
Graph Data Selection for Domain Adaptation: A Model-Free Approach

Ting-Wei Li

University of Illinois

Urbana-Champaign, IL USA

twli@illinois.edu

Ruizhong Qiu

University of Illinois

Urbana-Champaign, IL USA

rq5@illinois.edu

Hanghang Tong

University of Illinois

Urbana-Champaign, IL USA

htong@illinois.edu

Abstract

Graph domain adaptation (GDA) is a fundamental task in graph machine learning, with techniques like shift-robust graph neural networks (GNNs) and specialized training procedures to tackle the distribution shift problem. Although these *model-centric* approaches show promising results, they often struggle with severe shifts and constrained computational resources. To address these challenges, we propose a novel *model-free* framework, GRADATE (*GRApH DATA sElector*), that selects the best training data from the source domain for the classification task on the target domain. GRADATE picks training samples without relying on any GNN model’s predictions or training recipes, leveraging optimal transport theory to capture and adapt to distribution changes. GRADATE is *data-efficient*, scalable and meanwhile complements existing model-centric GDA approaches. Through comprehensive empirical studies on several real-world graph-level datasets and multiple covariate shift types, we demonstrate that GRADATE outperforms existing selection methods and enhances off-the-shelf GDA methods with much fewer training data.

1 Introduction

Graphs have emerged as a fundamental data structure for representing complex relationships across diverse domains, from modeling molecular interactions in biological networks [23, 6, 37, 85] to capturing user behaviors in recommendation systems [15, 19, 8, 9, 73]. In graph-level classification tasks [64, 66, 81, 20, 38], where the goal is to categorize graph structures, the distribution shift between source and target domains poses significant challenges. While numerous graph neural network (GNN) methods have been proposed for graph domain adaptation (GDA) [61, 12, 62, 55, 34], they predominantly relies on model architecture design and training strategies, which are inherently *model-dependent* and brittle. This model-centricity introduces practical challenges: (i) the need for extensive resources to train and validate different architectural variants and (ii) the ignorance of source data quality. To address these aforementioned issues, rather than relying on sophisticated GNN architectures or training procedures, we aim to answer a fundamental question:

How to select the most relevant source domain data, based on available validation data, for better graph-level classification accuracy evaluated on the target domain ?

In this paper, we propose a *model-free* method, GRADATE (*GRApH DATA sElector*), that selects a subset of important training data in the source domain independently of any specific GNN model design, making it both *data-efficient* and versatile. GRADATE reduces computational overhead and enables quick adaptation to unseen graph domains based on available validation data. Conceptually, GRADATE first leverages Fused Gromov-Wasserstein (FGW) distance [57] to compare graph samples. We provide a theoretical justification to demonstrate FGW’s unique advantage over multi-layer GNNs for graph comparison. Then, FGW is used as a building block to measure the dataset-level distance between training and validation sets, which is termed as Graph Dataset Distance (GDD). Through

theoretical analysis, we demonstrate that the domain generalization gap between source and target domain is upper-bounded by GDD, which naturally motivates us to minimize this GDD between training and validation sets by identifying the optimal subset of training graph data. At the core of GRADATE lies our novel optimization procedure, GREAT (*GDD shRinkagE via spArse projecTion*), that interleaves between (i) optimal transport-based distribution alignment with gradient updates on training sample weights and (ii) projection of training sample weights to sparse probability simplex. This dual-step process systematically increases the weights of beneficial training samples while eliminating the influence of detrimental samples that could harm generalization performance.

By extensive experiments on six real-world graph-level datasets and two types of covariate shifts, we first show that GRADATE significantly outperforms existing data-efficient selection methods. When coupled with vanilla GNNs that are only trained on its selected data, GRADATE even surpasses state-of-the-art GDA methods. Intriguingly, the practical implications of GRADATE extend beyond mere data selection. By operating independently of model architecture, GRADATE can seamlessly complement existing off-the-shelf GDA methods, further enhancing their performance through better data curation while significantly improving the *data-efficiency*.

In summary, our main contributions in this paper are as follows:

1. **Theoretical justification of FGW distance.** We show in Theorem 3.1 that the output distance of multi-layer GNNs is upper-bounded by Fused Gromov–Wasserstein distance [57] between graphs, which motivates us to use FGW as a building block to compare graphs.
2. **Novel graph dataset distance formulation.** Through Theorem 3.3, we prove that the graph domain generalization gap is upper-bounded by a notion of Graph Dataset Distance (GDD).
3. **A strong model-free graph selector.** We introduce GRADATE (*GRApH DATa sElector*) as the first *model-free* method tailored for domain shift problem for graph-level classification tasks, complementing the predominant model-centric GDA methods (see Section 4.2).
4. **A data-efficient and powerful GDA method.** We show that trained with data selected by GRADATE, even vanilla GNN can beat sophisticated GDA baselines (see Section 4.3).
5. **A universal GDA model enhancer.** We demonstrate that the data selected by GRADATE can be combined with GDA methods to further enhance their performance and efficiency (see Section 4.4).

The rest of the paper is organized as follows. Section 2 introduces needed background knowledge. Section 3 details our definition of Graph Dataset Distance (GDD) and proposed GRADATE, followed by experimental results in Section 4. Section 5 presents related works. Finally, the conclusion is provided in Section 6.

2 Preliminaries

In this section, we briefly introduce optimal transport and graph optimal transport, which are fundamental background related to our proposed method. We also provide the formal problem formulation of graph domain adaptation (GDA) in Appendix K.

2.1 Optimal Transport

Optimal transport (OT) [27] defines a distance between probability distributions. It is defined as follows: given cost function $d(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ and probability distributions $\mathbf{p}, \mathbf{q} \in \mathcal{P}(\mathcal{X})$, where \mathcal{X} is a metric space, $\text{OT}(\mathbf{p}, \mathbf{q}, d) \triangleq \min_{\pi \in \Pi(\mathbf{p}, \mathbf{q})} \int_{\mathcal{X} \times \mathcal{X}} d(x, x') \pi(x, x') dx dx'$, where $\Pi(\mathbf{p}, \mathbf{q})$ is the set of couplings with marginals \mathbf{p} and \mathbf{q} . For supervised learning scenarios, we consider the empirical measures: $\mathbf{p} = \frac{1}{n} \sum_{i=1}^n \delta_{x_i}$ and $\mathbf{q} = \frac{1}{m} \sum_{j=1}^m \delta_{x'_j}$, where δ is the Dirac delta function. With the pairwise cost matrix $\mathbf{M} = [d(x_i, x'_j)]_{ij}$, we can re-formulate the OT problem as $\text{OT}(\mathbf{p}, \mathbf{q}, d) = \text{OT}(\mathbf{p}, \mathbf{q}, \mathbf{M}) \triangleq \min_{\pi \in \Pi(\mathbf{p}, \mathbf{q})} \sum_{i=1}^n \sum_{j=1}^m \mathbf{M}_{ij} \pi_{ij}$.

Optimal Transport Dataset Distance (OTDD). Alvarez-Melis and Fusi [2] construct a distance metric between *datasets*, where each dataset is represented as a collection of feature-label pairs $z = (x, y) \in \mathcal{Z} (= \mathcal{X} \times \mathcal{Y})$. The authors propose *label-specific distributions*, which can be seen as distributions over features X of data samples with a specific label y , i.e. $\alpha_y(X) \triangleq P(X|Y=y)$. The metric on the space of feature-label pairs can thus be defined as a combination of *feature distance* and *label distance*: $d_{\mathcal{Z}}((x, y), (x', y')) \triangleq (d_{\mathcal{X}}(x, x')^r + c \cdot d(\alpha_y, \alpha_{y'})^r)^{1/r}$, where $d_{\mathcal{X}}$ is a metric on \mathcal{X} ,

$d(\alpha_y, \alpha_{y'})$ is the *label distance* between distributions of features associated with labels y and y' , $r \geq 1$ is the order of the distances and $c \geq 0$ is a pre-defined weight parameter. Consider two datasets $D_s = \{z_i^s : (x_i^s, y_i^s)\}_{i \in [n_1]}$, $D_t = \{z_i^t : (x_i^t, y_i^t)\}_{i \in [n_2]}$ and their corresponding uniform empirical distributions $\mathbf{p}, \mathbf{q} \in \mathcal{Z}$, where $\mathbf{p} = \sum_{i=1}^{n_1} \frac{1}{n_1} \delta_{z_i^s}$ and $\mathbf{q} = \sum_{i=1}^{n_2} \frac{1}{n_2} \delta_{z_i^t}$, OTDD between D_s and D_t is computed as: $\text{OTDD}(D_s, D_t) = \text{OT}(\mathbf{p}, \mathbf{q}, d_Z) = \text{OT}(\mathbf{p}, \mathbf{q}, \mathbf{M}) = \min_{\pi \in \Pi(\mathbf{p}, \mathbf{q})} \mathbb{E}_{(z^s, z^t) \sim \pi} [d_Z(z^s, z^t)]$, where $\Pi(\mathbf{p}, \mathbf{q})$ is the set of valid couplings and $\mathbf{M} = [d_Z(z_i^s, z_j^t)]_{ij}$ is the pairwise cost matrix.

2.2 Graph Optimal Transport

The Fused Gromov-Wasserstein (FGW) distance [57] integrates the Wasserstein distances [52] and the Gromov-Wasserstein distances [54, 42]. Formally, a graph \mathcal{G} with n nodes can be represented as a distribution over vectors in a d -dimensional space, where d is the dimension of the node feature. The *features* are represented as $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the *structure* can be summarized in an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. We further augment \mathcal{G} a probability distribution $\mathbf{p} \in \mathbb{R}^n$ over nodes in the graph, where $\sum_{i=1}^n p_i = 1$ and $p_i \geq 0, \forall i \in [n]$. To compute FGW distance between two attributed graphs ($\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{X}_1, \mathbf{p}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{X}_2, \mathbf{p}_2\}$), we use pairwise feature distance as inter-graph distance matrix and adjacency matrices as intra-graph similarity matrices. The FGW distance is defined as the solution of the following optimization problem:

$$\text{FGW}_\alpha(\mathcal{G}_1, \mathcal{G}_2) \triangleq \text{FGW}_\alpha(\|\mathbf{X}_1[i] - \mathbf{X}_2[j]\|_r)_{ij}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{p}_1, \mathbf{p}_2, \alpha)$$

$$= \left(\min_{\pi \in \Pi(\mathbf{p}_1, \mathbf{p}_2)} \sum_{i,j,k,l} (1-\alpha) \|\mathbf{X}_1[i] - \mathbf{X}_2[j]\|_2^r + \alpha |\mathbf{A}_1(i, k) - \mathbf{A}_2(j, l)|^r \pi(i, j) \pi(k, l) \right)^{\frac{1}{r}},$$

where $\Pi(\mathbf{p}_1, \mathbf{p}_2) \triangleq \{\pi | \pi \mathbf{1}_{n_2} = \mathbf{p}_1, \pi^T \mathbf{1}_{n_1} = \mathbf{p}_2, \pi \geq 0\}$ is the collection of feasible couplings between \mathbf{p}_1 and \mathbf{p}_2 , $\alpha \in [0, 1]$ acts as a trade-off parameter, and $r \geq 1$ is the order of the distances.

3 Methodology

In this section, we propose our framework, *GRApH DATa sElector*, abbreviated as GRADATE. In Section 3.1, we first introduce Theorem 3.1 to motivate our use of LinearFGW [46] for graph distance computation. After that, we define the Graph Dataset Distance (GDD), which measures the discrepancy between graph sets. In Section 3.2, we provide Theorem 3.3 to bound the domain generalization gap using GDD and then propose a GDD minimization problem that is solved by a novel optimization algorithm, GREAT (*GDD shRinkagE via spArse projecTion*). Finally, we introduce GRADATE that combines GDD computation and GREAT to select the most important subset of the training data to solve graph domain adaptation problem (definition is detailed in Appendix K).

3.1 Graph Dataset Distance (GDD): A Novel Notion to Compare Graph Datasets

3.1.1 FGW Distance For Graph Comparison

Challenges in Graph Distance Computation. To find the optimal samples that can achieve better performance on the target domain, we first need an efficient way that can accurately capture the discrepancy among graphs. To achieve this, most methods rely on Graph Neural Networks (GNNs) [62, 12, 55, 72] to obtain meaningful representations of these structured objects. However, these approaches face critical limitations: (i) high computational complexity to train on full training set and (ii) sensitivity to GNN hyper-parameters. To address these drawbacks, we propose to use FGW distance [57] for replacing GNNs. As demonstrated in the following Theorem 3.1, FGW offers provable advantages that make it more suitable than GNNs to compare attributed graphs.

Theorem 3.1. *Given two graphs $\mathcal{G}_1 = (\mathbf{A}_1, \mathbf{X}_1)$ and $\mathcal{G}_2 = (\mathbf{A}_2, \mathbf{X}_2)$, for a k -layer graph neural network (GNN) f with ReLU activations, under regularity assumptions in Appendix F.1.1, we have*

$$d_W(f(\mathcal{G}_1), f(\mathcal{G}_2)) \leq C \cdot \text{FGW}_\beta(\mathcal{G}_1, \mathcal{G}_2), \quad (1)$$

where d_W denotes the r -Wasserstein distance, C and β are constants depending on GNN f , regularity constants and k .

Proof. The proof is in Appendix F.1. □

Implication of Theorem 3.1. We have the following two main insights: (i) since the theorem holds for any possible k -layer GNNs, with ReLU activations, FGW is provably able to capture the differences between attributed graphs in a way that upper-bounds the discrepancy between learned GNN representations; (ii) to the best of our knowledge, this is the first time that FGW is proved to be the distance metric that enjoys this theoretical guarantee, and hence we adopt it as the major basis for our graph data selection method.

Practical Consideration. We utilize LinearFGW [46] as an efficient approximation of FGW distance. Formally, LinearFGW defines a distance metric $d_{\text{LinearFGW}}(\cdot, \cdot)$ where $d_{\text{LinearFGW}}(\mathcal{G}_i, \mathcal{G}_j)$ is the LinearFGW distance between any pair of graphs $\mathcal{G}_i, \mathcal{G}_j$. LinearFGW offers an approximation to FGW with linear time complexity with respect to the number of training graphs. While we omit the details of LinearFGW here for brevity, they can be found in Appendix A and Algorithm 1.

3.1.2 Graph-Label Distance

With FGW as a theoretically grounded metric for *model-free* comparison between individual attributed graphs, we further extend it to compare sets of labeled graphs across domains, which aids our domain adaptation process. Inspired by OTDD [2] (detailed in Section 2.1), we extend the original definition of *label distance* to incorporate distributions in graph subsets S, S' , namely α_y^S and $\alpha_{y'}^{S'}$. Given a set of labeled graphs $\mathcal{D} = \{\mathcal{G}_i, y_i\}_{i=1}^N$ and label set \mathcal{Y} , we formulate the *graph-label distance* between label $y \in \mathcal{Y}$ in graph subset $S = \{\mathcal{G}_i^S, y_i^S\}_{i \in |S|} \subseteq \mathcal{D}$ and label $y' \in \mathcal{Y}$ in graph subset $S' = \{\mathcal{G}_j^{S'}, y_j^{S'}\}_{j \in |S'|} \subseteq \mathcal{D}$ as follows:

$$d(\alpha_y^S, \alpha_{y'}^{S'}) = \text{OT}(\mathbf{p}_y^S, \mathbf{q}_{y'}^{S'}, d_{\text{LinearFGW}}), \quad (2)$$

where $\mathbf{p}_y^S = \frac{1}{|\{i: y_i^S = y\}|} \sum_{i: y_i^S = y} \delta_{\mathcal{G}_i^S}$, $\mathbf{q}_{y'}^{S'} = \frac{1}{|\{j: y_j^{S'} = y'\}|} \sum_{j: y_j^{S'} = y'} \delta_{\mathcal{G}_j^{S'}}$ are label-specific uniform empirical measures with the distance metric measured by LinearFGW. Intuitively, we collect graphs in subset S with label y and graphs in subset S' with label y' as distributions. Then, we compute the optimal transport distance between these distributions and define the distance as *graph label distance*.

3.1.3 Graph Dataset Distance (GDD)

Building upon the aforementioned graph-label distance, we then propose the notion of Graph Dataset Distance (GDD), which compares two graph subsets at a dataset-level. Specifically, based on Equation (2), we can define a distance metric $d_{g\mathcal{Z}}^c$ between graph subsets S, S' :

$$d_{g\mathcal{Z}}^c((\mathcal{G}_i^S, y_i^S), (\mathcal{G}_j^{S'}, y_j^{S'})) = d_{\text{LinearFGW}}(\mathcal{G}_i^S, \mathcal{G}_j^{S'}) + c \cdot d(\alpha_y^S, \alpha_{y'}^{S'}), \quad (3)$$

which is a combination of LinearFGW distance and graph-label distance with a weight parameter $c \geq 0$ balancing the importance of two terms. GDD can thus be defined as:

$$\text{GDD}(\mathcal{D}^S, \mathcal{D}^{S'}) = \text{OT}(\mathbf{p}^S, \mathbf{q}^{S'}, d_{g\mathcal{Z}}^c), \quad (4)$$

where $\mathbf{p}^S = \frac{1}{|S|} \sum_{i \in [|S|]} \delta_{(\mathcal{G}_i^S, y_i^S)}$ and $\mathbf{q}^{S'} = \frac{1}{|S'|} \sum_{j \in [|S'|]} \delta_{(\mathcal{G}_j^{S'}, y_j^{S'})}$ are uniform empirical measures. We summarize the computation of GDD in Appendix B and Algorithm 2.

