Model-Agnostic Graph Dataset Compression with the Tree Mover's Distance

Anonymous Authors[1]

Abstract

Graph neural networks have demonstrated remarkable success across a variety of domains. However, the acquisition and management of large-scale graph datasets poses several challenges. Acquiring graph-level labels can be prohibitively costly, especially for applications in the biosciences and combinatorial optimization. Storage and privacy constraints can pose additional challenges. In this work, we propose an approach for data subset selection for graph datasets, which downsamples graphs and nodes based on the Tree Mover's Distance. We provide new efficient methods for computing the TMD in our setting; empirical results showing our approach outperforms other node and graph sampling methods; and theoretical results bounding the decrease in accuracy caused by training on the downsampled graphs. Surprisingly, we find that with our method, we can subsample down to 1% of the number of graphs and 10% of the number of nodes on some datasets, with minimal degradation in model accuracy.

1. Introduction

Graph neural networks (GNNs) are a popular architecture for learning over graph-structured data. Despite their widespread popularity and demonstrated success (Bongini et al., 2022; Zhou et al., 2020; Peng et al., 2021; Fan et al., 2019; Peng et al., 2021; Zhang & Chen, 2018), training GNNs remains challenging because the datasets can be massive, both in terms of the size (number of nodes) of each graph, as well as the number of graphs in the dataset.

Large-scale datasets can pose many challenges to training GNNs. For instance, training may be computationally intractable (Hashemi et al., 2024; Yan et al., 2020; Zeng et al., 2021). Likewise, storing and loading massive graphs into memory can be slow and/or expensive (Huang et al., 2024; Hamilton et al., 2017; Da San Martino et al., 2012) or may raise privacy concerns (Wu et al., 2022). In addition, in many applications, labeling many graphs or labeling graphs with many nodes can be expensive. This is the case, for example, when GNNs are used to model molecular properties for computational drug discovery (Igarashi et al., 2024; Shen et al., 2020; Jiang et al., 2021). GNNs have also been employed to circumvent expensive simulation, e.g., in molecular dynamics (Gasteiger et al., 2021; Park et al., 2021; Li et al., 2022), where the cost of labeling larger instances grows quickly with graph size. Similarly, GNNs have been applied for disease prediction (Sun et al., 2020; Zheng et al., 2022), in which case labeling a single graph may require expensive clinical or diagnostic procedures. Additionally, there is growing interest in using GNNs to solve challenging or costly optimization problems, in which case the runtime to collect training labels grows with the graph size (e.g., Bengio et al., 2021; Cappart et al., 2023; Vesselinova et al., 2020).

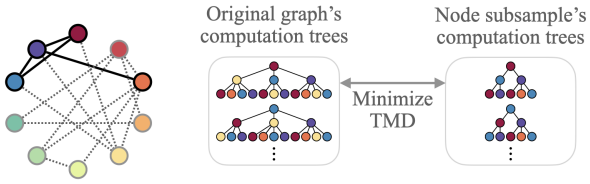

Figure 1. Node subsampling. Given node budget k, we want to find a subgraph of k nodes with minimum TMD from the original.

Subsampling is a natural approach to tackle these multifaceted challenges. In node and graph subsampling, we aim to reduce the number of nodes or graphs, respectively, in a graph dataset while preserving the performance of a downstream GNN trained on the dataset. Existing work on node and graph subsampling focuses primarily on node-level tasks (e.g., node classification), and often makes strong assumptions about the GNN model architecture—and can even assume access to the final model or the ability to approximate the training trajectories on the original, unsampled dataset.

In this paper, we propose and theoretically motivate new approaches for (1) subsampling nodes from each training graph and (2) subsampling entire graphs in the training set for graph-level learning tasks (e.g., graph classification

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Submitted to the Workshop on Advancing Neural Network Training at International Conference on Machine Learning (WANT@ICML 2024). Do not distribute.

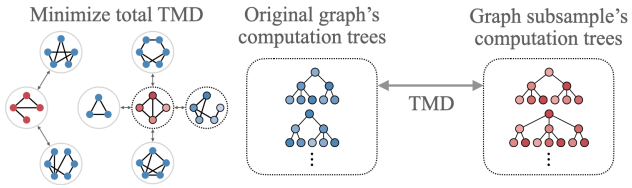

Figure 2. Graph subsampling. Given a graph budget k, the goal is to find a sample of k graphs that minimizes the total distance (TMD) from every original graph to its closest sample.

and regression). We must ensure that training a GNN on the subsampled data does not significantly reduce accuracy compared to the original training set. Moreover, in practice, we often do not know *a priori* (i.e., before collecting, compressing, and labeling our graph dataset) the architecture or hyperparameters of the GNN. Consequently, it is crucial that the original and subsampled datasets maintain *model-agnostic similarity*, i.e., they produce similar model behavior regardless of the downstream model.

For many GNNs, specific *structural* similarities—tied to the GNN's computations—are essential to successfully transfer a GNN out of distribution (Yehudai et al., 2021; Chuang & Jegelka, 2022; Ruiz et al., 2021; Levie et al., 2021; Le & Jegelka, 2023). A GNN uses *message-passing* to compute predictions about graphs. Initially, each node has an embedding, which it updates by aggregating its neighbors' embeddings and processing the result through a neural network. This process repeats over L iterations—or *layers*—and the nodes' final embeddings are used to compute the GNN's predictions. The L-hop neighbors involved in a node's final embedding can be visualized via the node's *computation tree*, illustrated in Figure 3. Computation trees play a critical role in the transfer and stability of GNNs (Yehudai et al., 2021; Chuang & Jegelka, 2022; Georgiev et al., 2022). Two nodes with similar computation trees will likely have similar embeddings. To capture this intuition, Chuang & Jegelka (2022) introduced a graph distance—the *Tree Mover's Distance (TMD)*—as an optimal transport distance between sets of computation trees, and demonstrated its correlation with the performance and stability of GNNs under data perturbations, including generalization bounds.

Based on the strong theoretical foundations provided by TMD, our primary goal is to ensure that the TMD between the original and subsampled graphs is small. However, achieving this goal requires computing the TMD between many graphs while subsampling, which poses a significant challenge. TMD is a combinatorially complex function: the dynamic programming algorithm developed in prior work for computing the TMD between graphs with n nodes has a runtime of $\mathcal{O}(Ln^4)$ (Chuang & Jegelka, 2022). Given our objective to reduce the size of large graphs, this runtime is prohibitively large and presents a computational challenge that we overcome in our methodological contributions.

Contributions. In this work, we make the following contributions: We present an approach for node subsampling that, given a budget k and a training graph, returns a subgraph on k nodes with low TMD from the original graph (Figure 1). Moreover, we present an approach for graph subsampling that, given a training set of graphs, returns k graphs such that the total TMD between each original graph and its closest sampled graph is small (Figure 2).

We show that if h is a GNN with minimal loss on the training set subsampled using our methods, then h also has minimal loss on the original training set. When subsampling nodes in each training graph, we bound the resulting increase in loss by the average TMD between the original and subsampled graphs. When subsampling entire graphs, we bound the increase in loss by the average TMD between each training graph and the nearest graph in the subsampled set.

Given a set of candidate subgraphs, we develop a faster algorithm for selecting a subgraph with minimum TMD from a given graph. It computes the depth-L TMD between a graph $G = (V, E)$ and any of its subgraphs in just $\mathcal{O}(L|E|)$ time. This algorithm leverages structural properties of the TMD that we uncover, which may be of independent interest.

Our experiments demonstrate our methods outperform or perform competitively with other standard approaches on real-world graph learning benchmarks. For graph sampling, our method consistently outperforms the baselines on four benchmark datasets, achieving over 90% of the test accuracy of the full training set for three of the datasets using 1% of the graphs. For node subsampling, it is consistently optimal or near-optimal across all datasets, unlike other competitive baselines, whose performance varies significantly with the fraction of deleted data.

2. Related work

Graph reduction, graph sparsification, and graph condensation approaches aim to to reduce the size of a graph while minimizing the impact on test loss. Existing approaches focus on node subsampling (e.g., Hashemi et al., 2024; Tanaka et al., 2020, and references therein), primarily for *node classification* and graph signal processing. These approaches involve subsampling nodes, and do not extend to graph subsampling, a critical focus of this work, and a setting which remains largely unstudied in the graph reduction literature. Two existing approaches to *model-agnostic* graph reduction for graph classification are Herding (Welling, 2009) and k-Centers (Farahani & Hekmatfar, 2009), which subsample k nodes by clustering the nodes' features into k clusters and selecting the centers. A key advantage of our approach is that, unlike these approaches, we leverage for the node features *and* the graph structure.

Gradient-matching approaches (Jin et al., 2022; Zhang et al., 2024; Fang et al., 2024) have been used for graph dataset subsampling and ensure that the loss's gradients remain similar when trained on the small and large graphs. While these are powerful methods that can aid in developing reduced graph datasets that are smaller and more interpretable, they are fundamentally incomparable in our setting because (1) they assume knowledge of a downstream GNN architecture and therefore are not *model-agnostic*; and (2) they require training on the complete, unreduced graphs to generate the gradient trajectories. This obviates several of the advantages of graph reduction with regard to labeling, training, and storage costs.

Georgiev et al. (2022) uses a normalized variant of TMD to select training datasets for neural algorithmic reasoning tasks. However, this is not agnostic to the downstream model and task and is not directly applicable to graph or node subsampling.

3. Notation and background

Notation. An undirected, unweighted graph $G = (V, E, f)$ has a node set V, edges E, and node features $f \in \mathbb{R}^{p\times|V|}$, where f^v is the feature vector of $v \in V$. We use $N_G(v)$ to denote the neighborhood of $v \in V$ and $\mathcal{G}$ to denote a set of graphs. An r-rooted tree is denoted by $T = (V, E, f, r)$. We use $\mathbf{1}, \mathbf{0}, \boldsymbol{I}$ to denote the all ones and zeros vectors and identity matrix, respectively. For norms, $\|\cdot\|$ denotes any norm over $\mathbb{R}^p$ and $\|\cdot\|_1$ is the ℓ_1 norm.

Message-passing neural networks (GNNs). We analyze *message-passing graph neural networks (GNNs)*, which operate on graphs $G = (V, E, f)$. For simplicity, we define *Graph Isomorphism Networks (GINs)* (Xu et al., 2019) here; however, the results of Chuang & Jegelka (2022) and, therefore, our results also apply to other GNNs. A GIN is defined by L (learnable) functions $\phi^{(\ell)} : \mathbb{R}^d \to \mathbb{R}^d$, such as (shallow) feed-forward neural networks, and a fixed weight $\eta > 0$. Each node begins with an initial embedding $z_v^{(0)} = f^v$ and updates its embedding using the following message-passing routine:

Message passing layers: for $\ell \in [L-1]$,

$$z_v^{(\ell)} = \phi^{(\ell)}\left(z_v^{(\ell-1)} + \eta \sum\nolimits_{u\in N(v)} z_u^{(\ell-1)}\right),$$

Graph readout: $h(G) = \phi^{(L)}\left(\sum_{v\in V} z_u^{(L-1)}\right)$.

