may lead to sub-optimal performance on graphs.
- We propose a model, *JHGDA*, to improve the performance of GDA models in cross-network node classification task. It jointly minimizes both marginal and class conditional distribution shifts based on hierarchical graph structures. Through theoretical proofs, we demonstrate its interpretability and effectiveness.
- We conduct extensive experiments on various real-world datasets to prove the superiority of *JHGDA*.

2 RELATED WORK

The scarcity of high-quality labels and the sparsity of graph structure in many real-world applications pose challenges to the effectiveness of GNNs. To mitigate these issues and enhance performance, researchers have proposed cross-network learning tasks to transfer abundant label knowledge and structural knowledge from relevant source graphs to target graphs [34, 35]. They cover various types including node classification [6, 17, 27, 42], link prediction [19, 39, 44] and graph classification [2, 4, 37]. Graph domain adaptation is among the important approaches to solving cross-network learning tasks. Most GDA models apply discrepancy measurement, including MMD, Jensen–Shannon divergence and Wasserstein distance to node embedding distributions for measuring domain discrepancy [6, 11, 34, 35, 39, 40, 46, 47]. Such discrepancy is minimized together with source task errors during training. Among them, UDAGCN [40] develops a dual graph convolutional network which jointly exploits local and global consistency for better adaptation. GRADE-N [39] characterizes graph domain discrepancy from the perspective of Weisfeiler-Lehman graph isomorphism test. Given Wasserstein distance as discrepancy measurement, SpecReg [46] proposes a theory-grounded spectral regularization method for a tightened generalization bound. ASN [47] and DGDA [4] are disentanglement-based models which only measure the discrepancy between domain-related components. These models neither consider hierarchical graph structures nor class conditional distribution shift, leading to a sub-optimal performance. In this work, we focus on cross-network node classification tasks and jointly minimize both marginal and class conditional distribution shifts on each level of network hierarchy.

Graph pooling is an essential tool for obtaining a holistic graph-level representation of the entire graph, and it could be roughly divided into flat pooling and hierarchical pooling [20]. Flat pooling models iteratively delete nodes with comparatively lower importance according to scoring functions [9, 13]. In this work, we utilize

hierarchical pooling models to extract meaningful and adaptive network hierarchy. It preserves the hierarchical structural information by iteratively coarsening the graph into a new graph of smaller size [7, 21, 45].

3 PROBLEM STATEMENT

A graph G is composed of node set V and edge set E, which can be represented by $G = (A, X, Y)$. $A \in \mathbb{R}^{N\times N}$, $X \in \mathbb{R}^{N\times F}$ and $Y \in \mathbb{R}^{N\times C}$ denote adjacency matrix, node feature matrix and label matrix, respectively. $N = |V|$ is the size of node set, F is the dimension of node feature space and C is the number of node classes. $Y_{i,c} = 1$ indicates node i is associated with class c.

In the context of cross-network node classification, we assume one source graph G^s and one target graph G^t could be accessed during training. The relevance between G^s and G^t is defined from two aspects: 1) X^s and X^t are sampled from the same feature space but have different distributions, and 2) the classification task keeps unchanged across domains [29, 49]. G^s is associated with the label matrix Y^s, while G^t is unlabeled. Given $G^s = (A^s, X^s, Y^s)$ and $G^t = (A^t, X^t)$, GDA model h aims to predict Y^t by jointly learning on source and target domains: $Y^t = h(G^s, G^t)$.

4 METHODOLOGY

4.1 Model Overview

As previously stated, it is necessary to extract hierarchical structures for graphs in each domain for better characterizing domain discrepancy. Furthermore, such hierarchy should adapt to downstream tasks and can be optimized through training. Inspired by DiffPool [45], we adaptively coarsen the input graph and learn the optimal hierarchical representations by modeling both structural and label information on graphs. Subsequently, we aggregate the domain discrepancy calculated between each hierarchical level of source and target graphs to obtain a comprehensive domain discrepancy. In addition, a pseudo labeling module is applied to the original target graph for generating pseudo labels. The overall framework is illustrated by Figure 2.

4.2 Hierarchical Pooling

By alternatively stacking graph convolution blocks (*gconv*) and graph pooling blocks (*gpool*) in each domain, we obtain a sequence of graphs as hierarchical structures: $(G^{(0)}, G^{(1)}, ..., G^{(L)})$. Each block comprises several graph convolution layers [15], and $G^{(0)}$ indicates the original graph. First of all, we update node embeddings on $G^{(l)}$ with *gconv*. Given node feature matrix $X^{(l)}$ and adjacency matrix $A^{(l)}$, *gconv* learns a function $GCN_{emb,l}$ to update node embeddings:

$$Z^{(l)} = GCN_{emb,l}(X^{(l)}, A^{(l)}) \tag{1}$$

Meanwhile, the pooling is achieved by assigning a group of nodes on $G^{(l)}$ into a cluster, which appears to be a node at $G^{(l+1)}$. Thus, we apply *gpool* to learn a function $GCN_{pool,l}$ which computes the cluster assignment matrix $S^{(l)} \in \mathbb{R}^{|V^{(l)}|\times|V^{(l+1)}|}$:

$$S^{(l)} = softmax(GCN_{pool,l}(X^{(l)}, A^{(l)})) \tag{2}$$

where row-wise $softmax$ is applied. By setting pooling rate $r^{(l)} \in (0, 1)$ in advance, $|V^{(l+1)}| = r^{(l)}|V^{(l)}|$. Generally, $S^{(l)}_{i,j}$ indicates the probability of assigning node i to cluster j.

Given updated node embedding $Z^{(l)}$ and pooling mapping $S^{(l)}$, the node feature $X^{(l+1)}$ and adjacency matrix $A^{(l+1)}$ on $G^{(l+1)}$ is computed as:

$$X^{(l+1)} = S^{(l)^T} Z^{(l)} \tag{3}$$

$$A^{(l+1)} = S^{(l)^T} A^{(l)} S^{(l)} \tag{4}$$

To further compute the class conditional distribution shift, label information on $G^{(l)}$ is needed. Thus, given soft label matrix $\widetilde{Y}^{(l)}$, where $\widetilde{Y}^{(l)}_{i,c}$ indicates $P(y_i = c|v_i)$ and each row sums up to 1 , $\widetilde{Y}^{(l+1)}$ is computed as:

$$\widetilde{Y}^{(l+1)} = S^{(l)^T} \widetilde{Y}^{(l)} \tag{5}$$

At $l = 0$, $\widetilde{Y}^{s,(0)}$ equals ground-truth labels Y^s, and $\widetilde{Y}^{t,(0)}$ is given by pseudo label strategy proposed in section 4.4.