Remark 3.2. If we set $c = 0$, GDD will omit the label information and only consider the distributional differences of graph data themselves, which matches the setting of unsupervised GDA problem.

3.2 GRADATE: A Model-Free Graph Data Selector

3.2.1 GDD Bounds Domain Generalization Gap

We give the following Theorem 3.3 to elucidate the utility of GDD and its relation to model performance discrepancy between graph domains. In short, we seek to utilize empirical observations in the source domain to minimize the expected risk calculated on the target domain P_t , namely, $\mathbb{E}_{(G, y) \sim P_t} [\mathcal{L}(f(\mathcal{G}), y)]$, which promotes the model performance (i.e., lower expected risk) on the target domain.

Theorem 3.3 (Graph Domain Generalization Gap). *Define the cost function among graph-label pairs as $d_{g\mathcal{Z}}^c$ with some positive c (via Equation (3)). Let \mathbf{w} denote the source distribution weight. Suppose that $d_{g\mathcal{Z}}^c$ is C -Lipschitz. Then for any model f trained on a training set, we have*

$$\mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{q}^{\text{val}}} [\mathcal{L}(f(\mathcal{G}), y)] \leq \mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{p}^{\text{train}}(\mathbf{w})} [\mathcal{L}(f(\mathcal{G}), y)] + C \cdot \text{GDD}(\mathcal{D}_{\mathbf{w}}^{\text{train}}, \mathcal{D}^{\text{val}}),$$

where $\text{GDD}(\mathcal{D}_{\mathbf{w}}^{\text{train}}, \mathcal{D}^{\text{val}}) = \text{OT}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, d_{g\mathcal{Z}}^c)$ is the graph dataset distance between the weighted training dataset (defined by \mathbf{w}) and target dataset.

Proof. The proof can be found in Appendix F.2. \square

Implication of Theorem 3.3. If we can lower the GDD between training and validation data, the discrepancy in the model performance with respect to training and validation sets may also decrease. Specifically, under the scenario where distribution shift occurs, some source data might be irrelevant or even harmful when learning a GNN model that needs to generalize well on the target domain. This motivates us to present our main framework, GRADATE (*GRApH DATa sElectoR*), which selects the best training data from the source domain for graph domain adaptation.

3.2.2 GDD Minimization Problem

Guided by the implication of Theorem 3.3, we formulate the GDD minimization problem as follows.

Definition 3.4 (Graph Dataset Distance Minimization). Given training and validation sets $\mathcal{D}^{\text{train}}$ and \mathcal{D}^{val} , we aim to find the best distribution weight \mathbf{w}^* that minimizes GDD between the training and validation sets under the sparsity constraint. Namely,

$$\mathbf{w}^* = \min_{\mathbf{w}} \text{OT}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, d_{g\mathcal{Z}}^c), \quad \text{s.t. } \sum_i \mathbf{w}[i] = 1, \mathbf{w} \geq 0, \|\mathbf{w}\|_0 \leq \lfloor n \cdot \tau \rfloor, \quad (5)$$

where $\mathbf{p}^{\text{train}}(\mathbf{w}) = \sum_{i \in [n]} \mathbf{w}_i \delta_{(\mathcal{G}_i^{\text{train}}, y_i^{\text{train}})}$ and $\mathbf{q}^{\text{val}} = \frac{1}{m} \sum_{j \in [m]} \delta_{(\mathcal{G}_j^{\text{val}}, y_j^{\text{val}})}$.

Optimization Procedure (GREAT algorithm). To solve the above GDD minimization problem, we propose GREAT (*GDD shRinkagE via spArse projecTion*) to optimize the weight \mathbf{w} over the entire training set. Starting from a uniform training weight vector \mathbf{w} , GREAT iteratively refines the importance of training samples through two key steps. In each iteration, it first computes the Graph Dataset Distance (GDD) between the reweighted training distribution $\mathbf{p}^{\text{train}}(\mathbf{w})$ and the validation distribution \mathbf{q}^{val} , using a pairwise cost matrix $\tilde{\mathbf{D}} \in \mathbb{R}^{n \times m}$ that encodes distances between individual training and validation samples. The gradient of GDD with respect to \mathbf{w} , denoted $\mathbf{g}_{\mathbf{w}}$, is then used to update the weights¹. Following this, the weight vector \mathbf{w} is sparsified by retaining only the top- k largest entries and re-normalized to remain on the probability simplex. After T such iterations, the non-zero indices in the final \mathbf{w} define the selected training subset \mathbf{S} . The detailed algorithm procedure is presented in Appendix C and Algorithm 3.

3.2.3 GRADATE

Combining GDD computation and optimization module GREAT, we summarize the main procedure of GRADATE in Algorithm 4 (details can be found in Appendix D). In short, given training and validation data, GRADATE iteratively calculates GDD based on current training weight and searches for a better one through GREAT. The final output of GRADATE corresponds to the selected subset of training data that is best suitable for adaptation to the target domain. We further provide the time complexity analysis of GRADATE as follows.

Time Complexity of GRADATE. Let N be the number of training graphs, M be the number of validation graphs, n be the number of nodes in each graph (WLOG, we assume all graph share the same size), L be the largest class size, τ is the approximation error introduced by approximate OT solvers², K be the number of iterations for solving LinearFGW, and T the number of update steps used in GREAT. The runtime complexity can be summarized in the following proposition.³

¹Note that we leverage the conclusion introduced in [26] to compute this gradient (stated as Theorem F.3).

²This is due to the entropic regularization in Sinkhorn iterations for empirical OT calculation.

³For empirical runtime behavior, we refer readers to Appendix M.

Proposition 3.5 (Time Complexity Analysis [2, 26, 1]). *The off-line procedure of GRADATE (i.e. can be computed before accessing the test set) has the time complexity $\mathcal{O}(NMKn^3 + NML^3 \log L)$ and the on-line procedure of GRADATE has the time complexity $\mathcal{O}(TNM \log(\max(N, M))\tau^{-3})$.*

4 Experiments

We conduct extensive experiments to evaluate the effectiveness of GRADATE across six real-world graph classification settings under two different types of distribution shift. Our experiments are designed to answer the following research questions:

- **(RQ1):** How does GRADATE compare to existing data selection methods?
- **(RQ2):** How does GRADATE + vanilla GNNs compare to model-centric GDA methods?
- **(RQ3):** To what extent can GRADATE enhance the effectiveness of model-centric GDA methods?

We will answer these research questions in Section 4.2, 4.3 and 4.4, correspondingly.

4.1 General Setup

In this section, we state the details of datasets and settings of GRADATE and baseline methods.

Datasets and Graph Domains. We consider graph classification tasks conducted on six real-world graph-level datasets, including IMDB-BINARY [69], IMDB-MULTI [69], MSRC_21 [45], ogbg-molbace [22], ogbg-molbbbp [22] and ogbg-molhiv [22]. The former three datasets are from the TUDataset [44]; while the latter three datasets are from the OGB benchmark [22]. We define the graph domains by *graph density* and *graph size*, which are the types of covariate shift that are widely studied in the literature [72, 41, 56, 5, 71, 10, 84]. Specifically, graphs are sorted by corresponding properties in an *ascending* order and split into train/val/test sets with ratios 60%/20%/20%. For brevity, we provide results on *graph density* shift in the main content. Additional experiments on *graph size* shift can be found in Appendix G. We also include the empirical cumulative distribution function (ECDF) plots of all settings in Appendix P to demonstrate the shift level.

Details of GRADATE and Baselines. To compute LinearFGW within GRADATE, we follow the default parameter settings in its github repository.⁴ The trade-off parameter α is computed in $\{0.5, 0.9\}$ ⁵ and the order r is set to 2. The update step is fixed to $T = 10$ and the learning rate equal to $\eta = 10^{-4}$ across different settings. A popular model-free data valuation method is LAVA [26]. We apply LAVA for graph data selection and make the following modifications. We first leverage LinearFGW to form the pairwise distance matrix and compute GDD. LAVA then picks the smallest k entries of the calibrated gradients as output. For the computation of GDD, we consider label signal $c \in \{0, 5\}$. We also incorporate KiDD-LR [66] as a model-centric but data-efficient baseline, which is a state-of-the-art graph dataset distillation method.

4.2 GRADATE as a Model-Free Graph Selector

To answer **(RQ1)**, we compare GRADATE with other data-efficient methods, including (1) model-free techniques: random selection and LAVA [26] and (2) model-centric techniques: KiDD-LR [66].

Experiment Setup. To test the effectiveness of these selection methods, we fix the backbone GNN models in use to train the data selected by each method. Two popular GNN models are chosen, including GCN [30] and GIN [65] with default hyper-parameters following Zeng et al. [81] and the corresponding original papers. We also consider results on GAT [58] and GraphSAGE [18]. Please see Appendix G.1, G.2 and G.3 for more details. Here we consider selection ratio $\tau \in [0.1, 0.2, 0.5]$. More details on the architectures and training protocol can be found in Appendix I.

Results. As shown in Table 1, across all datasets, GRADATE outperforms the baseline methods under different selection ratios. It is also worth noting that even with few selected data, GRADATE

⁴<https://github.com/haidnguyen0909/LinearFGW>

⁵Since the datasets do not contain node features, we consider a larger α to place a greater emphasis on the structural properties.

Dataset	GNN Architecture →			GCN			GIN		
	Selection Method ↓	$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full	$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full
IMDB-BINARY	Random	0.737 \pm 0.056	0.660 \pm 0.012	0.868 \pm 0.009	0.822 \pm 0.012	0.600 \pm 0.019	0.710 \pm 0.049	0.770 \pm 0.053	0.783 \pm 0.031
	KIDD-LR	0.697 \pm 0.041	0.787 \pm 0.034	0.810 \pm 0.022		0.682 \pm 0.013	0.772 \pm 0.029	0.795 \pm 0.014	
	LAVA	0.620 \pm 0.000	0.620 \pm 0.000	0.620 \pm 0.000		0.777 \pm 0.019	0.795 \pm 0.007	0.800 \pm 0.007	
IMDB-MULTI	GRADATE	0.805 \pm 0.000	0.855 \pm 0.024	0.890 \pm 0.015	0.102 \pm 0.017	0.800 \pm 0.008	0.832 \pm 0.002	0.900 \pm 0.013	0.143 \pm 0.056
	Random	0.139 \pm 0.032	0.092 \pm 0.032	0.080 \pm 0.000		0.102 \pm 0.015	0.180 \pm 0.005	0.156 \pm 0.057	
	KIDD-LR	0.156 \pm 0.022	0.154 \pm 0.046	0.171 \pm 0.052		0.058 \pm 0.044	0.093 \pm 0.010	0.077 \pm 0.025	
MSRC_21	LAVA	0.183 \pm 0.000	0.183 \pm 0.000	0.183 \pm 0.000	0.860 \pm 0.007	0.190 \pm 0.009	0.177 \pm 0.019	0.193 \pm 0.025	0.883 \pm 0.015
	GRADATE	0.588 \pm 0.286	0.349 \pm 0.323	0.611 \pm 0.242		0.183 \pm 0.073	0.266 \pm 0.133	0.361 \pm 0.162	
	Random	0.576 \pm 0.029	0.702 \pm 0.045	0.830 \pm 0.004		0.427 \pm 0.035	0.801 \pm 0.046	0.857 \pm 0.011	
ogbg-molbace	KIDD-LR	0.702 \pm 0.007	0.766 \pm 0.015	0.848 \pm 0.004	0.617 \pm 0.073	0.763 \pm 0.025	0.792 \pm 0.017	0.863 \pm 0.015	0.560 \pm 0.063
	LAVA	0.623 \pm 0.007	0.819 \pm 0.012	0.895 \pm 0.012		0.667 \pm 0.012	0.851 \pm 0.007	0.933 \pm 0.004	
	GRADATE	0.719 \pm 0.007	0.797 \pm 0.008	0.906 \pm 0.004		0.787 \pm 0.046	0.860 \pm 0.007	0.942 \pm 0.008	
ogbg-molbbbp	Random	0.551 \pm 0.100	0.375 \pm 0.012	0.581 \pm 0.039	0.478 \pm 0.069	0.637 \pm 0.012	0.621 \pm 0.027	0.537 \pm 0.085	0.671 \pm 0.034
	KIDD-LR	0.592 \pm 0.054	0.484 \pm 0.020	0.592 \pm 0.008		0.613 \pm 0.090	0.456 \pm 0.041	0.589 \pm 0.035	
	LAVA	0.627 \pm 0.033	0.637 \pm 0.030	0.637 \pm 0.014		0.602 \pm 0.028	0.633 \pm 0.048	0.672 \pm 0.028	
ogbg-molhiv	GRADATE	0.655 \pm 0.046	0.578 \pm 0.035	0.614 \pm 0.042	0.625 \pm 0.001	0.642 \pm 0.083	0.660 \pm 0.026	0.684 \pm 0.026	0.596 \pm 0.015
	Random	0.567 \pm 0.037	0.488 \pm 0.088	0.478 \pm 0.021		0.534 \pm 0.084	0.648 \pm 0.045	0.623 \pm 0.019	
	KIDD-LR	0.428 \pm 0.025	0.477 \pm 0.080	0.457 \pm 0.013		0.424 \pm 0.005	0.450 \pm 0.070	0.464 \pm 0.052	
ogbg-molhiv	LAVA	0.596 \pm 0.058	0.566 \pm 0.021	0.547 \pm 0.044		0.619 \pm 0.044	0.642 \pm 0.120	0.747 \pm 0.024	
	GRADATE	0.604 \pm 0.065	0.601 \pm 0.047	0.557 \pm 0.001		0.657 \pm 0.039	0.677 \pm 0.072	0.715 \pm 0.015	
	Random	0.603 \pm 0.005	0.615 \pm 0.004	0.621 \pm 0.001		0.608 \pm 0.015	0.609 \pm 0.030	0.593 \pm 0.012	
ogbg-molhiv	KIDD-LR	0.590 \pm 0.005	0.608 \pm 0.001	0.595 \pm 0.011		0.597 \pm 0.042	0.595 \pm 0.039	0.608 \pm 0.020	0.596 \pm 0.015
	LAVA	0.531 \pm 0.035	0.594 \pm 0.013	0.601 \pm 0.013		0.614 \pm 0.002	0.638 \pm 0.020	0.641 \pm 0.010	
	GRADATE	0.607 \pm 0.018	0.599 \pm 0.012	0.622 \pm 0.004		0.640 \pm 0.013	0.651 \pm 0.022	0.658 \pm 0.018	

Table 1: Performance comparison across data selection methods for *graph density* shift. We use **bold**/underline to indicate the 1st/2nd best results. In most settings, GRADATE achieves the best performance across datasets.

can already achieve or exceed GNN performance trained with full data, showing the importance of data quality in the source domain. The main reason is that because under distribution shift, a certain number of harmful graphs exist in training set. These samples can mislead the model and degrade generalization performance.

4.3 GRADATE as a GDA Method

We answer (**RQ2**) by directly compare the combination of GRADATE and vanilla GNNs (including non-domain-adapted GCN, GIN, GAT and GraphSAGE) with state-of-the-art GDA models. We fix the sparsity ratio τ to 20% across all selection methods.

Experiment Setup. The four GDA methods we consider include AdaGCN [12], GRADE [61], ASN [83] and UDAGCN [62]. We conduct GDA experiments based on the codebase of OpenGDA [53]. We include more details of model-specific parameter settings in Appendix J. We set the training set as the source domain and the validation set as the target domain. For GDD computation, we set label signal $c = 0$ to match the requirement of unsupervised GDA methods. Results on *graph size* shift can be found in Appendix G.4.

Results. Results are in Table 2. For model-centric GDA methods trained with full training data, the severe domain shift prohibits these methods from learning rich knowledge to perform well on the test data in the target domain. Instead, GRADATE finds the most useful data in the training set that results in simple GNN models with extraordinary classification accuracy while maintaining data efficiency. In addition, similar to the observation in Section 4.2, GRADATE selects non-trivial training data that outperforms other model-free data selection methods. In Appendix

4.4 GRADATE as a Model-Free GDA Enhancer

In order to answer (**RQ3**), we combine GRADATE with off-the-shelf GDA methods to study whether fewer but better training data can lead to even stronger adaptation performance.

Experiment Setup. Coupled with 10%, 20%, 50% data selected by each model-free method (i.e., random, LAVA and GRADATE), four GDA baselines (considered in Section 4.3) are directly run on the shrunk training dataset with the same validation dataset under *graph density* shift. For results on *graph size* shift, we refer the readers to Appendix G.5.