Other GNN architectures may, for instance, replace the summations with an average (as in the case of Graph Convolutional Neural Networks (GCNs) (Kipf & Welling, 2016)) or other aggregation functions (Hamilton, 2020).

Tree Mover's Distance (TMD). The TMD was introduced by Chuang & Jegelka (2022) and has found applications for graph learning under distribution shifts (Yu et al., 2023; Georgiev et al., 2022) and adversarially robust graph learning (Schuchardt et al., 2024). TMD is a pseudometric on graphs based on the optimal transport distance. Like a message passing GNN, it views a graph as a set of *computation trees*. A node's computation tree is constructed by adding the node's neighbors to the tree level by level, as in Figures 1 and 3.

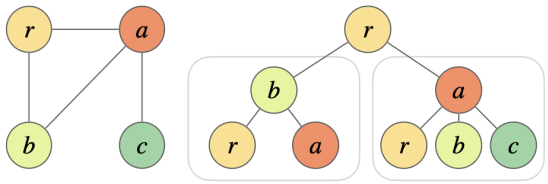

Figure 3. A graph and the depth-2 computation tree T of node r. The set $\mathcal{T}_r(T)$ consists of the two boxed subtrees.

Definition 3.1 (Computation tree)**.** Given $G = (V, E, f)$ and a node $v \in V$, let $T_v^1(G) = (v, \emptyset, \{f^v\}, v)$ be a singleton v-rooted tree that consists of only the node v, no edges, and the node feature vector f^v. Inductively, we define $T_v^L(G)$ to be the depth-L computation tree of node v. That is, $T_v^L(G)$ is a tree rooted at v obtained as follows. For each leaf ℓ of $T_v^{L-1}(G)$ and each neighbor $u \in N_G(\ell)$, we add a new node ℓ_u to $T_v^L(G)$ and add an edge from ℓ to ℓ_u. The multiset of depth-L computation trees defined by G is denoted by $\mathcal{T}_G^L := \{T_v^L(G)\}_{v\in V}$.

We can compare graphs by comparing their computation trees. TMD does this via optimal transport.

Definition 3.2 (OT distance)**.** Let $X = \{x_i\}_{i=1}^q$ and $Y = \{y_i\}_{i=1}^q$ be two multisets of q elements each. Given a distance metric $d : X \times Y \to \mathbb{R}$, let $C \in \mathbb{R}^{q\times q}$ be a matrix where $C_{i,j} = d(x_i, y_j)$ is the *transportation cost* between x_i and y_j. Moreover, let the set of all *transportation plans* between X and Y be $\Gamma(X, Y) := \{\gamma \in \mathbb{R}_+^{q\times q} \mid \gamma\mathbf{1} = \gamma^\top\mathbf{1} = \mathbf{1}\}$. The *transportation cost* of $\gamma \in \Gamma$ is $\langle\gamma, C\rangle := \sum_{i,j}\gamma_{i,j}C_{i,j}$. The (unnormalized) Wasserstein distance OT_d is defined as $\mathrm{OT}_d(X, Y) := q\min_{\gamma\in\Gamma(X,Y)}\langle C, \gamma\rangle$. We say $\gamma^\star$ *is an optimal transport (OT) plan* if $\gamma^\star \in \operatorname{argmin}_{\gamma\in\Gamma(X,Y)}\langle C, \gamma\rangle$.

We need a metric between trees to compute the OT distance between sets of computation trees. TMD compares two trees T_v, T_u by comparing their roots f_v, f_u and recursively comparing their subtrees.

Definition 3.3 (Tree multisets)**.** Given an r-rooted tree $T = (V, E, f, r)$, let $\ell = \mathrm{depth}(T)$. We use $\mathcal{T}_r(T)$ to denote the multiset of depth $(\ell - 1)$ computation trees of all neighbors $u \in N_T(r)$, illustrated in Figure 3. (In other words, if T_u is the subtree of T rooted at u, then

$\mathcal{T}_r(T) := \cup_{u \in N_T(r)} T_u^{(\ell-1)}(T_u)$, where the union denotes a multiset union).

TMD compares $\mathcal{T}_u(T)$ and $\mathcal{T}_v(T')$ via an OT distance. However, the number of subtrees could differ in $\mathcal{T}_u(T)$ and $\mathcal{T}_v(T')$. Chuang & Jegelka (2022) therefore augment the smaller set with *blank trees*. A blank tree $T_\mathbf{0}$ consists of a single node with node features $\mathbf{0} \in \mathbb{R}^d$ and no edges. Given two tree multisets $\mathcal{T}_v(T')$ and $\mathcal{T}_u(T)$, the function $\rho(\mathcal{T}_v(T'), \mathcal{T}_u(T))$ returns two multisets of the same size, where the smaller is padded with blank trees (see Appendix A.3 for more details).

We are now ready to define a distance metric between trees.

Definition 3.4 (Tree distance)**.** Let $T_a = (V_a, E_a, f_a, r_a)$ and $T_b = (V_b, E_b, f_b, r_b)$ be two trees with $\ell := \max(\text{depth}(T_a), \text{depth}(T_b))$. Moreover, let $w : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ be a depth-dependent weighting function. The *tree distance* (TD) between T_a and T_b is defined as

$$\text{TD}_w(T_a, T_b) := \|f_a^{r_a} - f_b^{r_b}\| + \begin{cases} w(\ell) \cdot \text{OT}_{\text{TD}_w}(\rho(\mathcal{T}_{r_a}(T_a), \mathcal{T}_{r_b}(T_b))), & \text{if } \ell > 1 \\ 0, & \text{otherwise.} \end{cases}$$

Here, OT_{TD_w} is the OT distance with metric TD_w. We now define the tree mover's distance (TMD), which calculates the cost of the optimal transport plan between two graphs' computation trees.

Definition 3.5 (Tree mover's distance (TMD) (Chuang & Jegelka, 2022))**.** Given graphs G and G', weight function $w : \mathbb{Z}_{\geq 0} \to \mathbb{R}$, and a depth parameter $L > 0$, we define $\text{TMD}_w^L(G, G') := \text{OT}_{\text{TD}_w}(\rho(\mathcal{T}_G^L, \mathcal{T}_{G'}^L))$. The *tree norm* is $\|G\|_{\text{TN}_w^L} := \text{TMD}_w^L(G, \emptyset)$ where we overload $\emptyset$ to denote the *empty graph*.

4. Node subsampling

In this section, we present an approach to subsampling the nodes of a graph dataset while preserving the performance of a downstream GNN trained on the subsampled dataset. Full proofs for all results in this section can be found in Appendix B. To this end, suppose we are given a graph dataset $X = \{G_1, ..., G_n\}$ where $G_i = (V_i, E_i)$ together with a node budget k. Given a subset of k nodes $S_i \subseteq V_i$, we use $G_i[S_i]$ to denote the subgraph of G_i induced by S_i. Our goal is to select these subsets so that a GNN obtains similar readouts on the original training dataset X and on the induced subgraphs $X' = \{G_1[S_1], ..., G_n[S_n]\}$. Although quantifying the stability of the GNN readouts with respect to node deletion directly is analytically intractable, it is possible to bound the Lipschitz constant of a message-passing GNN's readouts in terms of the TMD.[1]

Theorem 4.1 (Informal, Theorem 8 in (Chuang & Jegelka, 2022), restated)**.** *There exists a weight function w such that given an $(L-1)$-layer message-passing GNN with readout function $h : \mathcal{G} \mapsto \mathbb{R}^d$ and ℓ-th layer Lipschitz constants ϕ_ℓ, for any two graphs G_a, G_b, $\|h(G_a) - h(G_b)\| \leq \left(\prod_{\ell=1}^{L-1} \phi_\ell\right) \cdot \text{TMD}_w^L(G_a, G_b)$.*

Theorem 4.1 implies that if the Lipschitz constants of the GNN layers and the distances $\text{TMD}(G_i[S_i], G_i)$ are small, then minimizing a Lipschitz loss over the subsampled training set X' yields a nearly optimal hypothesis over X. The following corollary summarizes this observation.

Corollary 4.2. *Let $\mathcal{H}$ be a hypothesis class of $(L-1)$-layer GNNs $h : \mathcal{G} \to \mathbb{R}^d$ with ℓ-th layer Lipschitz constant upper-bounded by Φ_ℓ. Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be an M-Lipschitz loss function. Let $X = (G_1, ..., G_n)$ and $(y_1, ..., y_n)$ be a set of (training) graphs and their labels. For each $i \in [n]$, let $G_i = (V_i, E_i)$ and $S_i \subset V_i$. Suppose that $M\left(\prod_{\ell \in [L-1]} \Phi_\ell\right) \leq c$ and $\text{TMD}_w^L(G_i, G_i[S_i]) \leq \varepsilon_i$ for all $i \in [n]$. Finally, let $\hat{h} \in \mathcal{H}$ be a hypothesis with minimum loss over the subsampled training set:*

$$\hat{h} = \text{argmin}_{h \in \mathcal{H}} \sum\nolimits_{i \in [n]} \mathcal{L}(h(G_i[S_i]); y_i).$$

Then $\hat{h}$ has nearly optimal loss over the original training set as well:

$$\frac{1}{n} \sum\nolimits_{i \in [n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \min_{h \in \mathcal{H}} \frac{1}{n} \sum\nolimits_{i \in [n]} \mathcal{L}(h(G_i); y_i) + \frac{2c}{n} \sum\nolimits_{i \in [n]} \varepsilon_i.$$

Corollary 4.2 illustrates that the loss incurred by training on X' as opposed to X is bounded by an error proportional to the average of the distances $\text{TMD}_w^L(G_i, G_i[S_i])$. Motivated by this bound, our ideal goal would be to compute the optimal solution to the following optimization problem:

$$\min_{S \subset V : |S| \leq k} \text{TMD}_w^L(G, G[S]). \tag{1}$$

We face two key challenges in computing (approximately) optimal solutions to Equation (1).

Challenge 1: exponentially large feasible set. The first challenge is that the number of candidate subsets $\{S \subset V : |S| \leq k\}$ grows exponentially with k. Therefore, we restrict Equation (1) to an appropriately chosen feasible set $\mathcal{S}$. We discuss a natural heuristic for selecting $\mathcal{S}$ in Section 4.3.

[1] While Theorem 8 as stated in (Chuang & Jegelka, 2022) is specific to GINs, Chuang & Jegelka (2022) generalize it to other GNNs.