To sum up, the hierarchical pooling model extracts network hierarchy as a series of graphs for each domain and attaches each hierarchy with node feature matrix, adjacency matrix and soft label matrix.

4.3 Hierarchical Domain Discrepancy

As our objective is to measure domain discrepancy from different hierarchical levels, we calculate discrepancy $d_b(G^{s,(l)}, G^{t,(l)})$ at each level and aggregate the results to obtain a comprehensive domain discrepancy. Consequently, the hierarchical domain discrepancy is formulated as follows:

$$d(G^s, G^t) = \sum_{l=0}^{L} d_b(G^{s,(l)}, G^{t,(l)}) \tag{6}$$

where d_b represents base discrepancy measurement, such as MMD. In this work, we initially adopt the exponential form of MMD to measure marginal distribution shift:

$$d_b(G^{s,(l)}, G^{t,(l)}) = e^{-\alpha MMD(Z^{s,(l)}, Z^{t,(l)})} \tag{7}$$

Empirically, the exponential form has additional advantages, including smoother loss curves which are easier to optimize. The basic idea behind MMD is that if the generating distributions are identical, all the statistics are the same as well [10]. Formally, MMD defines the difference between two distributions with their mean embeddings in the reproducing kernel Hilbert space (RKHS). Given kernel mapping $\phi(\cdot)$ and kernel k in the corresponding RKHS, MMD is empirically estimated by comparing the square distance between the empirical kernel mean embeddings:

$$\begin{aligned} &MMD(Z^{s,(l)}, Z^{s,(l)}) \\ &= \| \frac{1}{n^{s,(l)}} \sum_i \phi(Z_i^{s,(l)}) - \frac{1}{n^{t,(l)}} \sum_i \phi(Z_i^{t,(l)}) \|_2^2 \\ &= \sum_i^{n^{s,(l)}} \sum_j^{n^{s,(l)}} ak_{i,j} + \sum_i^{n^{t,(l)}} \sum_j^{n^{t,(l)}} bk_{i,j} - \sum_i^{n^{s,(l)}} \sum_j^{n^{t,(l)}} ck_{i,j} \end{aligned} \tag{8}$$

where $k_{i,j} = k(Z_i^{s,(l)}, Z_j^{s,(l)})$, $n^{(l)}$ indicates the number of nodes, a,b and c represent $1/{n^{s,(l)}}^2$,$1/{n^{t,(l)}}^2$ and $2/n^{s,(l)}n^{t,(l)}$, respectively.

For better characterizing domain discrepancy, we further integrate class conditional MMD, named CMMD, to measure class conditional distribution shift [23, 24]. CMMD is formulated as the summation of MMD calculated for each class:

$$\begin{aligned} &CMMD(Z^{s,(l)}, Z^{s,(l)}) \\ &= \sum_{c}^{C} \| \frac{1}{n^{s,c,(l)}} \sum_{i} \phi(Z_i^{s,(l)}) - \frac{1}{n^{t,c,(l)}} \sum_{i} \phi(Z_i^{t,(l)}) \|_2^2 \\ &= \sum_{c}^{C} (\sum_{i=1}^{n^{s,c,(l)}} \sum_{j=1}^{n^{s,c,(l)}} a' k_{i,j} + \sum_{i=1}^{n^{t,c,(l)}} \sum_{j=1}^{n^{t,c,(l)}} b' k_{i,j} - \sum_{i=1}^{n^{s,c,(l)}} \sum_{j=1}^{n^{t,c,(l)}} c' k_{i,j}) \end{aligned} \tag{9}$$

where $n^{c,(l)}$ indicates number of nodes belonging to class c on $G^{(l)}$. It is counted after assigning the most probable label to the node, and such probability is given by $\widetilde{Y}^{(l)}$. a', b' and c' are similarly defined as they are in Equation 8.

By integrating MMD and CMMD, we could jointly minimize the domain shifts of both marginal and class conditional distribution:

$$d_b(G^{s,(l)}, G^{t,(l)}) = pe^{-\alpha MMD^{(l)}} + (1-p)e^{-\alpha CMMD^{(l)}} \tag{10}$$

where p indicates balancing coefficient, and $MMD^{(l)}$, $CMMD^{(l)}$ are short for MMD or CMMD computed at hierarchical level l.

4.4 Pseudo Labeling

Pseudo labeling strategy is proposed for unlabeled target graph to obtain class conditional information. We generate pseudo labels on $G^{t,(0)}$ and propagate them to the successive hierarchical levels through Equation 5. Finally, nodes at all hierarchical levels of the target graph are labeled.

Motivated by the principle that target nodes should align closely with source nodes of the same class in the embedding space, we propose a non-parametric strategy that transfers abundant label information from source graph to target graph for generating pseudo labels. Specifically, we initially cluster target nodes into C groups by KMeans algorithm, and $O^{t,k}$ denotes the center of the k^{th} cluster, which is computed by averaging node embeddings inside. Subsequently, We compute C class centers $O^{s,k}$ on source graph. As there are C target centers and C source centers, we formulate pseudo labeling as linear sum assignment problem:

$$\begin{aligned} \min_{M} \quad & \sum_{i}^{C} \sum_{j}^{C} D_{i,j} M_{i,j} \\ s.t. \quad & \sum_{j=1}^{C} M_{i,j} = 1, \quad i = 1, \ldots, C \\ & \sum_{i=1}^{C} M_{i,j} = 1, \quad j = 1, \ldots, C \\ & M_{i,j} \in \{0, 1\}, \quad i, j = 1, \ldots, C \end{aligned}$$

where D is the cost matrix and cosine dissimilarity is adopted for D: $D_{i,j} = \frac{1}{2}(1 - cos(O^{t,i}, O^{s,j}))$. M is the assignment matrix to be solved, and $M_{i,j} = 1$ indicates pseudo label j is assigned to all target nodes in the i^{th} cluster. The generated pseudo labels are propagated to the successive hierarchies through pooling (i.e., Equation 5).

4.5 Proposed Model

In this section, we propose *JHGDA* by putting all the ingredients introduced above together, and we analyze algorithm complexity subsequently.

