A Gromov–Wasserstein Geometric View of Spectrum-Preserving Graph Coarsening

Yifan Chen [1] Rentian Yao [2] Yun Yang [2] Jie Chen [3]

Abstract

Graph coarsening is a technique for solving large-scale graph problems by working on a smaller version of the original graph, and possibly interpolating the results back to the original graph. It has a long history in scientific computing and has recently gained popularity in machine learning, particularly in methods that preserve the graph spectrum. This work studies graph coarsening from a different perspective, developing a theory for preserving graph distances and proposing a method to achieve this. The geometric approach is useful when working with a collection of graphs, such as in graph classification and regression. In this study, we consider a graph as an element on a metric space equipped with the Gromov–Wasserstein (GW) distance, and bound the difference between the distance of two graphs and their coarsened versions. Minimizing this difference can be done using the popular weighted kernel K-means method, which improves existing spectrum-preserving methods with the proper choice of the kernel. The study includes a set of experiments to support the theory and method, including approximating the GW distance, preserving the graph spectrum, classifying graphs using spectral information, and performing regression using graph convolutional networks. Code is available at https://github.com/ychen-stat-ml/GW-Graph-Coarsening.

1. Introduction

Modeling the complex relationship among objects by using graphs and networks is ubiquitous in scientific applications (Morris et al., 2020). Examples range from analysis of chemical and biological networks (Debnath et al., 1991; Helma et al., 2001; Dobson & Doig, 2003; Irwin et al., 2012; Sorkun et al., 2019), learning and inference with social interactions (Oettershagen et al., 2020), to image understanding (Dwivedi et al., 2020). Many of these tasks are faced with large graphs, the computational costs of which can be rather high even when the graph is sparse (e.g., computing a few extreme eigenvalues and eigenvectors of the graph Laplacian can be done with a linear cost by using the Lanczos method, while computing the round-trip commute times requiring the pesudoinverse of the Laplacian, which admits a cubic cost). Therefore, it is practically important to develop methods that can efficiently handle graphs as their size grows. In this work, we consider one type of methods, which resolve the scalability challenge through working on a smaller version of the graph. The production of the smaller surrogate is called *graph coarsening*.

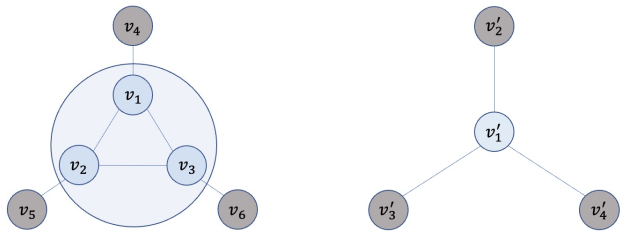

Figure 1: An example of graph coarsening. Three nodes v_1, v_2, and v_3 are merged to a supernode v_1' in the coarsened graph, while each of the other nodes (v_4, v_5, and v_6) is a separate supernode.

Graph coarsening is a methodology for solving large-scale graph problems. Depending on the problem itself, one may develop a solution based on the coarsened graph, or interpolate the solution back to the original one. Figure 1 illustrates a toy example of graph coarsening, whereby a cluster of nodes (v_1, v_2, and v_3) in the original graph are merged to a so-called *supernode* (v_1') in the coarsened graph. If the problem is graph classification (predicting classes of multiple graphs in a dataset), one may train a classifier by using these smaller surrogates, if it is believed that they inherit the characteristics of the original graphs necessary for classifica-

[1]Hong Kong Baptist University [2]University of Illinois Urbana-Champaign [3]MIT-IBM Watson AI Lab, IBM Research. Correspondence to: Yifan Chen <yifanc@comp.hkbu.edu.hk>.

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

tion. On the other hand, if the problem is node regression,[1] one may predict the targets for the nodes in the coarsened graph and interpolate them in the original graph (Huang et al., 2021).

Graph coarsening has a long history in scientific computing, while it gains attraction in machine learning recently (Chen et al., 2022). A key question to consider is what characteristics should be preserved when reducing the graph size. A majority of work in machine learning focuses on the spectral properties (Loukas & Vandergheynst, 2018; Loukas, 2019; Hermsdorff & Gunderson, 2019; Jin et al., 2020). That is, one desires that the eigenvalues of the original graph G are close to those of the coarsened graph $G^{(c)}$. While being attractive, it is unclear why this objective is effective for problems involving a collection of graphs (e.g., graph classification), where the distances between graphs shape the classifier and the decision boundary. Hence, in this work, we consider a different objective, which desires that the distance of a pair of graphs, $\text{dist}(G_1, G_2)$, is close to that of their coarsened versions, $\text{dist}(G_1^{(c)}, G_2^{(c)})$.

To achieve so, we make the following contributions.

- We consider a graph as an element on the metric space endowed with the Gromov–Wasserstein (GW) distance (Chowdhury & Mémoli, 2019), which formalizes the distance of graphs with different sizes and different node weights. We analyze the distance between G and $G^{(c)}$ as a major lemma for subsequent results.
- We establish an upper bound on the difference of the GW distance between the original graph pair and that of the coarsened pair. Interestingly, this bound depends on only the respective spectrum change of each of the original graphs. Such a finding explains the effectiveness of spectrum-preserving coarsening methods for graph classification and regression problems.
- We bridge the connection between the upper bound and weighted kernel K-means, a popular clustering method, under a proper choice of the kernel. This connection leads to a graph coarsening method that we demonstrate to exhibit attractive inference qualities for graph-level tasks, compared with other spectrum-preserving methods.

2. Related Work

Graph coarsening was made popular by scientific computing many decades ago, where a major problem was to solve large, sparse linear systems of equations (Saad, 2003). (Geometric) Multigrid methods (Briggs et al., 2000) were used to solve partial differential equations, the discretization of which leads to a mesh and an associated linear system. Multigrid methods solve a smaller system on the coarsened mesh and interpolate the solution back to the original mesh. When the linear system is not associated with a geometric mesh, algebraic multigrid methods (Ruge & Stüben, 1987) were developed to treat the coefficient matrix as a general graph, so that graph coarsening results in a smaller graph and the procedure of "solving a smaller problem then interpolating the solution on the larger one" remains applicable.

Graph coarsening was introduced to machine learning as a methodology of graph summarization (Liu et al., 2018) through the concepts of "graph cuts," "graph clustering," and "graph partitioning." A commonality of these concepts is that graph nodes are grouped together based on a certain objective. Graph coarsening plays an important role in multilevel graph partitioning (Karypis & Kumar, 1999). The normalized cut (Shi & Malik, 2000) is a well-known and pioneering method for segmenting an image treated as a graph. Dhillon et al. (2007) compute weighted graph cuts by performing clustering on the coarsest graph resulting from hierarchical coarsening and refining the clustering along the reverse hierarchy. Graph partitioning is used to form convolutional features in graph neural networks (Defferrard et al., 2016) and to perform neighborhood pooling (Ying et al., 2018). Graph coarsening is used to learn node embeddings in a hierarchical manner (Chen et al., 2018). For a survey of graph coarsening with comprehensive accounts on scientific computing and machine learning, see Chen et al. (2022).

A class of graph coarsening methods aim to preserve the spectra of the original graphs. Loukas & Vandergheynst (2018) and Loukas (2019) introduce the notion of "restricted spectral similarity" (RSS), requiring the eigenvalues and eigenvectors of the coarsened graph Laplacian, when restricted to the principal eigen-subspace, to approximate those of the original graph. Local variation algorithms are developed therein to achieve RSS. Jin et al. (2020) suggest the use of a spectral distance as the key metric for measuring the preservation of spectral properties. The authors develop two coarsening algorithms to maximally reduce the spectral distance between the original and the coarsened graph. Hermsdorff & Gunderson (2019) develop a probabilistic algorithm to coarsen a graph while preserving the Laplacian pseudoinverse, by using an unbiased procedure to minimize the variance.

Graph coarsening is increasingly used in deep learning. One limitation of traditional coarsening methods is that they mean to be universally applicable to any graphs, without the flexibility of adapting to a particular dataset or distribution of its own characteristics. Cai et al. (2021) address this limitation by adjusting the edge weights in the coarsened graph through graph neural networks (GNNs). Conversely, graph

[1]Machine learning problems on graphs could be on the graph level (e.g., classifying the toxicity of a protein graph) or on the node level (e.g., predicting the coordinates of each node (atom) in a molecular graph). In this work, our experiments focus on graph-level problems.

coarsening techniques can be applied to scale up GNNs by preprocessing the graphs (Huang et al., 2021). On a separate note, graph condensation (Jin et al., 2021) is another technique to accelerate GNNs, which uses supervised learning to condense node features and the graph structure. Furthermore, graph pooling can assign a node in the original graph to multiple supernodes (Grattarola et al., 2022), similar to the relaxation-based graph coarsening strategy explored in the algebraic multigrid literautre (Ron et al., 2011).

3. Notations and Preliminaries

In this section, we set up the notations for graph coarsening and review the background of Gromov–Wasserstein distance. Additional information can be found in Appendix A.

3.1. Graphs and Laplacians

We denote an undirected graph as G, whose node set is $\mathcal{V} := \{v_i\}_{i=1}^N$ with size $N = |\mathcal{V}|$ and whose symmetric weighted adjacency is $\boldsymbol{A} := [a_{ij}]$. The N-by-N (combinatorial) Laplacian matrix is defined as $\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D} := \operatorname{diag}(\boldsymbol{A}\mathbf{1}_N)$ is the degree matrix. The normalized Lapalcian is $\boldsymbol{\mathcal{L}} := \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$. Without loss of generality, we assume that G is connected, in which case the smallest eigenvalue of $\boldsymbol{L}$ and $\boldsymbol{\mathcal{L}}$ is zero and is simple.

3.2. Graph coarsening and coarsened graphs

Given a graph G, graph coarsening amounts to finding a smaller graph $G^{(c)}$ with $n \leq |\mathcal{V}|$ nodes to approximate G. One common coarsening approach obtains the coarsened graph from a parititioning $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ of the node set $\mathcal{V}$ (Loukas & Vandergheynst, 2018). In this approach, each subset $\mathcal{P}_j$ of nodes are collapsed to a supernode in the coarsened graph and the edge weight between two supernodes is the sum of the edge weights crossing the two corresponding partitions. Additionally, the sum of the edge weights in a partition becomes the weight of the supernode. In the matrix notation, we use $\boldsymbol{C}_p \in \{0,1\}^{n\times N}$ to denote the membership matrix induced by the partitioning $\mathcal{P}$, with the (k,i) entry being $\boldsymbol{C}_p(k,i) = 1_{\{v_i \in \mathcal{P}_k\}}$, where $1_{\{\cdot\}}$ is the indicator function. For notational convenience, we define the adjacency matrix of the coarsened graph to be $\boldsymbol{A}^{(c)} = \boldsymbol{C}_p\boldsymbol{A}\boldsymbol{C}_p^T$. Note that $\boldsymbol{A}^{(c)}$ includes not only edge weights but also node weights.

If we similarly define $\boldsymbol{D}^{(c)} := \operatorname{diag}\left(\boldsymbol{A}^{(c)}\mathbf{1}_n\right)$ to be the degree matrix of the coarsened graph $G^{(c)}$, then it can be verified that the matrix $\boldsymbol{L}^{(c)} = \boldsymbol{C}_p\boldsymbol{L}\boldsymbol{C}_p^\intercal = \boldsymbol{D}^{(c)} - \boldsymbol{A}^{(c)}$ is a (combinatorial) Laplacian (because its smallest eigenvalue is zero and is simple). Additionally, $\boldsymbol{\mathcal{L}}^{(c)} = \left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}}\boldsymbol{L}^{(c)}\left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}}$ is the normalized Laplacian.

In the literature, the objective of coarsening is often to minimize some difference between the original and the coarsened graph. For example, in spectral graph coarsening, Loukas (2019) proposes that the $\boldsymbol{L}$-norm of any N-dimensional vector $\boldsymbol{x}$ be close to the $\boldsymbol{L}^{(c)}$-norm of the n-dimensional vector $(\boldsymbol{C}_p^\intercal)^+\boldsymbol{x}$; while Jin et al. (2020) propose to minimize the difference between the ordered eigenvalues of the Laplacian of G and those of the lifted Laplacian of $G^{(c)}$ (such that the number of eigenvalues matches). Such objectives, while being natural and interesting in their own right, are not the only choice. Questions may be raised; for example, it is unclear why the objective is to preserve the Laplacian spectrum but not the degree distribution or the clustering coefficients. Moreover, it is unclear how preserving the spectrum benefits the downstream use. In this paper, we consider a different objective—preserving the graph distance—which may help, for example, maintain the decision boundary in graph classification problems. To this end, we review the Gromov–Wasserstein (GW) distance.

3.3. Gromov–Wasserstein distance and its induced metric space

The GW distance was originally proposed by Mémoli (2011) to measure the distance between two metric measure spaces $M_{\mathcal{X}}$ and $M_{\mathcal{Y}}$.[2] However, using metrics to characterize the difference between elements in sets $\mathcal{X}$ and $\mathcal{Y}$ can be too restrictive. Peyré et al. (2016) relaxed the metric notion by proposing the GW discrepancy, which uses dissimilarity, instead of metric, to characterize differences. Chowdhury & Mémoli (2019) then extended the concept of GW distance/discrepancy from metric measure spaces to measure networks, which can be considered a generalization of graphs.

Definition 3.1 (Measure network)**.** A measure network is a triple $(\mathcal{X}, \omega_X, \mu_X)$, where $\mathcal{X}$ is a Polish space (a separable and completely metrizable topological space), μ_X is a fully supported Borel probability measure, and ω_X is a bounded measurable function on $\mathcal{X}^2$.

In particular, a graph G (augmented with additional information) can be taken as a discrete measure network. We let $\mathcal{X}$ be the set of graph nodes $\{v_i\}_{i=1}^N$ and associate with it a probability mass $\mu_X = \boldsymbol{m} = [m_1, \ldots, m_N]^\intercal \in \mathbb{R}_+^N$, $\sum_{i=1}^N m_i = 1$. Additionally, we associate G with the node similarity matrix $\boldsymbol{S} = [s_{ij}] \in \mathbb{R}^{N\times N}$, whose entries are induced from the measurable map ω_X: $s_{ij} = \omega_X(v_i, v_j)$. Note that the mass $\boldsymbol{m}$ and the similarity $\boldsymbol{S}$ do not necessarily need to be related to the node weights and edge weights, although later we will justify and advocate the use of some variant of the graph Lapalcian as $\boldsymbol{S}$.