Type	Model	Data	Dataset					
			IMDB-BINARY	IMDB-MULTI	MSRC_21	ogbg-molbace	ogbg-molbbbp	ogbg-molhiv
GDA	AdaGCN	Full	0.808 \pm 0.015	0.073 \pm 0.000	0.319 \pm 0.032	0.607 \pm 0.068	0.778 \pm 0.002	0.428 \pm 0.011
	GRADE	Full	0.822 \pm 0.012	0.123 \pm 0.061	0.804 \pm 0.011	0.683 \pm 0.016	0.489 \pm 0.005	0.564 \pm 0.005
	ASN	Full	0.782 \pm 0.030	0.119 \pm 0.047	0.833 \pm 0.033	0.580 \pm 0.065	0.476 \pm 0.027	0.516 \pm 0.021
	UDAGCN	Full	0.807 \pm 0.013	0.114 \pm 0.049	0.351 \pm 0.019	0.541 \pm 0.034	0.522 \pm 0.015	0.451 \pm 0.030
GCN	GCN	Random 20%	0.660 \pm 0.012	0.092 \pm 0.032	0.702 \pm 0.045	0.529 \pm 0.124	0.528 \pm 0.030	0.598 \pm 0.003
		LAVA 20%	0.620 \pm 0.000	0.092 \pm 0.032	0.819 \pm 0.011	0.541 \pm 0.067	0.503 \pm 0.043	0.591 \pm 0.030
		GRADATE 20%	0.830 \pm 0.021	0.349 \pm 0.323	0.797 \pm 0.008	0.585 \pm 0.074	0.571 \pm 0.035	0.583 \pm 0.006
GIN	GIN	Random 20%	0.710 \pm 0.049	0.180 \pm 0.005	0.801 \pm 0.046	0.622 \pm 0.028	0.480 \pm 0.041	0.590 \pm 0.033
		LAVA 20%	0.778 \pm 0.045	0.170 \pm 0.009	0.851 \pm 0.012	0.655 \pm 0.067	0.644 \pm 0.021	0.638 \pm 0.012
		GRADATE 20%	0.832 \pm 0.025	0.266 \pm 0.133	0.860 \pm 0.007	0.662 \pm 0.006	0.665 \pm 0.053	0.644 \pm 0.017
Vanilla	Vanilla	Random 20%	0.662 \pm 0.029	0.067 \pm 0.005	0.713 \pm 0.008	0.472 \pm 0.034	0.486 \pm 0.041	0.593 \pm 0.012
		LAVA 20%	0.835 \pm 0.002	0.790 \pm 0.002	0.842 \pm 0.026	0.515 \pm 0.019	0.511 \pm 0.069	0.602 \pm 0.017
		GRADATE 20%	0.858 \pm 0.005	0.800 \pm 0.133	0.857 \pm 0.008	0.518 \pm 0.026	0.538 \pm 0.098	0.598 \pm 0.004
GAT	GAT	Random 20%	0.738 \pm 0.059	0.132 \pm 0.036	0.731 \pm 0.027	0.459 \pm 0.057	0.472 \pm 0.016	0.602 \pm 0.006
		LAVA 20%	0.835 \pm 0.005	0.570 \pm 0.292	0.827 \pm 0.015	0.514 \pm 0.132	0.491 \pm 0.095	0.537 \pm 0.067
		GRADATE 20%	0.855 \pm 0.005	0.580 \pm 0.281	0.842 \pm 0.007	0.536 \pm 0.062	0.533 \pm 0.037	0.541 \pm 0.014

Table 2: Performance comparison across GDA and vanilla methods for *graph density* shift. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE can consistently achieve top-2 performance across all datasets.

Results. As shown in Table 3, for most of the settings, GRADATE selects data that is the most beneficial to adapting to the target set. Notably, across many settings, only 10% or 20% GRADATE-selected data can outperform naively applying GDA methods on the full training data. This suggests that GRADATE can indeed improve *data-efficiency* by promoting the quality of training data. Furthermore, by effective data selection performed by GRADATE, the difficulty of addressing the domain shift can be lowered significantly and thus result in better adaptation performance.

4.5 Further Discussion

LAVA vs GRADATE. The modified version of LAVA utilizes LinearFGW to compare graphs and selects the training data with the smallest gradient value w.r.t. GDD. In contrast, GRADATE aims at finding optimal training data that directly minimizes GDD, which has a complete different motivation and enjoys a theoretical justification. Empirically, we also observe the superiority of GRADATE in most cases. Occasionally, LAVA achieve marginally better results than GRADATE, which may be attributed to the approximation error of LinearFGW and thus over-optimization on GDD.

Random vs GRADATE. From Table 2, we found GRADATE occasionally underperforms random selection with GraphSAGE, possibly because the neighbor sampling strategy introduces noise into global representations, weakening the supervision signal even for well-chosen training graphs.

Selection Ratio vs GNN Performance. From Tables 1 & 3, we find that a larger selection ratio may not always guarantee a better performance for selection methods including LAVA and GRADATE. This is because, under severe distribution shift between domains, a larger portion of training data may actually contain patterns that are irrelevant or even harmful to the target domain.

Effect of label signal c . While we treat c as a tunable hyper-parameter that can be optimized for different settings (i.e. various combinations of dataset, shift types and selection ratio), we empirically find that searching within $\{0, 5\}$ can already lead to good performance throughout all experiments in this paper. We also provide additional experiments under a label-free setting (i.e. c is forced to be equal to 0) in Appendix G.6 .

5 Related Work

Data Selection. Recent advancements in data selection have focused on optimizing data utilization, mainly on text and vision data to facilitate efficient training for large language/image models [29, 28, 43, 3, 63, 13, 70, 35]. For general model-free data selection, LAVA [26] offers a learning-agnostic data valuation method by seeking the data point that contributes the most to the distance between training and validation datasets. However, the paper studies predominantly on raw image datasets such

Dataset	GDA Method →				GRADE				
	Selection Method ↓		AdaGCN		τ = 10%	τ = 20%	τ = 50%	Full	
IMDB-BINARY	Random	0.763±0.040	0.773±0.019	0.798±0.002		0.683±0.010	0.792±0.002	0.780±0.015	
	LAVA	0.623±0.005	0.617±0.005	0.617±0.005	0.808±0.015	0.620±0.073	0.627±0.009	0.680±0.047	
	GRADATE	0.810 ±0.032	0.817 ±0.024	0.822 ±0.017		0.782 ±0.009	0.832 ±0.013	0.848 ±0.009	
IMDB-MULTI	Random	0.100±0.000	0.168±0.072	0.116±0.048		0.106±0.055	0.112±0.050	0.149±0.049	
	LAVA	0.191±0.007	0.183±0.000	0.184±0.002	0.073±0.000	0.183 ±0.000	<u>0.189</u> ±0.008	0.186 ±0.003	
	GRADATE	0.333 ±0.229	0.373 ±0.285	0.391 ±0.294		0.131 ±0.074	0.386 ±0.286	0.173 ±0.100	
MSRC_21	Random	0.208±0.027	0.374±0.011	0.307±0.087		0.512±0.041	0.626±0.055	0.708±0.023	
	LAVA	0.398±0.004	0.456 ±0.012	<u>0.480</u> ±0.061	0.319±0.032	0.608±0.018	0.743±0.021	0.860±0.014	
	GRADATE	0.415 ±0.112	0.406±0.043	0.532 ±0.039		0.664 ±0.021	0.778 ±0.015	0.865 ±0.027	
ogbg-molbace	Random	0.436±0.021	0.485±0.038	0.565±0.085		0.538±0.023	0.554±0.025	0.611±0.015	
	LAVA	0.574±0.017	0.589±0.074	0.607 ±0.071	0.607±0.068	0.557±0.055	0.653 ±0.054	<u>0.625</u> ±0.015	
	GRADATE	0.598 ±0.066	0.614 ±0.043	0.572 ±0.047		0.599 ±0.044	0.636 ±0.035	0.634 ±0.006	
ogbg-molbbbp	Random	0.494±0.014	0.469±0.031	0.527±0.035		0.511±0.032	0.433±0.001	0.495±0.041	
	LAVA	0.583±0.075	0.556±0.015	0.561 ±0.040	0.778±0.002	0.549±0.013	0.579 ±0.041	0.543 ±0.013	
	GRADATE	0.593 ±0.038	0.596 ±0.022	0.546±0.026		0.582 ±0.077	0.503 ±0.012	0.490±0.006	
ogbg-molhiv	Random	0.407±0.022	0.429±0.032	0.417±0.013		0.581±0.008	0.544±0.001	0.581±0.009	
	LAVA	0.453±0.016	0.428±0.013	0.440 ±0.003	0.428±0.011	0.566±0.011	0.571 ±0.005	0.572±0.019	
	GRADATE	0.463 ±0.041	0.473 ±0.021	0.447 ±0.038		0.584 ±0.012	0.589 ±0.003	0.586 ±0.003	
Dataset	GDA Method →				ASN				
	Selection Method ↓	τ = 10%	τ = 20%	τ = 50%	Full	τ = 10%	τ = 20%	τ = 50%	Full
IMDB-BINARY	Random	0.660±0.043	0.707±0.017	0.678±0.031		0.620±0.041	0.763 ±0.008	<u>0.823</u> ±0.005	
	LAVA	0.733±0.081	0.620±0.000	0.620±0.000	0.782±0.030	0.620 ±0.000	0.643±0.033	0.620±0.000	0.807±0.013
	GRADATE	0.748 ±0.037	0.818 ±0.016	0.855 ±0.011		0.770 ±0.023	0.847 ±0.012	0.852 ±0.005	
IMDB-MULTI	Random	0.126±0.013	0.101±0.058	0.156±0.039		0.150±0.024	0.101±0.045	0.076±0.003	
	LAVA	0.183±0.000	0.183±0.000	0.190±0.009	0.119±0.047	0.183 ±0.000	0.183 ±0.000	0.182 ±0.002	0.114±0.049
	GRADATE	0.292 ±0.352	0.588 ±0.286	0.381 ±0.301		0.093±0.066	0.554 ±0.263	0.339 ±0.337	
MSRC_21	Random	0.421±0.026	0.673±0.011	0.661±0.032		0.287±0.018	0.178±0.039	0.287±0.075	
	LAVA	0.635±0.015	0.746±0.019	0.868±0.014	0.833±0.033	0.453 ±0.035	0.447 ±0.052	0.623 ±0.059	0.351±0.019
	GRADATE	0.687 ±0.048	0.804 ±0.021	0.904 ±0.012		0.444 ±0.048	0.453 ±0.011	0.664 ±0.029	
ogbg-molbace	Random	0.539±0.074	0.637 ±0.009	0.507±0.061		0.478±0.037	0.581 ±0.018	<u>0.513</u> ±0.028	
	LAVA	0.578±0.036	0.603±0.009	0.646 ±0.050	0.580±0.065	0.562 ±0.039	0.578 ±0.015	0.513±0.077	0.541±0.034
	GRADATE	0.636 ±0.022	0.596±0.053	0.651 ±0.036		0.533 ±0.041	0.565±0.039	0.531 ±0.051	
ogbg-molbbbp	Random	0.504±0.015	0.533±0.025	0.497±0.032		0.538±0.026	0.529±0.040	0.530±0.051	
	LAVA	0.567±0.040	0.616 ±0.073	0.573 ±0.035	0.476±0.027	0.579±0.031	0.547 ±0.021	0.558 ±0.021	0.522±0.015
	GRADATE	0.573 ±0.088	0.596±0.100	0.535 ±0.027		0.591 ±0.040	0.575 ±0.030	0.570 ±0.009	
ogbg-molhiv	Random	0.436±0.038	0.483±0.044	0.455±0.059		0.453±0.015	0.406±0.015	0.464 ±0.024	
	LAVA	0.511±0.018	0.540 ±0.010	0.482 ±0.023	0.516±0.021	0.458±0.029	0.427±0.007	0.445 ±0.018	0.451±0.030
	GRADATE	0.527 ±0.041	0.491±0.080	0.491 ±0.050		0.453 ±0.011	0.445 ±0.018	0.444±0.020	

Table 3: Performance comparison across combinations of GDA methods and data selection methods for *graph density shift*. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE achieves the best performance in most settings.

as CIFAR-10 [31] and MNIST [32], where they already have high-quality pixel-value representations for computation. Unlike text or images, graphs lack a natural and uniform representation, making the development of model-free data selection more intricate. Tailored for graph-level tasks, graph dataset distillation is also a related topic. For example, Jin et al. [25] and Xu et al. [66] both propose to formulate a bi-level optimization problem to train a graph-level classifier. Jain et al. [24], on the other hand, utilizes Tree Mover Distance [11] to conduct graph-level sub-sampling with theoretical guarantees. However, these non-model-free methods might not be able to combat severe downstream distribution changes.

Graph Domain Adaptation (GDA). For grpah classification, GDA focuses on transferring knowledge from a source domain with labeled graph to a target domain. Model-centric GDA methods relying on GNNs have been pivotal in this area. For instance, Wu et al. [62] introduce UDAGCN, which integrates domain adaptation with GNNs to align feature distributions between domains. AdaGCN [12] addresses cross-network node classification leveraging adversarial domain adaptation to transfer label information between domains. Wu et al. [61] explore cross-network transfer learning through Weisfeiler-Lehman graph isomorphism test and introduce the GRADE algorithm that minimizes distribution shift to perform adaptation. ASN [83] explicitly separates domain-private and domain-shared information while capturing network consistency. More recently, Liu et al. [34] argue that excessive message passing exacerbates domain bias and propose A2GNN as a refined propagation scheme that disentangles transferable and domain-specific information; Chen et al. [7]

highlight the critical role of graph smoothness, presenting TDSS that enforces cross-domain consistency through spectral alignment. Meanwhile, Liu et al. [36] introduce a pairwise alignment strategy that leverages node-level relational matching to enhance inter-domain correspondence. However, these approaches mostly focus on designing architectures or training procedures and often rely heavily on the assumption that provided data in the training set is already optimal for the task, which is often invalid in real-world scenarios.

6 Conclusion

We introduce GRADATE, a model-free framework for graph classification that addresses distribution shift by solving a Graph Dataset Distance (GDD) minimization problem. By selecting the most beneficial data from the source domain, it offers a novel approach to improving GNN performance without relying on specific model predictions or training procedures. We also establish theoretical analysis on Fused Gromov–Wasserstein distance as a meaningful upper bound on GNN representation differences, and further justifies GDD as an optimization target to improve generalization performance. Across multiple real-world datasets and shift types, GRADATE consistently outperforms existing selection methods and GDA methods with better data efficiency. For future directions, we consider graph continual learning and multi-source domain adaptation.

Acknowledgement

This work is supported by NSF (2416070) and AFOSR (FA9550-24-1-0002). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

- [1] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems*, 30, 2017.
- [2] David Alvarez-Melis and Nicolo Fusi. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33:21428–21439, 2020.
- [3] Tianyi Bai, Ling Yang, Zhen Hao Wong, Jiahui Peng, Xinlin Zhuang, Chi Zhang, Lijun Wu, Jiantao Qiu, Wentao Zhang, Binhang Yuan, et al. Multi-agent collaborative data selection for efficient llm pretraining. *arXiv preprint arXiv:2410.08102*, 2024.
- [4] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.
- [5] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *International Conference on Machine Learning*, pages 837–851. PMLR, 2021.
- [6] Pietro Bongini, Niccolò Pancino, Franco Scarselli, and Monica Bianchini. Biognn: how graph neural networks can solve biological problems. In *Artificial Intelligence and Machine Learning for Healthcare: Vol. 1: Image and Data Analytics*, pages 211–231. Springer, 2022.
- [7] Wei Chen, Guo Ye, Yakun Wang, Zhao Zhang, Libang Zhang, Daixin Wang, Zhiqiang Zhang, and Fuzhen Zhuang. Smoothness really matters: A simple yet effective approach for unsupervised graph domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 15875–15883, 2025.
- [8] Zhiyong Cheng, Sai Han, Fan Liu, Lei Zhu, Zan Gao, and Yuxin Peng. Multi-behavior recommendation with cascading graph convolution networks. In *Proceedings of the ACM Web Conference 2023*, pages 1181–1189, 2023.
- [9] Nikzad Chizari, Keywan Tajfar, and María N Moreno-García. Bias assessment approaches for addressing user-centered fairness in gnn-based recommender systems. *Information*, 14(2):131, 2023.
- [10] Xu Chu, Yujie Jin, Xin Wang, Shanghang Zhang, Yasha Wang, Wenwu Zhu, and Hong Mei. Wasserstein barycenter matching for graph size generalization of message passing neural networks. In *International Conference on Machine Learning*, pages 6158–6184. PMLR, 2023.
- [11] Ching-Yao Chuang and Stefanie Jegelka. Tree mover’s distance: Bridging graph metrics and stability of graph neural networks. *Advances in Neural Information Processing Systems*, 35:2944–2957, 2022.
- [12] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4908–4922, 2022.
- [13] Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. *arXiv preprint arXiv:2310.15393*, 2023.
- [14] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.
- [15] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. Graph neural networks for recommender system. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 1623–1625, 2022.
- [16] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=H1gL-2A9Ym>.