Following the settings in DiffPool, we add two side loss terms to improve the performance of pooling. Initially, we encourage the model to learn clear cluster assignment between the successive hierarchical levels by adding regularization $L_e^{(l)} = \frac{1}{n^{(l)}} \sum_i^{n^{(l)}} H(S_i)$, where $H(\cdot)$ denotes entropy function and S_i is the i-th row of S. In addition, the model should preserve local structure during pooling. As a result, we encourage the model to assign neighboring nodes to the same cluster through regularizing $L_{sp}^{(l)} = \|A^{(l)} - S^{(l)} {S^{(l)}}^T\|_F$, where $\|\cdot\|_F$ denotes Frobenius norm. $L_s^{(l)} = L_e^{(l)} + L_{sp}^{(l)}$ indicates the sum of side loss terms at hierarchical level l.

However, $L_s^{(l)}$ is not enough to guarantee a good computation of class conditional distribution shift, for the label distribution $\widetilde{Y}_i^{(l)}$ of node i tends to be uniform, and such class information is ambiguous. In the contrary, we expect $\widetilde{Y}_i^{(l)}$ to be deterministic (i.e., $\widetilde{Y}_i^{(l)}$ should be similar to one-hot vector). Therefore, we regularize the entropy of label distribution for each node by minimizing label entropy loss $L_{le}^{(l)}$:

$$L_{le}^{(l)} = \frac{1}{n^{(l)}} \sum_{i}^{n^{(l)}} H(\widetilde{Y}_i^{(l)}) \tag{11}$$

In addition, the model should preserve local patterns related to class distribution. In practice, we encourage nodes of the same class to be pooled to the same cluster by minimizing class pattern preserving loss $L_{cpp}^{(l)}$:

$$L_{cpp}^{(l)} = \|S^{(l)} - \widetilde{Y}^{(l)} \widetilde{Y}^{(l+1)^T}\|_F \tag{12}$$

Consequently, we denote class-relevant side loss term at level l as $L_{cs}^{(l)} = L_{le}^{(l)} + L_{cpp}^{(l)}$. All sides loss terms are computed on both domains, and they are aggregated from all hierarchical levels: $L_s = \sum_l L_s^{(l)}, L_{cs} = \sum_l L_{cs}^{(l)}$.

Finally, *JHGDA* is jointly trained on source and target graphs, and it is optimized with the following objective function consisting of four integral parts:

$$L = L_{CE}(\hat{Y}^s, Y^s) + \beta d(G^s, G^t) + \gamma_1 L_s + \gamma_2 L_{cs} \tag{13}$$

where L_{CE} represents cross entropy loss computed between ground-truth source labels Y^s and predicted source labels $\hat{Y}^s$, which are generated by applying classifier in model h to $Z^{s,(0)}$. $d(G^s, G^t)$ is defined from Equation 6 and 10. β, γ_1 and γ_2 are coefficients. After training, we could apply the trained model to $G^{t,(0)}$ for obtaining target node embeddings and predicting node labels.

5 OPTIMIZATION

Given two graphs with approximately n nodes and m edges each, the time complexity of *JHGDA* with one pooling block and two graph convolution blocks is $O(m + r^2 n^2)$. This complexity is comparable

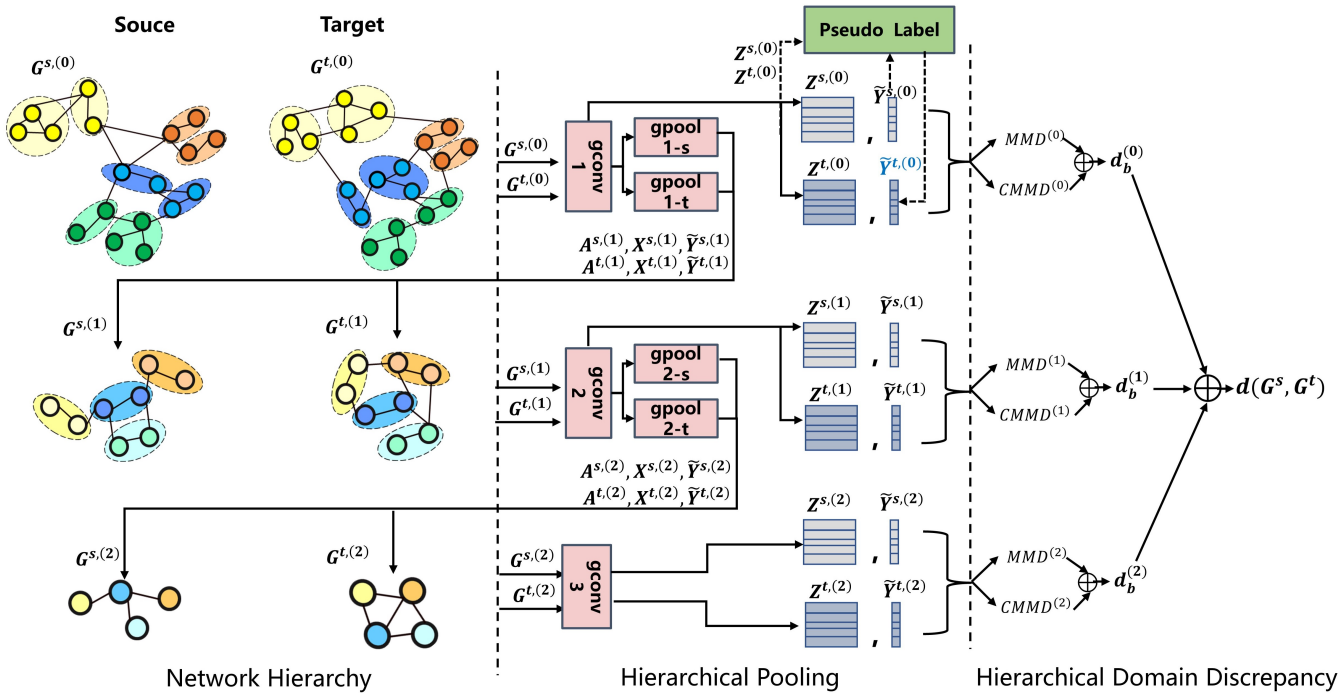

Figure 2: Overview of model framework to compute hierarchical domain discrepancy, where an alternative stacking of *gconv* and *gpool* blocks is adopted to extract hierarchical structures, and the domain discrepancy computed at each level is aggregated. *gpool-s* and *gpool-t* indicates separate *gpools* for source and target graphs. $d_b^{(l)}$ is short for $d_b(G^{s,(l)}, G^{t,(l)})$.