Definition 4.3 (Relaxed node subsampling problem)**.** Let $G = (V, E, f)$ be a graph, k be a node budget, and $\mathcal{S} \subset \{S \subset V : |S| \leq k\}$ be a set of candidate node subsets. The *relaxed node subsampling problem* is defined as: $\min_{S \in \mathcal{S}} \text{TMD}_w^L(G, G[S])$.

Challenge 2: computing $\text{TMD}_w^L(G, G[S])$ is computationally expensive. Unfortunately, even this relaxed node subsampling problem is computationally expensive, as the TMD is expensive to compute even for a single candidate $S \in \mathcal{S}$. The original dynamic-programming algorithm for computing $\text{TMD}_w^L(G, G[S])$ requires $\mathcal{O}(L|V|^4)$-time in general (Chuang & Jegelka, 2022) (regardless of $|S|$), bringing the total runtime to $\mathcal{O}(|V|^4 L |\mathcal{S}|)$. Since we aim to reduce the size of large graphs, this runtime is likely prohibitive in practice. To address this challenge, we prove that, surprisingly, Definition 4.3 is equivalent to a much simpler optimization problem which *can* be solved efficiently.

Theorem 4.4. *Let $G = (V, E, f)$ be a graph, k be a node budget, and $\mathcal{S} \subset \{S \subset V : |S| \leq k\}$. Then $S^\star$ is an optimal solution to the* relaxed node subsampling problem *if and only if*

$$S^\star \in \operatorname*{argmax}_{S \in \mathcal{S}; |S|=k} \|G[S]\|_{\text{TN}_w^L}. \tag{2}$$

We prove Theorem 4.4 in Section 4.1. Along the way, we prove several structural properties of the TMD, which may be of independent interest. Finally, since the algorithm of Chuang & Jegelka (2022) would take $\mathcal{O}(L|S|^4)$-time to compute the tree norm of $G[S]$, we provide a new linear-time algorithm for computing tree norms—Algorithm 1—that better leverages the graph's sparsity.

Theorem 4.5. *Given a graph $G = (V, E, f)$, Algorithm 1 computes $\|G\|_{\text{TN}_w^L}$ in $O(|E| L)$ time.*

We discuss the algorithm and proof sketch in Section 4.2; however, we first point out that this implies a more tractable algorithm for the relaxed node subsampling problem.

Remark 4.6. There is an algorithm that solves the relaxed node subsampling problem in $O(k^2 |\mathcal{S}|)$ time. It computes $\|G[S]\|_{\text{TN}_w^L}$ for each $S \subset \mathcal{S}$ and outputs the set S with the largest tree norm.

4.1. Analysis of TMD between graphs and subgraphs

In this section, we sketch our proof of Theorem 4.4. Along the way, we prove several properties of the TMD, which may be of independent interest given the broad applications of TMD to out-of-distribution generalization. Our analysis crucially relies on the following technical lemma.

Lemma 4.1. *Let $w : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ be a weight function and $G = (V, E, f)$ be a graph. Then, the identity transportation plan $\boldsymbol{I}$ that maps $T_v(G)$ to $T_v(G[S])$ for $v \in S$, and $T_u(G)$ to $\emptyset$ for $u \notin S$ is an optimal transport plan for the OT problem*

$$\text{TMD}_w^L(G, G[S]) = \text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_G^L, \mathcal{T}_{G[S]}^L\right)\right). \tag{3}$$

Consequently, we can decompose the right-hand-side of (3) into three terms: a sum over the feature norms of deleted nodes $v \notin S$; the cost of transporting the deleted nodes' computation trees (that is, $\{\mathcal{T}_v(T_v^L(G))\}_{v \notin S}$) to the empty set; and the cost of transporting the computation trees of the remaining nodes $v \in S$ (that is, $\{\mathcal{T}_v(T_v^L(G))\}_{v \in S}$) to their computation trees in $G[S]$. Formally,

$$\begin{aligned}
\text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_G^L, \mathcal{T}_{G[S]}^L\right)\right) &= \sum_{v \notin S} \|f^v\| \\
&+ w(L-1) \sum_{v \notin S} \text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_v\left(T_v^L(G)\right), \emptyset\right)\right) \\
&+ w(L-1) \sum_{v \in S} \text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_v\left(T_v^L(G)\right), \mathcal{T}_v\left(T_v^L(G[S])\right)\right)\right).
\end{aligned}$$

Proof sketch. We prove the statement by induction on L. In the base case ($L = 1$), the statement is equivalent to Definition 3.5. This is because by the definition of ρ and the tree distance TD, $\text{OT}_{\text{TD}_w}(\rho(\mathcal{T}_G^1, \mathcal{T}_{G[S]}^1))$ is equal to the sum of features of the nodes that are in G but not in $G[S]$. That is, $\text{OT}_{\text{TD}_w}(\rho(\mathcal{T}_G^1, \mathcal{T}_{G[S]}^1)) = \sum_{v \notin S} \|f^v\| = \langle C, \boldsymbol{I} \rangle$. This shows that the lemma holds in the base case. We prove the inductive step by using the recursive formulation of the TD to reduce OT problems of depth L to OT problems of depth $L - 1$, which allows us to apply the inductive hypothesis. ■

Using Lemma 4.1, we can precisely characterize the TMD induced by dropping a sequence of nodes from the graph. First, we prove the following chain rule.

Lemma 4.2. *For $T \subset S \subset V$, $\text{TMD}_w^L(G, G[S \setminus T]) = \text{TMD}_w^L(G, G[S]) + \text{TMD}_w^L(G[S], G[S \setminus T])$.*

Proof sketch. To prove this lemma, we apply the expression in Lemma 4.1 inductively over L to decompose $\text{TMD}_w^L(G, G[S \setminus T])$ into $\text{TMD}_w^L(G, G[S])$ and $\text{TMD}_w^L(G[S], G[S \setminus T])$. To highlight the intuition, we discuss the base case here and discuss the inductive step in the full proof. If $L = 1$, then the second and third terms in the expression of Lemma 4.1 are always 0, because $\mathcal{T}_G^1$ and $\mathcal{T}_{G[S]}^1$ are multisets of depth 1, so $\mathcal{T}_G(T_v^L(G))$ and $\mathcal{T}_{G[S]}(T_v^L(G[S]))$ are empty sets. Thus,

$$\begin{aligned}
\text{TMD}_w^1(G, G[S \setminus T]) &= \sum\nolimits_{v \notin (S \setminus T)} \|f^v\| \\
&= \sum\nolimits_{v \notin S} \|f^v\| + \sum_{v \in T} \|f^v\|,
\end{aligned}$$

Algorithm 1: TreeNorm(G, L, w)

Input: Graph $G = (V, E, f)$ with adjacency matrix A, weights $w : \{1, ..., L-1\} \to \mathbb{R}_+$, $L \geq 1$

Compute $x \in \mathbb{R}^{|V|}$ to be the vector such that $x_v = \|f^v\|$;

Initialize $z^{(0)} = x$;

for $\ell \in [L-1]$ **do**

  $z^{(\ell)} \leftarrow Az^{(\ell-1)}$;

$b \leftarrow z^{(0)} + \sum_{\ell=1}^{L-1} \left(\prod_{t=1}^{\ell} w(L-t) \right) \cdot z^{(\ell)}$;

return: $\|b\|_1$

(since $T \subset S$, $\{v \in V : v \notin (S \setminus T)\} = \{v \in V : v \notin S\} \sqcup \{v \in T\}$). Similarly applying Lemma 4.1 to $\text{TMD}_w^L(G, G[S \setminus T])$ and $\text{TMD}_w^1(G[S], G[S \setminus T])$ implies that the base case holds:

$$\text{TMD}_w^1(G, G[S]) = \sum\nolimits_{v \notin S} \|f^v\|, \text{ and}$$

$$\text{TMD}_w^1(G[S], G[S \setminus T]) = \sum\nolimits_{v \in T} \|f^v\|.$$

The intuition is similar for $L > 1$ but requires care to handle the recursive OT terms. ■

Finally, we can prove Theorem 4.4.

Proof of Theorem 4.4. Let $S \in \mathcal{S}$. Applying Lemma 4.2 with $T = V \setminus S$, we have that $\|G\|_{\text{TN}_w^L} = \text{TMD}_w^L(G, G[S]) + \|G[S]\|_{\text{TN}_w^L}$. Consequently,

$$\begin{aligned} &\max_{S \in \mathcal{S}; |S|=k} \|G[S]\|_{\text{TN}_w^L} \\ &\equiv \max_{S \in \mathcal{S}; |S|=k} \|G\|_{\text{TN}_w^L} - \text{TMD}_w^L(G, G[S]) \\ &\equiv \min_{S \in \mathcal{S}; |S|=k} \text{TMD}_w^L(G, G[S]). \end{aligned}$$ ■

4.2. Faster algorithm for computing tree norms

We next leverage Lemma 4.1 to obtain Algorithm 1, a faster algorithm for computing $\|G\|_{\text{TN}_w^L}$ for $G = (V, E, f)$. The original implementation would require $O(L|V|^4)$ time (Chuang & Jegelka, 2022), whereas ours has runtime $O(L|E|)$. Algorithm 1 is based on the fact that by Lemma 4.1, is that $\|G\|_{\text{TN}_w^L}$ is essentially a weighted sum of the number of vertices in all of its L-hop computation trees. For each tree, nodes in level ℓ receive weight $w(L - \ell + 1)$. Algorithm 1 computes this weighted sum. Its runtime is dominated by the cost of L matrix-vector multiplies with the adjacency matrix, which requires $O(|E|)$-time. The proof of correctness and runtime (Theorem 4.5) is in Appendix B.

4.3. Heuristic for selecting $\mathcal{S}$

We now present our heuristic for constructing the set of candidate subsets $\mathcal{S}$ in our experiments. Most real-world networks are scale-free and small-world, and hence can be well-modeled by random graphs, such as preferential attachment, configuration models, and inhomogeneous random graphs (Bollobás & Riordan, 2004; Newman & Watts, 1999). These graphs converge *locally* to graph limits (Van Der Hofstad, 2024, Vol. 2, Ch. 2). By a "transitive" argument, we can view real-world networks as parts of sequences converging to graph limits in the same sense. Alimohammadi et al. (2023) showed that for such graphs, sampling nodes' local breadth-first search (BFS) trees asymptotically preserves critical motifs of the graph limit (see Appendix A). Thus, we define the set of candidate subsets $\mathcal{S}$ as a set of BFS trees rooted at the graph's nodes.