[2] A metric measure space $M_{\mathcal{X}}$ is the triple $(\mathcal{X}, d_X, \mu_X)$, where $(\mathcal{X}, d_X)$ is a metric space with metric d_X and μ_X is a Borel probability measure on $\mathcal{X}$.

For a source graph G_s with N_s nodes and mass $\boldsymbol{m}_s$ and a target graph G_t with N_t nodes and mass $\boldsymbol{m}_t$, we can define a transport matrix $\boldsymbol{T} = [t_{ij}] \in \mathbb{R}^{N_s \times N_t}$, where t_{ij} specifies the probability mass transported from v_i^s (the i-th node of G_s) to v_j^t (the j-th node of G_t). We denote the collection of all feasible transport matrices as $\Pi_{s,t}$, which includes all $\boldsymbol{T}$ that satisfy $\boldsymbol{T}\mathbf{1} = \boldsymbol{m}_s$ and $\boldsymbol{T}^\intercal \mathbf{1} = \boldsymbol{m}_t$. Using the ℓ_p transportation cost $L(a,b) = (a-b)^p$, Chowdhury & Mémoli (2019) define the GW_p distance for graphs as

$$\begin{aligned}\mathrm{GW}_p^p(G_s, G_t) &= \min_{\boldsymbol{T}\in\Pi_{s,t}} \sum_{i,j=1}^{N_S} \sum_{i',j'=1}^{N_t} \left|s_{ij}^s - s_{i'j'}^t\right|^p \boldsymbol{T}_{ii'}\boldsymbol{T}_{jj'} \\ &= \min_{\boldsymbol{T}\in\Pi_{s,t}} \langle \boldsymbol{M}, \boldsymbol{T}\rangle, \qquad (1)\end{aligned}$$

whre the cross-graph dissimilarity matrix $\boldsymbol{M} \in \mathbb{R}^{N_s \times N_t}$ has entries $\boldsymbol{M}_{jj'} = \sum_{i,i'} \left|s_{ij}^s - s_{i'j'}^t\right|^p \boldsymbol{T}_{ii'}$ (which by themselves are dependent on $\boldsymbol{T}$).

The computation of the GW_p distance (GW distance for short) can be thought of as finding a proper alignment of nodes in two graphs, such that the aligned nodes v_j^s and $v_{j'}^t$ have similar interactions with other nodes in their respective graphs. Intuitively, this is achieved by assigning large transport mass $\boldsymbol{T}_{jj'}$ to a node pair $(v_j^s, v_{j'}^t)$ with small dissimilarity $\boldsymbol{M}_{jj'}$, making the GW distance a useful tool for graph matching (Xu et al., 2019b). We note that variants of the GW distance exist; for example, Titouan et al. (2019) proposed a fused GW distance by additioanlly taking into account the dissimilarity of node features. We also note that computing the GW distance is NP-hard, but several approximate methods were developed (Xu et al., 2019a; Zheng et al., 2022). In this work, we use the GW distance as a theoretical tool and may not need to compute it in action.

On closing this section, we remark that Chowdhury & Mémoli (2019, Theorem 18) show that the GW distance is indeed a metric for measure networks, modulo weak isomorphism.[3] Therefore, we can formally establish the metric space of interest.

Definition 3.2 (GW_p space)**.** Let $\mathcal{N}$ be the collection of all measure networks. For $p \geq 1$, we denote by $(\mathcal{N}, \mathrm{GW}_p)$ the metric space of measure networks endowed with the GW_p distance defined in (1) and call it the GW_p space.

4. Graph Coarsening from a Gromov–Wasserstein Geometric View

In this section, we examine how the GW geometric perspective can shape graph coarsening. In Section 4.1, we provide a framework that unifies many variants of the coarsening matrices through the use of the probability mass $\boldsymbol{m}$ introduced in the context of measure networks. Then, in Section 4.2, we analyze the variant associated with the similarity matrix $\boldsymbol{S}$. In particular, we establish an upper bound of the difference of the GW distances before and after coarsening. Based on the upper bound, in Section 4.3 we connect it with some spectral graph techniques and in Section 4.4 we advocate a choice of $\boldsymbol{S}$ in practice.

[3]The weak isomorphism allows a node to be split into several identical nodes with the mass preserved. See Definition 3 of Chowdhury & Mémoli (2019).

4.1. A unified framework for coarsening matrices

The membership matrix $\boldsymbol{C}_p$ is a kind of coarsening matrices: it connects the adjacency matrix $\boldsymbol{A}$ with the coarsened version $\boldsymbol{A}^{(c)}$ through $\boldsymbol{A}^{(c)} = \boldsymbol{C}_p \boldsymbol{A} \boldsymbol{C}_p^\intercal$. There are, however, different variants of coarsening matrices. We consider three here, all having a size $n \times N$.

(i) Accumulation. This is the matrix $\boldsymbol{C}_p$. When multiplied to the left of $\boldsymbol{A}$, the i-th row of the product is the sum of all rows of $\boldsymbol{A}$ corresponding to the partition $\mathcal{P}_i$.

(ii) Averaging. A natural alternative to summation is (weighted) averaging. We define the diagonal mass matrix $\boldsymbol{W} = \mathrm{diag}\,(m_1, \cdots, m_N)$ and for each partition $\mathcal{P}_i$, the accumulated mass $c_i = \sum_{j\in\mathcal{P}_i} m_j$ for all $i \in [n]$. Then, the averaging coarsening matrix is

$$\bar{\boldsymbol{C}}_w := \mathrm{diag}(c_1^{-1}, \cdots, c_n^{-1}) \boldsymbol{C}_p \boldsymbol{W}.$$

This matrix takes the effect of averaging because of the division over c_i. Moreover, when the probability mass $\boldsymbol{m}$ is uniform (i.e., all m_j's are the same), we have the relation $\bar{\boldsymbol{C}}_w^+ = \boldsymbol{C}_p^\intercal$.

(iii) Projection. Neither $\boldsymbol{C}_p$ nor $\bar{\boldsymbol{C}}_w$ is orthogonal. We define the projection coarsening matrix as

$$\boldsymbol{C}_w := \mathrm{diag}(c_1^{-1/2}, \cdots, c_n^{-1/2}) \boldsymbol{C}_p \boldsymbol{W}^{\frac{1}{2}},$$

by noting that $\boldsymbol{C}_w \boldsymbol{C}_w^\intercal$ is the identity and hence $\boldsymbol{C}_w$ has orthonoral rows. Therefore, the N-by-N matrix $\boldsymbol{\Pi}_w := \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{C}_p^\intercal \bar{\boldsymbol{C}}_w \boldsymbol{W}^{-\frac{1}{2}} = \boldsymbol{C}_w^\intercal \boldsymbol{C}_w$ is a projection operator.

In Section 3.2, we have seen that the combinatorial Laplacian $\boldsymbol{L}$ takes $\boldsymbol{C}_p$ as the coarsening matrix, because the relationship $\boldsymbol{L}^{(c)} = \boldsymbol{C}_p \boldsymbol{L} \boldsymbol{C}_p^\intercal$ inherits from $\boldsymbol{A}^{(c)} = \boldsymbol{C}_p \boldsymbol{A} \boldsymbol{C}_p^\intercal$. On the other hand, it can be proved that, if we take the diagonal mass matrix $\boldsymbol{W}$ to be the degree matrix $\boldsymbol{D}$, the normalized Laplacian defined in Section 3.2 can be written as $\boldsymbol{\mathcal{L}}^{(c)} = \boldsymbol{C}_w \boldsymbol{\mathcal{L}} \boldsymbol{C}_w^\intercal$ (see Appendix B.1). In other words, the normalized Laplacian uses $\boldsymbol{C}_w$ as the coarsening matrix. The matrix $\boldsymbol{\mathcal{L}}^{(c)}$ is called a doubly-weighted Laplacian (Chung & Langlands, 1996).

For a general similarity matrix $\boldsymbol{S}$ (not necessarily a Laplacian), we use the averaging coarsening matrix $\bar{\boldsymbol{C}}_w$ and define $\boldsymbol{S}^{(c)} := \bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal$. This definition appears to be more natural in the GW setting; see a toy example in Appendix B.2. It is interesting to note that Vincent-Cuaz et al.

(2022) proposed the concept of semi-relaxed GW (srGW) divergence, wherein the first-order optimality condition of the constrained srGW barycenter problem is exactly the equality $\boldsymbol{S}^{(c)} = \bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal$. See Appendix B.3.

In the next subsection, we will consider the matrix $\boldsymbol{U} := \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{S} \boldsymbol{W}^{\frac{1}{2}}$. We define the coarsened version by using the projection coarsening matrix $\boldsymbol{C}_w$ as in $\boldsymbol{U}^{(c)} := \boldsymbol{C}_w \boldsymbol{U} \boldsymbol{C}_w^\intercal$. We will bound the distance of the original and the coarsened graph by using the eigenvalues of $\boldsymbol{U}$ and $\boldsymbol{U}^{(c)}$. The reason why $\boldsymbol{U}$ and $\boldsymbol{S}$ use different coarsening matrices lies in the technical subtlety of $\boldsymbol{W}^{\frac{1}{2}}$: $\boldsymbol{C}_w$ absorbs this factor from $\bar{\boldsymbol{C}}_w$.

4.2. Graph distance on the GW_2 space

Now we consider the GW distance between two graphs. For theoretical and computational convenience, we take the ℓ_2 transportation cost (i.e., taking $p = 2$ in (1)). When two graphs G_1 and G_2 are concerned, we inherit the subscripts $_1$ and $_2$ to all related quantities, such as the probability masses $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, avoiding verbatim redundancy when introducing notations. This pair of subscripts should not cause confusion with other subscripts, when interpreted in the proper context. Our analysis can be generalized from the GW_2 distance to other distances, as long as the transportation cost satisfies the following decomposable condition.

Proposition 4.1. *(Peyré et al., 2016) Let $\boldsymbol{T}^* \in \mathbb{R}^{N_1 \times N_2}$ be the optimal transport plan from $\boldsymbol{m}_1$ to $\boldsymbol{m}_2$,[4] and $\boldsymbol{S}_k \in \mathbb{R}^{N_k \times N_k}$ be similarity matrices for $k = 1, 2$. If the transport cost can be written as $L(a, b) = f_1(a) + f_2(b) - h_1(a)h_2(b)$ for some element-wise functions (f_1, f_2, h_1, h_2), then we can write $\boldsymbol{M}$ in (1) as*

$$f_1(\boldsymbol{S}_1)\boldsymbol{m}_1 \mathbf{1}_{N_2}^\intercal + \mathbf{1}_{N_1} \boldsymbol{m}_2^\intercal f_2(\boldsymbol{S}_2)^\intercal - h_1(\boldsymbol{S}_1)\boldsymbol{T}^* h_2(\boldsymbol{S}_2)^\intercal.$$

Clearly, for the squared cost $L(a, b) = (a - b)^2$, we may take $f_1(a) = a^2$, $f_2(b) = b^2$, $h_1(a) = a$, and $h_2(b) = 2b$.

We start the analysis by first bounding the distance between G and $G^{(c)}$.

Theorem 4.2 (Single graph). *Consider a graph G with positive semi-definite (PSD) similarity matrix $\boldsymbol{S}$ and diagonal mass matrix $\boldsymbol{W}$ and similarly the coarsened graph $G^{(c)}$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ be the sorted eigenvalues of $\boldsymbol{U} = \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{S} \boldsymbol{W}^{\frac{1}{2}}$ and $\lambda_1^{(c)} \geq \cdots \geq \lambda_n^{(c)}$ be the sorted eigenvalues of $\boldsymbol{U}^{(c)} = \boldsymbol{C}_w \boldsymbol{U} \boldsymbol{C}_w^\intercal$. Then,*

$$\mathrm{GW}_2^2(G, G^{(c)}) \leq \lambda_{N-n+1} \sum_{i=1}^{n} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},n}, \quad (2)$$

where $C_{\boldsymbol{U},n} = \sum_{i=1}^{n} \lambda_i(\lambda_i - \lambda_{N-n+i}) + \sum_{i=n+1}^{N} \lambda_i^2$ is non-negative and is independent of coarsening.

[4]The existence of $\boldsymbol{T}^*$ is guaranteed by the fact that the feasible region of $\boldsymbol{T}$ is compact and the object function $\langle \boldsymbol{M}, \boldsymbol{T} \rangle$ is continuous with respect to $\boldsymbol{T}$.

Remark 4.3. (i) The bound is tight when $n = N$ because the right-hand side is zero in this case. (ii) The choice of coarsening only affects the spectral difference

$$\Delta := \sum_{i=1}^{n} (\lambda_i - \lambda_i^{(c)}), \quad (3)$$

because $C_{\boldsymbol{U},n}$ is independent of it. Each term $\lambda_i - \lambda_i^{(c)}$ in Δ is non-negative due to the Poincaré separation theorem (see Appendix D.1). (iii) Δ is a generalization of the spectral distance proposed by Jin et al. (2020), because our matrix $\boldsymbol{U}$ is not necessarily the normalized Laplacian. For additional discussions, see Appendix B.4. (iv) When $\boldsymbol{U}$ is taken as the normalized Laplacian, our bound is advantageous over the bound established by Jin et al. (2020) in the sense that Δ is the only term impacted by coarsening and that no assumptions on the K-means cost are imposed.

We now bound the difference of distances. The following theorem suggests that the only terms dependent on coarsening are Δ_1 and Δ_2, counterparts of Δ in Theorem 4.2, for graphs G_1 and G_2 respectively.