- [17] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- [18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.
- [20] Xiaobin Hong, Wenzhong Li, Chaoqun Wang, Mingkai Lin, and Sanglu Lu. Label attentive distillation for gnn-based graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
- [21] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.
- [22] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.
- [23] Kexin Huang, Cao Xiao, Lucas M Glass, Marinka Zitnik, and Jimeng Sun. Skipgnn: predicting molecular interactions with skip-graph networks. *Scientific reports*, 10(1):21092, 2020.
- [24] Mika Sarkin Jain, Stefanie Jegelka, Ishani Karmarkar, Luana Ruiz, and Ellen Vitercik. Subsampling graphs with gnn performance guarantees. *arXiv preprint arXiv:2502.16703*, 2025.
- [25] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 720–730, 2022.
- [26] Hoang Anh Just, Feiyang Kang, Jiachen T Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. Lava: Data valuation without pre-specified learning algorithms. *arXiv preprint arXiv:2305.00054*, 2023.
- [27] Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.
- [28] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.
- [29] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8110–8118, 2021.
- [30] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [31] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report.
- [32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [33] Ting Wei Li, Qiaozhu Mei, and Jiaqi Ma. A metadata-driven approach to understand graph neural networks. *Advances in Neural Information Processing Systems*, 36:15320–15340, 2023.
- [34] Meihan Liu, Zeyu Fang, Zhen Zhang, Ming Gu, Sheng Zhou, Xin Wang, and Jiajun Bu. Rethinking propagation for unsupervised graph domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

- [35] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv preprint arXiv:2407.01492*, 2024.
- [36] Shikun Liu, Deyu Zou, Han Zhao, and Pan Li. Pairwise alignment improves graph domain adaptation. In *International Conference on Machine Learning*, pages 32552–32575. PMLR, 2024.
- [37] Tianyu Liu, Yuge Wang, Rex Ying, and Hongyu Zhao. Muse-gnn: Learning unified gene representation from multimodal biological graph data. *Advances in neural information processing systems*, 36:24661–24677, 2023.
- [38] Xingyu Liu, Juan Chen, and Quan Wen. A survey on graph classification and link prediction based on gnn. *arXiv preprint arXiv:2307.00865*, 2023.
- [39] Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. Topological augmentation for class-imbalanced node classification, 2023.
- [40] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. Class-imbalanced graph learning without class rebalancing. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.
- [41] Junyu Luo, Zhiping Xiao, Yifan Wang, Xiao Luo, Jingyang Yuan, Wei Ju, Langechuan Liu, and Ming Zhang. Rank and align: towards effective source-free graph domain adaptation. *arXiv preprint arXiv:2408.12185*, 2024.
- [42] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11:417–487, 2011.
- [43] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.
- [44] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.
- [45] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245, 2016.
- [46] Dai Hai Nguyen and Koji Tsuda. On a linear fused gromov-wasserstein distance for graph structured data. *Pattern Recognition*, 138:109351, 2023.
- [47] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 35, pages 25531–25546, 2022.
- [48] Ruizhong Qiu, Gaotang Li, Tianxin Wei, Jingrui He, and Hanghang Tong. Saffron-1: Safety inference scaling, 2025.
- [49] Ruizhong Qiu, Ting-Wei Li, Gaotang Li, and Hanghang Tong. Graph homophily booster: Rethinking the role of discrete features on heterophilic graphs. *arXiv preprint arXiv:2509.12530*, 2025.
- [50] Ruizhong Qiu, Zhe Xu, Wenzuan Bao, and Hanghang Tong. Ask, and it shall be given: On the Turing completeness of prompting. In *13th International Conference on Learning Representations*, 2025.
- [51] Ruizhong Qiu, Weiliang Will Zeng, Hanghang Tong, James Ezick, and Christopher Lott. How efficient is LLM-generated code? A rigorous & high-standard benchmark. In *13th International Conference on Learning Representations*, 2025.

- [52] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision*, 40:99–121, 2000.
- [53] Boshen Shi, Yongqing Wang, Fangda Guo, Jiangli Shao, Huawei Shen, and Xueqi Cheng. Opengda: Graph domain adaptation benchmark for cross-network learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 5396–5400, 2023.
- [54] Karl-Theodor Sturm. *The space of spaces: curvature bounds and gradient flows on the space of metric measure spaces*, volume 290. American Mathematical Society, 2023.
- [55] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. Adagcn: Adaboosting graph convolutional networks into deep models. *arXiv preprint arXiv:1908.05081*, 2019.
- [56] Yuhao Tang, Junyu Luo, Ling Yang, Xiao Luo, Wentao Zhang, and Bin Cui. Multi-view teacher with curriculum data fusion for robust unsupervised domain adaptation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 2598–2611. IEEE, 2024.
- [57] Titouan Vayer, Laetitia Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. Fused gromov-wasserstein distance for structured objects. *Algorithms*, 13(9):212, 2020.
- [58] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [59] Wei Wang, Dejan Slepčev, Saurav Basu, John A Ozolek, and Gustavo K Rohde. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision*, 101:254–269, 2013.
- [60] Fang Wu, Nicolas Courty, Shuteng Jin, and Stan Z Li. Improving molecular representation learning with metric learning-enhanced optimal transport. *Patterns*, 4(4), 2023.
- [61] Jun Wu, Jingrui He, and Elizabeth Ainsworth. Non-iid transfer learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10342–10350, 2023.
- [62] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of the web conference 2020*, pages 1457–1467, 2020.
- [63] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36, 2024.
- [64] Yu Xie, Yanfeng Liang, Maoguo Gong, A Kai Qin, Yew-Soon Ong, and Tiantian He. Semisupervised graph neural networks for graph classification. *IEEE Transactions on Cybernetics*, 53(10):6222–6235, 2022.
- [65] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [66] Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Kernel ridge regression-based graph dataset distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2850–2861, 2023.
- [67] Zhe Xu, Kaveh Hassani, Si Zhang, Hanqing Zeng, Michihiro Yasunaga, Limei Wang, Dongqi Fu, Ning Yao, Bo Long, and Hanghang Tong. Language models are graph learners, 2024.
- [68] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.
- [69] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

- [70] Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*, 2024.
- [71] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.
- [72] Nan Yin, Mengzhu Wang, Zhenghan Chen, Li Shen, Huan Xiong, Bin Gu, and Xiao Luo. Dream: Dual structured exploration with mixup for open-set graph domain adaption. In *The Twelfth International Conference on Learning Representations*, 2024.
- [73] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.
- [74] Hyunsik Yoo, Zhichen Zeng, Jian Kang, Ruizhong Qiu, David Zhou, Zhining Liu, Fei Wang, Charlie Xu, Eunice Chan, and Hanghang Tong. Ensuring user-side fairness in dynamic recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pages 3667–3678, 2024.
- [75] Hyunsik Yoo, SeongKu Kang, Ruizhong Qiu, Charlie Xu, Fei Wang, and Hanghang Tong. Embracing plasticity: Balancing stability and plasticity in continual recommender systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.
- [76] Hyunsik Yoo, Ruizhong Qiu, Charlie Xu, Fei Wang, and Hanghang Tong. Generalizable recommender system during temporal popularity distribution shifts. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.
- [77] Qi Yu, Zhichen Zeng, Yuchen Yan, Lei Ying, R Srikant, and Hanghang Tong. Joint optimal transport and embedding for network alignment. In *Proceedings of the ACM on Web Conference 2025*, pages 2064–2075, 2025.
- [78] Zhichen Zeng, Si Zhang, Yinglong Xia, and Hanghang Tong. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM web conference 2023*, pages 372–382, 2023.
- [79] Zhichen Zeng, Ruike Zhu, Yinglong Xia, Hanqing Zeng, and Hanghang Tong. Generative graph dictionary learning. In *International Conference on Machine Learning*, pages 40749–40769. PMLR, 2023.
- [80] Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16660–16668, 2024.
- [81] Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. Graph mixup on approximate gromov–wasserstein geodesics. In *Forty-first International Conference on Machine Learning*, 2024.
- [82] Zhichen Zeng, Ruizhong Qiu, Wenxuan Bao, Tianxin Wei, Xiao Lin, Yuchen Yan, Tarek F Abdelzaher, Jiawei Han, and Hanghang Tong. Pave your own path: Graph gradual domain adaptation on fused gromov-wasserstein geodesics. *arXiv preprint arXiv:2505.12709*, 2025.
- [83] Xiaowen Zhang, Yuntao Du, Rongbiao Xie, and Chongjun Wang. Adversarial separation network for cross-network node classification. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 2618–2626, 2021.
- [84] Yangze Zhou, Gitta Kutyniok, and Bruno Ribeiro. Ood link prediction generalization capabilities of message-passing gnns in larger test graphs. *Advances in Neural Information Processing Systems*, 35:20257–20272, 2022.
- [85] Marinka Zitnik, Michelle M Li, Aydin Wells, Kimberly Glass, Deisy Morselli Gysi, Arjun Krishnan, T_M Murali, Predrag Radivojac, Sushmita Roy, Anaïs Baudot, et al. Current and future directions in network biology, 2024.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: Our main contributions are summarized in five bullet points in Section 1 and we point the readers to corresponding sections to see justification (including theorems, methodologies and empirical results).

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We discuss potential limitations of our presented work in Appendix N.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide all proofs and assumptions of our theorem results in Appendix F.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all experimental details needed in Section 4, Appendix I and Appendix J.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide the code package during submission and make the code available upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not

including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide all experimental details needed in Section 4, Appendix I and Appendix J.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We include std in all of our main tables (see Section 4).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We specify the compute resources in Appendix I.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We confirm that this work is conducted with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We provide related discussion in Section O.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We confirm that this work does not pose safety risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring

that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We provide the related details in Appendix H.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: We did not introduce new assets in this paper.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLM is not an important part of the core methods in this paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

Appendix

The content of appendix is organized as follows:

1. Algorithms:

- Appendix A talks about the details of LinearFGW [46] that we omit in the main text. We summarize the overall procedure of LinearFGW in Algorithm 1.
- Appendix B goes through the steps to compute Graph Dataset Distance (GDD). The entire procedure is included in Algorithm 2.
- Appendix C summarizes the submodule GREAT used in our main algorithm (Algorithm 3).
- Appendix D summarizes our main algorithm GRADATE (Algorithm 4).

2. Proofs:

- Appendix F provides the proofs for all the theorems in the main text.

3. Additional Experiments:

- Appendix G.1 compares GRADATE with other data selection methods under *graph size* shift with additional GNN backbones.
- Appendix G.2 compares GRADATE with other data selection methods under *graph density* shift.
- Appendix G.3 compares GRADATE with other data selection methods under *graph size* shift with additional GNN backbones.
- Appendix G.4 compares the combination of GRADATE and vanilla GNNs with other GDA methods under *graph size* shift.
- Appendix G.5 compares the combination of GDA methods and GRADATE against other data selection methods under *graph size* shift.
- Appendix G.6 ablates on the validation-label-free setting.
- Appendix G.7 includes results on additional graph backbones.
- Appendix G.8 includes results on additional GDA methods.

4. Discussions:

- Appendix E discusses FGW and Graph Dataset Distance (GDD) in relation to prior notions such as Tree-Mover Distance (TMD) [11] and Maximum Mean Discrepancy (MMD) [17].
- Appendix N discusses potential limitations and future direction of our work.

5. Reproducibility:

- Appendix H provides the dataset statistics and licenses used in this work.
- Appendix I introduces the overall settings of GNN to use for the graph data selection evaluation, including the model we select and the training protocols.
- Appendix J includes GDA method-specific parameter settings, where we follow the default settings of the OpenGDA package [53].

6. Others:

- Appendix K includes problem definition of Graph Domain Adaptation (GDA).
- Appendix L provides additional related work.
- Appendix M contains the empirical runtime of GRADATE.
- Appendix P includes the ECDF plots of graph properties across datasets.

A Details of LinearFGW (Algorithm 1)

Formally, consider a set of N graphs $\mathcal{D} = \{\mathcal{G}_i\}_{i=1}^N$, where each $\mathcal{G} = (\mathbf{A}, \mathbf{X}) \in \mathcal{D}$ represents an attributed graph with adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Note that n is the number of nodes of \mathcal{G} and d is the dimension of node features. LinearFGW first requires a reference graph $\bar{\mathcal{G}} = (\bar{\mathbf{A}}, \bar{\mathbf{X}})$ where $\bar{\mathbf{A}} \in \mathbb{R}^{\bar{n} \times \bar{n}}$, $\bar{\mathbf{X}} \in \mathbb{R}^{\bar{n} \times d}$, \bar{n} is the number of nodes and d is the dimension of node features. Typically, $\bar{\mathcal{G}}$ is obtained by solving an *FGW barycenter problem*, which aims to find a “center” graph that has the minimum sum of pairwise graph distances over the entire graph set \mathcal{D} .

Following the notation used in Section 2.2, we define the inter-graph distance matrix $\mathbf{M}_{(\mathcal{G}_1, \mathcal{G}_2)}$ between any pair of graphs (named as $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{X}_1, \mathbf{p}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{X}_2, \mathbf{p}_2\}$) to be the pairwise Euclidean distance of node features. Namely, $\mathbf{M}_{(\mathcal{G}_1, \mathcal{G}_2)} = [\|\mathbf{X}_1[i] - \mathbf{X}_2[j]\|]_{ij}$. In addition, the intra-graph similarity matrix is chosen to be defined as their corresponding adjacency matrices

(i.e., $\mathbf{C}_{\mathcal{G}_1} = \mathbf{A}_1$ and $\mathbf{C}_{\mathcal{G}_2} = \mathbf{A}_2$). Together with uniform distributions⁶ $\mathbf{p}_{\mathcal{G}_1} = \frac{\mathbf{1}_{n_1}}{n_1}$ and $\mathbf{p}_{\mathcal{G}_2} = \frac{\mathbf{1}_{n_2}}{n_2}$ over the nodes of \mathcal{G}_1 and \mathcal{G}_2 (with sizes n_1 and n_2), correspondingly, the *FGW barycenter problem*⁷ can be formulated as follows:

$$\bar{\mathcal{G}} = \arg \min_{\mathcal{G}} \sum_{i=1}^N \text{FGW}(\mathbf{M}_{(\mathcal{G}_i, \mathcal{G})}, \mathbf{C}_{\mathcal{G}_i}, \mathbf{C}_{\mathcal{G}}, \mathbf{p}_{\mathcal{G}_i}, \mathbf{p}_{\mathcal{G}}, \alpha), \quad (6)$$

where $\alpha \in [0, 1]$ is the pre-defined trade-off parameter.

After calculating the reference graph $\bar{\mathcal{G}}$, we then obtain N optimal transport plans $\{\pi_i\}_{i \in [n]}$ as the solutions by computing $\text{FGW}(\mathcal{G}, \bar{\mathcal{G}})$ for each $\mathcal{G} \in \mathcal{D}$ (via solving Equation (1)). Then, the barycentric projection [46] of each graph's node edge with respect to $\bar{\mathcal{G}}$ can be written as

$$\mathbf{T}_{\text{node}}(\boldsymbol{\pi}_i) = \bar{n} \cdot \boldsymbol{\pi}_i \mathbf{X}_i \in \mathbb{R}^{\bar{n} \times d}, \quad (7)$$

$$\mathbf{T}_{\text{edge}}(\boldsymbol{\pi}_i) = \bar{n}^2 \cdot \boldsymbol{\pi}_i \mathbf{C}_i \boldsymbol{\pi}_i^\top \in \mathbb{R}^{\bar{n} \times \bar{n}}. \quad (8)$$

Finally, we can define the LinearFGW distance based on these barycentric projections. Namely, for any pair of graphs $(\mathcal{G}_i, \mathcal{G}_j)$, we define a distance metric $d_{\text{LinearFGW}}(\cdot, \cdot)$ over the graph set \mathcal{D} :

$$\begin{aligned} d_{\text{LinearFGW}}(\mathcal{G}_i, \mathcal{G}_j) = & \\ & (1 - \alpha) \|\mathbf{T}_{\text{node}}(\boldsymbol{\pi}_i) - \mathbf{T}_{\text{node}}(\boldsymbol{\pi}_j)\|_F^2 \\ & + \alpha \|\mathbf{T}_{\text{edge}}(\boldsymbol{\pi}_i) - \mathbf{T}_{\text{edge}}(\boldsymbol{\pi}_j)\|_F^2, \end{aligned} \quad (9)$$

for $i, j \in [N]$. Note that $\|\cdot\|_F$ represents the Frobenius norm.

Algorithm 1 LinearFGW [46]

```

1: Input:  $N$  graphs  $\mathcal{D} = \{\mathcal{G}_i\}_{i=1}^N$ , trade-off parameter  $\alpha$ .
2: Initialize pairwise distance matrix  $\mathbf{D} \in \mathbb{R}^{N \times N}$ 
3: Solve the FGW barycenter problem in Equation (6) and obtain the reference graph  $\bar{\mathcal{G}}$ ;
4: for graph  $\mathcal{G}_i$  in  $\mathcal{D}$  do
5:   Compute  $\text{FGW}(\mathcal{G}_i, \bar{\mathcal{G}})$  via solving Equation (1) and obtain  $\boldsymbol{\pi}_i$ ;
6:   Compute  $\mathbf{T}_{\text{node}}(\boldsymbol{\pi}_i)$  and  $\mathbf{T}_{\text{edge}}(\boldsymbol{\pi}_i)$  via Equation (7)(8);
7: end for
8: for  $\mathcal{G}_i$  in  $\mathcal{D}$  do
9:   for  $\mathcal{G}_j$  in  $\mathcal{D}$  do
10:    Compute  $\mathbf{D}[i, j] = d_{\text{LinearFGW}}(\mathcal{G}_i, \mathcal{G}_j)$  via Equation (9);
11:   end for
12: end for
13: return LinearFGW pairwise distance matrix  $\mathbf{D}$ .
```

B Summarization of GDD (Algorithm 2)

⁶Since we have no prior over the node importance in either graphs, the probability simplex will typically be set as uniform.

⁷The optimization algorithm for solving this problem is omitted. Please refer to the original paper for more details.