Definition 4.7 (k-BFS subset)**.** Given $G = (V, E, f)$ and $v \in V$, let ℓ_k be the deepest level such that the v-rooted BFS tree of depth ℓ_k has at most k nodes (breaking ties in a fixed but arbitrary way). The k-BFS subset of v, denoted $S_{\text{BFS}(v;k)}$, is the set of nodes at distance at most ℓ_k from v.

We define the set of candidate subsets $\mathcal{S} = \{S_{\text{BFS}(v;k)}\}_{v \in V}$ as the nodes' k-BFS subsets.

5. Graph subsampling

We now present an approach that uses TMD to subsample the number of graphs in a dataset while preserving the performance of a downstream GNN trained on the subsampled dataset. Omitted proofs are in Section B. Let $X = \{G_1, ..., G_n\}$ be a graph dataset. Given a budget k, the goal is to identify a subset $\mathcal{I} \subset [n]$ of size k such that a GNN obtains similar readouts and loss on the original training set X and on the subsampled set $\{G_i\}_{i \in \mathcal{I}}$. We must ensure that the selected set $\mathcal{I}$ is representative of the full dataset. To this end, we select $\mathcal{I}$ by optimizing the medoids objective, which captures how well each graph $G_i \in X$ is represented by $\mathcal{I}$.

Definition 5.1 (Medoids objective)**.** Let $X = \{G_1, ..., G_n\}$ be a graph dataset and $\mathcal{I} \subset [n]$ with $|\mathcal{I}| \leq k$. The medoids objective value of $\mathcal{I}$ on X with respect to TMD_w^L is defined

$$f_{\text{TMD}_w^L}(\mathcal{I}; X) = \frac{1}{|X|} \sum\nolimits_{i \in [n]} \min_{j \in \mathcal{I}} \text{TMD}_w^L(G_i, G_j).$$

For each $j \in \mathcal{I}$, let τ_j be the number of graphs in X closest to G_j (i.e., the medoid's cluster size): $\tau_j := \left| \{i \in [n] : \text{TMD}_w^L(G_i, G_j) < \text{TMD}_w^L(G_i, G_k),\ \forall k \neq j \in \mathcal{I}\} \right|$, breaking ties arbitrarily.

The following lemma shows how we can bound the difference in a GNN's loss on the subsampled dataset $\mathcal{I} \subset X$ and the original dataset X in terms of this medoids objective.

Lemma 5.1. *Let $\mathcal{H}$ be a hypothesis class of $(L-1)$-layer GNNs $h : \mathcal{G} \to \mathbb{R}^d$ with ℓ-th layer Lipschitz constant upper-bounded by Φ_ℓ. Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be an M-Lipschitz loss*

function. Let $y = (y_1, ..., y_n)$ be labels for X and $\mathcal{I}$ be a subset of $[n]$. Then for any GNN $h \in \mathcal{H}$ and $G \in \mathcal{G}$,

$$\frac{1}{n}\Big|\sum_{i\in\mathcal{I}} \tau_i \mathcal{L}(h(G_i); y_i) - \sum_{i\in[n]} \mathcal{L}(h(G_i); y_i)\Big| \leq \quad (4)$$

$$M\left(\prod_{\ell\in[L-1]} \Phi_\ell\right) f_{\mathrm{TMD}^L_w}(\mathcal{I}; X). \quad (5)$$

Proof sketch. We use the Lipschitz constant M of $\mathcal{L}$ and Theorem 4.1 to bound the average deviation in the loss on X and the (weighted) loss on $\mathcal{I}$. ■

Lemma 5.1 implies that if the right-hand-side of Equation (5) is small and we minimize loss over the subsampled training set (the left-most term in Equation (5)), we will obtain a hypothesis with nearly optimal loss over the entire training set. We summarize this observation as follows.

Corollary 5.2. *Suppose that $M\left(\prod_{\ell\in[L-1]} \Phi_\ell\right) \leq c$ and $f_{\mathrm{TMD}^L_w}(\mathcal{I}; X) \leq \varepsilon$. Let $\hat{h} \in \mathcal{H}$ be the hypothesis that minimizes loss on the subsampled graphs $\sum_{i\in\mathcal{I}} \tau_i \mathcal{L}(h(G_i); y_i)$. Then*

$$\frac{1}{n}\sum_{i\in[n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_i); y_i) + 2c\varepsilon.$$

Corollary 5.2 illustrates that the additional loss incurred by training on the subsampled graphs as opposed to X is bounded by an error proportional to $f_{\mathrm{TMD}^L_w}(\mathcal{I}; X)$. This motivates us to formulate the graph subsampling problem as the problem of computing $\mathcal{I}$ that minimizes $f_{\mathrm{TMD}^L_w}(\mathcal{I}; X)$.

Definition 5.3 (Graph subsampling problem)**.** Let $X = \{G_1, ..., G_n\}$ be a set of graphs and k be a graph budget. In the graph *graph subsampling problem*, we must select a subset $\mathcal{I} \subset X$ of k graphs that minimizes the k-medoids objective value of $f_{\mathrm{TMD}^L_w}(\mathcal{I}; X)$.

The k-medoids clustering problem is NP-hard (Kazakovtsev & Rozhnov, 2020), but there are efficient approximation algorithms. We use Python's k-medoids implementation in the sklearn package (Buitinck et al., 2013).

6. Experiments

We evaluate our sampling methods on four graph benchmark datasets from the TUDataset graph benchmark library (Morris et al., 2020). All GNNs were trained for 50 epochs with an 80%-train set and 20%-test split on CPUs. For MUTAG, COX2, and BZX we used a 3-layer Graph Isomorphism Network (Xu et al., 2019). For IMDB-BINARY we used a 3-layer Graph Convolutional Neural Network (GCN) (Kipf & Welling, 2016). Hyper-parameter details are in Appendix A.4. In all figures, the line plots and shadows indicate the means and 95% confidence intervals across 256 random trials. Figure 4 compares our TMD-based k-medoids graph subsampling procedure from Section 5 against several alternative benchmarks and against training on the full dataset.

We are unaware of any previous work that studies graph subsampling, but we nonetheless compare against two clustering approaches based on Herding (Welling, 2009) and k-Centers (Farahani & Hekmatfar, 2009). These approaches are typically used for node subsampling and coreset selection in other domains (see Section 2). We adapt them to graph subsampling by defining distances between graphs using three popular graph kernels: Weisfeiler Lehman kernel (WL) (Shervashidze et al., 2011), the WL optimal assignment kernel (WL-OT) (Kriege et al., 2016), graphlet sampling (GS) kernel (Borgwardt & Kriegel, 2005), and the shortest paths (SP) kernel. Figure 4 shows that our method consistently outperforms uniform random sampling, Herding (with respect to all kernels), and k-Centers (with respect to all kernels) on all four benchmarks. Our approach achieves over 90% of the test accuracy of the full training set for MUTAG, COX2, and BZR using just 1% of the graphs. Where labeling is costly, this could be a powerful approach.

Node subsampling. Figure 5 compares our TMD-based node subsampling procedure from Section 4 against several alternative benchmarks and against training on the full dataset. For node subsampling, we only need to compute tree norms, so we use Algorithm 1. We compare against model-agnostic node selection policies Herding (Welling, 2009) and k-Centers (Farahani & Hekmatfar, 2009), and approaches by Salha et al. (2022) and Razin et al. (2023). While other approaches exhibit significant variability across datasets and subsampling levels, our approach consistently has the strongest or near-strongest performance across all datasets, sometimes requiring as little as 10% of the original number of nodes to achieve similar test accuracy.

7. Discussion

We proposed new approaches for compressing graph datasets that preserve the performance of a downstream GNN. Our methods apply to both node subsampling and graph subsampling. By leveraging the TMD, our methods ensure that the subsampled data maintains structural similarities with the original dataset, and we could bound the loss incurred by subsampling. Our methods are enabled by an algorithm we developed for selecting subgraphs with minimal TMD from a given graph, providing significant runtime speed-up compared to prior work. Our experiments demonstrated that our method outperforms comparable methods.

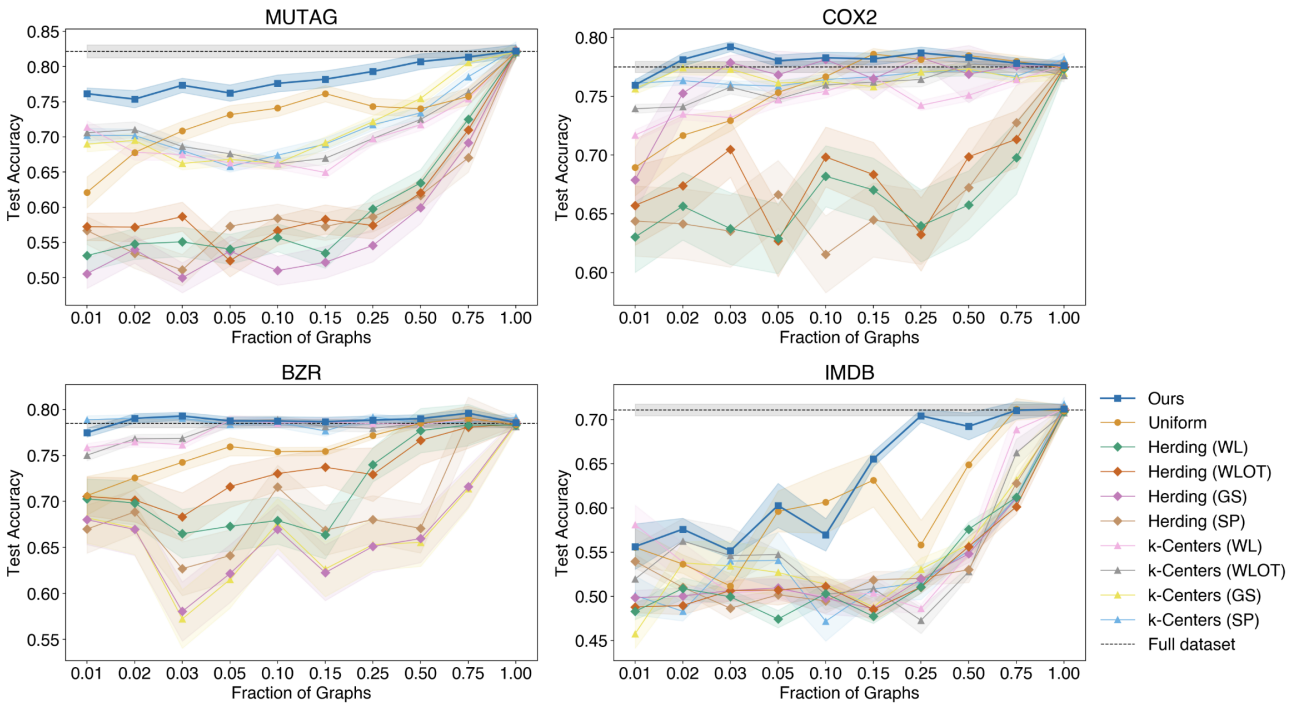

Figure 4. Graph subsampling. Accuracy vs. fraction of graphs in the training set, subsampled with our approach and alternative methods.