Theorem 4.4. *Given a pair of graphs G_1 and G_2, we extend all notations in Theorem 4.2 by adding subscripts $_1$ and $_2$ respectively for G_1 and G_2. We denote the optimal transport plan induced by $\mathrm{GW}_2(G_1, G_2)$ as $\boldsymbol{T}^*$ and let the normalized counterpart be $\boldsymbol{P} = \boldsymbol{W}_1^{-\frac{1}{2}} \boldsymbol{T}^* \boldsymbol{W}_2^{-\frac{1}{2}}$. Additionally, we define $\boldsymbol{V}_1 := \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal$ with eigenvalues $\nu_{1,1} \geq \nu_{1,2} \geq \cdots \geq \nu_{1,N_1}$ and $\boldsymbol{V}_2 := \boldsymbol{P}^\intercal \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P}$ with eigenvalues $\nu_{2,1} \geq \nu_{2,2} \geq \cdots \geq \nu_{2,N_2}$, both independent of coarsening. Then, $\left|\mathrm{GW}_2^2(G_1^{(c)}, G_2^{(c)}) - \mathrm{GW}_2^2(G_1, G_2)\right|$ is upper bounded by*

$$\begin{aligned}\max\Big\{&\lambda_{1,N_1-n_1+1} \cdot \Delta_1 + C_{\boldsymbol{U}_1,n_1} \\ &\quad + \lambda_{2,N_2-n_2+1} \cdot \Delta_2 + C_{\boldsymbol{U}_2,n_2}, \\ &2 \cdot [\nu_{1,N_1-n_1+1} \cdot \Delta_1 + C_{\boldsymbol{U}_1,\boldsymbol{V}_1,n_1} \\ &\quad + \nu_{2,N_2-n_2+1} \cdot \Delta_2 + C_{\boldsymbol{U}_2,\boldsymbol{V}_2,n_2}]\Big\},\end{aligned}$$

where $C_{\boldsymbol{U}_1,n_1}$ is from Theorem 4.2 and the other coarsening-independent terms $\boldsymbol{U}_2, C_{\boldsymbol{U}_2,n_2}, C_{\boldsymbol{U}_2,\boldsymbol{V}_2,n_2}, C_{\boldsymbol{U}_2,\boldsymbol{V}_2,n_2}$ are introduced in Lemma E.5 in Appendix E.

Remark 4.5. (i) The above bound takes into account both the differences between two graphs and their respective coarsenings. Even when the two graphs are identical $G_1 = G_2$, the bound can still be nonzero if the coarsened graphs $G_1^{(c)}$ and $G_2^{(c)}$ do not match. (ii) The decoupling of Δ_1 and Δ_2 offers an algorithmic benefit when one wants to optimize the differences of distances for all graph pairs in a dataset: it suffices to optimize the distance between G and $G^{(c)}$ for each graph individually. This benefit is in line with the prior practice of directly applying spectrum-preserving coarsening methods for graph-level tasks (Jin et al., 2020; Huang et al., 2021). Their experimental results and our numerical verification in Section 6 show that our bound is useful and it partly explains the empirical success of spectral graph coarsening.

4.3. Connections with truncated SVDs and Laplacian eigenmaps

The spectral difference $\Delta = \sum_{i=1}^{n}\left(\lambda_i - \lambda_i^{(c)}\right)$ in Theorem 4.2 can be used as the loss function for defining an optimal coarsening. Because $\Delta + \sum_{i=n+1}^{N}\lambda_i = \mathrm{Tr}(\boldsymbol{U}) - \mathrm{Tr}(\boldsymbol{C}_w\boldsymbol{U}\boldsymbol{C}_w^\intercal)$ and because $\sum_{i=n+1}^{N}\lambda_i$ is independent of coarsening, minimizing Δ is equivalent to the following problem

$$\min_{\boldsymbol{C}_w} \mathrm{Tr}\left(\boldsymbol{U} - \boldsymbol{\Pi}_w\boldsymbol{U}\boldsymbol{\Pi}_w\right), \tag{4}$$

by recalling that $\boldsymbol{\Pi}_w = \boldsymbol{C}_w^\intercal\boldsymbol{C}_w$ is a projector. The problem (4) is a well-known trace optimization problem, which has rich connections with many spectral graph techniques (Kokiopoulou et al., 2011).

Connection with truncated SVD. At first sight, a small trace difference does not necessarily imply the two matrices are close. However, because $\boldsymbol{\Pi}_w$ is a projector, the Poincaré separation theorem (see Appendix D.1) suggests that their eigenvalues can be close. A well-known example of using the trace to find optimal approximations is the truncated SVD, which retains the top singular values (equivalently eigenvalues for PSD matrices). The truncated SVD is a technique to find the optimal rank-n approximation of a general matrix $\boldsymbol{U}$ in terms of the spectral norm or the Frobenius norm, solving the problem

$$\min_{\boldsymbol{C}\in\mathcal{O}_n} \mathrm{Tr}\left(\boldsymbol{U} - \boldsymbol{C}^\intercal\boldsymbol{C}\boldsymbol{U}\boldsymbol{C}^\intercal\boldsymbol{C}\right),$$

where $\mathcal{O}_n$ is the class of all rank-n orthogonal matrices. The projection coarsening matrix $\boldsymbol{C}_w$ belongs to this class.

Connection with Laplacian eigenmaps. The Laplacian eigenmap (Belkin & Niyogi, 2003) is a manifold learning technique that learns an n-dimensional embedding for N points connected by a graph. The embedding matrix $\boldsymbol{Y}$ solves the trace problem $\min_{\boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^\intercal=\boldsymbol{I}_n}\mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^\intercal\right)$. If we let $\bar{\boldsymbol{Y}} = \boldsymbol{Y}\boldsymbol{D}^{-\frac{1}{2}}$, the problem is equivalent to

$$\min_{\bar{\boldsymbol{Y}}\in\mathcal{O}_n} \mathrm{Tr}\left(\bar{\boldsymbol{Y}}\boldsymbol{\mathcal{L}}\bar{\boldsymbol{Y}}^\intercal\right).$$

Different from truncated SVD, which uses the top singular vectors (eigenvectors) to form the solution, the Laplacian eigenmap uses the bottom eigenvectors of $\boldsymbol{\mathcal{L}}$ to form the solution.

4.4. Signless Laplacians as similarity matrices

The theory established in Section 4.2 is applicable to any PSD matrix $\boldsymbol{S}$, but for practical uses we still have to define it. Based on the foregoing exposition, it is tempting to let $\boldsymbol{S}$ be the Laplacian $\boldsymbol{L}$ or the normalized Laplacian $\boldsymbol{\mathcal{L}}$, because they are PSD and they reveal important information of the graph structure (Tsitsulin et al., 2018). In fact, as a real example, Chowdhury & Needham (2021) used the heat kernel as the similarity matrix to define the GW distance (and in this case the GW framework is related to spectral clustering). However, there are two problems that make such a choice troublesome. First, the (normalized) Laplacian is sparse but its off-diagonal, nonzero entries are negative. Thus, when $\boldsymbol{S}$ is interpreted as a similarity matrix, a pair of nodes not connected by an edge becomes more similar than a pair of connected nodes, causing a dilema. Second, under the trace optimization framework, one is lured to intuitively look for solutions toward the bottom eigenvectors of $\boldsymbol{S}$, like in Laplacian eigenmaps, an opposite direction to the true solution of (4), which is toward the top eigenvectors instead.

To resolve these problems, we propose to use the signless Laplacian (Cvetković et al., 2007), $\boldsymbol{D}+\boldsymbol{A}$, or its normalized version, $\boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, as the similarity matrix $\boldsymbol{S}$. These matrices are PSD and their nonzero entries are all positive. With such a choice, the spectral difference Δ in (3) has a fundamentally different behavior from the spectral distance used by Jin et al. (2020) when defining the coarsening objective.

5. Computational Equivalence between Graph Coarsening and Weighted Kernel K-means

By recalling that $\boldsymbol{U} = \boldsymbol{W}^{\frac{1}{2}}\boldsymbol{S}\boldsymbol{W}^{\frac{1}{2}}$ and that $\boldsymbol{\Pi}_w = \boldsymbol{C}_w^\intercal\boldsymbol{C}_w$ is a projector, we rewrite the coarsening objective in (4) as

$$\mathrm{Tr}\left(\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{S}\boldsymbol{W}^{\frac{1}{2}}\right) - \mathrm{Tr}\left(\boldsymbol{C}_w\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{S}\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{C}_w^\intercal\right). \tag{5}$$

When $\boldsymbol{S}$ is PSD, it could be interpreted as a kernel matrix such that there exists a set of feature vectors $\{\boldsymbol{\phi}_i\}$ for which $\boldsymbol{S}_{ij} = \langle\boldsymbol{\phi}_i, \boldsymbol{\phi}_j\rangle$ for $i, j = 1, \ldots, N$. Then, with simple algebraic manipulation, we see that (5) is equivalent to the well-known clustering objective:

$$\sum_k\sum_{i\in\mathcal{P}_k} m_i\|\boldsymbol{\phi}_i - \boldsymbol{\mu}_k\|^2 \quad \text{with} \quad \boldsymbol{\mu}_k = \sum_{i\in\mathcal{P}_k}\frac{m_i}{c_k}\boldsymbol{\phi}_i, \tag{6}$$

where the norm $\|\cdot\|$ is induced by the inner product $\langle\cdot,\cdot\rangle$. Here, $\boldsymbol{\mu}_k$ is the weighted center of $\boldsymbol{\phi}_i$ for all nodes i belonging to cluster k. Hence, the weighted kernel K-means algorithm (Dhillon et al., 2004; 2007) can be applied to minimize (6) by iteratively recomputing the centers and updating cluster assignments according to the distance of a node to all centers. We denote the squared distance between

Algorithm 1 Kernel graph coarsening (KGC).

Input: $\boldsymbol{S}$: similarity matrix, $\boldsymbol{m}$: node mass vector, n: number of clusters
Output: $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^n$: node partition

```
 1: function KGC $\{\boldsymbol{S}, \boldsymbol{m}, n\}$
 2:   Initialize the $n$ clusters: $\mathcal{P}^{(0)} = \left\{\mathcal{P}_1^{(0)}, \dots, \mathcal{P}_n^{(0)}\right\}$;
      $c_j^{(0)} = \sum_{k \in \mathcal{P}_j^{(0)}} m_k, \forall j \in [n]$.
 3:   Set the counter of iterations $t = 0$.
 4:   for node $i = 1$ to $N$ do
 5:     Find its new cluster index by (7):
          $\mathrm{idx}(i) = \arg\min_{j \in [n]} \mathrm{dist}_j^{(t)}(i)$.
 6:   end for
 7:   Update the clusters: for all $j \in [n]$,
        $\mathcal{P}_j^{(t+1)} = \{i : \mathrm{idx}(i) = j\}, c_j^{(t+1)} = \sum_{k \in \mathcal{P}_j^{(t+1)}} m_k$.
 8:   if the partition $\mathcal{P}^{(t+1)}$ is invariant then
 9:     return $\mathcal{P}^{(t+1)}$
10:   else
11:     Set $t = t + 1$ and go to Line 4.
12:   end if
13: end function
```

$\boldsymbol{\phi}_i$ and any $\boldsymbol{\mu}_j$ by $\mathrm{dist}_j^2(i)$, which is

$$\boldsymbol{S}_{ii} - 2\sum_{k \in \mathcal{P}_j} m_k \boldsymbol{S}_{ki}/c_j + \sum_{k_1, k_2 \in \mathcal{P}_j} m_{k_1} m_{k_2} \boldsymbol{S}_{k_1 k_2}/c_j^2. \quad (7)$$

The K-means algorithm is summarized in Algorithm 1 and we call this method *kernel graph coarsening* (KGC).

KGC as a post-refinement of graph coarsening. KGC can be used as a standalone coarsening method. A potential drawback of this usage is that the K-means algorithm is subject to initialization and the clustering quality varies significantly sometimes. Even advanced initialization techniques, such as K-means++ (Arthur & Vassilvitskii, 2006), are not gauranteed to work well in practice. Moreover, KGC, in the vanilla form, does not fully utilize the graph information (such as node features), unless additional engineering of the similarity matrix $\boldsymbol{S}$ is conducted. Faced with these drawbacks, we suggest a simple alternative: initialize KGC by the output of another coarsening method and use KGC to improve it (Scrucca & Raftery, 2015). In our experience, KGC almost always monotonically reduces the spectral difference Δ and improves the quality of the initial coarsening.

Time complexity. Let T be the number of K-means iterations. The time complexity of KGC is $\mathcal{O}(T(M + Nn))$, where $M = \mathrm{nnz}(\boldsymbol{S})$ is the number of nonzeros in $\boldsymbol{S}$. See Appendix B.5 for the derivation of this cost and a comparison with the cost of spectral graph coarsening (Jin et al., 2020).

6. Numerical Experiments

We evaluate graph coarsening methods, including ours, on eight benchmark graph datasets: MUTAG (Debnath et al., 1991; Kriege & Mutzel, 2012), PTC (Helma et al., 2001), PROTEINS (Borgwardt et al., 2005; Schomburg et al., 2004), MSRC (Neumann et al., 2016), IMDB (Yanardag & Vishwanathan, 2015), Tumblr (Oettershagen et al., 2020), AQSOL (Sorkun et al., 2019; Dwivedi et al., 2020), and ZINC (Irwin et al., 2012). Information of these datasets is summarized in Table 1.

Table 1: Summary of datasets. $|V|$ and $|E|$ denote the average number of nodes and edges, respectively. All graphs are treated undirected. $\mathrm{R}(1)$ represents a regression task.

Dataset	Classes	Size	|V|	|E|
MUTAG	2	188	17.93	19.79
PTC	2	344	14.29	14.69
PROTEINS	2	1113	39.06	72.82
MSRC	8	221	39.31	77.35
IMDB	2	1000	19.77	96.53
Tumblr	2	373	53.11	199.78
AQSOL	R(1)	9823	17.57	17.86
ZINC	R(1)	12000	23.16	49.83

We compare our method with the following baseline methods (Loukas, 2019; Jin et al., 2020): ① **Variation Neighborhood Graph Coarsening (VNGC)**; ② **Variation Edge Graph Coarsening (VEGC)**; ③ **Multilevel Graph Coarsening (MGC)**; and ④ **Spectral Graph Coarsening (SGC)**. For our method, we consider the vanilla KGC, which uses K-means++ for initialization, and the variant KGC(A), which takes the output of the best-performing baseline method for initialization. More implementation details are provided in Appendix C.

6.1. GW distance approximation

For a sanity check, we evaluate each coarsening method on the approximation of the GW distance, to support our motivation of minimizing the upper bound in Theorem 4.2.