Algorithm 2 (Training-Validation) GDD Computation

- 1: **Input:** labeled training graphs $\mathcal{D}^{\text{train}} = \{\mathcal{G}_i^{\text{train}}, y_i^{\text{train}}\}_{i=1}^n$, labeled validation graphs $\mathcal{D}^{\text{val}} = \{\mathcal{G}_i^{\text{val}}, y_i^{\text{val}}\}_{i=1}^m$, trade-off parameter α , label signal strength $c \geq 0$, a shared label set \mathcal{Y} .
- 2: Compute pairwise LinearFGW distance matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ via Algorithm 1 with the graph set $\mathcal{D} = \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}}$ and parameter α ;
- 3: Initialize new pairwise distance matrix $\tilde{\mathbf{D}} = \mathbf{D}$;
- 4: Initialize uniform empirical measures:
 $\mathbf{p}^{\text{train}} = \frac{1}{n} \sum_{i \in [n]} \delta_{(\mathcal{G}_i^{\text{train}}, y_i^{\text{train}})}$, $\mathbf{q}^{\text{val}} = \frac{1}{m} \sum_{j \in [m]} \delta_{(\mathcal{G}_j^{\text{val}}, y_j^{\text{val}})}$;
- 5: **for** training label ℓ_t **in** \mathcal{Y} **do**
- 6: **for** validation label ℓ_v **in** \mathcal{Y} **do**
- 7: Collect training index set with label ℓ_t :
 $\mathbf{I}_{\ell_t} = \{i | y_i^{\text{train}} = \ell_t\}$;
- 8: Collect validation data set with label ℓ_v :
 $\mathbf{I}_{\ell_v} = \{j | y_j^{\text{val}} = \ell_v\}$;
- 9: Compute *graph-label distance* in Equation (2):
 $d(\ell_t, \ell_v) = \text{OT}(\mathbf{p}_{\ell_t}^{\text{train}}, \mathbf{q}_{\ell_v}^{\text{val}}, d_{\text{LinearFGW}})$;
- 10: Update distance sub-matrix $\tilde{\mathbf{D}}[i \in \mathbf{I}_{\ell_t}, j \in \mathbf{I}_{\ell_v}] = \mathbf{D}[i \in \mathbf{I}_{\ell_t}, j \in \mathbf{I}_{\ell_v}] + c \cdot d(\ell_t, \ell_v)$;
- 11: **end for**
- 12: **end for**
- 13: Compute $\text{OTDD}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) = \text{OT}(\mathbf{p}^{\text{train}}, \mathbf{q}^{\text{val}}, \tilde{\mathbf{D}})$ via the equation in Section 2.1.
- 14: **return** $\text{GDD}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) = \text{OTDD}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}})$.

C Summarization of GREAT (Algorithm 3)

Starting from a uniform training weight \mathbf{w} , GREAT alternates between two subroutines: (i) computes GDD between the two sets using pairwise distances $\tilde{\mathbf{D}} \in \mathbb{R}^{n \times m}$ as the cost matrix (Line 4) and obtains the gradient $\mathbf{g}_w = \nabla_w \text{GDD}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, \tilde{\mathbf{D}})$ for updating \mathbf{w} (Line 5) and (ii) gradually sparsifies \mathbf{w} by retaining only the top- k entries followed by normalization to ensure \mathbf{w} is on the probability simplex (Line 6-9). After T iterations, we extract the non-zero entries from the resulting \mathbf{w} and name this training index set as \mathbf{S} .

Algorithm 3 GREAT

- 1: **Input:** pairwise LinearFGW distance matrix $\tilde{\mathbf{D}} \in \mathbb{R}^{n \times m}$, selection ratio τ , update step T , learning rate η .
- 2: Initialize uniform training weights: $\mathbf{w} = \frac{1}{n}$;
- 3: **for** $t = 1$ **to** $T - 1$ **do**
- 4: Compute $\text{GDD}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, \tilde{\mathbf{D}})$ via Algorithm 2;
- 5: Compute $\mathbf{g}_w = \nabla_w \text{GDD}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, \tilde{\mathbf{D}})$ via Theorem F.3;
- 6: Compute current sparsity level:
 $k = n \cdot \max(\tau, \frac{T-t+1}{T-1} + \frac{\tau t}{T-1})$;
- 7: Update data weight: $\mathbf{w} = \max(\mathbf{w} - \eta \cdot \mathbf{g}_w, \mathbf{0})$;
- 8: Sparsify data weight: $\mathbf{w} = \mathbf{w} \odot \text{Top-}k(\mathbf{w})$;
- 9: Apply ℓ_1 -normalization: $\mathbf{w} = \mathbf{w} / \|\mathbf{w}\|_1$;
- 10: **end for**
- 11: **return** training data index set $\mathbf{S} = \text{nonzero}(\mathbf{w})$.

D Summarization of GRADATE (Algorithm 4)

E Discussions on FGW & GDD and Previous Measures

Comparison between FGW [57] and TMD [11]. Specifically, FGW has the following advantages over Tree Mover Distance (TMD) [11]. Firstly, Linear optimal transport theory [59, 46] can be

Algorithm 4 GRADATE

- 1: **Input:** labeled training graphs $\mathcal{D}^{\text{train}} = \{\mathcal{G}_i^{\text{train}}, y_i^{\text{train}}\}_{i=1}^n$, labeled validation graphs $\mathcal{D}^{\text{val}} = \{\mathcal{G}_i^{\text{val}}, y_i^{\text{val}}\}_{i=1}^m$, trade-off parameter α , label signal strength $c \geq 0$, selection ratio τ , update step T , learning rate η .
 - 2: Compute pairwise LinearFGW distance matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ via Algorithm 1 with the graph set $\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}$ and parameter α ;
 - 3: Compute the (label-informed) pairwise distance matrix $\tilde{\mathbf{D}}$ with label signal c (line 3-12) in Algorithm 2;
 - 4: Compute $\mathbf{S} = \text{GREAT}(\tilde{\mathbf{D}}, \tau, T, \eta)$;
 - 5: **return** selected training data index set \mathbf{S} .
-

utilized to bring down the costs for pairwise FGW distance computation while TMD does not have similar technique. Secondly, a single-pair FGW computation (with time complexity $\mathcal{O}(|\mathcal{V}|^3)$) is cheaper than a single-pair TMD computation (with time complexity $\mathcal{O}(|\mathcal{L}|\mathcal{V}|^4)$), where \mathcal{V} is graph size and \mathcal{L} is the depth of TMD. While cheaper, FGW can achieve similar theoretical results as TMD.

Comparison between GDD and MMD [17]. GDD offers a more flexible and expressive notion of graph dataset similarity than Max Mean Discrepancy (MMD) [17], which solely compares aggregated graph embeddings. To be more specific, GDD has the following three advantages over MMD. Firstly, unlike MMD, which often depends on model-specific representations (such as pre-trained encoder) or require training, GDD does not involve training and is model-free. This makes it broadly applicable across various graph-level datasets without the need for task-specific models. Secondly, GDD can optionally incorporate *auxiliary label information* (when available), enabling more fine-grained and task-relevant comparisons between data distributions in classification settings. Finally, GDD is based on Wasserstein distance. Although not explicitly stated in our paper, this results in interpretable correspondences between data points across datasets that can directly be used for data selection or data re-weighting algorithms for domain adaptation applications.

F Proofs of Theorems

F.1 Proof of Theorem 3.1

In this section, we prove Theorem 3.1. We first focus on a simplified case with $k = 1$, which implies that the underlying GNN has only one layer. Then, based on this result, we use induction to generalize the conclusion to any positive k , which represents multi-layer GNNs.

F.1.1 Assumptions

With a slight abuse of notation, let a graph \mathcal{G} denote its node set as well. We assume that f only uses one-hop information followed by a linear transformation. Specifically, for any graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ and any node $u \in \mathcal{G}$, the output $f(\mathcal{G})_u$ depends only on the local neighborhood of u , defined as $\mathcal{N}_{\mathcal{G}}(u) := \{\mathbf{A}[u, v], \mathbf{X}[v]\}_{v \in \mathcal{G}}$. This localized aggregation is first computed by a convolution function g , and the result is then passed through a linear transformation with weights \mathbf{W} and bias \mathbf{b} , giving:

$$f(\mathcal{G})_u = \mathbf{W} \cdot g(\mathcal{N}_{\mathcal{G}}(u)) + \mathbf{b} = \mathbf{W} \cdot g(\{\mathbf{A}[u, v], \mathbf{X}[v]\}_{v \in \mathcal{G}}) + \mathbf{b}. \quad (10)$$

We assume the convolution function g is C_W -Lipschitz w.r.t. the following FGW distance $d_{W;\alpha}$: for any nodes $u_1 \in \mathcal{G}_1$ and $u_2 \in \mathcal{G}_2$,

$$d_{W;\alpha}(\mathcal{N}_{\mathcal{G}_1}(u_1), \mathcal{N}_{\mathcal{G}_2}(u_2)) \quad (11)$$

$$:= \left(\inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r}, \quad (12)$$

where we use $\mu_1 := \text{Unif}(\mathcal{G}_1)$ and $\mu_2 := \text{Unif}(\mathcal{G}_2)$ in this work.

F.1.2 Proof for $k = 1$

Proof. Let $\mu_1 := \text{Unif}(\mathcal{G}_1)$, $\mu_2 := \text{Unif}(\mathcal{G}_2)$.

For any coupling $\pi \in \Pi(\mu_1, \mu_2)$, by Jensen's inequality w.r.t. the concave function $x \mapsto x^{1/r}$,

$$\mathbb{E}_{(u_1, u_2) \sim \pi} [\|f(\mathcal{G}_1)_{u_1} - f(\mathcal{G}_2)_{u_2}\|] \quad (13)$$

$$= \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\|[\mathbf{W} \cdot g(\{(\mathbf{A}_1[u_1, v_1], \mathbf{X}_1[v_1])\}_{v_1 \in \mathcal{G}_1}) + \mathbf{b}] - [\mathbf{W} \cdot g(\{(\mathbf{A}_2[u_2, v_2], \mathbf{X}_2[v_2])\}_{v_2 \in \mathcal{G}_2}) + \mathbf{b}]\| \right] \quad (14)$$

$$\leq \mathbb{E}_{(u_1, u_2) \sim \pi} \left[C_W \|\mathbf{W}\| \cdot d_{W; \alpha}(\{(\mathbf{A}_1[u_1, v_1], \mathbf{X}_1[v_1])\}_{v_1 \in \mathcal{G}_1}, \{(\mathbf{A}_2[u_2, v_2], \mathbf{X}_2[v_2])\}_{v_2 \in \mathcal{G}_2}) \right] \quad (15)$$

$$= C_W \|\mathbf{W}\| \cdot \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\left(\inf_{\pi' \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(v_1, v_2) \sim \pi'} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \right] \quad (16)$$

$$\leq C \cdot \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\left(\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \right] \quad (17)$$

$$\leq C \cdot \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right]^{1/r}, \quad (18)$$

where $C_1 = C_W \|\mathbf{W}\|$. We explain the inequalities as follows. The first inequality is from our smoothness assumption stated in the previous subsection. The second is by removing the infimum. The third is another use of Jensen's inequality.

Since the above inequality holds for any valid coupling π , we can take infimum on both side. Thus, it follows that $d_W(f(\mathcal{G}_1), f(\mathcal{G}_2))$ is at most

$$\inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(u_1, u_2) \sim \pi} [\|f(\mathcal{G}_1)_{u_1} - f(\mathcal{G}_2)_{u_2}\|] \quad (19)$$

$$\leq \inf_{\pi \in \Pi(\mu_1, \mu_2)} C_1 \cdot \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right]^{1/r} \quad (20)$$

$$= C_1 \cdot \inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right]^{1/r} \quad (21)$$

$$= C_1 \cdot \inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right]^{1/r} \quad (22)$$

$$= C_1 \cdot \left(\inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right] \right)^{1/r} \quad (23)$$

$$= C_1 \cdot \left(\inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} [(1 - \alpha) \|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r + \alpha |\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right] \right)^{1/r} \quad (24)$$

$$= C_1 \cdot \text{FGW}_\alpha(\mathcal{G}_1, \mathcal{G}_2), \quad (25)$$

which completes the proof the case of $k = 1$. Note that the inequality is from our smoothness assumption stated in the previous section and the last equality is due to the definition of FGW distance with trade-off parameter $\beta = \alpha$. \square

F.1.3 Proof for general $k > 1$

Proof. For general $k > 1$, we can iteratively apply similar logic as in the case of $k = 1$ to bound the output distance with multi-layer GNNs. Specifically, we can write a k -layer GNN f as a composite function that concatenates multiple convolution layer (i.e. f_1, \dots, f_k)⁸ with ReLU activation functions (i.e. $\sigma_1, \dots, \sigma_{k-1}$): $f = f_k \circ \sigma_{k-1} \circ f_{k-1} \circ \dots \circ \sigma_1 \circ f_1$, where $\sigma_j = \text{ReLU}(\cdot)$. For any $m \leq k$, define $h_m := f_m \circ \sigma_{m-1} \circ h_{m-1} = f_m \circ h'_{m-1}$, where $h'_{m-1} = \sigma_{m-1} \circ h_{m-1}$.

⁸We assume all these convolution functions $\{f_m\}_{1 \leq m \leq k}$ satisfy the assumption we made in Section F.1.1 with constant C_W .

Note that we have $f = h_k$. Then, for any coupling $\pi \in \Pi(\mu_1, \mu_2)$, we have:

$$\mathbb{E}_{(u_1, u_2) \sim \pi} [\|f(\mathcal{G}_1)_{u_1} - f(\mathcal{G}_2)_{u_2}\|] \quad (26)$$

$$= \mathbb{E}_{(u_1, u_2) \sim \pi} [|h_k(\mathcal{G}_1)_{u_1} - h_k(\mathcal{G}_2)_{u_2}|] \quad (27)$$

$$= \mathbb{E}_{(u_1, u_2) \sim \pi} [|(f_k \circ h'_{k-1})(\mathcal{G}_1)_{u_1} - (f_k \circ h'_{k-1})(\mathcal{G}_2)_{u_2}|] \quad (28)$$

$$\leq \mathbb{E}_{(u_1, u_2) \sim \pi} [\|C_1 \cdot d_{W; \alpha}(h'_{k-1}(\mathcal{G}_1)_{u_1}, h'_{k-1}(\mathcal{G}_2)_{u_2})\|] \quad (29)$$

$$= C_1 \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\left(\inf_{\pi' \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(v_1, v_2) \sim \pi'} [(1-\alpha) [\|h'_{k-1}(\mathcal{G}_1)_{v_1} - h'_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r]] \right)^{1/r} \right] \quad (30)$$

$$\leq C_1 \mathbb{E}_{(u_1, u_2) \sim \pi} \left[\mathbb{E}_{(v_1, v_2) \sim \pi} \left((1-\alpha) [\|h'_{k-1}(\mathcal{G}_1)_{v_1} - h'_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \right] \quad (31)$$

$$\leq C_1 \left((1-\alpha) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|h'_{k-1}(\mathcal{G}_1)_{v_1} - h'_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \quad (32)$$

$$\leq C_1 \left((1-\alpha) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|h'_{k-1}(\mathcal{G}_1)_{v_1} - h'_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \quad (33)$$

$$\leq C_1 \left((1-\alpha) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|h_{k-1}(\mathcal{G}_1)_{v_1} - h_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \quad (34)$$

$$\leq C_1 \left((1-\alpha) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|h_{k-1}(\mathcal{G}_1)_{v_1} - h_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \quad (35)$$

$$\leq C_1 \left((1-\alpha) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|h_{k-1}(\mathcal{G}_1)_{v_1} - h_{k-1}(\mathcal{G}_2)_{v_2}\|^r] + \alpha \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r}. \quad (36)$$

Note that the first inequality is from the smoothness assumption, the second is by removing infimum, the third is by Jensen's inequality and the fourth is because ReLU(\cdot) is a contraction function.

Here, we can iteratively apply the regularity assumption specified in Section F.1.1 to expand the term above: $\|h_{m-1}(\mathcal{G}_1)_{v_1} - h_{m-1}(\mathcal{G}_2)_{v_2}\|^r, \forall m \in \{k, \dots, 1\}$ to have the following deduction.

$$\mathbb{E}_{(u_1, u_2) \sim \pi} [\|f(\mathcal{G}_1)_{u_1} - f(\mathcal{G}_2)_{u_2}\|] \quad (37)$$

$$\leq C_1 \left(C_1^{k-1} (1-\alpha)^k \mathbb{E}_{(v_1, v_2) \sim \pi} [\|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r] + \alpha \sum_{m=0}^{k-1} [C_1 (1-\alpha)]^m \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \quad (38)$$

$$= C \cdot \left((1-\beta) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r] + \beta \mathbb{E}_{(u_1, u_2) \sim \pi} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r}, \quad (39)$$

where $C = C_1 \frac{\alpha + (1-\alpha)^k (C_1)^{k-1} (1-C_1)}{1-C_1 (1-\alpha)}$ and $\beta = \frac{\alpha (1-C_1^k (1-\alpha)^k)}{\alpha + (1-\alpha)^k (C_1)^{k-1} (1-C_1)}$.