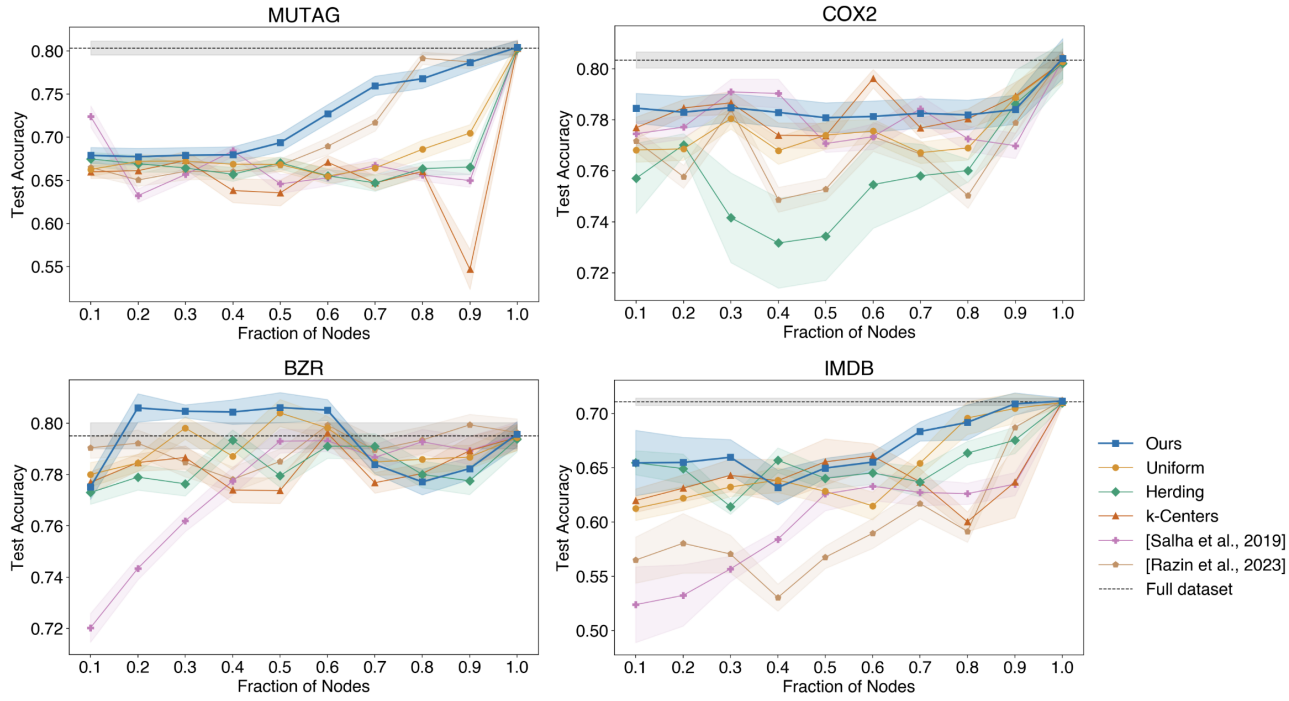

Figure 5. Node subsampling. Accuracy vs. fraction of nodes in the training set, subsampled with our approach and existing model-agnostic methods.

References

Alimohammadi, Y., Ruiz, L., and Saberi, A. A local graph limits perspective on sampling-based gnns. *arXiv preprint arXiv:2310.10953*, 2023.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Bollobás, B. and Riordan, O. Coupling scale-free and classical random graphs. *Internet Mathematics*, 1(2):215–225, 2004.

Bongini, P., Pancino, N., Scarselli, F., and Bianchini, M. Biognn: how graph neural networks can solve biological problems. In *Artificial Intelligence and Machine Learning for Healthcare*, pp. 211–231. Springer, 2022.

Borgwardt, K. M. and Kriegel, H.-P. Shortest-path kernels on graphs. pp. 8–pp. IEEE, 2005.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Veličković, P. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Chuang, C.-Y. and Jegelka, S. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 2944–2957, 2022.

Da San Martino, G., Navarin, N., and Sperduti, A. A memory efficient graph kernel. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *Proceedings of the International World Wide Web Conference (WWW)*, pp. 417–426, 2019.

Fang, J., Li, X., Sui, Y., Gao, Y., Zhang, G., Wang, K., Wang, X., and He, X. Exgc: Bridging efficiency and explainability in graph condensation. In *Proceedings of the International World Wide Web Conference (WWW)*, 2024.

Farahani, R. Z. and Hekmatfar, M. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Gasteiger, J., Becker, F., and Günnemann, S. Gemnet: Universal directional graph neural networks for molecules. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Georgiev, D., Lio, P., Bachurski, J., Chen, J., Shi, T., and Giusti, L. Beyond Erdos-Renyi: Generalization in algorithmic reasoning on graphs. In *Learning on Graphs (LoG)*, 2022.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. 30, 2017.

Hamilton, W. L. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

Hashemi, M., Gong, S., Ni, J., Fan, W., Prakash, B. A., and Jin, W. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *arXiv preprint arXiv:2402.03358*, 2024.

He, Y., Zhang, X., Huang, J., Rozemberczki, B., Cucuringu, M., and Reinert, G. Pytorch geometric signed directed: A software package on graph neural networks for signed and directed graphs. In *Learning on Graphs Conference*, pp. 12–1. PMLR, 2024.

Huang, K., Jiang, H., Wang, M., Xiao, G., Wipf, D., Song, X., Gan, Q., Huang, Z., Zhai, J., and Zhang, Z. Freshgnn: Reducing memory access via stable historical embeddings for graph neural network training. 17(6):1473–1486, 2024.

Igarashi, Y., Kojima, R., Matsumoto, S., Iwata, H., Okuno, Y., and Yamada, H. Developing a gnn-based ai model to predict mitochondrial toxicity using the bagging method. 49(3):117–126, 2024.

Jiang, D., Wu, Z., Hsieh, C.-Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., and Hou, T. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. 13:1–23, 2021.

Jin, W., Tang, X., Jiang, H., Li, Z., Zhang, D., Tang, J., and Yin, B. Condensing graphs via one-step gradient matching. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 720–730, 2022.

KAWANO, K., KOIDE, S., SHIOKAWA, H., and AMAGASA, T. Multi-dimensional fused gromov wasserstein discrepancy for edge-attributed graphs. pp. 683–693, 2024.

Kazakovtsev, L. A. and Rozhnov, I. Application of algorithms with variable greedy heuristics for k-medoids problems. *Informatica*, 44(1), 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. 2016.

Kriege, N. M., Giscard, P.-L., and Wilson, R. On valid optimal assignment kernels and applications to graph classification. 29, 2016.

Le, T. and Jegelka, S. Limits, approximation and size transferability for gnns on sparse graphs via graphops. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Levie, R., Huang, W., Bucci, L., Bronstein, M. M., and Kutyniok, G. Transferability of spectral graph convolutional neural networks. *Journal of Machine Learning Research*, 2021.

Li, Z., Meidani, K., Yadav, P., and Farimani, A. B. Graph neural networks accelerated molecular dynamics. *J. Chem. Phys.*, 156, 2022.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Newman, M. E. and Watts, D. J. Scaling and percolation in the small-world network model. *Physical review E*, 60 (6):7332, 1999.

Park, C. W., Kornbluth, M., Vandermause, J., Wolverton, C., Kozinsky, B., and Mailoa, J. P. Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture. *npj Computational Materials*, 7(73), 2021.

Peng, Y., Choi, B., and Xu, J. Graph learning for combinatorial optimization: a survey of state-of-the-art. pp. 119–141, 2021.

Razin, N., Verbin, T., and Cohen, N. On the ability of graph neural networks to model interactions between vertices. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Ruiz, L., Gama, F., and Ribeiro, A. Graph neural networks: Architectures, stability, and transferability. *Proceedings of the IEEE*, 109(5):660–682, 2021.

Salha, G., Hennequin, R., Tran, V. A., and Vazirgiannis, M. A degeneracy framework for scalable graph autoencoders, 2022.

Schuchardt, J., Scholten, Y., and Günnemann, S. (Provable) adversarial robustness for group equivariant tasks: Graphs, point clouds, molecules, and more. 2024.

Shen, X., Liu, Y., Wu, Y., and Xie, L. Molgnn: Self-supervised motif learning graph neural network for drug discovery. In *Machine Learning for Molecules Workshop at NeurIPS*, pp. 4, 2020.

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

Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., and Vazirgiannis, M. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

Sun, Z., Yin, H., Chen, H., Chen, T., Cui, L., and Yang, F. Disease prediction via graph neural networks. 25(3): 818–826, 2020.

Tanaka, Y., Eldar, Y. C., Ortega, A., and Cheung, G. Sampling signals on graphs: From theory to applications. *IEEE Signal Processing Magazine*, 37(6):14–30, 2020.

Van Der Hofstad, R. *Random graphs and complex networks*. Cambridge University Press, 2024.

Vesselinova, N., Steinert, R., Perez-Ramirez, D. F., and Boman, M. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 8:120388–120416, 2020.

Welling, M. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1121–1128, 2009.

Wu, B., Li, J., Yu, J., Bian, Y., Zhang, H., Chen, C., Hou, C., Fu, G., Chen, L., Xu, T., et al. A survey of trustworthy graph learning: Reliability, explainability, and privacy protection. *arXiv preprint arXiv:2205.10014*, 2022.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Yan, B., Wang, C., Guo, G., and Lou, Y. Tinygnn: Learning efficient graph neural networks. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1848–1856, 2020.

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

Yu, J., Liang, J., and He, R. Mind the label shift of augmentation-based graph ood generalization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11620–11630, 2023.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Accurate, efficient and scalable training of graph neural networks. *Journal of Parallel and Distributed Computing*, 147:166–183, 2021.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. 31, 2018.

Zhang, T., Zhang, Y., Wang, K., Wang, K., Yang, B., Zhang, K., Shao, W., Liu, P., Zhou, J. T., and You, Y. Two trades is not baffled: Condense graph via crafting rational gradient matching. *arXiv preprint arXiv:2402.04924*, 2024.

Zheng, S., Zhu, Z., Liu, Z., Guo, Z., Liu, Y., Yang, Y., and Zhao, Y. Multi-modal graph learning for disease prediction. 41(9):2207–2216, 2022.