Algorithm 1 Training *JHGDA* for one epoch

Input: $G^s = (A^s, X^s, Y^s)$, $G^t = (A^t, X^t)$, model h with initialized parameters θ_h, number of *gpool* L and other related parameters
Output: Optimized h

Initialize L_s, L_{cs}, $d(G^s, G^t)$ with 0
Initialize $A^{s,(0)}, A^{t,(0)}, X^{s,(0)}, X^{t,(0)}$ and $\widetilde{Y}^{s,(0)}$
for $l < L + 1$ **do**
  Update node embeddings $Z^{s,(l)}$ and $Z^{t,(l)}$ //Eq. 1
  if $l == 0$ **then**
    Update target pseudo labels $\widetilde{Y}^{t,(0)}$ // Sect. 4.4
    Compute source cross-entropy loss L_{CE}
  end if
  $d(G^s, G^t) += d_b^{(l)}$ // domain discrepancy at level l
  if $l < L$ **then**
    for $k \in \{s, t\}$ **do**
      Compute $S^{k,(l)}, A^{k,(l+1)}, X^{k,(l+1)}, \widetilde{Y}^{k,(l+1)}$ // Pooling
      $L_s += L_s^{(l)}$, $L_{cs} += L_{cs}^{(l)}$ // side loss terms at level l
    end for
  end if
end for
$Loss = L_{CE}(\hat{Y}^s, Y^s) + \beta d(G^s, G^t) + \gamma_1 L_s + \gamma_2 L_{cs}$
$\theta_h \leftarrow \theta_h - lr * \nabla_{\theta_h} Loss$

to that of baseline GDA models applying GCN to K-hop PPMI matrix [40, 47].

In our model, the high complexity mainly comes from the dense adjacency matrix at higher hierarchical levels. One way to improve the practicality is setting a moderately low but efficient value for r. While a higher value of r keeps richer structural information, it also significantly increases the size of adjacency matrix. We propose two additional solutions to achieve a balance between effectiveness and efficiency. First of all, we sparsify cluster assignment matrix $S^{(l)}$. As dense $S^{(l)}$ leads to a fully connected $G^{(l+1)}$, which significantly increases model complexity, we sparsify it by only keeping top-K assignment for each node. Such sparsification converts $S_i^{(l)}$ into a one-hot vector, indicating which cluster the node i should be mapped to. Besides, we further share *gpool* between source and target graphs to reduce space complexity. Experimental results in Section 7.5 validate the effectiveness of the proposed optimization methods.

6 THEORETICAL ANALYSIS

In this section we first provide a theoretical insight of *JHGDA* by proving the proposed domain discrepancy (i.e., Equation 6, 10) could be viewed as the combination of graph kernels, each of which compares source and target graphs at different levels of hierarchy. Such specially designed graph kernels have additional merits including task-adaptive and differentiable compared to conventional graph kernels. Subsequently, we prove the effectiveness of *JHGDA* by demonstrating its target risk is bounded, and such upper bound could be progressively reduced.

Definition 1. *(Convolution Graph Kernel) Graph kernels are widely adopted to measure similarity between graphs. We consider graph kernels that perform pairwise comparisons between local sub-structures centered at every node [16, 28]. Given mapping R^{-1} which decompose graph into such sub-structures, the graph kernel K could be defined as:*

$$K(G, G') = \sum_{x \in R^{-1}(G)} \sum_{y \in R^{-1}(G')} k_{base}(x, y)$$

By using the definition of graph kernel, we first prove that $d(G^s, G^t)$ defined without class conditional distribution shit (i.e., defined by Equation 7) equals the combination of graph kernels.

Theorem 1.

$$d(G^s, G^t) = \sum_{l=0}^{L} Q^l e^{\alpha c^l K^l(G^s, G^t)}$$

where K^l, Q^l and c^l denotes graph kernels, terms unrelated to discrepancy and constant terms at level l, respectively.

Proof 1. *Firstly, we rewrite Equation 8 as:*

$$MMD(Z^{s,(l)}, Z^{t,(l)}) = A^l + B^l - c^l \sum_{i}^{n^{s,(l)}} \sum_{j}^{n^{t,(l)}} k_{i,j} \tag{14}$$

where A and B are unrelated to discrepancy.

When $l = 0$, substituting Equation 14 into $d(G^s, G^t)$ would yield:

$$d_b(G^{s,(0)}, G^{t,(0)}) = Q^0 * e^{\alpha c^0 \sum_{i=1}^{n^{s,(0)}} \sum_{j=1}^{n^{t,(0)}} k(Z_i^{s,(0)}, Z_j^{t,(0)})} \tag{15}$$

where $Q^0 = e^{-\alpha(A^0 + B^0)}$.

As $G^{(0)}$ is the original input graph, we use $\mathcal{V}$ as R^{-1} which decomposes graph into node sets. Therefore, $d(G^{s,(0)}, G^{t,(0)})$ could be rewritten by graph kernel K^0:

$$\begin{aligned} d_b(G^{s,(0)}, G^{t,(0)}) &= Q^0 e^{\alpha c^0 \sum_{u \in \mathcal{V}(G^{s,(0)})} \sum_{v \in \mathcal{V}(G^{t,(0)})} k(u,v)} \\ &= Q^0 e^{\alpha c^0 K^0(G^s, G^t)} \end{aligned} \tag{16}$$

When $l = 1$, R^{-1} is represented by pooling mapping $\mathcal{S}^{(0)}$ which maps nodes to clusters. Thus, Equation 14-16 could be rewritten accordingly and $d(G^{s,(1)}, G^{t,(1)})$ is represented by:

$$\begin{aligned} d(G^{s,(1)}, G^{t,(1)}) &= Q^1 e^{\alpha c^1 \sum_{u \in \mathcal{S}^{(0)}(G^{s,(0)})} \sum_{v \in \mathcal{S}^{(0)}(G^{t,(0)})} k(u,v)} \\ &= Q^1 e^{\alpha c^1 K^1(G^s, G^t)} \end{aligned} \tag{17}$$

When extending hierarchical level from l to L, R^{-1} could be represented by the composite function: $\mathcal{S}^{(l-1)} \circ \mathcal{S}^{(l-2)} \circ \cdots \mathcal{S}^{(0)}$, and $K^l(G^s, G^t)$ is computed accordingly.