We first compare the average squared GW distance, by using the normalized signless Laplacian $\boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ as the similarity matrix $\boldsymbol{S}$ (see Section 4.4). We vary the coarsened graph size $n = \lceil c * N \rceil$ for $c = 0.3, \cdots, 0.9$. For each graph in PTC and IMDB, we compute $\mathrm{GW}_2^2(G, G^{(c)})$ and report the average in Table 5, Appendix C.1. We obtain similar observations across the two datasets. In particular, KGC and KGC(A) outperform baselines. When c is small, it is harder

Table 2: Change of the matrix of GW distances (computed as Frobenius norm error) and coarsening time. Dataset: PTC. Results are averged over 10 runs. MGC is a deterministic method and therefore standard deviation is 0. KGC(A) is initialized with the MGC output.

Coars. Mat.	Methods	Frob. Error↓	Time↓
Projection	VNGC	182.85 ± 0.02	6.38 ± 0.01
	VEGC	54.81 ± 0.02	$3.81 \pm 0.$
	MGC	$13.69 \pm 0.$	6.71 ± 0.01
	SGC	12.41 ± 0.04	30.24 ± 0.07
Averaging	VNGC	17.34 ± 0.01	6.55 ± 0.18
	VEGC	9.22 ± 0.02	3.75 ± 0.01
	MGC	$5.31 \pm 0.$	6.59 ± 0.02
	SGC	6.06 ± 0.02	28.06 ± 0.10
	KGC	$\mathbf{4.45 \pm 0.03}$	$\mathbf{1.34 \pm 0.33}$
	KGC(A)	$5.28 \pm 0.$	$\mathbf{0.27 \pm 0.}$

for K-means clustering to find the best clustering plan and hence KGC(A) works better. When c gets larger, KGC can work better, probably because the baseline outputs are poor intiializations.

We then report the average gap between the left- and right-hand sides of the bound (2) in Table 6, Appendix C.1. For all methods, including the baselines, the gap is comparable to the actual squared distance shown in Table 5, showcasing the quality of the bound. Moreover, the gap decreases when c increases, as expected.

We next compute the matrix of GW distances, $\boldsymbol{Z}$ (before coarsening) and $\boldsymbol{Z}^{(c)}$ (after coarsening), and compute the change $\|\boldsymbol{Z} - \boldsymbol{Z}^{(c)}\|_F$. Following previous works (Chan & Airoldi, 2014; Xu et al., 2020), we set $n = \lfloor N_{\max}/\log(N_{\max})\rfloor$. The similarity matrix for the coarsened graph uses the averaging coarsening matrix as advocated in Section 4.1; that is, $\boldsymbol{S}^{(c)} = \bar{\boldsymbol{C}}_w \left(\boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right)\bar{\boldsymbol{C}}_w^{\intercal}$. Additionally, we use a variant of the similarity matrix, $\boldsymbol{S}^{(c)} = \boldsymbol{I}_n + \left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}}\boldsymbol{A}^{(c)}\left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}}$, resulting from the projection coarsening matrix, for comparison.

Table 2 summarizes the results for PTC. It confirms that using "Averaging" is better than using "Projection" for the coarsening matrix. Additionally, KGC(A) initialized with MGC (the best baseline) induces a significantly small change (though no better than the change caused by KGC), further verifying that minimizing the loss (4) will lead to a small change in the GW distance. The runtime reported in the table confirms the analysis in Section 5, showing that KGC is efficient. Moreover, the runtime of KGC(A) is nearly negligible compared with that of the baseline method used for initializing KGC(A).

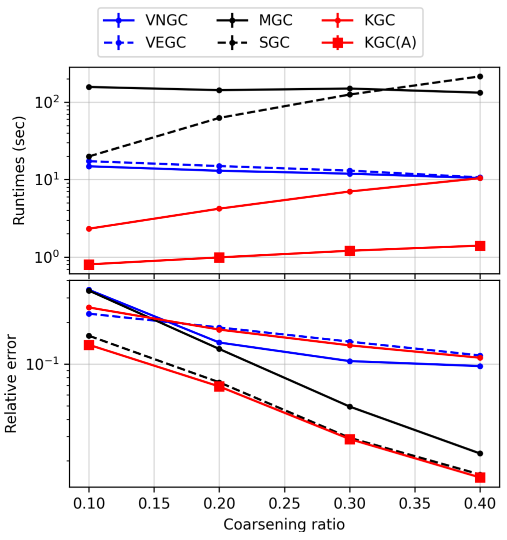

Figure 2: Runtime of coarsening methods and relative error for spectrum preservation on Tumblr. KGC(A) is initialized with SGC (the black dashed curve); its error curve (the red solid curve) is slightly below SGC.

6.2. Laplacian spectrum preservation

We evaluate the capability of the methods in preserving the Laplacian spectrum, by noting that the spectral objectives of the baseline methods differ from that of ours (see remarks after Theorem 4.2). We coarsen each graph to its $10\%, 20\%, 30\%$, and 40% size and evaluate the metric $\frac{1}{5}\sum_{i=1}^{5}\frac{\lambda_i - \lambda_i^{(c)}}{\lambda_i}$ (the relative error of the first five largest eigenvalues). The experiment is conducted on Tumblr and the calculation of the error metric is repeated ten times.

In Figure 2, we see that KGC has a comparable error to the variational methods VNGC and VEGC, while incurring a lower time cost, especially when the coarsening ratio is small. Additionally, KGC(A) consistently improves the initialization method SGC and attains the lowest error.

6.3. Graph classification using Laplacian spectrum

We test the performance of the various coarsening methods for graph classification. We follow the setting of Jin et al. (2020) and adopt the Network Laplacian Spectral Descriptor (Tsitsulin et al., 2018, NetLSD) along with a 1-NN classifier as the classification method. We set $n = 0.2N$. For evaluation, we select the best average accuracy of 10-fold cross validations among three different seeds. Additionally, we add two baselines, EIG and FULL, which respectively use the first n eigenvalues and the full spectrum of $\boldsymbol{\mathcal{L}}$ in NetLSD. Similarly to before, we initialize KGC(A) with the best baseline.

Table 3 shows that preserving the spectrum does not neces-

Table 3: Classification accuracy with coarsened graphs five times smaller than the original graphs.

Datasets	**MUTAG**	**PTC**	**PROTEINS**	**MSRC**	**IMDB**	**Tumblr**
VNGC	76.11 ± 2.25	56.69 ± 2.52	65.44 ± 1.57	14.92 ± 1.57	53.90 ± 0.50	50.43 ± 2.62
VEGC	84.59 ± 2.02	56.39 ± 2.03	64.08 ± 1.11	16.80 ± 2.15	64.20 ± 1.90	48.26 ± 1.71
MGC	84.15 ± 3.14	54.66 ± 3.59	66.16 ± 1.64	15.36 ± 1.80	$\mathbf{69.50 \pm 1.42}$	50.14 ± 2.67
SGC	84.44 ± 2.86	53.79 ± 2.28	63.91 ± 1.51	16.76 ± 2.50	66.00 ± 1.26	48.53 ± 2.35
KGC	81.90 ± 2.74	$\mathbf{61.58 \pm 2.49}$	63.45 ± 0.83	$\mathbf{19.84 \pm 2.23}$	67.80 ± 1.65	52.52 ± 2.81
KGC(A)	$\mathbf{86.23 \pm 2.69}$	57.25 ± 2.16	$\mathbf{66.43 \pm 0.92}$	17.17 ± 2.91	69.20 ± 1.37	$\mathbf{52.57 \pm 2.22}$
EIG	85.61 ± 1.69	56.08 ± 2.28	64.35 ± 1.43	12.19 ± 2.79	68.70 ± 1.71	49.57 ± 1.95
FULL	84.59 ± 2.51	54.37 ± 2.12	67.51 ± 0.82	23.58 ± 2.50	69.90 ± 1.40	52.57 ± 3.36

Table 4: Graph regression results on AQSOL and ZINC. The asterisk symbol $*$ indicates the TestMAE improvement of KGC(A) over its VEGC initialization is statistically significant at the 95% significance level in an paired t-test.

(a) AQSOL.

Methods	**TestMAE±s.d.**	**TrainMAE±s.d.**	**Epochs**
VNGC	1.403 ± 0.005	0.629 ± 0.018	135.75
VEGC	1.390 ± 0.005	0.702 ± 0.003	107.75
MGC	1.447 ± 0.005	0.628 ± 0.012	111.00
SGC	1.489 ± 0.010	0.676 ± 0.021	107.00
KGC	1.389 ± 0.015	0.678 ± 0.013	112.00
KGC(A)	$\mathbf{1.383 \pm 0.005}^*$	0.657 ± 0.013	124.75
FULL	1.372 ± 0.020	0.593 ± 0.030	119.50

(b) ZINC.

Methods	**TestMAE±s.d.**	**TrainMAE±s.d.**	**Epochs**
VNGC	0.709 ± 0.005	0.432 ± 0.012	120.00
VEGC	0.646 ± 0.001	0.418 ± 0.008	138.25
MGC	0.677 ± 0.002	0.414 ± 0.006	112.50
SGC	0.649 ± 0.007	0.429 ± 0.008	111.75
KGC	0.737 ± 0.010	0.495 ± 0.012	113.50
KGC(A)	$\mathbf{0.641 \pm 0.003}^*$	0.433 ± 0.013	126.50
FULL	0.416 ± 0.006	0.313 ± 0.011	159.50

sarily leads to the best classification, since EIG and FULL, which use the original spectrum, are sometime outperformed by other methods. On the other hand, our proposed method, KGC or KGC(A), is almost always the best.

6.4. Graph regression using GCNs

We follow the setting of Dwivedi et al. (2020) to perform graph regression on AQSOL and ZINC (see Appendix C.4 for details). For evaluation, we pre-coarsen the whole datasets, reduce the graph size by 70%, and run GCN on the coarsened graphs. Table 4 suggests that KGC(A) initialized with VEGC (the best baseline) always returns the best test MAE. On ZINC, KGC sometimes suffers from the initialization, but its performance is still comparable to a reported MLP baseline (Dwivedi et al., 2020, TestMAE 0.706 ± 0.006).

7. Conclusions

In this work, we propose a new perspective to study graph coarsening, by analyzing the distance of graphs on the GW space. We derive an upper bound on the change of the distance caused by coarsening, which depends on only the spectral difference $\Delta = \sum_{i=1}^{n} \left(\lambda_i - \lambda_i^{(c)}\right)$. This bound in a way justifies the idea of preserving the spectral information as the main objective of graph coarsening, although our definition of "spectral-preserving" differs from prior spectral coarsening techniques. More importantly, we point out the equivalence between the bound and the objective of weighted kernel K-means clustering. This equivalence leads to a new coarsening method we termed KGC. Our experiment results validate the theoretical analysis, showing that KGC preserves the GW distance between graphs and improves the accuracy of graph-level classification and regression tasks.

Acknowledgements

We wish to appreciate all the constructive and insightful comments from the anonymous reviewers. Yun Yang's research was supported in part by U.S. NSF grant DMS-2210717. Jie Chen acknowledges supports from the MIT-IBM Watson AI Lab.

References

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Bellman, R. *Introduction to matrix analysis*. SIAM, 1997.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Briggs, W. L., Henson, V. E., and McCormick, S. F. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2nd edition, 2000.

Cai, C., Wang, D., and Wang, Y. Graph coarsening with neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Chan, S. and Airoldi, E. A consistent histogram estimator for exchangeable graph models. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 208–216, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/chan14.html.

Chen, H., Perozzi, B., Hu, Y., and Skiena, S. HARP: Hierarchical representation learning for networks. In *AAAI*, 2018.

Chen, J., Saad, Y., and Zhang, Z. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, 2022.

Chowdhury, S. and Mémoli, F. The gromov–wasserstein distance between networks and stable network invariants. *Information and Inference: A Journal of the IMA*, 8(4): 757–787, 2019.

Chowdhury, S. and Needham, T. Generalized spectral clustering via gromov-wasserstein learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 712–720. PMLR, 2021.

Chung, F. R. and Langlands, R. P. A combinatorial laplacian with vertex weights. *journal of combinatorial theory, Series A*, 75(2):316–327, 1996.

Cvetković, D., Rowlinson, P., and Simić, S. K. Signless laplacians of finite graphs. *Linear Algebra and its applications*, 423(1):155–171, 2007.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34 (2):786–797, 1991.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845, 2016.

Dhillon, I. S., Guan, Y., and Kulis, B. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 551–556, 2004.

Dhillon, I. S., Guan, Y., and Kulis, B. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Grattarola, D., Zambon, D., Bianchi, F. M., and Alippi, C. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Helma, C., King, R. D., Kramer, S., and Srinivasan, A. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

Hermsdorff, G. B. and Gunderson, L. M. A unifying framework for spectrum-preserving graph sparsification and coarsening. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7734–7745, 2019.

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

Huang, Z., Zhang, S., Xi, C., Liu, T., and Zhou, M. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 675–684, 2021.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Ivashkin, V. and Chebotarev, P. Do logarithmic proximity measures outperform plain ones in graph clustering? In *International Conference on Network Analysis*, pp. 87–105. Springer, 2016.

Jin, W., Zhao, L., Zhang, S., Liu, Y., Tang, J., and Shah, N. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580*, 2021.

Jin, Y., Loukas, A., and JáJá, J. Graph coarsening with preserved spectral properties. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4452–4462. PMLR, 2020.

Karypis, G. and Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Kokiopoulou, E., Chen, J., and Saad, Y. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18(3):565–602, 2011.

Kriege, N. M. and Mutzel, P. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

Liu, Y., Safavi, T., Dighe, A., and Koutra, D. Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3), jun 2018. ISSN 0360-0300. doi: 10.1145/3186727. URL https://doi.org/10.1145/3186727.

Loukas, A. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Loukas, A. and Vandergheynst, P. Spectrally approximating large graphs with smaller graphs. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3243–3252. PMLR, 2018.

Mémoli, F. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

Minka, T. P. Old and new matrix algebra useful for statistics, 2000. https://tminka.github.io/papers/matrix/.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

Oettershagen, L., Kriege, N. M., Morris, C., and Mutzel, P. Temporal graph kernels for classifying dissemination processes. In *Proceedings of the 2020 SIAM International Conference on Data Mining, SDM 2020, Cincinnati, Ohio, USA, May 7-9, 2020*, pp. 496–504. SIAM, 2020. doi: 10.1137/1.9781611976236.56.

Peyré, G., Cuturi, M., and Solomon, J. Gromov-wasserstein averaging of kernel and distance matrices. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2664–2672. JMLR.org, 2016.

Ron, D., Safro, I., and Brandt, A. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011.