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

A. Additional background

A.1. Random graph limits

As we discussed in Section 4.3, most real networks are scale-free and small-world, and hence well-modeled by random graph models, such as preferential attachment, configuration models and inhomogenous random graphs. As shown by (Van Der Hofstad, 2024), such random graph models produce graphs that can be shown to converge *locally* to a limit (G, o), where G is a graph to which we assign a root node o. By a "transitive" argument, it makes sense to see real networks as parts of sequences converging to graph limits in a similar sense.

To be precise, let $\mathcal{G}_*$ be the set of all possible rooted graphs. A limit graph is defined as a measure over the space $\mathcal{G}_*$ with respect to the local metric

$$d_{loc}((G_1, o_1), (G_2, o_2)) = \frac{1}{1 + \inf_k\{k : B_k(G_1, o_1) \not\simeq B_k(G_2, o_2)\}}$$

where $B_k(G, v)$ is the k-hop neighborhood of node v, and $\simeq$ is the graph isomorphism. A sequence of graphs converging to this limit is defined as follows.

Definition A.1 (Local convergence (Alimohammadi et al., 2023))**.** Let $G_n = (V_n, E_n)$ denote a finite connected graph. Let (G_n, o_n) be the rooted graph obtained by letting $o_n \in V_n$ be chosen uniformly at random. We say that (G_n, o_n) converges locally to the connected rooted graph (G, o), which is a (possibly random) element of $\mathcal{G}_*$ having law μ, when, for every bounded and continuous function $h : \mathcal{G}_* \to \mathbb{R}$,

$$\mathbb{E}[h(G_n, o_n)] \to \mathbb{E}_\mu(G, o)$$

where the expectation on the right-hand-side is with respect to (G, o) having law μ, while the expectation on the left-hand-side is with respect to the random vertex o_n.

A.2. Other graph kernels

TMD is one recently proposed pseudometric on graphs, or graph kernel (Chuang & Jegelka, 2022). It is appealing in our setting because it characterizes the robustness of GNNs (see Theorem 4.1.) Other popular graph kernels include the Weisfeiler Lehman (WL) kernel (Shervashidze et al., 2011), Weisfeiler Lehman Optimal Transport (WL-OT) (Kriege et al., 2016), Shortest Paths kernel ((Borgwardt & Kriegel, 2005)), and FGW (KAWANO et al., 2024). To our knowledge, these kernels do not have comparable robustness guarantees to TMD.

A.3. Additional details about the TMD

We use $T_{\mathbf{0}}^n$ to denote n disjoint copies of $T_{\mathbf{0}}$. Given two tree multisets $\mathcal{T}_u(T)$, $\mathcal{T}_v(T')$, we define ρ to be the following augmentation function, which returns two multi-sets of the same size:

$$\rho : (\mathcal{T}_v(T'), \mathcal{T}_u(T)) \mapsto \left(\mathcal{T}_v(T') \cup {T_{\mathbf{0}}}^{\max(|\mathcal{T}_u(T)|-|\mathcal{T}_v(T')|,0)}, \mathcal{T}_u(T) \cup {T_{\mathbf{0}}}^{\max(|\mathcal{T}_v(T)|-|\mathcal{T}_u(T')|,0)}\right).$$

A.4. Additional details about experiments

All experiments were run on a 16-core machine with 64 GB of RAM. The GNNs were implemented using Pytorch Geometric (He et al., 2024). The TMD weight function was set according to Pascal's triangle rule (Chuang & Jegelka, 2022, Theorem 8) and our implementation of TMD was based on (Chuang & Jegelka, 2022).[2] We used the Graph Kernel Library (Siglidis et al., 2020) for the kernel distances in our experiments. We will be making the code for all of our experiments publicly available.

For MUTAG, COX2, and BZX we used a GIN model with the following hyperparameters:

- Number of layers: 3
- Learning rate: 0.01
- Batch size: 128

[2]One consideration with is the runtime required to compute the k-medoids. However, reduced labeling cost, reduced data storage, and privacy are often more significant priorities.

- Number of hidden channels per layer: 32
- Aggregation function: global add pool
- Weight regularization: N/A
- Optimizer: Adam
- Loss: BCE with logits
- Dropout: N/A

For IMDB we used a GCN model with the following hyperparameters:

- Number of layers: 3
- Learning rate: 0.01
- Batch size: 32
- Number of hidden channels per layer: 32
- Aggregation function: global add pool
- Weight regularization: N/A
- Optimizer: Adam
- Loss: BCE with logits
- Dropout: N/A

We refrained from using batch normalization in both models. Models were implemented using PytorchGeometric (Fey & Lenssen, 2019).

B. Appendix

Throughout this section, we use $\deg_G(v)$ to denote the degree of $v \in V$.

B.1. Omitted proofs from Section 4

Corollary 4.2. *Let $\mathcal{H}$ be a hypothesis class of $(L-1)$-layer GNNs $h : \mathcal{G} \to \mathbb{R}^d$ with ℓ-th layer Lipschitz constant upper-bounded by Φ_ℓ. Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be an M-Lipschitz loss function. Let $X = (G_1, ..., G_n)$ and $(y_1, ..., y_n)$ be a set of (training) graphs and their labels. For each $i \in [n]$, let $G_i = (V_i, E_i)$ and $S_i \subset V_i$. Suppose that $M\left(\prod_{\ell \in [L-1]} \Phi_\ell\right) \leq c$ and $\mathrm{TMD}^L_w(G_i, G_i[S_i]) \leq \varepsilon_i$ for all $i \in [n]$. Finally, let $\hat{h} \in \mathcal{H}$ be a hypothesis with minimum loss over the subsampled training set:*

$$\hat{h} = \mathrm{argmin}_{h \in \mathcal{H}} \sum\nolimits_{i \in [n]} \mathcal{L}(h(G_i[S_i]); y_i).$$

Then $\hat{h}$ has nearly optimal loss over the original training set as well:

$$\frac{1}{n} \sum\nolimits_{i \in [n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \min_{h \in \mathcal{H}} \frac{1}{n} \sum\nolimits_{i \in [n]} \mathcal{L}(h(G_i); y_i) + \frac{2c}{n} \sum\nolimits_{i \in [n]} \varepsilon_i.$$

Proof. Consider any $h \in \mathcal{H}$. By Theorem 4.1, we have that for each $i \in [n]$,

$$\|h(G_i; y_i) - h(G_i[S_i]; y_i)\| \leq \left(\prod_{\ell \in [L-1]} \Phi_\ell\right) \mathrm{TMD}^L_w(G_i, G_i[S_i]). \tag{6}$$

Consequently, using the fact that $\mathcal{L}$ is M-Lipschitz and (6),

$$\begin{aligned}|\mathcal{L}\left(h(G_i); y_i\right) - \mathcal{L}\left(h(G_i[S_i])\right)| &\leq M \left\|h(G_i; y_i) - h(G_i[S_i]; y_i)\right\| \\ &\leq M \left(\prod_{\ell \in [L-1]} \Phi_\ell\right) \text{TMD}_w^L(G_i, G_i[S_i]) \\ &= c\text{TMD}_w^L(G_i, G_i[S_i]) \leq c\varepsilon_i.\end{aligned}$$

Consequently, by the triangle inequality,

$$\begin{aligned}\left|\frac{1}{n}\sum_{i\in[n]} \mathcal{L}\left(h(G_i); y_i\right) - \frac{1}{n}\sum_{i\in[n]} \mathcal{L}\left(h(G_i[S_i])\right)\right| &\leq \frac{1}{n}\sum_{i\in[n]} |\mathcal{L}\left(h(G_i); y_i\right) - \mathcal{L}\left(h(G_i[S_i]; y_i)\right)| \\ &\leq \frac{c}{n}\sum_{i\in[n]} \varepsilon_i.\end{aligned}$$

So, we know that for any $h \in \mathcal{H}$

$$\frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_i; y_i) \leq \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_i[S_i]; y_i) + \frac{c}{n}\sum_{i\in[n]} \varepsilon_i. \tag{7}$$

Let $h^\star$ be the hypothesis that minimizes average loss across the original dataset X:

$$h^\star = \operatorname*{argmin}_{h\in\mathcal{H}} \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_i); y_i).$$

By definition of $\hat{h}$, this means that

$$\frac{1}{n}\sum_{i\in[n]} \mathcal{L}(\hat{h}(G_i[S_i]; y_i) \leq \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h^\star(G_i[S_i]); y_i) + \frac{c}{n}\sum_{i\in[n]} \varepsilon_i.$$

Applying (7) again to $h^\star$, we have that

$$\frac{1}{n}\sum_{i\in[n]} \mathcal{L}(\hat{h}(G_i[S_i]; y_i) \leq \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h^\star(G_i); y_i) + \frac{c}{n}\sum_{i\in[n]} \varepsilon_i.$$

■

Lemma 4.1. *Let $w : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ be a weight function and $G = (V, E, f)$ be a graph. Then, the identity transportation plan $\boldsymbol{I}$ that maps $T_v(G)$ to $T_v(G[S])$ for $v \in S$, and $T_u(G)$ to $\emptyset$ for $u \notin S$ is an optimal transport plan for the OT problem*

$$\text{TMD}_w^L(G, G[S]) = \text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_G^L, \mathcal{T}_{G[S]}^L\right)\right). \tag{3}$$

Consequently, we can decompose the right-hand-side of (3) into three terms: a sum over the feature norms of deleted nodes $v \notin S$; the cost of transporting the deleted nodes' computation trees (that is, $\{\mathcal{T}_v(T_v^L(G))\}_{v\notin S}$) to the empty set; and the cost of transporting the computation trees of the remaining nodes $v \in S$ (that is, $\{\mathcal{T}_v(T_v^L(G))\}_{v\in S}$) to their computation trees in $G[S]$. Formally,

$$\begin{aligned}&\text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_G^L, \mathcal{T}_{G[S]}^L\right)\right) = \sum_{v\notin S} \|f^v\| \\ &+ w(L-1)\sum_{v\notin S} \text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_v\left(T_v^L(G)\right), \emptyset\right)\right) \\ &+ w(L-1)\sum_{v\in S} \text{OT}_{\text{TD}_w}\left(\rho\left(\mathcal{T}_v\left(T_v^L(G)\right), \mathcal{T}_v\left(T_v^L(G[S])\right)\right)\right).\end{aligned}$$

Proof. Let $\bar{G}[S]$ be the graph obtained by augmenting $G[S]$ with node features $\mathbf{0}$ for each $v \in V \setminus S$. That is, $\bar{G}[S] = (V, E[S], f')$ where $f'^v = f^v$ if $v \in S$ and $\mathbf{0}$ otherwise By the definition of the augmentation function ρ, we can instead consider the following OT problem, which is equivalent to (3):

$$\text{OT}_{\text{TD}_w}\left(\rho(\mathcal{T}_G^L, \mathcal{T}_{\bar{G}[S]}^L)\right).$$

In the base case where $L = 1$, we see that *any* transportation plan between the nodes of $\bar{G}[S]$ and G must transport the excess node feature mass $\sum_{v \notin S} \|f^v\|$ in G to some nodes in $G[S]$. The transportation plan $\boldsymbol{I}$ has cost exactly equal to $\sum_{v \in V} \|f^v - f'^v\| = \sum_{v \notin S} \|f^v\|$, so $\boldsymbol{I}$ must be an optimal transportation plan.