As K^l represents graph kernel, the proposed domain discrepancy is in fact a combination of graph kernels. K^l compares sub-structures extracted by $\mathcal{S}^{(l-1)} \circ \mathcal{S}^{(l-2)} \circ \cdots \mathcal{S}^{(0)}$. All K^ls are differentiable and adaptive to tasks. Following the similar idea, we could prove that $d(G^s, G^t)$ defined with class conditional distribution shit (i.e., defined by Equation 10) is also a combination of graph kernels.

Theorem 2.

$$d(G^s, G^t) = \sum_{l=0}^{L} \Pi_c W^{c,l} e^{\alpha K^{c,l}(G^s, G^t)}$$

where $K^{c,l}$ and $W^{c,l}$ denote graph kernel and terms unrelated to discrepancy for each class c at level l.

Proof 2. *As most of the proof is similar to Proof 1, here we only emphasize the formulation of graph kernel $K^{c,l}$ for simplicity. We define graph kernel for each class at each hierarchical level, and the mapping R^{-1} for $K^{c,l}$ is defined as $C \circ \mathcal{S}^{(l-1)} \circ \mathcal{S}^{(l-2)} \circ \cdots \mathcal{S}^{(0)}$ where mapping C maps $G^{(l)}$ to a set of nodes belonging to class c.*

Finally, we show that target risk of the proposed model is bounded based on $\mathcal{H}\Delta\mathcal{H}$-divergence [1]. We use $\epsilon(h)$ to denote task risks on both domains.

Theorem 3.

$$\begin{aligned} \epsilon_T(h) \leq \hat{\epsilon_S}(h) &+ \frac{p}{2} \sum_{l=0}^{L} Q^l e^{\alpha c^l K^l(G^s, G^t)} \\ &+ \frac{1-p}{2} \sum_{l=0}^{L} \Pi_c W^{c,l} e^{\alpha K^{c,l}(G^s, G^t)} \\ &+ \lambda^* + \mathbf{C} \end{aligned}$$

where $\hat{\epsilon_S}(h)$ approximates $\epsilon_S(h)$ by cross entropy loss L_{CE} in practice. The second and the third term on the RHS represent domain discrepancy. λ^* is the ideal joint error on both domains, and **C** *is constant.*

Therefore, Theorem 3 provides insights on the effectiveness of *JHGDA* because it progressively reduce the first three terms of target risk upper bound.

7 EXPERIMENTS

We evaluate *JHGDA* against several state-of-the-art GDA models and other possible baselines, with the aim of addressing the following questions:

Q1 How does *JHGDA* perform compared to other baselines on multiple datasets? Is there a way to predetermine in which scenarios the model will perform better?

Q2 What is the effectiveness of 1) extracting hierarchical graph structures and 2) jointly modeling marginal and class conditional distribution shifts?

Q3 What is the sensitivity of hyper-parameters?

Q4 What is the effectiveness of the proposed optimization solutions?

7.1 Experimental Settings

To prove the superiority of *JHGDA* on cross-network node classification tasks, we evaluate it on various types of datasets, including *Airport* [32], *Citation* [35] and *Blog* [35]. *Airport* contains airport traffic networks from three countries: USA, Brazil and Europe, in which the node indicates airport and the edge indicates the existence of commercial fights. *Citation* contains three citation networks Acmv9 (A), Citationv1 (C) and Dblpv7 (D) which are extracted from the ArnetMiner datasets [38]. In these networks a node represents a paper and an edge indicates the citation relationship. *Blog* contains two disjoint social networks Blog1 (B1) and Blog2 (B2) which are extracted from the BlogCatalog dataset [18]. Each node represents a blogger and each edge indicates the friendship between two bloggers. These datasets are widely adopted by this line of research,

and we build cross-network node classification tasks between every two graphs within the same dataset.

Baselines. We consider baselines from four categories:

Inductive Graph Neural Network GCN [15] and GraphSAGE [12] are used in an inductive manner, where they are trained on source graph and tested directly on target graph.

Conventional Domain Adaptation DANN [8] is selected from conventional domain adaptation models which ignore the intrinsic relationship between samples.

Graph Domain Adaptation A branch of SOTA GDA models based on GNN is selected, including AdaGCN [6], UDAGCN [40], ASN [47], GRADE-N [39] and SpecReg [46].

GNN with self-supervised learning EGI [48] and GCC [30] are chosen as they are representative works in the field of self-supervised graph models [22]. In practice, EGI is trained on source graph and tested on target graph. For GCC, we load the pre-trained model provided by authors to infer source and target node embeddings, after which we train an SVM on source embeddings for node classification and apply it to target embeddings.

Model Configuration. *JHGDA* adopts an architecture *gconv1-gpool1-gconv2* to each graph, and *gconv* is always shared between graphs. That is, *gconv1* is firstly applied to the original graphs for updating node embeddings, then *gpool1* is stacked on *gconv1* for pooling. Finally, *gconv2* is applied to the pooled graphs for update embeddings. Each block comprises two GCN layers, in which hidden dimensions follow the same setting of baselines. We use Gaussian RBF kernel in Equation 8-9, and α is empirically set to 0.1. The coefficients β and γ_1 are both set to 0.1 following conventional settings [39, 45]. The pooling rate r equals 0.5 on *Airport* and is reduced to 0.3 and 0.2 on *Citation* and *Blog*, because they are much larger. We train the model using Adam optimizer with learning rates tuned from {5e-1, 5e-2, 5e-3}. We report classification accuracy averaged over 10 runs.

7.2 Results for Cross-network Node Classification

Table 1 and 2 compare the performance of *JHGDA* to baselines, and the results provide positive answers to the first part of **Q1**. Across all datasets, *JHGDA* achieves the highest average performance, surpassing the SOTA baselines by 8.3%, 3.9% and 1.8% on *Airport*, *Citation* and *Blog*, respectively. Observably, the naive inductive GNN and conventional DA models often underperform due to their inability to adequately capture domain shifts or structural information. The performance of GCC is hindered by a gap between pretraining and testing, while EGI's limitations stem from an inability to model domain discrepancy. In contrast, GDA models achieve the best overall performance across most scenarios, indicating the superiority of GDA in addressing cross-network node classification tasks. It is also noteworthy that *JHGDA* consistently performs well in all datasets, which is hard to achieve by baseline models. The optimal GDA baselines vary across datasets with GRADE-N, ASN, and AdaGCN. The superior performance of *JHGDA* can be attributed to its ability to model the intrinsic hierarchical property of networks, which widely exists across various scenarios. Furthermore, the underperformance of GDA baselines can be linked to their neglect of class conditional distribution shift.