Since the above equation holds for any coupling π , we can take infimum from both sides to get:

$$d_W(f(\mathcal{G}_1), f(\mathcal{G}_2)) \quad (40)$$

$$= \inf_{\pi \in \Pi(\mu_1, \mu_2)} \mathbb{E}_{(u_1, u_2) \sim \pi} [\|f(\mathcal{G}_1)_{u_1} - f(\mathcal{G}_2)_{u_2}\|] \quad (41)$$

$$\leq C \cdot \inf_{\pi \in \Pi(\mu_1, \mu_2)} \left((1 - \beta) \mathbb{E}_{(v_1, v_2) \sim \pi} [\|\mathbf{X}_1[v_1] - \mathbf{X}_2[v_2]\|^r] + \right. \quad (42)$$

$$\left. \beta \mathbb{E}_{\substack{(u_1, u_2) \sim \pi \\ (v_1, v_2) \sim \pi}} [|\mathbf{A}_1[u_1, v_1] - \mathbf{A}_2[u_2, v_2]|^r] \right)^{1/r} \quad (43)$$

$$= C \cdot \text{FGW}_\beta(\mathcal{G}_1, \mathcal{G}_2), \quad (44)$$

which completes the proof. \square

Remark F.1. To justify the smoothness assumption on g , we note that it is an abstraction of GNN aggregation functions. For example, aggregation operations such as *mean*, *max* and *sum* all satisfy our assumption.

Remark F.2. Note that our technical assumption and the results of Theorem 3.1 are independent. Firstly, the assumption on the convolution function g is about the smoothness property *between node representations within a single graph*; while the results of Theorem 3.1 is bounding the FGW distance between *sets of node representations between two graphs*.

F.2 Proof of Theorem 3.3

For any coupling $\pi \in \Pi(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}})$, by Jensen's inequality and the Lipschitzness assumption,

$$\left| \mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{p}^{\text{train}}(\mathbf{w})} [\mathcal{L}(f(\mathcal{G}), y)] - \mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{q}^{\text{val}}} [\mathcal{L}(f(\mathcal{G}), y)] \right| \quad (45)$$

$$= \left| \mathbb{E}_{(\mathcal{G}^{\text{train}}, y^{\text{train}}) \sim \mathbf{p}^{\text{train}}(\mathbf{w})} [\mathcal{L}(f(\mathcal{G}^{\text{train}}), y^{\text{train}})] - \mathbb{E}_{(\mathcal{G}^{\text{val}}, y^{\text{val}}) \sim \mathbf{q}^{\text{val}}} [\mathcal{L}(f(\mathcal{G}^{\text{val}}), y^{\text{val}})] \right| \quad (46)$$

$$= \left| \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [\mathcal{L}(f(\mathcal{G}^{\text{train}}), y^{\text{train}})] - \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [\mathcal{L}(f(\mathcal{G}^{\text{val}}), y^{\text{val}})] \right| \quad (47)$$

$$= \left| \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [\mathcal{L}(f(\mathcal{G}^{\text{train}}), y^{\text{train}}) - \mathcal{L}(f(\mathcal{G}^{\text{val}}), y^{\text{val}})] \right| \quad (48)$$

$$\leq \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [|\mathcal{L}(f(\mathcal{G}^{\text{train}}), y^{\text{train}}) - \mathcal{L}(f(\mathcal{G}^{\text{val}}), y^{\text{val}})|] \quad (49)$$

$$\leq \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [C \cdot d_{g\mathcal{Z}}^c((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}}))] \quad (50)$$

$$= C \cdot \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [d_{g\mathcal{Z}}^c((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}}))]. \quad (51)$$

Since this holds for any coupling $\pi \in \Pi(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}})$, then we have

$$\left| \mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{p}^{\text{train}}(\mathbf{w})} [\mathcal{L}(f(\mathcal{G}), y)] - \mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{q}^{\text{val}}} [\mathcal{L}(f(\mathcal{G}), y)] \right| \quad (52)$$

$$\leq C \cdot \inf_{\pi \in \Pi(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}})} \mathbb{E}_{((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}})) \sim \pi} [d_{g\mathcal{Z}}^c((\mathcal{G}^{\text{train}}, y^{\text{train}}), (\mathcal{G}^{\text{val}}, y^{\text{val}}))] \quad (53)$$

$$= C \cdot \text{OT}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, d_{g\mathcal{Z}}^c) \quad (54)$$

$$= C \cdot \text{GDD}(\mathcal{D}_{\mathbf{w}}^{\text{train}}, \mathcal{D}^{\text{val}}). \quad (55)$$

It follows that

$$\mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{q}^{\text{val}}} [\mathcal{L}(f(\mathcal{G}), y)] \leq \mathbb{E}_{(\mathcal{G}, y) \sim \mathbf{p}^{\text{train}}(\mathbf{w})} [\mathcal{L}(f(\mathcal{G}), y)] + C \cdot \text{GDD}(\mathcal{D}_{\mathbf{w}}^{\text{train}}, \mathcal{D}^{\text{val}}), \quad (56)$$

which completes the proof. Note that the first inequality follows from Jensen's inequality (w.r.t. the absolute function). The second and third inequalities are both due to the smoothness assumption stated in Theorem 3.3.

F.3 Proof of Theorem F.3

Theorem F.3 (Gradient of GDD w.r.t. Training Weights; 26). *Given a distance matrix \mathbf{D} , a validation empirical measure \mathbf{q}^{val} and a training empirical measure $\mathbf{p}^{\text{train}}(\mathbf{w})$ based on the weight \mathbf{w} . Let $\beta(\pi^*)$ be the dual variables with respect to $\mathbf{p}^{\text{train}}(\mathbf{w})$ for the GDD problem defined in Equation (5). The gradient of GDD($\mathbf{p}^{\text{train}}(\mathbf{w})$, \mathbf{q}^{val} , \mathbf{D}) with respect to \mathbf{w} can be computed as:*

$$\nabla_{\mathbf{w}} \text{GDD}(\mathbf{p}^{\text{train}}(\mathbf{w}), \mathbf{q}^{\text{val}}, \mathbf{D}) = \beta^*(\pi^*),$$

where $\beta^*(\pi^*)$ is the optimal solution w.r.t. $\mathbf{p}^{\text{train}}(\mathbf{w})$ to the dual of the GDD problem.

Proof. Omitted. Please see the Sensitivity Theorem stated by Bertsekas [4]. \square

G Additional Experiments

G.1 Comparing data selection methods for graph size shift on GCN & GIN

We conduct the same evaluation as Table 1 on *graph size* shift with GCN and GIN as backbone model in Table 4.

Dataset	GNN Architecture → Selection Method ↓	GCN				GIN			
		$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full	$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full
IMDB-BINARY	Random	0.573±0.041	0.612±0.008	0.645±0.051		0.620±0.007	0.582±0.009	0.605±0.019	
	KIDD-LR	0.592±0.015	0.540±0.014	0.652±0.008	0.630±0.008	0.553±0.013	0.555±0.012	0.577±0.012	0.602±0.010
	LAVA	0.824±0.008	0.823±0.019	0.837±0.006		0.822±0.005	0.830±0.011	0.848±0.002	
	GRADATE	0.826±0.009	0.825±0.018	0.830±0.007		0.823±0.002	0.820±0.008	0.832±0.008	
IMDB-MULTI	Random	0.374±0.031	0.354±0.008	0.366±0.008		0.351±0.008	0.372±0.039	0.369±0.019	
	KIDD-LR	0.329±0.010	0.416±0.064	0.432±0.010	0.386±0.006	0.346±0.048	0.371±0.010	0.412±0.018	0.368±0.010
	LAVA	0.314±0.006	0.426±0.003	<u>0.600±0.005</u>		0.341±0.049	<u>0.388±0.018</u>	<u>0.563±0.007</u>	
	GRADATE	0.353±0.000	0.524±0.016	0.602±0.004		0.349±0.046	0.497±0.015	0.604±0.006	
MSRC_21	Random	0.450±0.008	0.497±0.011	0.781±0.019		0.149±0.007	0.418±0.008	0.690±0.015	
	KIDD-LR	0.725±0.017	0.819±0.015	0.857±0.008	0.816±0.026	0.649±0.012	0.743±0.008	0.781±0.050	0.749±0.023
	LAVA	0.617±0.015	0.825±0.014	0.918±0.018		0.617±0.008	<u>0.810±0.004</u>	<u>0.889±0.011</u>	
	GRADATE	0.670±0.017	0.836±0.017	0.953±0.011		0.629±0.011	0.813±0.008	0.901±0.008	
ogbg-molbace	Random	0.443±0.014	0.504±0.022	0.476±0.011		0.479±0.070	0.471±0.092	0.578±0.030	
	KIDD-LR	0.446±0.040	0.489±0.049	0.483±0.011	0.434±0.033	0.547±0.080	0.523±0.060	0.571±0.013	0.548±0.028
	LAVA	0.563±0.045	0.574±0.067	<u>0.535±0.044</u>		0.645±0.035	0.641±0.027	0.648±0.025	
	GRADATE	0.570±0.080	0.599±0.037	0.575±0.056		0.646±0.033	0.618±0.061	0.630±0.020	
ogbg-molbbbp	Random	0.499±0.041	0.635±0.042	0.648±0.031		0.698±0.010	0.633±0.043	0.691±0.040	
	KIDD-LR	0.639±0.025	0.599±0.013	0.611±0.023	0.618±0.037	0.546±0.108	0.656±0.038	0.609±0.081	0.779±0.017
	LAVA	0.667±0.015	0.675±0.013	0.691±0.017		0.859±0.019	<u>0.889±0.016</u>	<u>0.893±0.011</u>	
	GRADATE	0.677±0.007	0.671±0.015	0.673±0.041		0.866±0.016	0.890±0.011	0.895±0.012	
ogbg-molhiv	Random	0.576±0.008	0.579±0.004	0.594±0.001		0.613±0.004	0.617±0.045	0.624±0.015	
	KIDD-LR	0.556±0.001	0.551±0.027	0.595±0.003	0.592±0.000	0.586±0.055	0.586±0.014	0.629±0.019	0.664±0.027
	LAVA	0.669±0.001	0.683±0.004	0.659±0.002		0.769±0.014	0.737±0.012	0.796±0.025	
	GRADATE	0.640±0.002	0.638±0.006	0.629±0.000		0.731±0.017	0.767±0.004	0.805±0.024	

Table 4: Performance comparison across data selection methods for *graph size* shift on GCN and GIN. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE achieves top-2 performance across all datasets and is the best-performer in most settings.

G.2 Comparing data selection methods for graph density shift on GAT & GraphSAGE

We conduct the same evaluation as Table 1 on *graph density* shift with GAT and GraphSAGE as backbone model in Table 5.

G.3 Comparing data selection methods for graph size shift on GAT & GraphSAGE

We conduct the same evaluation as Table 1 on *graph size* shift with GAT and GraphSAGE as backbone model in Table 6.

G.4 Comparing GDA and vanilla methods for graph size shift

We conduct the same evaluation as Table 2 on *graph size* shift in Table 7.

Dataset	GNN Architecture →		GAT			Full	GraphSAGE			
	Selection Method ↓		$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$		$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full
IMDB-BINARY	Random	0.602 ± 0.005	0.695 ± 0.035	0.797 ± 0.005			0.730 ± 0.014	0.637 ± 0.039	0.762 ± 0.027	
	KIDD-LR	0.683 ± 0.041	0.803 ± 0.005	0.817 ± 0.024	0.807 ± 0.033	0.662 ± 0.054	0.785 ± 0.025	0.775 ± 0.054		
	LAVA	0.818 ± 0.010	0.857 ± 0.009	0.885 ± 0.018		0.827 ± 0.008	0.840 ± 0.021	0.883 ± 0.012	0.823 ± 0.009	
IMDB-MULTI	GRADATE	0.850 ± 0.023	0.865 ± 0.008	0.892 ± 0.012		0.835 ± 0.015	0.852 ± 0.035	0.907 ± 0.005		
	Random	0.087 ± 0.014	0.071 ± 0.006	0.076 ± 0.003	0.080 ± 0.000	0.090 ± 0.005	0.203 ± 0.061	0.126 ± 0.064		
	KIDD-LR	0.176 ± 0.024	0.121 ± 0.044	0.158 ± 0.036		0.154 ± 0.028	0.124 ± 0.068	0.054 ± 0.011	0.097 ± 0.024	
MSRC_21	LAVA	0.597 ± 0.273	0.599 ± 0.294	<u>0.341</u> ± 0.049		0.341 ± 0.049	0.307 ± 0.164	<u>0.328</u> ± 0.317		
	GRADATE	0.790 ± 0.000	0.589 ± 0.287	0.776 ± 0.039	0.860 ± 0.007	0.306 ± 0.216	0.299 ± 0.282	0.363 ± 0.238		
	Random	0.462 ± 0.029	0.763 ± 0.007	0.857 ± 0.018		0.617 ± 0.017	0.725 ± 0.033	0.842 ± 0.029		
ogbg-molbace	KIDD-LR	0.661 ± 0.030	0.778 ± 0.015	0.860 ± 0.025		0.681 ± 0.073	0.787 ± 0.025	0.857 ± 0.004	0.874 ± 0.004	
	LAVA	0.699 ± 0.047	<u>0.816</u> ± 0.037	<u>0.912</u> ± 0.007	0.583 ± 0.042	0.766 ± 0.029	<u>0.857</u> ± 0.015	<u>0.918</u> ± 0.011		
	GRADATE	0.716 ± 0.017	0.822 ± 0.004	0.921 ± 0.007		0.781 ± 0.026	0.877 ± 0.026	0.944 ± 0.011		
ogbg-molbbbp	Random	0.480 ± 0.040	0.606 ± 0.085	0.637 ± 0.075	0.569 ± 0.030	0.459 ± 0.149	0.478 ± 0.097	0.503 ± 0.034		
	KIDD-LR	0.558 ± 0.012	0.443 ± 0.029	0.628 ± 0.023		0.606 ± 0.023	0.596 ± 0.079	0.607 ± 0.047	0.622 ± 0.119	
	LAVA	0.564 ± 0.097	0.519 ± 0.007	<u>0.696</u> ± 0.031		0.620 ± 0.073	0.649 ± 0.004	0.651 ± 0.059		
ogbg-molhiv	GRADATE	0.501 ± 0.017	0.541 ± 0.048	0.720 ± 0.004	0.571 ± 0.030	0.621 ± 0.067	0.587 ± 0.078	0.568 ± 0.126		
	Random	0.511 ± 0.034	0.529 ± 0.027	0.513 ± 0.018		0.463 ± 0.012	0.385 ± 0.032	0.468 ± 0.008		
	KIDD-LR	0.444 ± 0.050	0.405 ± 0.021	0.434 ± 0.025		0.392 ± 0.002	0.415 ± 0.028	0.466 ± 0.034	0.447 ± 0.008	
GRADATE	LAVA	0.584 ± 0.054	0.552 ± 0.018	<u>0.603</u> ± 0.021		0.526 ± 0.087	0.612 ± 0.005	0.495 ± 0.029		
	Random	0.601 ± 0.017	0.591 ± 0.011	0.581 ± 0.016	0.571 ± 0.030	0.577 ± 0.016	0.591 ± 0.007	0.594 ± 0.005		
	KIDD-LR	0.620 ± 0.001	0.616 ± 0.003	0.615 ± 0.007		0.607 ± 0.008	0.534 ± 0.057	0.603 ± 0.018	0.588 ± 0.003	
GRADATE	LAVA	0.621 ± 0.001	0.631 ± 0.003	0.624 ± 0.014		0.575 ± 0.012	0.607 ± 0.008	0.608 ± 0.007		
	Random	0.638 ± 0.001	0.620 ± 0.002	<u>0.619</u> ± 0.004	0.599 ± 0.021	<u>0.598</u> ± 0.009	0.610 ± 0.006			
	KIDD-LR									
LAVA	GRADATE									

Table 5: Performance comparison across data selection methods for *graph density* shift on GAT and GraphSAGE. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE is the best-performer in most settings.