Now, if $L > 1$, consider the OT problem 3, and let C and Γ be the distance matrix and feasible transportation plans for this OT problem. Let $\gamma \in \Gamma$ be any feasible transportation plan.

By the definition of TD, for any $i, j \in V$, transporting $\gamma_{i,j}$ mass from i to j incurs two costs:

(1) $\gamma_{i,j}$ incurs the cost of transporting f^i to f'^j.

(2) $\gamma_{i,j}$ incurs the cost of taking all of the depth $(L-1)$ computation trees of all neighbors of i in $T_i^L(G)$, and transporting them to the depth $(L-1)$ computation trees of all neighbors of j in $T_j^L(G[S])$ (after appropriately augmenting with isolated nodes of feature $\mathbf{0}$ for each $v \notin S$.

To quantify these two costs, let us define $\bar{\mathcal{T}}_j(T_j^L(G[S]))$ to be the multiset of computation trees obtained by augmenting $\mathcal{T}_j(T_j L(G[S]))$ with isolated nodes of feature vectors $\mathbf{0}$ for each $j \notin S$. That is, let $\bar{\mathcal{T}}_j(T_j^L(G[S]))$ be defined as the second output of ρ when applied to $\left(\mathcal{T}_j(T_j^L(G)), \mathcal{T}_j(T_j^L(G[S]))\right)$:

$$\left(\mathcal{T}_j(T_j^L(G)), \bar{\mathcal{T}}_j(T_j^L(G[S]))\right) = \rho\left(\mathcal{T}_j(T_j^L(G)), \mathcal{T}_j(T_j^L(G[S]))\right).$$

We can now quantify the cost of transportation plan γ (3) as follows. In the following equation, the first term corresponds to item (1) above and the second term corresponds to item (2) discussed above.

$$\sum_{i,j \in V} \gamma_{i,j} C_{i,j} = \sum_{i,j \in V} \gamma_{i,j} \left\|f^i - f'^j\right\| + w(L-1) \sum_{i,j \in V} \gamma_{i,j} \text{OT}_{\text{TD}_w}(\mathcal{T}_i(T_i^L(G)), \bar{\mathcal{T}}_j(T_j^L(G[S]))).$$

Now, we can immediately lower bound the cost $\sum_{i,j} \gamma_{i,j} C_{i,j}$ of γ as follows:

$$\begin{aligned}\sum_{i,j \in V} \gamma_{i,j} C_{i,j} \geq \min_{\nu \in \Gamma} \sum_{i,j \in V} \nu_{i,j} \left\|f^i - f'^j\right\| \\ + w(L-1) \min_{\tau \in \Gamma} \sum_{i,j \in V} \tau_{i,j} \text{OT}_{\text{TD}_w}(\mathcal{T}_i(T_i^L(G)), \bar{\mathcal{T}}_j(T_j^L(G[S]))).\end{aligned}$$

For the first term, it is easy to see that $\nu = \boldsymbol{I}$ is an optimal solution, by the same argument as in the base case: any transportation plan ν must incur the cost of transporting the excess mass $\{\|f^v\|\}_{v \notin S}$ in the node features of G which is not present in the node features of S.

For the second term, we can notice that $\text{OT}_{\text{TD}_w}(\mathcal{T}_i(T_i^L(G)), \bar{\mathcal{T}}_j(T_j^L(G[S])))$ is also an TMD between graphs which contain the corresponding multisets of depth $L-1$ trees! So, by the inductive hypothesis, $\tau = \boldsymbol{I}$ is an optimal solution for this OT problem as well.

Consequently, substituting $\tau = \nu = \boldsymbol{I}$, we can further simplify the lower bound to

$$\begin{aligned}
\sum_{i,j\in V} \gamma_{i,j} C_{i,j} &\geq \sum_{i,j\in V} \boldsymbol{I}_{i,j} \left\| f^i - f'^j \right\| + w(L-1) \sum_{i,j\in V} \boldsymbol{I}_{i,j} \mathrm{OT}_{\mathrm{TD}_w}(\mathcal{T}_i(T_i^L(G)), \bar{\mathcal{T}}_j(T_j^L(G[S]))) \\
&= \sum_{i\in V} \left\| f^i - f'^j \right\| + w(L-1) \sum_{i\in V} \mathrm{OT}_{\mathrm{TD}_w}\left(\mathcal{T}_i(T_i^L(G), \bar{\mathcal{T}}_i(T_i^L(G[S])))\right) \\
&= \sum_{v\notin S} \|f^v\| + w(L-1) \sum_{v\notin S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\big(\mathcal{T}_v(T_v^L(G)), \emptyset\big)\right) \\
&+ w(L-1) \sum_{v\in S} \mathrm{OT}_{\mathrm{TD}_w}\left(\mathcal{T}_v(T_v^L(G)), \bar{\mathcal{T}}_v(T_v^L(G[S]))\right) \\
&= \sum_{v\notin S} \left\| f^i \right\| + w(L-1) \sum_{v\notin S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_v(T_v^L(G)), \emptyset\right)\right) \\
&+ w(L-1) \sum_{v\in S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_v(T_v^L(G)), \mathcal{T}_v(T_v^L(G[S]))\right)\right).
\end{aligned}$$

Finally, the inequality is tight, because $\gamma = \boldsymbol{I} \in \Gamma$ is clearly a feasible transportation plan. So, by induction, the lemma holds. ■

Remark B.1. The terms $\mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_v(T_v^L(G)), \mathcal{T}_v(T_v^L(G[S]))\right)\right)$ in the expression of Lemma 4.1 can themselves be viewed as the TMD between two graphs $\bar{G}$ and $\bar{G}'$, where $\bar{G}$ is the graph consisting of all elements in $\mathcal{T}_v(T_v^L(G))$ as a disjoint trees, and $\bar{G}'$ is the graph consisting of all elements in $\mathcal{T}_v(T_v^L(G[S]))$ as disjoint trees. In this sense, Lemma 4.1 precisely characterizes the recursive structure of TMD between a graph and its subgraph. We leverage this fact in later proofs.

Lemma 4.2. *For $T \subset S \subset V$, $\mathrm{TMD}_w^L(G, G[S \setminus T]) = \mathrm{TMD}_w^L(G, G[S]) + \mathrm{TMD}_w^L(G[S], G[S \setminus T])$.*

Proof. Let $Z = \{z \in V : z \notin S\}$. Since $T \subset V$, T and Z are disjoint. So, Lemma 4.1 shows that

$$\begin{aligned}
\mathrm{TMD}_w^L(G, G[S \setminus T]) &= \sum_{z\in Z} \|f^z\| + w(L-1) \sum_{z\in Z} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_z(T_z^L(G)), \emptyset\right)\right) \\
&+ \sum_{t\in T} \left\| f^t \right\| + w(L-1) \sum_{t\in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \emptyset\right)\right) \\
&+ w(L-1) \sum_{u\in S\setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right).
\end{aligned}$$

Now, each term in $\mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right)$ inside the summation is, in fact a depth $L-1$ TMD between two graphs corresponding to the two disjoint forests of computation trees (as we discussed in Remark B.1). Thus, by the inductive hypothesis, we can expand the last term as follows:

$$\begin{aligned}
\sum_{u\in S\setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right) &= \sum_{u\in S\setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S]))\right)\right) \\
&+ \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G[S])), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right).
\end{aligned}$$

Consequently, we obtain:

$$\begin{aligned}
\mathrm{TMD}_w^L(G, G[S \setminus T]) &= \sum_{z\in Z} \|f^z\| + w(L-1) \sum_{z\in Z} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_z(T_z^L(G)), \emptyset\right)\right) \\
&+ \sum_{t\in T} \left\| f^t \right\| + w(L-1) \sum_{t\in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \emptyset\right)\right) \\
&+ w(L-1) \sum_{u\in S\setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S]))\right)\right) \\
&+ w(L-1) \sum_{u\in S\setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G[S])), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right).
\end{aligned}$$

Next, we can add and subtract terms as follows.

$$\begin{aligned}
\mathrm{TMD}_w^L(G, G[S \setminus T]) = &\sum_{z \in Z} \|f^z\| + w(L-1) \sum_{z \in Z} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_z(T_z^L(G)), \emptyset\right)\right) \\
&+ \sum_{t \in T} \|f^t\| + w(L-1) \sum_{t \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \emptyset\right)\right) \\
&+ w(L-1) \sum_{u \in S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S]))\right)\right) \\
&- w(L-1) \sum_{u \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S]))\right)\right) \\
&+ w(L-1) \sum_{u \in S \setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G[S])), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right).
\end{aligned}$$

Now, $\mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[V \setminus S]))\right)\right)$ is also a depth $L-1$ TMD between two graphs corresponding to two disjoint forests of computation trees (again, see Remark B.1). So, we can apply the inductive hypothesis to see that for each $t \in T$,

$$\begin{aligned}
\mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \emptyset\right)\right) = &\mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \mathcal{T}_t(T_t^L(G[S]))\right)\right) \\
&+ \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G[S])), \emptyset\right)\right).
\end{aligned}$$

Thus, summing over $t \in T$, we have

$$\begin{aligned}
\sum_{t \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G[S])), \emptyset\right)\right) = &\sum_{t \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \emptyset\right)\right) \\
&- \sum_{t \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G)), \mathcal{T}_t(T_t^L(G[S]))\right)\right).
\end{aligned}$$

By substituting this into the equation above, we obtain

$$\begin{aligned}
\mathrm{TMD}_w^L(G, G[S \setminus T]) = &\sum_{z \in Z} \|f^z\| + w(L-1) \sum_{z \in Z} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_z(T_z^L(G)), \emptyset\right)\right) \\
&+ \sum_{t \in T} \|f^t\| + w(L-1) \sum_{t \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G[S])), \emptyset\right)\right) \\
&+ w(L-1) \sum_{u \in S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S]))\right)\right) \\
&+ w(L-1) \sum_{u \in S \setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G[S])), \mathcal{T}_u(T_u^L(G[S \setminus T]))\right)\right).
\end{aligned}$$

Rearranging in the form of Lemma 4.1, we obtain

$$\begin{aligned}
\mathrm{TMD}_w^L(G, G[S \setminus T]) = &\sum_{u \notin S} \|f^u\| + w(L-1) \sum_{u \notin S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathbf{0}\right)\right) \\
&+ w(L-1) \sum_{u \in S} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_u(T_u^L(G)), \mathcal{T}_u(T_u^L(G[S]))\right)\right) \\
&+ \sum_{t \in T} \|f^t\| + w(L-1) \sum_{t \in T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G[S])), \mathbf{0}\right)\right) \\
&+ w(L-1) \sum_{t \in S \setminus T} \mathrm{OT}_{\mathrm{TD}_w}\left(\rho\left(\mathcal{T}_t(T_t^L(G[S])), \mathcal{T}_t(T_t^L(G[S \setminus T]))\right)\right) \\
&= \mathrm{TMD}_w^L(G, G[S]) + \mathrm{TMD}_w^L(G[S], G[S \setminus T]).
\end{aligned}$$

■

Theorem 4.5. *Given a graph $G = (V, E, f)$, Algorithm 1 computes $\|G\|_{\mathrm{TN}_w^L}$ in $O(|E|\, L)$ time.*

Proof. Consider Algorithm 1, TreeNorm. The runtime is immediate from the algorithm pseudocode, as matrix-vector products can be computed in $O(|E|)$. In this proof, let A_G denote the adjacency matrix of a graph G and let x_G be the vector in $\mathbb{R}^V$ such that $x_G(v) = \|f^v\|$.