To address the second part of **Q1**, we assume that the degree of a network's hierarchical property is directly proportional to the performance of *JHGDA*. In practice, we adopt the global clustering coefficient (CC) [14] as the measurement of network hierarchy [5, 31, 36]. We compute the Pearson correlation between the average performance gain by which we surpass the best baseline GDA model and the average CC on each dataset. The correlation coefficient equals 0.95, indicating a strong positive correlation. Therefore, *JHGDA* tend to perform better on datasets with a higher degree of hierarchy. Hopefully, many real-world networks evidently exhibit intrinsic hierarchy [31].

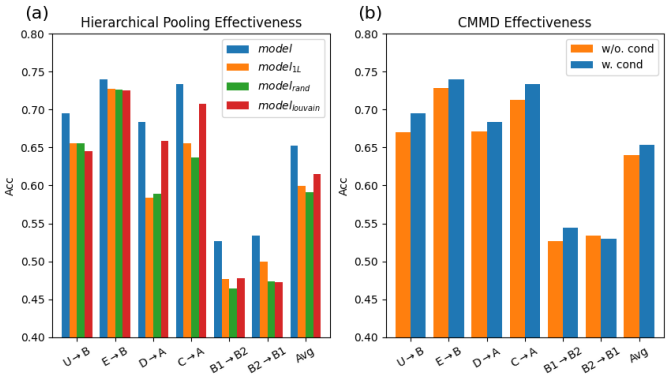

Figure 3: Effectiveness of adopting hierarchical pooling and class conditional distribution shift.

7.3 Effectiveness Analysis

To address **Q2**, we conduct ablation studies to independently investigate the effectiveness of hierarchical pooling and the adoption of CMMD. Initially, we create a variant, $model_{rand}$, in which a random pooling block substitutes the *gpool* in the hierarchical pooling module, resulting in a random distribution of nodes to clusters. We also create another variant $model_{louvain}$ by substituting *gpool* with louvain community detection method [3], which maps nodes to fixed communities. Lastly, We remove hierarchical pooling entirely, focusing solely on the original graph (denoted as $model_{1L}$). Figure 3(a) reveals the results, which suggest that 1) the inclusion of hierarchy generally enhances performance and 2) the hierarchy generated by *JHGDA* consistently proves to be the most effective and reliable. In addition, we investigate the incorporation of CMMD to assess the advantage of modeling class conditional distribution shift, and the results are shown in Figure 3(b). Comparing the two models based on Equation 7 and 10, the latter consistently outperforms, revealing the superiority of jointly modeling marginal and class conditional distribution shifts.

7.4 Parameter Sensitivity

To address **Q3**, we investigate hyper-parameters from the aspect of model architecture and loss-related coefficients.

Parameters related to model architecture. First, we conduct experiments to identify an effective yet low pooling rate $r^{(l)}$. Although a higher pooling rate keeps more structural information, it

Table 1: Classification accuracy on Airport

	USA → Brazil	USA → Europe	Brazil→USA	Brazil→Europe	Europe→USA	Europe→Brazil	Avg.
GCN	0.427	0.436	0.454	0.481	0.458	0.465	0.453
GSAGE	0.445	0.433	0.461	0.458	0.463	0.455	0.452
DANN	0.501	0.386	0.402	0.350	0.436	0.538	0.436
UDAGCN	0.607	0.488	0.497	0.510	0.434	0.477	0.502
AdaGCN	0.466	0.434	0.501	0.486	0.456	0.561	0.484
ASN	0.519	0.469	0.498	0.494	0.466	0.595	0.507
GRADE-N	0.550	0.457	0.497	0.506	0.463	0.588	0.510
SpecReg	0.481	0.444	0.480	0.546	0.436	0.527	0.486
EGI	0.523	0.451	0.417	0.454	0.452	0.588	0.481
GCC	0.351	0.331	0.272	0.396	0.275	0.366	0.332
JHGDA	**0.695**	**0.519**	**0.511**	**0.569**	**0.522**	**0.740**	**0.593**

Table 2: Classification accuracy on Citation and Blog

	Citation							Blog		
	A→D	D→A	A→C	C→A	C→D	D→C	Avg.	B1→B2	B2→B1	Avg.
GCN	0.623	0.578	0.675	0.635	0.666	0.654	0.638	0.501	0.467	0.484
GSAGE	0.665	0.593	0.717	0.653	0.701	0.685	0.669	0.460	0.446	0.453
DANN	0.488	0.436	0.520	0.518	0.518	0.465	0.491	0.410	0.419	0.415
AdaGCN	0.687	0.663	0.701	0.643	0.709	0.702	0.684	0.522	**0.532**	0.527
UDAGCN	0.684	0.623	0.728	0.663	0.712	0.645	0.676	0.522	0.517	0.519
ASN	0.729	0.723	0.752	0.678	0.752	0.754	0.731	0.515	0.498	0.506
GRADE-N	0.701	0.660	0.736	0.687	0.722	0.687	0.699	0.507	0.473	0.490
SpecReg	0.662	0.550	0.733	0.582	0.668	0.697	0.649	0.383	0.376	0.379
EGI	0.647	0.557	0.676	0.598	0.662	0.652	0.632	0.261	0.258	0.260
GCC	0.315	0.280	0.256	0.254	0.292	0.256	0.276	0.146	0.153	0.149
JHGDA	**0.755**	**0.737**	**0.814**	**0.757**	**0.762**	**0.794**	**0.770**	**0.544**	0.530	**0.537**

significantly increases both time and space complexity. The results on *Airport* and *Citation* are shown in Figure 4, from which we select the appropriate $r^{(l)}$ for each dataset. For *Citation* dataset, $r^{(l)}$ over 0.5 leads to OOM and thus is left under-explored. Generally, as there is no significant merit in increasing pooling rates, a heuristic setting could be around 0.5 for balancing performance and complexity.

Subsequently, we conduct experiments to explore the effect of involving hierarchies at higher levels on *Airport* dataset without loss of generality. Obviously, stacking more *gpool*s could extract higher network hierarchy, but it is both time and spacial consuming. We observe that the model which stacks two *gpool*s for each graph performs slightly better but increases ×3 times of GPU memory usage. Thus, a heuristic setting could be adopting single *gpool* for each graph.