Ruge, J. W. and Stüben, K. Algebraic multigrid. In *Multigrid methods*, pp. 73–130. SIAM, 1987.

Ruhe, A. Perturbation bounds for means of eigenvalues and invariant subspaces. *BIT Numerical Mathematics*, 10(3): 343–354, 1970.

Saad, Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., and Schomburg, D. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

Scrucca, L. and Raftery, A. E. Improved initialisation of model-based clustering using gaussian hierarchical partitions. *Advances in data analysis and classification*, 9(4): 447–460, 2015.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Sorkun, M. C., Khetan, A., and Er, S. Aqsoldb, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds. *Scientific data*, 6(1):1–8, 2019.

Titouan, V., Courty, N., Tavenard, R., and Flamary, R. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284. PMLR, 2019.

Tsitsulin, A., Mottin, D., Karras, P., Bronstein, A. M., and Müller, E. Netlsd: Hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 2347–2356. ACM, 2018. doi: 10.1145/3219819.3219991.

Vincent-Cuaz, C., Flamary, R., Corneli, M., Vayer, T., and Courty, N. Semi-relaxed gromov-wasserstein divergence and applications on graphs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RShaMexjc-x.

Xu, H., Luo, D., and Carin, L. Scalable gromov-wasserstein learning for graph partitioning and matching. *Advances in neural information processing systems*, 32, 2019a.

Xu, H., Luo, D., Zha, H., and Carin, L. Gromov-wasserstein learning for graph matching and node embedding. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6932–6941. PMLR, 2019b.

Xu, H., Luo, D., Carin, L., and Zha, H. Learning graphons via structured gromov-wasserstein barycenters. In *AAAI Conference on Artificial Intelligence*, 2020.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 1365–1374. ACM, 2015. doi: 10.1145/2783258.2783417.

Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

Zheng, L., Xiao, Y., and Niu, L. A brief survey on computational gromov-wasserstein distance. *Procedia Computer Science*, 199:697–702, 2022.

A. List of notations

Notations	**Meaning**
$\boldsymbol{L}(\boldsymbol{L}^{(c)})$	(coarsened) graph Laplacian matrix
$\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{L}}^{(c)})$	(coarsened) normalized graph Laplacian matrix
$G^{(c)}$	coarsened graph
$\boldsymbol{D}(\boldsymbol{D}^{(c)})$	(coarsened) degree matrix
$\boldsymbol{A}(\boldsymbol{A}^{(c)})$	(coarsened) adjacency matrix
$\boldsymbol{C}_p$	membership matrix, coarsening matrix with entries $\in \{0,1\}$
$\boldsymbol{C}_w$	orthogonal coarsening matrix, a variant of coarsening matrix
$\bar{\boldsymbol{C}}_w$	weighted averaging matrix, a variant of coarsening matrix
$\boldsymbol{\Pi}_w$	projection matrix induced by $\boldsymbol{C}_w$
$\boldsymbol{W}$	diagonal node mass matrix
$\boldsymbol{S}(\boldsymbol{S}^{(c)})$	(coarsened) similarity matrix

We elaborate more about the family of coarsening matrices here. Specifically, we define the matrix $\boldsymbol{C}_p \in \mathbb{R}^{n\times N}$ as

$$\boldsymbol{C}_p(k,i) = \begin{cases} 1 & v_i \in \mathcal{P}_k \\ 0 & otherwise \end{cases}.$$

Let $p_i = |\mathcal{P}_i|$ be the number of nodes in the i-th partition $\mathcal{P}_i$ and n be the number of partitioning. Then, we define the weight matrix $\boldsymbol{W} = \mathrm{diag}(m_1, \cdots, m_N)$ and let $c_i = \sum_{j\in\mathcal{P}_i} m_j$. We can thus give the definition of the weighted averaging matrix as

$$\bar{\boldsymbol{C}}_w = \begin{pmatrix} \frac{1}{c_1} & & & \\ & \frac{1}{c_2} & & \\ & & \cdots & \\ & & & \frac{1}{c_n} \end{pmatrix} \boldsymbol{C}_p \boldsymbol{W}.$$

and the orthogonal coarsening matrix

$$\boldsymbol{C}_w = \begin{pmatrix} \sqrt{\frac{1}{c_1}} & & & \\ & \sqrt{\frac{1}{c_2}} & & \\ & & \cdots & \\ & & & \sqrt{\frac{1}{c_n}} \end{pmatrix} \boldsymbol{C}_p \boldsymbol{W}^{\frac{1}{2}}$$

Let the projection matrix be $\boldsymbol{\Pi}_w = \boldsymbol{C}_w^\intercal \boldsymbol{C}_w$. We can check that $\boldsymbol{\Pi}_w^2 = \boldsymbol{\Pi}_w$.

B. Derivations Omitted in the Main Text

B.1. Weighted graph coarsening leads to doubly-weighted Laplacian

We show in the following that $\boldsymbol{C}_w \boldsymbol{\mathcal{L}} \boldsymbol{C}_w^\intercal$ can reproduce the normalized graph Laplacian $\boldsymbol{\mathcal{L}}^{(c)}$. In this case, $\boldsymbol{C}_w = \left(\boldsymbol{C}_p \boldsymbol{D} \boldsymbol{C}_p^\intercal\right)^{-\frac{1}{2}} \boldsymbol{C}_p \boldsymbol{D}^{\frac{1}{2}}$ and interestingly $\boldsymbol{C}_p \boldsymbol{D} \boldsymbol{C}_p^\intercal = \mathrm{diag}\left(\boldsymbol{C}_p \boldsymbol{A} \boldsymbol{C}_p^\intercal \boldsymbol{1}\right) = \mathrm{diag}\left(\boldsymbol{A}^{(c)}\boldsymbol{1}\right) = \boldsymbol{D}^{(c)}$, indicating

$$\begin{aligned} \boldsymbol{C}_w \boldsymbol{\mathcal{L}} \boldsymbol{C}_w^\intercal &= \boldsymbol{I}_n - \boldsymbol{C}_w \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{C}_w^\intercal \\ &= \boldsymbol{I}_n - \left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}} \boldsymbol{C}_p \boldsymbol{D}^{\frac{1}{2}} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{D}^{\frac{1}{2}} \boldsymbol{C}_p^\intercal \left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}} \\ &= \boldsymbol{I}_n - \left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}} \boldsymbol{A}^{(c)} \left(\boldsymbol{D}^{(c)}\right)^{-\frac{1}{2}} = \boldsymbol{\mathcal{L}}^{(c)}. \end{aligned}$$

The results above imply we can unify previous coarsening results under the weighted graph coarsening framework in this paper, with a proper choice of similarity matrix $\boldsymbol{S}$ and node measure μ.

B.2. A toy example of coarsening a 3-node graph

Consider a fully-connected 3-node graph with equal weights for nodes and the partition $\{\{v_1\}, \{v_2, v_3\}\}$. Its similarity matrix $\boldsymbol{S} = \boldsymbol{D} + \boldsymbol{A}$ and the three possible coarsened similarity matrices will respectively be

$$\boldsymbol{S} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}, \quad \bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal = \begin{pmatrix} 2 & 1 \\ 1 & 3/2 \end{pmatrix}, \quad \boldsymbol{C}_w \boldsymbol{S} \boldsymbol{C}_w^\intercal = \begin{pmatrix} 2 & \sqrt{2} \\ \sqrt{2} & 3 \end{pmatrix}, \quad \boldsymbol{C}_p \boldsymbol{S} \boldsymbol{C}_p^\intercal = \begin{pmatrix} 2 & 2 \\ 2 & 6 \end{pmatrix}.$$

It is clear that $\bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal$, which will have the most appropriate entry magnitude to give the minimal GW-distance, is different from $\boldsymbol{C}_p \boldsymbol{S} \boldsymbol{C}_p^\intercal$ proposed in Jin et al. (2020) and (Cai et al., 2021). We note in $\boldsymbol{S}^{(c)} = \bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal$ the edge weight (similarity) is still 1, since we explicitly specify the node weight of the supernode becomes 2. The new GW geometric framework thus decouples the node weights and similarity intensities.

B.3. Deriving the "Averaging" magnitude from the constrained srGW barycenter problem

We first introduce the definition of srGW divergence. For a given graph G with node mass distribution $\boldsymbol{m} \in \Delta^{N-1}$ and similarity matrix $\boldsymbol{S}$, we can construct another graph $G^{(c)}$ with given similarity matrix $\boldsymbol{S}^{(c)}$ and *unspecified* node mass distribution $\boldsymbol{m}^{(c)} \in \Delta^{n-1}$. To better reflect the dependence of $\mathrm{GW}_2^2\left(G, G^{(c)}\right)$ on node mass distribution and similarity matrix, we abuse the notation GW_2 as $\mathrm{GW}_2^2\left(\boldsymbol{S}, \boldsymbol{m}, \boldsymbol{S}^{(c)}, \boldsymbol{m}^{(c)}\right)$ and $\Pi\left(\boldsymbol{m}, \boldsymbol{m}^{(c)}\right) := \left\{\boldsymbol{T} \in \mathbb{R}_+^{N\times n} : \boldsymbol{T}\mathbf{1} = \boldsymbol{m}, \boldsymbol{T}^\intercal \mathbf{1} = \boldsymbol{m}^{(c)}\right\}$ in this subsection. Vincent-Cuaz et al. (2022) then defined

$$\mathrm{srGW}_2^2\left(\boldsymbol{S}, \boldsymbol{m}, \boldsymbol{S}^{(c)}\right) := \min_{\boldsymbol{m}^{(c)}} \mathrm{GW}_2^2\left(\boldsymbol{S}, \boldsymbol{m}, \boldsymbol{S}^{(c)}, \boldsymbol{m}^{(c)}\right),$$

and we can further find an optimal (w.r.t. the srGW divergence) by solving the following srGW barycenter problem with only one input $(\boldsymbol{S}, \boldsymbol{m})$, i.e.

$$\min_{\boldsymbol{S}^{(c)}} \mathrm{srGW}_2^2\left(\boldsymbol{S}, \boldsymbol{m}, \boldsymbol{S}^{(c)}\right). \tag{8}$$

We can then do the following transform to Equation (8) to reveal its connection with our proposed "Averaging" magnitude $\bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal$ for the coarsened similarity matrix $\boldsymbol{S}^{(c)}$.

$$\begin{aligned} \min_{\boldsymbol{S}^{(c)}} \mathrm{srGW}_2^2\left(\boldsymbol{S}, \boldsymbol{m}, \boldsymbol{S}^{(c)}\right) &= \min_{\boldsymbol{S}^{(c)}} \min_{\boldsymbol{m}^{(c)}} \mathrm{GW}_2^2\left(\boldsymbol{S}, \boldsymbol{m}, \boldsymbol{S}^{(c)}, \boldsymbol{m}^{(c)}\right) \\ &= \min_{\boldsymbol{S}^{(c)}} \min_{\boldsymbol{m}^{(c)}} \min_{\boldsymbol{T}\in\Pi\left(\boldsymbol{m}, \boldsymbol{m}^{(c)}\right)} \left\langle \boldsymbol{M}\left(\boldsymbol{S}, \boldsymbol{S}^{(c)}, \boldsymbol{T}\right), \boldsymbol{T}\left(\boldsymbol{m}, \boldsymbol{m}^{(c)}\right)\right\rangle \\ &= \min_{\boldsymbol{m}^{(c)}, \boldsymbol{T}\in\Pi\left(\boldsymbol{m}, \boldsymbol{m}^{(c)}\right)} \min_{\boldsymbol{S}^{(c)}} \left\langle \boldsymbol{M}\left(\boldsymbol{S}, \boldsymbol{S}^{(c)}, \boldsymbol{T}\right), \boldsymbol{T}\left(\boldsymbol{m}, \boldsymbol{m}^{(c)}\right)\right\rangle \\ &= \min_{\boldsymbol{T}\in\mathbb{R}_+^{N\times n} : \boldsymbol{T}\mathbf{1} = \boldsymbol{m}} \min_{\boldsymbol{S}^{(c)}} \left\langle \boldsymbol{M}\left(\boldsymbol{S}, \boldsymbol{S}^{(c)}, \boldsymbol{T}\right), \boldsymbol{T}\right\rangle. \end{aligned}$$

In the display above, we use $\boldsymbol{M}\left(\boldsymbol{S}, \boldsymbol{S}^{(c)}, \boldsymbol{T}\right)$ and $\boldsymbol{T}\left(\boldsymbol{m}, \boldsymbol{m}^{(c)}\right)$ to show the dependence terms of the cross-graph dissimilarity matrix $\boldsymbol{M}$ and the transport matrix $\boldsymbol{T}$. The last equation holds since for a given transport matrix $\boldsymbol{T}$ the new node mass distribution $\boldsymbol{m}^{(c)}$ is uniquely decided by $\boldsymbol{m}^{(c)} = \boldsymbol{T}^\intercal \mathbf{1}$.

Notably, there is a closed-form solution for the inner minimization problem $\min_{\boldsymbol{S}^{(c)}} \left\langle \boldsymbol{M}\left(\boldsymbol{S}, \boldsymbol{S}^{(c)}, \boldsymbol{T}\right), \boldsymbol{T}\right\rangle$. Peyré et al. (2016, Equation (14)) derive that the optimal $\boldsymbol{S}^{(c)}$ reads

$$\mathrm{diag}\left(\boldsymbol{m}^{(c)}\right)^{-1} \boldsymbol{T}^\intercal \boldsymbol{S} \boldsymbol{T} \mathrm{diag}\left(\boldsymbol{m}^{(c)}\right)^{-1}.$$

If we then enforce the restriction that the node mass transport must be performed in a clustering manner (i.e., the transport matrix $\boldsymbol{T} = \boldsymbol{W}\boldsymbol{C}_p^\intercal \in \mathbb{R}^{N\times n}$ for a certain membership matrix $\boldsymbol{C}_p$), we exactly have $\boldsymbol{S}^{(c)} = \bar{\boldsymbol{C}}_w \boldsymbol{S} \bar{\boldsymbol{C}}_w^\intercal$. The derivation above verifies the effectiveness of the "Averaging" magnitude we propose.