Dataset	GNN Architecture →		GAT			Full	GraphSAGE			
	Selection Method ↓		$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$		$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full
IMDB-BINARY	Random	0.678 ± 0.082	0.558 ± 0.022	0.660 ± 0.085			0.555 ± 0.011	0.563 ± 0.012	0.562 ± 0.012	
	KIDD-LR	0.683 ± 0.071	0.587 ± 0.081	0.665 ± 0.098	0.595 ± 0.007	0.663 ± 0.035	0.558 ± 0.018	0.595 ± 0.007		
	LAVA	0.808 ± 0.014	<u>0.830</u> ± 0.004	<u>0.835</u> ± 0.000		0.807 ± 0.026	0.808 ± 0.027	0.830 ± 0.004	0.567 ± 0.018	
IMDB-MULTI	GRADATE	0.835 ± 0.018	0.833 ± 0.005	0.837 ± 0.010		0.808 ± 0.016	0.828 ± 0.024	0.838 ± 0.016		
	Random	0.384 ± 0.014	0.408 ± 0.004	0.384 ± 0.034		0.336 ± 0.010	0.357 ± 0.005	0.381 ± 0.026		
	KIDD-LR	0.366 ± 0.020	0.434 ± 0.006	0.404 ± 0.010	0.374 ± 0.028	0.339 ± 0.030	0.418 ± 0.030	0.422 ± 0.011	0.391 ± 0.026	
MSRC_21	LAVA	0.333 ± 0.058	0.417 ± 0.015	<u>0.577</u> ± 0.014		0.374 ± 0.021	<u>0.389</u> ± 0.039	0.392 ± 0.036		
	GRADATE	0.342 ± 0.050	0.537 ± 0.003	0.616 ± 0.002		0.392 ± 0.032	0.360 ± 0.025	0.517 ± 0.071		
	Random	0.284 ± 0.025	0.614 ± 0.056	0.731 ± 0.018	0.787 ± 0.004	0.497 ± 0.023	0.412 ± 0.050	0.725 ± 0.022		
ogbg-molbace	KIDD-LR	0.626 ± 0.004	0.746 ± 0.026	0.830 ± 0.011		0.722 ± 0.030	0.798 ± 0.029	0.789 ± 0.026	0.810 ± 0.027	
	LAVA	0.620 ± 0.015	0.798 ± 0.012	<u>0.909</u> ± 0.018		0.693 ± 0.029	<u>0.827</u> ± 0.015	<u>0.918</u> ± 0.008		
	GRADATE	0.643 ± 0.039	0.860 ± 0.021	0.947 ± 0.014		0.760 ± 0.023	0.842 ± 0.007	0.944 ± 0.004		
ogbg-molbbbp	Random	0.515 ± 0.006	0.464 ± 0.056	0.488 ± 0.005	0.463 ± 0.004	0.523 ± 0.048	0.463 ± 0.020	0.583 ± 0.027		
	KIDD-LR	0.480 ± 0.011	0.452 ± 0.016	0.467 ± 0.022		0.507 ± 0.078	0.457 ± 0.029	0.467 ± 0.043	0.487 ± 0.108	
	LAVA	0.524 ± 0.037	0.545 ± 0.058	0.613 ± 0.080		0.650 ± 0.011	0.481 ± 0.007	<u>0.550</u> ± 0.060		
GRADATE	Random	0.666 ± 0.003	0.677 ± 0.007	0.684 ± 0.018	0.679 ± 0.004	0.634 ± 0.018	0.648 ± 0.028	0.641 ± 0.025		
	KIDD-LR	0.594 ± 0.017	0.596 ± 0.020	0.650 ± 0.002		0.518 ± 0.046	0.594 ± 0.004	0.602 ± 0.061	0.680 ± 0.010	
	LAVA	0.714 ± 0.011	0.731 ± 0.019	<u>0.710</u> ± 0.036		0.639 ± 0.022	0.602 ± 0.030	<u>0.645</u> ± 0.026		
ogbg-molhiv	GRADATE	0.735 ± 0.021	0.699 ± 0.027	0.713 ± 0.005	0.588 ± 0.005	0.623 ± 0.041	0.650 ± 0.019	0.652 ± 0.041		
	Random	0.584 ± 0.001	0.585 ± 0.002	0.589 ± 0.003		0.610 ± 0.024	0.491 ± 0.031	0.603 ± 0.001		
	KIDD-LR	0.586 ± 0.001	0.584 ± 0.001	0.584 ± 0.004		0.549 ± 0.071	0.588 ± 0.008	0.564 ± 0.011	0.596 ± 0.000	
LAVA	GRADATE	0.704 ± 0.005	0.773 ± 0.007	0.759 ± 0.002		0.721 ± 0.004	0.701 ± 0.008	0.686 ± 0.011		
	Random	0.663 ± 0.005	0.655 ± 0.009	0.660 ± 0.008	0.637 ± 0.002	0.646 ± 0.019	0.640 ± 0.007			
	KIDD-LR									
	LAVA	GRADATE								

Table 6: Performance comparison across data selection methods for *graph size* shift on GAT and GraphSAGE. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE achieves top-2 performance across most settings. The under-performance on ogbg-molhiv might due to the reason discussed in Section 4.5.

G.5 Enhancing GDA methods for *graph size* shift

We conduct the same evaluation as Table 3 on *graph size* shift in Table 8.

G.6 Validation-label-free setting

While we originally consider c as tunable parameter, we acknowledge that the existence of validation labels will be implicitly required when $c \neq 0$, which might not be practical under some real-world scenarios. Here, we pick two GNN backbones (i.e. GCN [30] and GIN [65]) evaluate the effectiveness

Type	Model	Data	Dataset					
			IMDB-BINARY	IMDB-MULTI	MSRC_21	ogbg-molbace	ogbg-molbbbp	ogbg-molhiv
GDA	AdaGCN	Full	0.593 \pm 0.012	0.362 \pm 0.017	0.202 \pm 0.075	0.513 \pm 0.018	0.625 \pm 0.137	0.412 \pm 0.011
	GRADE	Full	0.648 \pm 0.105	0.390 \pm 0.019	0.696 \pm 0.008	0.403 \pm 0.018	0.669 \pm 0.005	0.599 \pm 0.005
	ASN	Full	0.633 \pm 0.054	0.372 \pm 0.009	0.734 \pm 0.015	0.523 \pm 0.091	0.616 \pm 0.042	0.519 \pm 0.077
	UDAGCN	Full	0.688 \pm 0.049	0.392 \pm 0.046	0.260 \pm 0.049	0.448 \pm 0.020	0.513 \pm 0.024	0.439 \pm 0.034
GCN	GCN	Random 20%	0.612 \pm 0.008	0.354 \pm 0.008	0.497 \pm 0.011	0.504 \pm 0.022	0.635 \pm 0.042	0.579 \pm 0.004
		LAVA 20%	0.823 \pm 0.019	0.426 \pm 0.003	0.825 \pm 0.014	0.574 \pm 0.067	0.675 \pm 0.013	0.683 \pm 0.038
		GRADATE 20%	0.825 \pm 0.018	0.524 \pm 0.016	0.836 \pm 0.017	0.599 \pm 0.037	0.671 \pm 0.015	0.638 \pm 0.006
GIN	GIN	Random 20%	0.582 \pm 0.009	0.372 \pm 0.039	0.418 \pm 0.008	0.471 \pm 0.092	0.633 \pm 0.043	0.617 \pm 0.045
		LAVA 20%	0.830 \pm 0.011	0.388 \pm 0.018	0.810 \pm 0.004	0.641 \pm 0.027	0.889 \pm 0.016	0.737 \pm 0.012
		GRADATE 20%	0.820 \pm 0.008	0.497 \pm 0.015	0.813 \pm 0.008	0.618 \pm 0.061	0.890 \pm 0.011	0.767 \pm 0.004
Vanilla	Vanilla	Random 20%	0.558 \pm 0.022	0.408 \pm 0.004	0.614 \pm 0.056	0.464 \pm 0.056	0.677 \pm 0.007	0.585 \pm 0.002
		LAVA 20%	0.830 \pm 0.004	0.417 \pm 0.015	0.798 \pm 0.012	0.545 \pm 0.058	0.731 \pm 0.019	0.773 \pm 0.007
		GRADATE 20%	0.833 \pm 0.005	0.537 \pm 0.003	0.860 \pm 0.021	0.570 \pm 0.062	0.699 \pm 0.027	0.655 \pm 0.009
GAT	GAT	Random 20%	0.563 \pm 0.012	0.357 \pm 0.005	0.412 \pm 0.050	0.463 \pm 0.020	0.648 \pm 0.028	0.491 \pm 0.031
		LAVA 20%	0.808 \pm 0.027	0.389 \pm 0.039	0.827 \pm 0.015	0.481 \pm 0.007	0.602 \pm 0.030	0.701 \pm 0.008
		GRADATE 20%	0.828 \pm 0.024	0.360 \pm 0.025	0.842 \pm 0.007	0.561 \pm 0.090	0.650 \pm 0.019	0.646 \pm 0.019

Table 7: Performance comparison across GDA and vanilla methods for *graph size* shift. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE can consistently achieve top-2 performance across all datasets and is the best performer in most settings.

of GRADATE with $c = 0$ (i.e. validation-label-free). The result is shown in Table 9. For simplicity, we only report relative performance improvement (%) of GRADATE over the strongest baseline under all settings. We can observe that the advantage of GRADATE remains comparable to what is reported in the main text.

G.7 Additional backbones

In addition to typical GNN backbones, we also conduct experiments on a wider range of graph algorithms, including SGFormer [60] and APPNP [16]. In Table 10, we report relative performance improvement (%) of GRADATE over the strongest baseline under all settings. We can observe that GRADATE still outperforms other baselines mostly.

G.8 Additional GDA methods

We add additional experiments based on two variants of A2GNN [34] with different losses and TDSS [7]. In Table 11, we report relative performance improvement (%) of GRADATE over the strongest baseline under all settings. For simplicity, we report results with selection ratio equals to 20%. It can be observed that GRADATE consistently provides the most significant enhancements for the three newly evaluated GDA methods compared to other selection baselines.

H Datasets

In Table 12, we provide details of datasets used in this work. For # NODES and # EDGES, we report the mean sizes across all graphs in the dataset.

I Backbone GNN Settings for Graph Selection Evaluation

GNN Models. We consider four widely used graph neural network architecture, GCN [30], GIN [65], GAT [58] and GraphSAGE [18]. The detailed model architectures are described as follows: (i) For GCN, we use three GCN layers with number of hidden dimensions equal to 32. ReLU is used between layers and a global mean pooling layer is set as the readout layer to generate graph-level embedding. A dropout layer with probability $p = 0.5$ is applied after the GCN layers. Finally, a linear layer with softmax is placed at the end for graph class prediction. (ii) For GIN, we use three-layer GIN with 32 hidden dimensions. We use ReLU between layers and global mean pooling for readout. A dropout layer with probability 0.5 is placed after GIN layers and finally a linear layer with softmax for prediction. (iii) For GAT, we use two-layer GAT layers with four heads with global mean pooling

Dataset	GDA Method → AdaGCN				GRADE					
	Selection Method ↓		$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full	$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full
IMDB-BINARY	Random	0.582 ± 0.091	0.520 ± 0.103	0.455 ± 0.120		0.593 ± 0.012	0.572 ± 0.111	0.522 ± 0.045	0.613 ± 0.095	
	LAVA	0.818 ± 0.012	0.815 ± 0.005	0.810 ± 0.013			0.813 ± 0.007	0.814 ± 0.007	0.816 ± 0.005	0.648 ± 0.105
	GRADATE	0.834 ± 0.014	0.830 ± 0.010	0.822 ± 0.022			0.814 ± 0.013	0.826 ± 0.007	0.827 ± 0.013	
IMDB-MULTI	Random	0.261 ± 0.064	0.247 ± 0.052	0.252 ± 0.059		0.362 ± 0.017	0.312 ± 0.034	0.280 ± 0.000	0.282 ± 0.030	
	LAVA	0.374 ± 0.055	0.385 ± 0.080	0.368 ± 0.098			0.386 ± 0.047	0.407 ± 0.076	0.411 ± 0.050	0.390 ± 0.019
	GRADATE	0.386 ± 0.053	0.442 ± 0.085	0.509 ± 0.114			0.401 ± 0.076	0.411 ± 0.076	0.503 ± 0.112	
MSRC_21	Random	0.084 ± 0.028	0.079 ± 0.022	0.114 ± 0.030		0.202 ± 0.075	0.137 ± 0.012	0.379 ± 0.053	0.667 ± 0.055	
	LAVA	0.377 ± 0.029	0.472 ± 0.030	0.540 ± 0.103			0.532 ± 0.029	0.728 ± 0.038	0.854 ± 0.035	0.696 ± 0.008
	GRADATE	0.411 ± 0.075	0.465 ± 0.042	0.593 ± 0.071			0.553 ± 0.043	0.744 ± 0.044	0.867 ± 0.013	
ogbg-molbace	Random	0.498 ± 0.091	0.477 ± 0.038	0.498 ± 0.063		0.513 ± 0.018	0.478 ± 0.005	0.468 ± 0.074	0.475 ± 0.046	
	LAVA	0.510 ± 0.027	0.523 ± 0.066	0.529 ± 0.056			0.509 ± 0.055	0.559 ± 0.026	0.552 ± 0.033	0.403 ± 0.018
	GRADATE	0.524 ± 0.057	0.560 ± 0.053	0.550 ± 0.030			0.556 ± 0.069	0.508 ± 0.025	0.512 ± 0.022	
ogbg-molbbbp	Random	0.539 ± 0.020	0.600 ± 0.044	0.538 ± 0.083		0.625 ± 0.137	0.632 ± 0.006	0.648 ± 0.002	0.650 ± 0.002	
	LAVA	0.583 ± 0.075	0.653 ± 0.007	0.657 ± 0.004			0.631 ± 0.011	0.640 ± 0.001	0.637 ± 0.003	0.669 ± 0.005
	GRADATE	0.662 ± 0.020	0.654 ± 0.004	0.664 ± 0.010			0.636 ± 0.010	0.641 ± 0.005	0.639 ± 0.013	
ogbg-molhiv	Random	0.356 ± 0.023	0.358 ± 0.013	0.364 ± 0.011		0.412 ± 0.011	0.588 ± 0.014	0.615 ± 0.012	0.592 ± 0.010	
	LAVA	0.382 ± 0.035	0.403 ± 0.033	0.384 ± 0.041			0.673 ± 0.004	0.681 ± 0.002	0.668 ± 0.009	0.599 ± 0.005
	GRADATE	0.393 ± 0.040	0.387 ± 0.068	0.395 ± 0.040			0.658 ± 0.004	0.647 ± 0.005	0.642 ± 0.005	
Dataset	GDA Method → ASN				UDAGCN					
	Selection Method ↓	$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full	$\tau = 10\%$	$\tau = 20\%$	$\tau = 50\%$	Full	
IMDB-BINARY	Random	0.613 ± 0.110	0.568 ± 0.078	0.515 ± 0.024		0.633 ± 0.054	0.507 ± 0.077	0.467 ± 0.029	0.605 ± 0.067	
	LAVA	0.817 ± 0.012	0.810 ± 0.012	0.847 ± 0.017			0.817 ± 0.012	0.811 ± 0.005	0.837 ± 0.017	0.688 ± 0.049
	GRADATE	0.825 ± 0.007	0.819 ± 0.024	0.834 ± 0.012			0.840 ± 0.008	0.831 ± 0.009	0.816 ± 0.016	
IMDB-MULTI	Random	0.126 ± 0.013	0.101 ± 0.058	0.156 ± 0.039		0.372 ± 0.009	0.340 ± 0.080	0.306 ± 0.019	0.307 ± 0.033	
	LAVA	0.379 ± 0.050	0.445 ± 0.057	0.593 ± 0.004			0.348 ± 0.051	0.387 ± 0.093	0.519 ± 0.120	0.392 ± 0.046
	GRADATE	0.425 ± 0.015	0.455 ± 0.097	0.577 ± 0.006			0.390 ± 0.055	0.444 ± 0.089	0.451 ± 0.145	
MSRC_21	Random	0.481 ± 0.071	0.277 ± 0.039	0.556 ± 0.012		0.734 ± 0.015	0.151 ± 0.072	0.204 ± 0.065	0.209 ± 0.062	
	LAVA	0.661 ± 0.027	0.779 ± 0.039	0.867 ± 0.017			0.435 ± 0.024	0.498 ± 0.090	0.563 ± 0.099	0.260 ± 0.049
	GRADATE	0.686 ± 0.022	0.796 ± 0.020	0.868 ± 0.034			0.465 ± 0.051	0.470 ± 0.097	0.616 ± 0.055	
ogbg-molbace	Random	0.465 ± 0.048	0.440 ± 0.049	0.466 ± 0.060		0.523 ± 0.091	0.485 ± 0.017	0.503 ± 0.044	0.544 ± 0.011	
	LAVA	0.496 ± 0.077	0.560 ± 0.032	0.596 ± 0.052			0.499 ± 0.036	0.553 ± 0.041	0.517 ± 0.012	0.448 ± 0.020
	GRADATE	0.565 ± 0.073	0.596 ± 0.053	0.546 ± 0.023			0.521 ± 0.002	0.519 ± 0.022	0.555 ± 0.024	
ogbg-molbbbp	Random	0.537 ± 0.091	0.530 ± 0.076	0.545 ± 0.062		0.616 ± 0.042	0.549 ± 0.031	0.568 ± 0.043	0.536 ± 0.008	
	LAVA	0.606 ± 0.024	0.635 ± 0.008	0.646 ± 0.000			0.655 ± 0.005	0.649 ± 0.003	0.673 ± 0.011	0.513 ± 0.024
	GRADATE	0.621 ± 0.016	0.640 ± 0.019	0.650 ± 0.017			0.660 ± 0.008	0.674 ± 0.011	0.677 ± 0.027	
ogbg-molhiv	Random	0.385 ± 0.023	0.459 ± 0.086	0.397 ± 0.070		0.519 ± 0.077	0.446 ± 0.041	0.412 ± 0.021	0.409 ± 0.014	
	LAVA	0.449 ± 0.058	0.465 ± 0.088	0.399 ± 0.074			0.426 ± 0.021	0.431 ± 0.042	0.433 ± 0.017	0.439 ± 0.034
	GRADATE	0.435 ± 0.044	0.423 ± 0.096	0.474 ± 0.094			0.433 ± 0.020	0.434 ± 0.008	0.395 ± 0.015	

Table 8: Performance comparison across combinations of GDA methods and data selection methods for *graph size* shift. We use **bold**/underline to indicate the 1st/2nd best results. GRADATE achieves the best performance in most settings.

Dataset	GCN ($\tau=10\%$)	GCN ($\tau=20\%$)	GCN ($\tau=50\%$)	GIN ($\tau=10\%$)	GIN ($\tau=20\%$)	GIN ($\tau=50\%$)
IMDB-BINARY	+11.13	+9.92	+2.76	+4.63	+5.16	+13.37
IMDB-MULTI	+221.31	+201.09	+189.61	+45.78	+95.01	+87.04
MSRC_21	+1.99	+1.36	+3.01	+2.36	+0.09	+2.88
ogbg-molbace	+1.32	-4.39	+13.17	-16.32	+2.84	+7.48
ogbg-molbbbp	+0.00	-14.46	+0.00	-0.08	+2.18	+2.22
ogbg-molhiv	+3.05	+0.75	+0.48	-1.24	+0.35	+2.52

Table 9: Relative improvement (%) of GRADATE over the strongest baseline under different settings.

for readout. A dropout layer with probability 0.5 is placed after GIN layers and finally a linear layer with softmax for prediction. (iv) For GraphSAGE, we use two GraphSAGE layers with mean aggregation operation. The hidden dimension is set to 32. A dropout layer with probability $p = 0.5$ is applied after the GCN layers. Finally, a linear layer with softmax is placed at the end for graph class prediction.