Parameters related to loss terms. We analyze two crucial coefficients in loss terms, p and γ_2. p balances marginal distribution shift and class conditional distribution shift, which are modeled in Equation 10, while γ_2 determines the weight of conditional side loss L_{cs}. We conducted experiments on different tasks, with the results from *Citation* dataset presented in Figure 5, which typifies the phenomena observed across all datasets. First of all, we observe from Figure 5(a) that increasing p from 0 to 1 initially boosts performance, yet any further increases appear detrimental. This finding underlines that modeling marginal distribution shift is essential, and incorporating class conditional distribution shift benefits the task. Consequently, a heuristic setting of p could be slightly larger than 0.5. As for γ_2, the findings in Figure 5(b) suggest that regularizing L_{cs} is beneficial and one could heuristically set γ_2 slightly larger than 0.1.

7.5 Optimization Effectiveness

To address **Q4**, we conduct experiments on optimized models. First of all, results in Figure 6(a) reveal that sparsifying assignment matrix by keeping top-1 assignment is sufficient for a competitive performance, despite a slight performance decrease. Meanwhile, it leads to ×1.5 reduction in time consumption and ×1.4 reduction in GPU memory usage on *Citation* dataset (tested when $r^{(l)} = 0.2$). This optimization becomes even more significant when dealing with

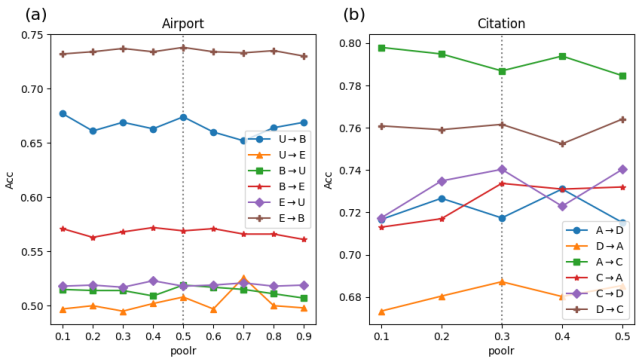

Figure 4: Correlation between pooling rate and task performance across different datasets. We mark the pooling rates which best balance complexity and efficiency in vertical dash lines.

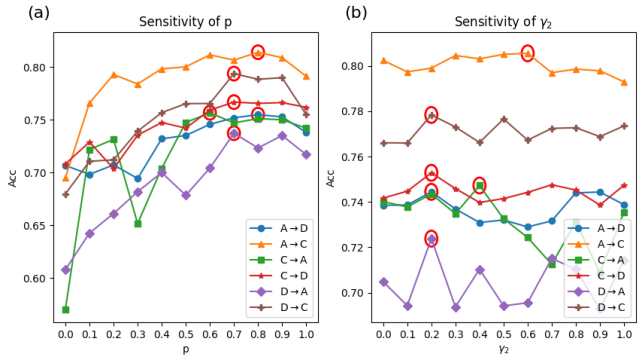

Figure 5: Sensitivity of coefficients in loss terms. We highlight the coefficient-performance pair that yields the best result for each task with a red circle.

larger pooled graphs. Secondly, although sharing *gpool* may overlook distinct hierarchies of each graph, our results in Figure 6(b) demonstrate similar effectiveness in sharing pooling blocks between graphs.

7.6 Visualization

In this section, we visualize node embeddings generated by competitive GDA models in the task $C \longrightarrow D$. The goal is to gain insight into model performance. We observe from Figure 7 with two phenomenons. Firstly, the nodes belonging to different classes are well-separated from each other. This shows that *JHGDA* is effective in distinguishing between different classes in the embedding space. Secondly, nodes belonging to the same class from different domains are mostly overlapping, which indicates that *JHGDA* could significantly reduce domain discrepancy. The first observation indicates good classification, while the latter suggests good adaptation.

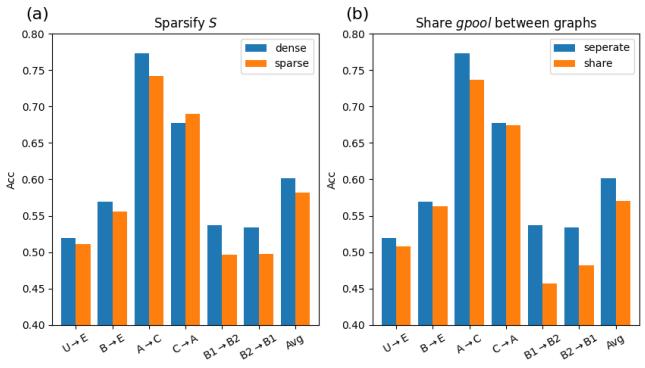

Figure 6: The performance of the proposed optimization solutions compared with the original design.

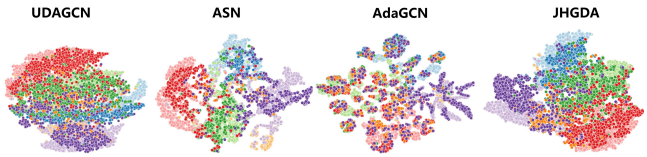

Figure 7: Visualizing source and target node embeddings via T-SNE. Each color indicates a class, while dark and light shades of the same color represent nodes belonging to the same class from source and target domains.

8 CONCLUSION

In this work, we mainly propose *JHGDA* to improve the performance of GDA models in cross-network node classification tasks. The key idea is to utilize the intrinsic hierarchical structures of graphs for improving the computation of domain discrepancy. By doing so, we can highlight subtle differences that may not be immediately apparent at the node-level. Additionally, we incorporate class-conditional distribution shift into discrepancy computing. This allows for more effective transfer of label information from source graphs to target graphs. Subsequently, we prove the model effectiveness and strengthen the interpretability through theoretical analysis. We also conduct experimental studies to validate the superiority of *JHGDA*. In the future, we may strive to handle the cases where multiple source graphs could be utilized to transfer knowledge and design new frameworks for other cross-network learning tasks, including link prediction and graph classification. We will also deep into the theory of graph domain adaptation for developing more powerful models.

ACKNOWLEDGMENTS

This work was funded by the National Key R&D Program Young Scientists Project under grant number 2022YFB3102200 and the National Natural Science Foundation of China under grant number U21B2046. This work is also supported by the fellowship of China Postdoctoral Science Foundation 2022M713206.

REFERENCES

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79, 1 (2010), 151–175.