B.4. Comparing spectral difference Δ to spectral distance in Jin et al. (2020)

Jin et al. (2020) proposed to specify the graph coarsening plan by minimizing the following *full spectral distance* term (c.f. their Equation (8)):

$$\begin{aligned} SD_{\text{full}}\left(G, G^{(c)}\right) &= \sum_{i=1}^{k_1} |\boldsymbol{\lambda}(i) - \boldsymbol{\lambda}_c(i)| + \sum_{i=k_1+1}^{k_2} |\boldsymbol{\lambda}(i) - 1| + \sum_{i=k_2+1}^{N} |\boldsymbol{\lambda}(i) - \boldsymbol{\lambda}_c(i-N+n)| \\ &= \sum_{i=1}^{k_1} (\boldsymbol{\lambda}_c(i) - \boldsymbol{\lambda}(i)) + \sum_{i=k_2+1}^{N} (\boldsymbol{\lambda}(i) - \boldsymbol{\lambda}_c(i-N+n)) + \sum_{i=k_1+1}^{k_2} |\boldsymbol{\lambda}(i) - 1|, \end{aligned}$$

where $\boldsymbol{\lambda}(i)$'s and $\boldsymbol{\lambda}_c(i)$'s correspond to the eigenvalues (in an ascending order) of the normalized Laplacian $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{L}}^{(c)}$, and k_1 and k_2 are defined as $k_1 = \arg\max_i \{i : \boldsymbol{\lambda}_c(i) < 1\}$, $k_2 = N - n + k_1$. The last equation holds due to their Interlacing Property 4.1 (similar to Theorem D.1).

We note (i) the two "spectral" loss functions, Δ and SD_{full}, refer to different spectra. We leverage the spectrum of $C_w U C_w^T$ while they focus on graph Laplacians. Our framework is more general and takes node weights into consideration. (ii) They actually divided $\boldsymbol{\lambda}_c(i)$'s into two sets and respectively compared them to $\boldsymbol{\lambda}(i)$ and $\boldsymbol{\lambda}(i+N-n)$; the signs for $\lambda(i) - \lambda_c(i)$ and $\lambda(i+N-n) - \lambda_c(i)$ are thus different.

B.5. Time complexity of Algorithm 1

We first recall the complexity of Algorithm 1 stated in Section 5. Let T be the upper bound of the K-means iteration, the time complexity of Algorithm 1 is $\mathcal{O}\left(T\left(M + Nn\right)\right)$, where $M = \text{nnz}(\boldsymbol{S})$ is the number of non-zero elements in $\boldsymbol{S}$.

The cost mainly comes from the computation of the "distance" (7), which takes $\mathcal{O}(M)$ time to obtain the second term in Equation (7) and pre-compute the third term (independent of i); obtaining the exact $\text{dist}_j(i)$ for all the Nn (i, j) pairs requires another $\mathcal{O}(Nn)$ time. Compared to the previous spectral graph coarsening method (Jin et al., 2020), Algorithm 1 removes the partial sparse eigenvalue decomposition, which takes $\mathcal{O}\left(R(Mn + Nn^2)\right)$ time using Lanczos iteration with R restarts. The K-means part of spectral graph coarsening takes $\mathcal{O}(TNn^2)$ for a certain choice of the eigenvector feature dimension k_1 (Jin et al., 2020, Section 5.2), while weighted kernel K-means clustering nested in our algorithm can better utilize the sparsity in the similarity matrix.

C. Details of Experiments

We first introduce the hardware and the codebases we utilize in the experiments. The algorithms tested are all implemented in unoptimized Python code, and run with one core of a server CPU (Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz) on Ubuntu 18.04. Specifically, our method KGC is developed based on the (unweighted) kernel K-means clustering program provided by Ivashkin & Chebotarev (2016); for the other baseline methods, the variation methods VNGC and VEGC are implemented by Loukas (2019), and the spectrum-preserving methods MGC and SGC are implemented by Jin et al. (2020); The S-GWL method (Xu et al., 2019a; Chowdhury & Needham, 2021) is tested as well (can be found in the GitHub repository of this work), while this algorithm cannot guarantee that the number of output partition is the same as specified. For the codebases, we implement the experiments mainly using the code by Jin et al. (2020); Dwivedi et al. (2020), for graph classification and graph regression respectively. More omitted experimental settings in Section 6 will be introduced along the following subsections.

C.1. Details of GW distance approximation in Section 6.1

We compute the pair-wise GW_2 distance matrix $\boldsymbol{G}$ for graphs in PTC and IMDB, with the normalized signless Laplacians set as the similarity matrices. For the computation of the GW_2 distance, we mainly turn to the popular OT package `POT` (Flamary et al., 2021, Python Optimal Transport).

The omitted tables of average squared GW distance $GW_2^2(G^{(c)}, G)$ and average empirical bound gaps are presented in Tables 5 and 6.

Regarding time efficiency, we additionally remark our method KGC works on the dense graph matrix, even though it has a better theoretical complexity using a sparse matrix; for small graphs in MUTAG, directly representing them by dense

Table 5: Average squared GW distance $GW_2^2(G^{(c)}, G)$ on PTC and IMDB dataset.

Dataset	**Methods**	$c = 0.3\downarrow$	$c = 0.5\downarrow$	$c = 0.7\downarrow$	$c = 0.9\downarrow$
PTC	VNGC	0.05558	0.04880	0.03781	0.03326
	VEGC	0.03064	0.02352	0.01614	0.00927
	MGC	0.05290	0.04360	0.02635	0.00598
	SGC	0.03886	0.03396	0.02309	0.00584
	KGC	0.03332	0.02369	**0.01255**	**0.00282**
	KGC(A)	**0.03055**	**0.02346**	0.01609	0.00392
IMDB	VNGC	0.05139	0.05059	0.05043	0.05042
	VEGC	0.02791	**0.02106**	0.01170	0.00339
	MGC	**0.02748**	0.02116	0.01175	0.00339
	SGC	0.02907	0.02200	0.01212	0.00352
	KGC	0.02873	0.02111	**0.01137**	**0.00320**
	KGC(A)	**0.02748**	**0.02106**	0.01170	0.00337

Table 6: Average empirical bound gaps (in Theorem 4.2) on PTC and IMDB dataset.

Dataset	**Methods**	$c = 0.3$	$c = 0.5$	$c = 0.7$	$c = 0.9$
PTC	VNGC	0.06701	0.06671	0.05393	0.04669
	VEGC	0.06246	0.06129	0.04424	0.02577
	MGC	0.03203	0.03200	0.02167	0.00540
	SGC	0.04599	0.04156	0.02488	0.00554
	KGC	0.05145	0.05173	0.03530	0.00852
	KGC(A)	0.06519	0.06402	0.04702	0.00372
IMDB	VNGC	0.009281	0.00927	0.009278	0.009268
	VEGC	0.016879	0.016735	0.01636	0.008221
	MGC	0.017307	0.01663	0.016309	0.008179
	SGC	0.015719	0.015793	0.015934	0.008049
	KGC	0.016054	0.016679	0.016687	0.008347
	KGC(A)	0.017307	0.016735	0.01636	0.008177

matrices would even be faster, when a modern CPU is used.

C.2. Details of Laplacian spectrum preservation in Section 6.2

We mainly specify the evaluation metric for spectrum preservation in this subsection. Following the eigenvalue notation in Theorem 4.2, we define the top-5 eigenvalue relative error as $\frac{1}{5}\sum_{i=1}^{5}\frac{\lambda_i-\lambda_i^{(c)}}{\lambda_i}$. Here for all coarsening methods $\lambda_i-\lambda_i^{(c)}$ is always non-negative thanks to Poincaré separation theorem (Theorem D.1).

C.3. Details of Graph classification with Laplacian spectrum in Section 6.3

We remark the graph classification experiments are mainly adapted from the similar experiments in Jin et al. (2020, Section 6.1). For MUTAG and PTC datasets, we apply Algorithm 1 to $\boldsymbol{D}+\boldsymbol{A}$; for the other four datasets, we utilize the normalized signless Laplacian $\boldsymbol{I}_N+\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$.

C.4. Details of Graph regression with GCNs in Section 6.4

We mainly follow the GCN settings in Dwivedi et al. (2020). More detailed settings are stated as follows.

Graph regression. The procedure of graph regression is as follows: Taking a graph G as input, a GCN will return a graph embedding $\boldsymbol{y}_G\in\mathbb{R}^d$; Dwivedi et al. (2020) then pass $\boldsymbol{y}_G$ to an MLP and compute the prediction score $y_{\text{pred}}\in\mathbb{R}$ as

$$y_{\text{pred}}=\boldsymbol{P}\cdot\text{ReLU}(\boldsymbol{Q}\boldsymbol{y}_{\mathcal{G}}),$$

where $\boldsymbol{P}\in\mathbb{R}^{1\times d},\boldsymbol{Q}\in\mathbb{R}^{d\times d}$. They will then use $|y_{\text{pred}}-y|$, the L_1 loss (the MAE metric in Table 4) between the predicted score y_{pred} and the groundtruth score y, both in training and performance evaluation.

Data splitting. They apply a scaffold splitting (Hu et al., 2020) to the AQSOL dataset in the ratio $8:1:1$ to have $7831, 996$, and 996 samples for train, validation, and test sets.

Training hyperparameters. For the learning rate strategy across all GCN models, we follow the existing setting to choose the initial learning rate as 1×10^{-3}, the reduce factor is set as 0.5, and the stopping learning rate is 1×10^{-5}. Also, all the GCN models tested in our experiments share the same architecture—the network has 4 layers and 108442 tunable parameters.

As for the concrete usage of graph coarsening to GCNs, we discuss the main idea in Appendix C.4.1 and leave the technical details to Appendix C.4.2.

C.4.1. Application of graph coarsening to GCNs

Motivated by the GW setting, we discuss the application of graph coarsening to a prevailing and fundamental graph model—GCN[5]. After removing the bias term for simplicity and adapting the notations in this paper, we take a vanilla 1-layer GCN as an example and formulate it as

$$\boldsymbol{y}^\intercal=\frac{\boldsymbol{1}_N^\intercal}{N}\sigma\left(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}\boldsymbol{W}_{\text{GCN}}\right),$$

where $\boldsymbol{y}$ is the graph representation of a size-N graph associated with the adjacency matrix $\boldsymbol{A}$, σ is an arbitrary activation function, $\boldsymbol{H}$ is the embedding matrix for nodes in the graph, and $\boldsymbol{W}_{\text{GCN}}$ is the weight matrix for the linear transform in the layer. We take the mean operation $\frac{\boldsymbol{1}_N^\intercal}{N}$ in the GCN as an implication of even node weights (therefore no w subscript for coarsening matrices), and intuitively incorporate graph coarsening into the computation by solely replacing $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ with $\boldsymbol{\Pi}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{\Pi}$ and denote $\boldsymbol{y}^{(c)}$ as the corresponding representation. We have

$$\left(\boldsymbol{y}^{(c)}\right)^\intercal=\boldsymbol{c}^\intercal\sigma\left(\bar{\boldsymbol{C}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{C}_p\bar{\boldsymbol{C}}\boldsymbol{H}\boldsymbol{W}_{\text{GCN}}\right), \tag{9}$$

and the graph matrix $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ is supposed to be coarsened as $\bar{\boldsymbol{C}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{C}_p$, which can be extended to multiple layers. Due to the space limit, we defer the derivation to Appendix C.4.2 and solely leave some remarks here.

[5]We omit the introduction to GCN here and refer readers to Kipf & Welling (2017) for more details.

The propagation above is guided by the GW setting along this paper: even the nodes in the original graph are equally-weighted, the supernodes after coarsening can have different weights, which induces the new readout operation $\boldsymbol{c}^\intercal$ ($\boldsymbol{c}$ is the mass vector of the supernodes). Furthermore, an obvious difference between Equation (9) and previous designs (Huang et al., 2021; Cai et al., 2021) of applying graph coarsening to GNNs is the asymmetry of the coarsened graph matrix $\bar{\boldsymbol{C}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{C}_p$. The design (9) is tested for the graph regression experiment in Section 6.4. More technical details are introduced in the subsequent subsection.

C.4.2. Derivation of the GCN propagation in the GW setting

We consider a general layer-L GCN used by Dwivedi et al. (2020). We first recall the definition (normalization modules are omitted for simplicity):

$$\begin{aligned}
&\boldsymbol{y}^\intercal = \frac{\mathbf{1}_N^\intercal}{N}\boldsymbol{H}^{(L)},\\
&\boldsymbol{H}^{(l)} = \sigma(\boldsymbol{Z}^{(l)}), \quad \boldsymbol{Z}^{(l)} = \left(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right)\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}, \quad \forall l \in [L],\\
&\boldsymbol{H}^{(0)} := \boldsymbol{X}, \quad \text{the embedding matrix for each nodes,}
\end{aligned}$$

where σ is an activation function and from now on we will abuse $\boldsymbol{P}$ here to denote $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$; $\boldsymbol{H}^{(l)}$ is the embedding matrix of the graph nodes in the l-th layer, and $\boldsymbol{W}^{(l)}$ is the weight matrix of the same layer.