To verify the correctness, we proceed by induction. We just need to show that for all L, $\|G\|_{\mathrm{TN}_w^L} = \|x_G\|_1 + \left\|\sum_{\ell=1}^{L-1}\left(\prod_{t=1}^{\ell} w(L-t)\right) A_G^\ell x_G\right\|_1$, as this is the exact computation computed by the algorithm pseudocode. Note that an equivalent expression is $\|G\|_{\mathrm{TN}_w^L} = \left\|\sum_{\ell=0}^{L-1}\left(\prod_{t=1}^{\ell} w(L-t)\right) A_G^\ell x_G\right\|_1$, where the product over the emptyset is defined as 1.

If $L = 1$, then $\|G\|_{\mathrm{TN}_w^L}$ is simply the sum of its feature values, so this is trivially true.

Consider the case of a depth-L tree-norm computation. Throughout this proof, all variables are positive, and consequently we can move the ℓ_1 norm in and out of sums freely (i.e., we have a triangle *equality*.)

By taking $S = \emptyset$ in Lemma 4.1, we see that $\|G\|_{\mathrm{TN}_w^L}$ is equal to the norm of its nodes plus $w(L-1)\sum_{v\in V}\left\|\mathcal{T}_v(T_v^L(G))\right\|_{\mathrm{TN}_w^{L-1}}$. That is,

$$\|G\|_{\mathrm{TN}_w^L} = \|x_G\|_1 + w(L-1)\sum_{v\in V}\left\|\mathcal{T}_v(T_v^L(G))\right\|_{\mathrm{TN}_w^{L-1}}.$$

To interpret this expression, let G'_v be the graph which is a forest of all the trees in $\mathcal{T}_v(T_v^L(G))$– for each $u \in N_G(v)$, we G'_v will contain a depth-$(L-1)$ tree $G'_{v,u}$. Then, by inductive hypothesis,

$$\begin{aligned}
\left\|\mathcal{T}_v(T_v^L(G))\right\|_{\mathrm{TN}_w^{L-1}} &= \sum_{u\in N_G(v)}\left\|(T_v^L(G))\right\|_{\mathrm{TN}_w^{L-1}} = \|G'_v\|_{\mathrm{TN}_w^{L-1}} = \sum_{u\in N_G(v)}\left\|G'_{u,v}\right\|_{\mathrm{TN}_w^{L-1}} \\
&= \sum_{u\in N_G(v)}\left\|\sum_{\ell=0}^{L-2}\left(\prod_{t=1}^{\ell} w(L-t-1)\right) A_{G_{u,v}}^\ell x_{G_{u,v}}\right\|_1 \\
&= \sum_{u\in N_G(v)}\sum_{\ell=0}^{L-2}\left(\prod_{t=1}^{\ell} w(L-t-1)\right)\left\|A_{G_{u,v}}^\ell x_{G_{u,v}}\right\|_1,
\end{aligned}$$

Letting $A_G(v,u)$ be the (v,u)-th entry of A_G,

$$\begin{aligned}
\left\|\mathcal{T}_v(T_v^L(G))\right\|_{\mathrm{TN}_w^{L-1}} &= \sum_{u\in N_G(v)}\sum_{\ell=0}^{L-2}\left(\prod_{t=1}^{\ell} w(L-t-1)\right)\left\|A_{G_{u,v}}^\ell x_{G_{u,v}}\right\|_1 \\
&= \left\|\sum_{u\in V}\sum_{\ell=0}^{L-2}\left(\prod_{t=1}^{\ell} w(L-t-1)\right) A_G(u,v) A_{G_{u,v}}^\ell x_{G_{u,v}}\right\|_1.
\end{aligned}$$

Now, by the way we constructed the trees $\mathcal{T}_v(T_v^L(G))$, u has the same $(L-1)$-hop neighborhood in G as it does in $G_{u,v}$. Consequently,

$$\begin{aligned}
\|G\|_{\mathrm{TN}_w^L} &= \|x_G\|_1 + w(L-1)\sum_{v\in V}\left\|\mathcal{T}_v(T_v^L(G))\right\|_{\mathrm{TN}_w^{L-1}} \\
&= \|x_G\|_1 + w(L-1)\left\|\sum_{\ell=0}^{L-2}\left(\prod_{t=1}^{\ell} w(L-t-1)\right)\sum_{v\in V}\sum_{u\in V} A_G(u,v) A_{G_{u,v}}^\ell x_{G_{u,v}}\right\|_1 \\
&= \|x_G\|_1 + \left\|\sum_{\ell=1}^{L-2}\left(\prod_{t=1}^{\ell} w(L-t-1)\right)\sum_{v\in V} A_G^{\ell+1} x_G\right\|_1 \\
&= \|x_G\|_1 + \left\|\sum_{\ell=1}^{L-1}\left(\prod_{t=1}^{\ell} w(L-t)\right) A_G^\ell x_G\right\|_1,
\end{aligned}$$

completing the induction and the proof of correctness.

■

B.2. Omitted proofs from Section 5

Lemma 5.1. *Let $\mathcal{H}$ be a hypothesis class of $(L-1)$-layer GNNs $h : \mathcal{G}_{\mathcal{I}} \to \mathbb{R}^d$ with ℓ-th layer Lipschitz constant upper-bounded by Φ_ℓ. Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be an M-Lipschitz loss function. Let $y = (y_1, ..., y_n)$ be labels for X and $\mathcal{I}$ be a subset of $[n]$. Then for any GNN $h \in \mathcal{H}$ and $G \in \mathcal{G}$,*

$$\frac{1}{n}\Big|\sum_{i\in\mathcal{I}} \tau_i \mathcal{L}(h(G_i); y_i) - \sum_{i\in[n]} \mathcal{L}(h(G_i); y_i)\Big| \leq \tag{4}$$

$$M\left(\prod_{\ell\in[L-1]} \Phi_\ell\right) f_{\mathrm{TMD}^L_w}(\mathcal{I}; X). \tag{5}$$

Proof. Let $\kappa : [n] \to \mathcal{I}$ map $i \in [n]$ to the index $j \in \mathcal{I}$ such that G_j is the closest graph in $\{G_i\}_{i\in\mathcal{I}}$ to G_i. That is,

$$\kappa(i) = \operatorname*{argmin}_{j\in\mathcal{I}} \mathrm{TMD}^L_w(G_i, G_j),$$

with ties broken arbitrarily (but consistently with the mapping τ.) First, we can rearrange the sums as follows

$$\begin{aligned}
\left|\frac{1}{n}\sum_{i\in\mathcal{I}} \tau_i \mathcal{L}(h(G_i); y_i) - \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_i); y_i)\right| &= \left|\frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_{\kappa(i)}); y_{\kappa(i)}) - \mathcal{L}(h(G_i); y_i)\right| \\
&\leq \frac{1}{n}\sum_{i\in[n]} \left|\mathcal{L}(h(G_{\kappa(i)}; y_{\kappa(i)})) - \mathcal{L}(h(G_i); y_i)\right| \\
&\leq \frac{M}{n}\sum_{i\in[n]} \left\|\mathcal{L}(h(G_{\kappa(i)}); y_{\kappa(i)}) - \mathcal{L}(h(G_i); y_i)\right\|,
\end{aligned}$$

where in the last inequality, we used that $\mathcal{L}$ is M-lipschitz. Now, by Theorem 4.1, we have

$$\begin{aligned}
\frac{M}{n}\sum_{i\in[n]} \left\|\mathcal{L}(h(G_{\kappa(i)}); y_{\kappa(i)}) - h(G_i; y_i)\right\| &\leq \frac{M}{n}\left(\prod_{\ell\in[L-1]} \Phi_\ell\right) \sum_{i\in[n]} \mathrm{TMD}^L_w(G_{\kappa(i)}, G_i) \\
&= M\left(\prod_{\ell\in[L-1]} \Phi_\ell\right) f_{\mathrm{TMD}^L_w}(S; X).
\end{aligned}$$

■

Corollary 5.2. *Suppose that $M\left(\prod_{\ell\in[L-1]} \Phi_\ell\right) \leq c$ and $f_{\mathrm{TMD}^L_w}(\mathcal{I}; X) \leq \varepsilon$. Let $\hat{h} \in \mathcal{H}$ be the hypothesis that minimizes loss on the subsampled graphs $\sum_{i\in\mathcal{I}} \tau_i \mathcal{L}(h(G_i); y_i)$. Then*

$$\frac{1}{n}\sum_{i\in[n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i\in[n]} \mathcal{L}(h(G_i); y_i) + 2c\varepsilon.$$

Proof. From Lemma 5.1, we know that

$$\frac{1}{n}\sum_{i\in[n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \frac{1}{n}\sum_{i\in S} \tau_i \mathcal{L}(\hat{h}(G_i); y_i) + c\varepsilon.$$

Let h^* be the hypothesis that minimizes average loss over the original dataset X:

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i \in [n]} \mathcal{L}(h(G_i); y_i).$$

By definition of $\hat{h}$, this means that

$$\frac{1}{n} \sum_{i \in [n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \frac{1}{n} \sum_{i \in S} \tau_i \mathcal{L}(h^*(G_i); y_i) + c\varepsilon.$$

Applying Lemma 5.1 again, we have that

$$\frac{1}{n} \sum_{i \in [n]} \mathcal{L}\left(\hat{h}(G_i); y_i\right) \leq \frac{1}{n} \sum_{i \in [n]} \mathcal{L}\left(h^*(G_i); y_i\right) + 2c\varepsilon.$$

■