Experiment Details. We perform all our methods in Python and GNN models are built-in modules of PyTorch Geometric [14]. The learning rate is set to 10^{-2} with weight decay $5 \cdot 10^{-4}$. We

Dataset	SGFormer ($\tau=10\%$)	SGFormer ($\tau=20\%$)	SGFormer ($\tau=50\%$)	APPNP ($\tau=10\%$)	APPNP ($\tau=20\%$)	APPNP ($\tau=50\%$)
IMDB-BINARY	+9.82	+0.59	-0.34	+7.28	+3.01	-0.35
IMDB-MULTI	+6.46	+57.40	+1.65	-13.46	-2.59	+13.71
MSRC_21	+14.26	+3.58	+2.13	+9.54	+3.18	+1.27
ogbg-molbace	-9.66	+0.05	+1.90	+4.03	-2.31	+2.84
ogbg-molbbbp	+2.93	+5.47	-3.71	+22.80	+6.33	-4.58
ogbg-molhiv	+1.51	-1.79	+1.65	-12.00	+6.07	+1.88

Table 10: Relative improvement (%) of GRADATE over the strongest baseline under different settings.

Dataset	A2GNN-ADV ($\tau=20\%$)	A2GNN-MMD ($\tau=20\%$)	TDSS ($\tau=20\%$)
IMDB-BINARY	+31.12	+11.40	+3.12
IMDB-MULTI	+26.54	+58.88	-42.10
MSRC_21	+20.63	-4.40	+25.09
ogbg-molbace	+1.46	-2.57	+2.88
ogbg-molbbbp	+17.34	+5.85	-5.71
ogbg-molhiv	+0.65	+0.81	+0.55

Table 11: Relative improvement (%) of GRADATE over three additionally added GDA methods under different datasets.

train 200 epochs for datasets IMDB-BINARY, IMDB-MULTI, MSRC_21 and 100 epochs for datasets ogbg-molbace, ogbg-molbbbp, ogbg-molhiv with early stopping, evaluating the test set on the model checkpoint that achieves the highest validation performance during training. For each combination of data and model, we report the mean and standard deviation of classification performance over 3-5 random trials. For TUDatasets, we use accuracy as the performance metric; for OGB datasets, we use AUCROC as the performance metric. The computation is performed on Linux with an NVIDIA Tesla V100-SXM2-32GB GPU. For graphs without node features, we also follow Zeng et al. [81] that generates degree-specific one-hot features for each node in the graphs.

J GDA Method-Specific Settings

We follow the default parameter settings in the code repository of OpenGDA [53]. We train 200 epochs for datasets IMDB-BINARY, IMDB-MULTI, MSRC_21 and 100 epochs for datasets ogbg-molbace, ogbg-molbbbp, ogbg-molhiv with early stopping, evaluating the test set on the model checkpoint that achieves the highest validation performance during training.

- AdaGCN [12]: We set the learning rate to 10^{-3} with regularization coefficient equal to 10^{-4} . Dropout rate is 0.3 and $\lambda_b = 1$, $\lambda_{gp} = 5$.
- ASN [83]: We set the learning rate to 10^{-3} with regularization coefficient equal to 10^{-4} . The dropout rate is 0.5. The difference loss coefficient, domain loss coefficient and the reconstruction loss coefficient is set to $10^{-6}, 0.1, 0.5$.
- GRADE [61]: We set the learning rate to 10^{-3} with regularization coefficient equal to 10^{-4} . Dropout rate is set to 0.1.
- UDAGCN [62]: We set the learning rate to 10^{-3} with regularization coefficient equal to 10^{-4} . The domain loss weight equals to 1.

K Additional Preliminary: Graph Domain Adaptation (GDA)

Consider a source domain $\mathcal{D}_s = (\mathcal{G}_i^s, y_i^s)_{i=1}^{n_s}$ and a target domain $\mathcal{D}_t = (\mathcal{G}_i^t, y_i^t)_{i=1}^{n_t}$, where each $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ represents an attributed graph with the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the dimension of node features. With a

DATASET	# GRAPHS	# NODES	# EDGES	#FEATURES	# CLASS	DATA SOURCE	LICENSE
IMDB-BINARY	1000	19.77	96.53	None	2	PyG [14]	MIT License
IMDB-MULTI	1500	12.74	53.88	None	3	PyG [14]	MIT License
MSRC_21	563	77.52	198.32	None	20	PyG [14]	MIT License
ogbg-molbace	1513	34.08	36.85	9	2	OGB [21]	MIT License
ogbg-molbbbp	2039	24.06	25.95	9	2	OGB [21]	MIT License
ogbg-molhiv	41127	25.51	27.46	9	2	OGB [21]	MIT License

Table 12: Dataset Statistics and Licenses.

shared label set \mathcal{Y} , the graphs in the source domain are labeled with $y_i^s \in \mathcal{Y}$. The two domains are drawn from shifted joint distributions of graph and label space, i.e., $P_s(\mathcal{G}, y) \neq P_t(\mathcal{G}, y)$. The goal of GDA is to learn a classifier $f : \mathcal{G} \rightarrow \mathcal{Y}$ with the source domain data that minimizes the expected risk on the target domain: $\mathbb{E}_{(\mathcal{G}, y) \sim P_t} [\mathcal{L}(f(\mathcal{G}), y)]$, where \mathcal{L} is a task-specific loss function.

L Additional Related Work

In the era of big data and AI [40, 39, 48–51, 47, 68, 74–76, 67, 33], efficient data utilization has become paramount for scalable machine learning. Beyond the data selection and domain adaptation methods discussed in the main text, our work also connects to several related research directions that provide complementary perspectives on graph learning and distribution matching. Specifically, there is a rich body of work at the intersection of optimal transport theory and graph data [24, 77–80, 82], which are closely related to our main methodology.

M Empirical Runtime of GRADATE

In Table 13 and Table 14, we provide the empirical runtime on datasets (IMDB-BINARY, IMDB-MULTI and MSRC_21) and datasets (ogbg-molbace, ogbg-molbbbp and ogbg-molhiv), respectively. We observe that the on-line runtime is insignificant compared to typical GNN training time. And the off-line computation is only run once, which can be pre-computed. Furthermore, we can achieve much better accuracy compared to LAVA with nearly no additional runtime.

	Procedure / Dataset	IMDB-BINARY	IMDB-MULTI	MSRC_21
Off-line Computation (GDD Computation)	FGW Pairwise distance	7.41	9.61	18.18
	Label-informed pairwise distance	0.04	0.06	0.24
On-line Computation	GREAT (Algorithm 3)	0.28	0.52	0.11
	LAVA	0.09	0.14	0.03
GNN Training Time	GCN (w/ 10% data)	13.45	16.36	9.59
	GCN (w/ 20% data)	17.64	21.40	13.85
	GCN (w/ 50% data)	29.92	45.82	19.57

Table 13: **Empirical run-time behavior for TUDatasets (in seconds).** We can observe that the off-line procedures can be run comparable to a single GNN training time and the on-line procedure has a negligible runtime compared to GNN training. In addition, we can achieve significantly better performance compared to LAVA with slight additional on-line runtime.

N Limitations and Outlook

1. *How to eliminate the dependence on validation data?* Although it is common in machine learning research to assume we have some validation set that represents the data distribution on the target set (or “statistically closer” to the target set), it might not be always available under certain extreme scenarios. Thus, our interesting future direction is to extend our framework to no-validation-data or test-time adaptation settings.
2. *Can our proposed method scale to extremely large settings?* When we have millions of *large* graphs in both training and validation set, the efficiency of GRADATE might be a concern. However, most of the computationally intensive sub-procedure of our method can be done off-line (see complexity analysis in Section 3.2 and empirical runtime in Appendix M) and

	Procedure / Dataset	ogbg-molbace	ogbg-molbbbp	ogbg-molhiv
Off-line Computation (GDD Computation)	FGW Pairwise distance	15.44	19.67	283.99
	Label-informed pairwise distance	0.12	0.17	53.82
On-line Computation	GREAT (Algorithm 3)	0.43	0.84	295.42
	LAVA	0.06	0.13	39.45
GNN Training Time	GCN (w/ 10% data)	12.76	13.91	190.34
	GCN (w/ 20% data)	13.08	22.59	312.08
	GCN (w/ 50% data)	22.21	30.48	845.46

Table 14: **Empirical run-time behavior for OGB datasets (in seconds).** We can observe that the off-line procedures can be run comparable to a single GNN training time and the on-line procedure has a negligible runtime compared to GNN training. In addition, we can achieve significantly better performance compared to LAVA with slight additional on-line runtime.

the online runtime is ignorable compared to typical GNN training on full dataset. One possible mitigation is to do data clustering using FGW distance before running our main algorithm GRADATE to avoid computational overhead.

3. *How to select the optimal amount of data?* We demonstrate that in Section 4.5, the relationship between selection ratio and GNN adaptation performance is not always trivial across different settings. To ease the comparison pipeline, we fix to some target selection ratios (i.e. 10%, 20%, 50%) for our main experiments, but we acknowledge that these ratios might not yield the best adaptation performance or serve as the best indicator to comparison across different methods. Thus, one potential extension of our method is to automate the process of selecting the optimal selection ratios when dealing different levels of domain shifts.
4. *How to extend GRADATE to node-level graph domain adaption setting?* One possible extension to solve node-level tasks is as follows: decomposing source/target graphs into set of ego-graphs where each of these ego-graphs represent the local topology of each node, and applying GRADATE directly to select the optimal nodes for adaptation. Future directions that require more investigation include (i) how to decide the size of local vicinity for each ego-graph, (ii) how to co-consider node/edge selection for optimal adaptation and (iii) how to mitigate the information loss when extracting ego-graphs.

O Impact Statement

This paper discusses the advancement of the field of Graph Machine Learning. While there are potential societal consequence of our work, none of which we feel must be highlighted .

P ECDF Plots of Different Covariate Shift Settings

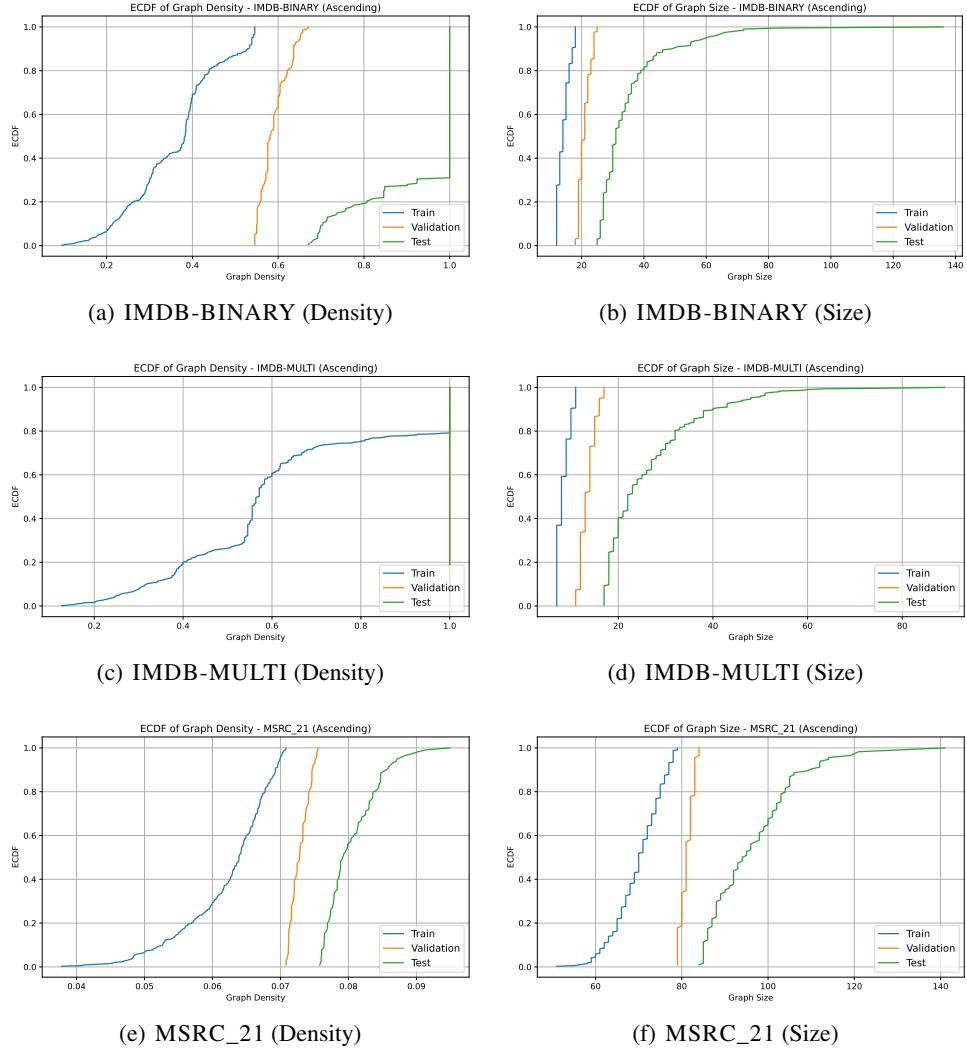


Figure 1: ECDF plots of graph density and size for IMDB-BINARY, IMDB-MULTI, and MSRC_21 datasets. The Blue, Orange, and Green curves represent the distributions of the training, validation, and test splits, respectively. Graphs are sorted in the ascending order by the specified shift (density or size).

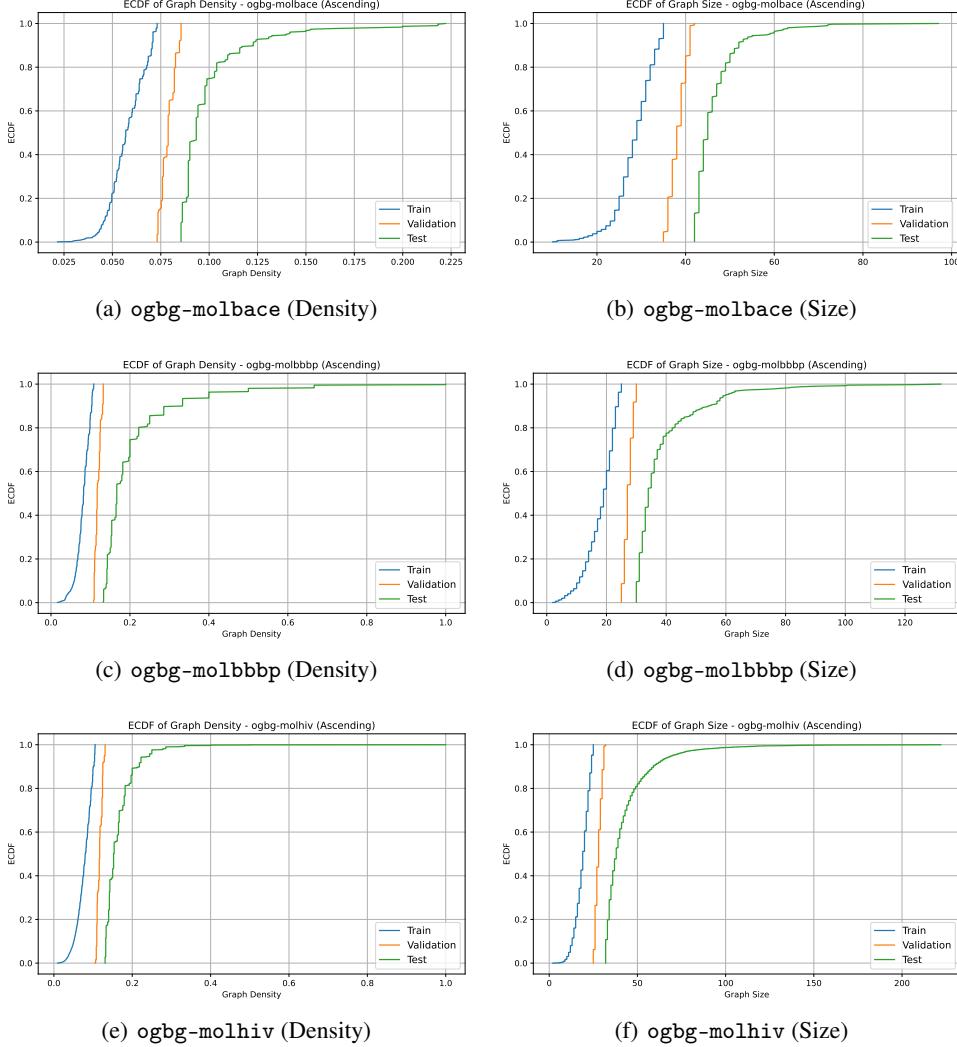


Figure 2: ECDF plots of graph density and size for ogbg-molbbbp, ogbg-molbace, and ogbg-molhiv datasets. The Blue, Orange, and Green curves represent the distributions of the training, validation, and test splits, respectively. Graphs are sorted in ascending order by the specified shift (density or size).