To apply the coarsened graph, we enforce the following regulations:

$$\boldsymbol{H}^{(c,l)} = \sigma(\boldsymbol{Z}^{(c,l)}), \quad \boldsymbol{Z}^{(c,l)} = (\boldsymbol{\Pi}\boldsymbol{P}\boldsymbol{\Pi})\,\boldsymbol{H}^{(c,l-1)}\boldsymbol{W}^{(l)}, \quad \forall l \in [L].$$

To reduce the computation in node aggregation, we utilize the two properties that $\boldsymbol{\Pi} = \boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}$ and $\sigma\left(\boldsymbol{C}_p^\intercal\boldsymbol{B}\right) = \boldsymbol{C}_p^\intercal\sigma\left(\boldsymbol{B}\right)$, for any element-wise activation function and matrix $\boldsymbol{B}$ with a proper shape (considering $\boldsymbol{C}_p^\intercal$ simply "copy" the rows from $\boldsymbol{B}$); for the top two layers, we then have

$$\begin{aligned}
\boldsymbol{y}^\intercal &= \frac{\mathbf{1}_N^\intercal}{N}\sigma(\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{\Pi}\boldsymbol{H}^{(c,L-1)}\boldsymbol{W}^{(L)}) = \frac{\mathbf{1}_N^\intercal}{N}\boldsymbol{C}_p^\intercal\sigma\left(\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{H}^{(c,L-1)}\boldsymbol{W}^{(L)}\right),\\
\boldsymbol{H}^{(c,L-1)} &= \sigma\left(\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{H}^{(c,L-2)}\boldsymbol{W}^{(L-1)}\right) = \boldsymbol{C}_p^\intercal\sigma\left(\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{H}^{(c,L-2)}\boldsymbol{W}^{(L-1)}\right).
\end{aligned}$$

Note $\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{C}_p^\intercal = \boldsymbol{C}_p^\intercal$; for the top two layers we finally obtain

$$\boldsymbol{y}^\intercal = \frac{\mathbf{1}_N^\intercal}{N}\boldsymbol{C}_p^\intercal\sigma\left(\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{H}^{(c,L-1)}\boldsymbol{W}^{(L)}\right) = \boldsymbol{c}^\intercal\sigma\left(\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\sigma\left(\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}\boldsymbol{H}^{(c,L-2)}\boldsymbol{W}^{(L-1)}\right)\boldsymbol{W}^{(L)}\right),$$

implying that we can replace $\boldsymbol{P}$ with $\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal$ in the propagation of the top two layers. The trick can indeed be repeated for each layer, and specifically, in the bottom layer we can have

$$\boldsymbol{H}^{(c,1)} = \boldsymbol{C}_p^\intercal\sigma\left[\left(\bar{\boldsymbol{C}}\boldsymbol{P}\boldsymbol{C}_p^\intercal\right)\left(\bar{\boldsymbol{C}}\boldsymbol{H}^{(0)}\right)\boldsymbol{W}^{(1)}\right],$$

which well corresponds to the node weight concept in the GW setting: $\bar{\boldsymbol{C}}\boldsymbol{H}^{(0)}$ uses the average embedding of the nodes within the cluster to represent the coarsened centroid. We then finish the justification of the new GCN propagation in Equation (9).

D. Useful Facts

D.1. Poincaré separation theorem

For convenience of the reader, we repeat Poincaré separation theorem (Bellman, 1997) in this subsection.

Theorem D.1 (Poincaré separation theorem)**.** *Let $\boldsymbol{A}$ be an $N \times N$ real symmetric matrix and $\boldsymbol{C}$ an $n \times N$ semi-orthogonal matrix such that $\boldsymbol{C}\boldsymbol{C}^\intercal = \boldsymbol{I}_n$. Denote by $\lambda_i, i = 1, 2, \ldots, N$ and $\lambda_i^{(c)}, i = 1, 2, \ldots, n$ the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^\intercal$, respectively (in descending order). We have*

$$\lambda_i \geq \lambda_i^{(c)} \geq \lambda_{N-n+i}, \tag{10}$$

D.2. Ruhe's trace inequality

For convenience of the reader, Ruhe's trace inequality (Ruhe, 1970) is stated as follows.

Lemma D.2 (Ruhe's trace inequality)**.** *If $\boldsymbol{A}, \boldsymbol{B}$ are both $N \times N$ PSD matrices with eigenvalues,*

$$\lambda_1 \geq \cdots \geq \lambda_N \geq 0, \quad \nu_1 \geq \cdots \geq \nu_N \geq 0,$$

then

$$\sum_{i=1}^{N} \lambda_i \nu_{N-i+1} \leq \operatorname{tr}(AB) \leq \sum_{i=1}^{N} \lambda_i \nu_i .$$

E. Proof of Theorem 4.2 and Theorem 4.4

We will first prove an intermediate result Lemma E.1 for coarsened graph $G_1^{(c)}$ and un-coarsend graph G_2 to introduce necessary technical tools for Theorem 4.2 and Theorem 4.4. The ultimate proof of Theorem 4.2 and Theorem 4.4 will be stated in Appendix E.2 and Appendix E.3 respectively.

E.1. Lemma E.1 and Its Proof

We first give the complete statement of Lemma E.1. In Lemma E.1, we consider the case in which only graph G_1 is coarsened, and the notations are slightly simplified: when the context is clear, the coarsening-related terms without subscripts specific to graphs, e.g. $\boldsymbol{C}_p, \bar{\boldsymbol{C}}_w$, are by default associated with G_1 unless otherwise announced. We follow the simplified notation in the statement and proof of Theorem 4.2, which focuses on solely $\mathrm{GW}_2(G^{(c)}, G)$ and the indices $1, 2$ are not involved. For Theorem 4.4, we will explicitly use specific subscripts, such as $\boldsymbol{C}_{p,1}, \bar{\boldsymbol{C}}_{w,2}$, for disambiguation.

Lemma E.1. *Let $\boldsymbol{P} = \boldsymbol{W}_1^{-\frac{1}{2}} \boldsymbol{T}^* \boldsymbol{W}_2^{-\frac{1}{2}}$. If both $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are PSD, we have*

$$\begin{aligned}|\mathrm{GW}_2^2(G_1, G_2) - \mathrm{GW}_2^2(G_1^{(c)}, G_2)| \leq \max \Bigg\{ &\lambda_{N_1-n_1+1} \sum_{i=1}^{n_1} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},n_1}, \\ &2\left(\nu_{N_1-n_1+1} \sum_{i=1}^{n_1} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},\boldsymbol{V},n_1}\right)\Bigg\}. \end{aligned} \tag{11}$$

Here, $\lambda_1^{(c)} \geq \cdots \geq \lambda_n^{(c)}$ are eigenvalues of $\boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w$, $\boldsymbol{U} = \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}}$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N_1}$, and $\boldsymbol{V} = \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal$ with eigenvalues $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_{N_1}$, and we let $C_{\boldsymbol{U},n_1} := \sum_{i=1}^{n_1} \lambda_i(\lambda_i - \lambda_{N_1-n_1+i}) + \sum_{i=n_1+1}^{N_1} \lambda_i^2$ and $C_{\boldsymbol{U},\boldsymbol{V},n_1} := \sum_{i=1}^{n_1} \lambda_i \left(\nu_i - \nu_{N_1-i+1}\right) + \sum_{i=n_1+1}^{N_1} \lambda_i \nu_i$ be two non-negative constants.

Remark E.2. Take $G_2 = G_1$, and we have $\boldsymbol{T}^* = \boldsymbol{W}_1$ which implies $\boldsymbol{P} = \boldsymbol{I}_N$ and $\boldsymbol{V} = \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}} = \boldsymbol{U}$. This directly leads to the bound in Theorem 4.2 though with an additional factor 2. In Appendix E.2 we will show the approach to obtain a slightly tighter bound removing the unnecessary factor 2.

To illustrate our idea more clearly, we will start from the following decomposition of GW_2 distance. The detailed proofs of all the lemmas in this section will be provided to Appendix E.4 for the readers interested.

Lemma E.3. *For any two graphs G_1 and G_2, we have*

$$\mathrm{GW}_2^2(G_1, G_2) = I_1 + I_2 - 2 I_3,$$

where

$$I_1 = \operatorname{Tr}\left(\left(\boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}}\right)\left(\boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}}\right)\right), \qquad I_2 = \operatorname{Tr}\left(\left(\boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}}\right)\left(\boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}}\right)\right)$$

do not depend on the choice of transport map, while

$$I_3 = \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal\right)$$

requires the optimal transport map $\boldsymbol{T}^*$ from the graph G_1 to the graph G_2.

Replacing the G_1-related terms in Equation (15) with their $G_1^{(c)}$ counterparts, we know that the distance between $G_1^{(c)}$ and G_2 is $\mathrm{GW}_2(G_1^{(c)}, G_2) = I_1' + I_2 - 2I_3'$, where:

$$I_1' = \operatorname{Tr}\left(\left[\left(\boldsymbol{W}_1^{(c)}\right)^{\frac{1}{2}} \boldsymbol{S}_1^{(c)} \left(\boldsymbol{W}_1^{(c)}\right)^{\frac{1}{2}}\right]^2\right), \qquad I_3' = \operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_2 \left(\boldsymbol{T}_{co}^*\right)^\intercal\right). \tag{12}$$

$\boldsymbol{T}_{co}^* \in \Pi(\mu_1^{(c)}, \mu_2)$ (co represents the transport from the "c"oarsened graph to the "o"riginal graph) here is the optimal transport matrix induced by the GW distance and $\boldsymbol{S}_1^{(c)} := \bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w^\intercal$.

The key step to preserve the GW distance is to control the difference $|I_1 - I_1'|$ and $|I_3 - I_3'|$, since $I_2 = I_2'$ will cancel each other. We will start from the bound of $|I_1 - I_1'|$.

Lemma E.4. *Let $\boldsymbol{U} := \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}}$. If the similarity matrix $\boldsymbol{S}_1 \in \mathbb{R}^{N \times N}$ is PSD, we have*

$$0 \le I_1 - I_1' \le \lambda_{N-n+1} \sum_{i=1}^{n} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},n}.$$

Here, $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_N$ are the eigenvalues of $\boldsymbol{U}$, and $C_{\boldsymbol{U},n} := \sum_{i=1}^{n} \lambda_i(\lambda_i - \lambda_{N-n+i}) + \sum_{i=n+1}^{N} \lambda_i^2$ is non-negative.

We can similarly bound $I_3 - I_3'$ with an additional tool Ruhe's trace inequality (a variant of Von Neumann's trace inequality specific to PSD matrices, c.f. Appendix D.2).

Lemma E.5. *Let $\boldsymbol{P} = \boldsymbol{W}_1^{-\frac{1}{2}} \boldsymbol{T}^* \boldsymbol{W}_2^{-\frac{1}{2}}$ be the normalized optimal transport matrix, and $\boldsymbol{V} := \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal$ with eigenvalues $\nu_1 \ge \nu_2 \ge \cdots \ge \nu_N$. If both $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are PSD, we have*

$$0 \le I_3 - I_3' \le \nu_{N-n+1} \sum_{i=1}^{n} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},\boldsymbol{V},n}.$$

Here, $C_{\boldsymbol{U},\boldsymbol{V},n} := \sum_{i=1}^{n} \lambda_i \left(\nu_i - \nu_{N-i+1}\right) + \sum_{i=n+1}^{N} \lambda_i \nu_i$ is non-negative.

Now we have

$$|\mathrm{GW}_2^2(G_1, G_2) - \mathrm{GW}_2^2(G_1^{(c)}, G_2)| = |I_1 - I_1' + 2(I_3' - I_3)| \le \max\{I_1 - I_1', 2(I_3 - I_3')\}$$

considering that $I_1 - I_1' \ge 0$ and $I_3' - I_3 \le 0$. Then, combining all the pieces above yields the bound in Lemma E.1.

E.2. Proof of Theorem 4.2

With the above dissection of the terms I_1', I_3', we can now give a finer control of the distance $\mathrm{GW}_2^2(G_1^{(c)}, G_1)$. We first expand $\mathrm{GW}_2^2(G_1^{(c)}, G_1)$ as

$$\mathrm{GW}_2^2(G_1^{(c)}, G_1) = I_1 + I_1' - 2 \operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_1 \left(\boldsymbol{T}_{co}^*\right)^\intercal\right),$$

where the notation $\boldsymbol{T}_{co}^*$ is now abused as the optimal transport matrix induced by $\mathrm{GW}_2(G_1^{(c)}, G_1)$. Applying the optimality inequality in Lemma E.7, we have

$$\mathrm{GW}_2^2(G_1^{(c)}, G_1) \le I_1' + I_1 - 2 \operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{C}_p \boldsymbol{W}_1 \boldsymbol{S}_1 \boldsymbol{W}_1 \boldsymbol{C}_p^\intercal\right),$$

where we remark $\boldsymbol{C}_p \boldsymbol{W}_1$ is a qualified transport matrix for $G_1^{(c)}$ and G_1.

To further simplify the upper bound above, we show the equivalence between I_1' and $\operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{C}_p \boldsymbol{W}_1 \boldsymbol{S}_1 \boldsymbol{W}_1 \boldsymbol{C}_p^\intercal\right)$ as follows:

$$\begin{aligned}\operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{C}_p \boldsymbol{W}_1 \boldsymbol{S}_1 \boldsymbol{W}_1 \boldsymbol{C}_p^\intercal\right) &= \operatorname{Tr}\left(\bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w \boldsymbol{C}_p \boldsymbol{W}_1 \boldsymbol{S}_1 \boldsymbol{W}_1 \boldsymbol{C}_p^\intercal\right) = \operatorname{Tr}\left(\boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w \boldsymbol{U}\right) = \operatorname{Tr}\left(\boldsymbol{\Pi}_w \boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w \boldsymbol{\Pi}_w \boldsymbol{U}\right) \\ &= \operatorname{Tr}\left(\boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w \boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w\right) = I_1'.\end{aligned}$$

Combing the above pieces together, we obtain

$$\mathrm{GW}_2^2(G_1^{(c)}, G_1) \le I_1 + I_1' - 2I_1' \le \lambda_{N-n+1} \sum_{i=1}^{n} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},n},$$

which uses Lemma E.4 for the last inequality. The proof of Theorem 4.2 is now complete.

Remark E.6. Theorem 4.2 can be leveraged to directly control $\left|\mathrm{GW}_2(G_1^{(c)}, G_2^{(c)}) - \mathrm{GW}_2(G_1, G_2)\right|$. Note that GW_2 is a pseudo-metric and satisfies triangular inequality (Chowdhury & Mémoli, 2019), which implies

$$\begin{aligned}\mathrm{GW}_2(G_1^{(c)}, G_2^{(c)}) - \mathrm{GW}_2(G_1, G_2) &\le \mathrm{GW}_2(G_1^{(c)}, G_1) + \mathrm{GW}_2(G_1, G_2) + \mathrm{GW}_2(G_2, G_2^{(c)}) - \mathrm{GW}_2(G_1, G_2) \\ &= \mathrm{GW}_2(G_1^{(c)}, G_1) + \mathrm{GW}_2(G_2, G_2^{(c)}) \\ &\le \sqrt{2}\left(\mathrm{GW}_2^2(G_1^{(c)}, G_1) + \mathrm{GW}_2^2(G_2, G_2^{(c)})\right)^{\frac{1}{2}},\end{aligned}$$

and similarly we have

$$\mathrm{GW}_2(G_1, G_2) - \mathrm{GW}_2(G_1^{(c)}, G_2^{(c)}) \le \mathrm{GW}_2(G_1^{(c)}, G_1) + \mathrm{GW}_2(G_2, G_2^{(c)}) \le \sqrt{2}\left(\mathrm{GW}_2^2(G_1^{(c)}, G_1) + \mathrm{GW}_2^2(G_2, G_2^{(c)})\right)^{\frac{1}{2}}.$$

This implies

$$\begin{aligned}&\left|\mathrm{GW}_2(G_1^{(c)}, G_2^{(c)}) - \mathrm{GW}_2(G_1, G_2)\right| \le \sqrt{2}\left(\mathrm{GW}_2^2(G_1^{(c)}, G_1) + \mathrm{GW}_2^2(G_2, G_2^{(c)})\right)^{\frac{1}{2}} \\ \le &\sqrt{2}\left(\lambda_{1,N_1-n_1+1} \sum_{i=1}^{n_1} \left(\lambda_{1,i} - \lambda_{1,i}^{(c)}\right) + C_{\boldsymbol{U}_1,n_1} + \lambda_{2,N_2-n_2+1} \sum_{i=1}^{n_2} \left(\lambda_{2,i} - \lambda_{2,i}^{(c)}\right) + C_{\boldsymbol{U}_2,n_2}\right)^{\frac{1}{2}}.\end{aligned}$$

Here, the last inequality is due to Theorem 4.2. We comment the above result obtained from a direct application of triangle inequality is indeed weaker than the result stated in Theorem 4.4; we will then devote the remaining part of this section to the proof thereof.