[2] Roy Benjamin, Uriel Singer, and Kira Radinsky. 2022. Graph Neural Networks Pretraining Through Inherent Supervision for Molecular Property Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2903–2912.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[4] Ruichu Cai, Fengzhu Wu, Zijian Li, Pengfei Wei, Lingling Yi, and Kun Zhang. 2021. Graph domain adaptation: A generative view. *arXiv preprint arXiv:2106.07482* (2021).

[5] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. 2007. Structural inference of hierarchies in networks. In *Statistical Network Analysis: Models, Issues, and New Directions: ICML 2006 Workshop on Statistical Network Analysis, Pittsburgh, PA, USA, June 29, 2006, Revised Selected Papers*. Springer, 1–13.

[6] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. 2022. Graph Transfer Learning via Adversarial Domain Adaptation with Graph Convolution. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[7] Alexandre Duval and Fragkiskos Malliaros. 2022. Higher-order clustering and pooling for graph neural networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 426–435.

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[9] Hongyang Gao, Yongjun Chen, and Shuiwang Ji. 2019. Learning graph pooling and hybrid convolutional operations for text representations. In *The world wide web conference*. 2743–2749.

[10] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.

[11] Gaoyang Guo, Chaokun Wang, Bencheng Yan, Yunkai Lou, Hao Feng, Junchao Zhu, Jun Chen, Fei He, and Philip Yu. 2022. Learning adaptive node embeddings across graphs. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[13] Takeshi D Itoh, Takatomi Kubo, and Kazushi Ikeda. 2022. Multi-level attention pooling for graph neural networks: Unifying graph representations with multiple localities. *Neural Networks* 145 (2022), 356–373.

[14] Andreas Kemper. 2009. *Valuation of network effects in software markets: A complex networks approach*. Springer Science & Business Media.

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. 2020. A survey on graph kernels. *Applied Network Science* 5, 1 (2020), 1–42.

[17] Haoran Li, Hanghang Tong, and Yang Weng. 2022. Domain Adaptation in Physical Systems via Graph Kernel. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 868–876.

[18] Jundong Li, Xia Hu, Jiliang Tang, and Huan Liu. 2015. Unsupervised streaming feature selection in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1041–1050.

[19] Xinhang Li, Zhaopeng Qiu, Xiangyu Zhao, Zihao Wang, Yong Zhang, Chunxiao Xing, and Xian Wu. 2022. Gromov-Wasserstein Guided Representation Learning for Cross-Domain Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1199–1208.

[20] Chuang Liu, Yibing Zhan, Chang Li, Bo Du, Jia Wu, Wenbin Hu, Tongliang Liu, and Dacheng Tao. 2022. Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321* (2022).

[21] Ning Liu, Songlei Jian, Dongsheng Li, and Hongzuo Xu. 2022. Unsupervised Hierarchical Graph Pooling via Substructure-Sensitive Mutual Information Maximization. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1299–1308.

[22] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2022), 5879–5900.

[23] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. 2013. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*. 2200–2207.

[24] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*. PMLR, 2208–2217.

[25] Bin Lu, Xiaoying Gan, Weinan Zhang, Huaxiu Yao, Luoyi Fu, and Xinbing Wang. 2022. Spatio-Temporal Graph Few-Shot Learning with Cross-City Knowledge Transfer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1162–1172.

[26] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *International conference on machine learning*. PMLR, 4212–4221.

[27] Xiaojun Ma, Qin Chen, Yi Wu, Guojie Song, Liang Wang, and Bo Zheng. 2023. Rethinking Structural Encodings: Adaptive Graph Transformer for Node Classification Task. In *Proceedings of the ACM Web Conference 2023*. 533–544.

[28] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. 2021. Graph kernels: A survey. *Journal of Artificial Intelligence Research* 72 (2021), 943–1027.

[29] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[30] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1150–1160.

[31] Erzsébet Ravasz and Albert-László Barabási. 2003. Hierarchical organization in complex networks. *Physical review E* 67, 2 (2003), 026112.

[32] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 385–394.

[33] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[34] Xiao Shen, Quanyu Dai, Fu-lai Chung, Wei Lu, and Kup-Sze Choi. 2020. Adversarial deep network embedding for cross-network node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2991–2999.

[35] Xiao Shen, Quanyu Dai, Sitong Mao, Fu-lai Chung, and Kup-Sze Choi. 2020. Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems* 32, 5 (2020), 1935–1948.

[36] Sara Nadiv Soffer and Alexei Vázquez. 2005. Network clustering coefficient without degree-correlation biases. *Phys. Rev. E* 71 (May 2005), 057101. Issue 5.

[37] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1717–1727.

[38] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 990–998.

[39] Jun Wu, Jingrui He, and Elizabeth Ainsworth. 2022. Non-IID Transfer Learning on Graphs. *arXiv preprint arXiv:2212.08174* (2022).

[40] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. 2020. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*. 1457–1467.

[41] Ning Wu, Xin Wayne Zhao, Jingyuan Wang, and Dayan Pan. 2020. Learning effective road network representation with hierarchical graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 6–14.

[42] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. 2022. Handling Distribution Shifts on Graphs: An Invariance Perspective. In *International Conference on Learning Representations*.

[43] Haotian Xue, Kaixiong Zhou, Tianlong Chen, Kai Guo, Xia Hu, Yi Chang, and Xin Wang. 2021. CAP: Co-Adversarial Perturbation on Weights and Features for Improving Generalization of Graph Neural Networks. *arXiv preprint arXiv:2110.14855* (2021).

[44] Jieyu Yang, Zhaoxin Huan, Yong He, Ke Ding, Liang Zhang, Xiaolu Zhang, Jun Zhou, and Linjian Mo. 2022. Task Similarity Aware Meta Learning for Cold-Start Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4630–4634.

[45] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems* 31 (2018).

[46] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2023. Graph Domain Adaptation via Theory-Grounded Spectral Regularization. In *The Eleventh International Conference on Learning Representations*.

[47] Xiaowen Zhang, Yuntao Du, Rongbiao Xie, and Chongjun Wang. 2021. Adversarial Separation Network for Cross-Network Node Classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2618–2626.

[48] Qi Zhu, Carl Yang, Yidan Xu, Haonan Wang, Chao Zhang, and Jiawei Han. 2021. Transfer learning of graph neural networks with ego-graph information maximization. *Advances in Neural Information Processing Systems* 34 (2021), 1766–1779.

[49] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A comprehensive survey on transfer learning. *Proc. IEEE* 109, 1 (2020), 43–76.