E.3. Proof of Theorem 4.4

For one side of the result, we can follow the derivation in Lemma E.3 and apply Theorem 4.2 to have

$$\begin{aligned}\mathrm{GW}_2^2(G_1, G_2) - \mathrm{GW}_2^2(G_1^{(c)}, G_2^{(c)}) =& I_1 - I_1' + I_2 - I_2' + 2(I_3' - I_3) \le I_1 - I_1' + I_2 - I_2' \\ \le& \lambda_{N_1-n_1+1} \sum_{i=1}^{n_1} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U}_1,n_1} + \lambda_{N_2-n_2+1} \sum_{i=1}^{n_2} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U}_2,n_2},\end{aligned}$$

where we recall $I_3 := \mathrm{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 (\boldsymbol{T}^*)^\intercal\right)$ and I_3' now is $\mathrm{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{T}_{cc}^* \boldsymbol{S}_2^{(c)} (\boldsymbol{T}_{cc}^*)^\intercal\right)$ (T_{cc}^* is the optimal transport matrix for $G_1^{(c)}$ and $G_2^{(c)}$).

For the other side, we still decompose the object $\mathrm{GW}_2^2(G_1^{(c)}, G_2^{(c)}) - \mathrm{GW}_2^2(G_1, G_2)$ as $(I_1' - I_1) + (I_2' - I_2) + 2(I_3 - I_3')$, and similarly we can follow Lemma E.4 to bound the first two terms. The next task is to disassemble $I_3 - I_3'$.

We first prepare some notations for the analysis. To clarify the affiliation, we redefine $\boldsymbol{U}_1 := \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}}$ with eigenvalues $\lambda_{1,1} \ge \lambda_{1,2} \ge \cdots \ge \lambda_{1,N_1}$, $\boldsymbol{V}_1 = \boldsymbol{P}\boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal$ with eigenvalues $\nu_{1,1} \ge \nu_{1,2} \ge \cdots \ge \nu_{1,N_1}$[6], $\boldsymbol{U}_2 = \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}}$ with eigenvalues $\lambda_{2,1} \ge \lambda_{2,2} \ge \cdots \ge \lambda_{2,N_2}$, $\boldsymbol{V}_2 = \boldsymbol{P}^\intercal \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P}$ with eigenvalues $\nu_{2,1} \ge \nu_{2,2} \ge \cdots \ge \nu_{2,N_2}$, and similarly re-introduce $C_{p,1}, \bar{\boldsymbol{C}}_{w,1}, \boldsymbol{C}_{w,1}, C_{p,2}, \bar{\boldsymbol{C}}_{w,2}, \boldsymbol{C}_{w,2}$, and

$$\boldsymbol{\Pi}_{w,1} := \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{C}_{p,1}^\intercal \bar{\boldsymbol{C}}_{w,1} \boldsymbol{W}_1^{-\frac{1}{2}} = \boldsymbol{C}_{w,1}^\intercal \boldsymbol{C}_{w,1}, \qquad \boldsymbol{\Pi}_{w,2} := \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{C}_{p,2}^\intercal \bar{\boldsymbol{C}}_{w,2} \boldsymbol{W}_2^{-\frac{1}{2}} = \boldsymbol{C}_{w,2}^\intercal \boldsymbol{C}_{w,2}.$$

[6]We index $\boldsymbol{P}\boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal$ as $\boldsymbol{V}_1$ since it is associated with $\boldsymbol{U}_1$ and is an $N_1 \times N_1$ matrix.

Recalling Lemma E.7, we can analogously obtain $I_3' - I_3 \leq 0$; replacing $\boldsymbol{T}_{cc}^*$ with $\boldsymbol{C}_{p,1}\boldsymbol{T}^*\boldsymbol{C}_{p,2}^\intercal$, we have

$$\begin{aligned} I_3 - I_3' &\leq \operatorname{Tr}\left(\boldsymbol{S}_1\boldsymbol{T}^*\boldsymbol{S}_2\left(\boldsymbol{T}^*\right)^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{S}_1^{(c)}\boldsymbol{C}_{p,1}\boldsymbol{T}^*\boldsymbol{C}_{p,2}^\intercal\boldsymbol{S}_2^{(c)}\boldsymbol{C}_{p,2}\left(\boldsymbol{T}^*\right)^\intercal\boldsymbol{C}_{p,1}^\intercal\right) \\ &= \operatorname{Tr}\left(\boldsymbol{U}_1\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{\Pi}_{w,1}\boldsymbol{U}_1\boldsymbol{\Pi}_{w,1}\boldsymbol{P}\boldsymbol{\Pi}_{w,2}\boldsymbol{U}_2\boldsymbol{\Pi}_{w,2}\boldsymbol{P}^\intercal\right), \end{aligned}$$

where we apply the same derivation as in Equation (16) to obtain the second line above. For simplicity, we let $\boldsymbol{U}_1^\Pi := \boldsymbol{\Pi}_{w,1}\boldsymbol{U}_1\boldsymbol{\Pi}_{w,1}$ and $\boldsymbol{U}_2^\Pi := \boldsymbol{\Pi}_{w,2}\boldsymbol{U}_2\boldsymbol{\Pi}_{w,2}$. We can now bound $I_3 - I_3'$ as

$$\begin{aligned} I_3 - I_3' &\leq \operatorname{Tr}\left(\boldsymbol{U}_1\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{U}_1^\Pi\boldsymbol{P}\boldsymbol{U}_2^\Pi\boldsymbol{P}^\intercal\right) \\ &= \operatorname{Tr}\left(\boldsymbol{U}_1\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{U}_1^\Pi\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right) + \operatorname{Tr}\left(\boldsymbol{U}_1^\Pi\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{U}_1^\Pi\boldsymbol{P}\boldsymbol{U}_2^\Pi\boldsymbol{P}^\intercal\right) \\ &= \operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right] + \operatorname{Tr}\left[\boldsymbol{U}_1^\Pi\boldsymbol{P}\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right] \\ &= \operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{P}\boldsymbol{U}_2\boldsymbol{P}^\intercal\right] + \operatorname{Tr}\left[\boldsymbol{U}_1\boldsymbol{P}\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right] \\ &\qquad - \operatorname{Tr}\left[\boldsymbol{U}_1\boldsymbol{P}\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right] + \operatorname{Tr}\left[\boldsymbol{U}_1^\Pi\boldsymbol{P}\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right] \\ &= \operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{V}_1\right] + \operatorname{Tr}\left[\boldsymbol{V}_2\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\right] - \operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{P}\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right]. \end{aligned} \tag{13}$$

The first two terms in the last line above can be directly addressed by Lemma E.5; for the last term, we can bound it as

$$\begin{aligned} \left|\operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{P}\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right]\right| &\leq \left\|\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{P}\right\|_F \left\|\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\boldsymbol{P}^\intercal\right\|_F \\ &\leq \left\|\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right\|_F \|\boldsymbol{P}\| \left\|\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right\|_F \|\boldsymbol{P}\| \leq \left\|\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right\|_F \left\|\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right\|_F, \end{aligned}$$

where Lemma E.8 shows $\|\boldsymbol{P}\| \leq 1$ and justifies the last inequality.

We now give another useful fact that $I_1 - I_1' = \left\|\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right\|_F^2$:

$$\begin{aligned} I_1 - I_1' &\overset{(i)}{=} \operatorname{Tr}\left[\boldsymbol{U}_1^2 - \left(\boldsymbol{U}_1^\Pi\right)^2\right] = \operatorname{Tr}\left[\boldsymbol{U}_1^2 + \left(\boldsymbol{U}_1^\Pi\right)^2\right] - 2\operatorname{Tr}\left[\left(\boldsymbol{\Pi}_{w,1}\boldsymbol{U}_1\boldsymbol{\Pi}_{w,1}\right)^2\right] \\ &= \operatorname{Tr}\left[\boldsymbol{U}_1^2 + \left(\boldsymbol{U}_1^\Pi\right)^2\right] - \operatorname{Tr}\left(\boldsymbol{\Pi}_{w,1}\boldsymbol{U}_1\boldsymbol{\Pi}_{w,1}\boldsymbol{U}_1\right) - \operatorname{Tr}\left(\boldsymbol{U}_1\boldsymbol{\Pi}_{w,1}\boldsymbol{U}_1\boldsymbol{\Pi}_{w,1}\right) = \operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)^2\right] \\ &= \left\|\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right\|_F^2, \end{aligned}$$

where (i) comes from Equation (17). Analogously we have $I_2 - I_2' = \left\|\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right\|_F^2$. Combining the above pieces together we can bound the object as

$$\begin{aligned} \mathrm{GW}_2^2(G_1^{(c)}, G_2^{(c)}) - \mathrm{GW}_2^2(G_1, G_2) \leq &- \left\|\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right\|_F^2 - \left\|\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right\|_F^2 + \\ &2\operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{V}_1\right] + 2\operatorname{Tr}\left[\boldsymbol{V}_2\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\right] + 2\left\|\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right\|_F\left\|\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right\|_F \\ \leq &2\operatorname{Tr}\left[\left(\boldsymbol{U}_1 - \boldsymbol{U}_1^\Pi\right)\boldsymbol{V}_1\right] + 2\operatorname{Tr}\left[\boldsymbol{V}_2\left(\boldsymbol{U}_2 - \boldsymbol{U}_2^\Pi\right)\right] \\ \overset{(i)}{\leq} &2 \cdot \left[\nu_{1,N_1-n_1+1}\sum_{i=1}^{n_1}\left(\lambda_{1,i} - \lambda_{1,i}^{(c)}\right) + C_{\boldsymbol{U}_1,\boldsymbol{V}_1,n_1} + \right. \\ &\left. \nu_{2,N_2-n_2+1}\sum_{i=1}^{n_2}\left(\lambda_{2,i} - \lambda_{2,i}^{(c)}\right) + C_{\boldsymbol{U}_2,\boldsymbol{V}_2,n_2}\right], \end{aligned}$$

where we reuse Inequality (17) to attain (i). The proof of Theorem 4.4 is then completed.

E.4. Proof of some technical results

E.4.1. Proof of Lemma E.3

Proof. Following the definition in Equation (1), we rewrite the GW distance in the trace form and have

$$\begin{aligned} \langle\boldsymbol{M}, \boldsymbol{T}^*\rangle &= \left\langle f_1(\boldsymbol{S}_1)\boldsymbol{m}_1\boldsymbol{1}_{N_2}^\intercal, \boldsymbol{T}^*\right\rangle + \left\langle\boldsymbol{1}_{N_1}\boldsymbol{m}_2^\intercal f_2(\boldsymbol{S}_2)^\intercal, \boldsymbol{T}^*\right\rangle - \left\langle h_1(\boldsymbol{S}_1)\boldsymbol{T}^*h_2(\boldsymbol{S}_2)^\intercal, \boldsymbol{T}^*\right\rangle \\ &= \operatorname{Tr}\left(f_1(\boldsymbol{S}_1)\boldsymbol{m}_1\boldsymbol{1}_{N_2}^\intercal\left(\boldsymbol{T}^*\right)^\intercal\right) + \operatorname{Tr}\left(f_2(\boldsymbol{S}_2)\boldsymbol{m}_2\boldsymbol{1}_{N_1}^\intercal\boldsymbol{T}^*\right) - \operatorname{Tr}\left(h_1(\boldsymbol{S}_1)\boldsymbol{T}^*h_2(\boldsymbol{S}_2)^\intercal\left(\boldsymbol{T}^*\right)^\intercal\right) \\ &= \operatorname{Tr}\left(\boldsymbol{m}_1^\intercal f_1(\boldsymbol{S}_1)\boldsymbol{m}_1\right) + \operatorname{Tr}\left(\boldsymbol{m}_2^\intercal f_2(\boldsymbol{S}_2)\boldsymbol{m}_2\right) - \operatorname{Tr}\left(h_1(\boldsymbol{S}_1)\boldsymbol{T}^*h_2(\boldsymbol{S}_2)^\intercal\left(\boldsymbol{T}^*\right)^\intercal\right); \end{aligned} \tag{14}$$

the third equation above holds because for any $\boldsymbol{T} \in \Pi(\mu_1, \mu_2)$, $\boldsymbol{T}\mathbf{1}_{N_2} = \boldsymbol{m}_1$ and $\boldsymbol{T}^\intercal \mathbf{1}_{N_1} = \boldsymbol{m}_2$.

In the classical square loss case, we can immediately have $f_1(\boldsymbol{S}_1) = \boldsymbol{S_1} \odot \boldsymbol{S_1}$, $f_2(\boldsymbol{S}_2) = \boldsymbol{S_2} \odot \boldsymbol{S_2}$, $h_1(\boldsymbol{S}_1) = \boldsymbol{S}_1$ and $h_2(\boldsymbol{S}_2) = 2\boldsymbol{S}_2$, where $\odot$ denotes the Hadamard product of two matrices with the same size. We can accordingly expand the first term in Equation (14) as

$$\operatorname{Tr}\left(\boldsymbol{m}_1^\intercal f_1(\boldsymbol{S}_1)\boldsymbol{m}_1\right) = \boldsymbol{m}_1^\intercal \left(\boldsymbol{S_1} \odot \boldsymbol{S_1}\right)\boldsymbol{m}_1 = \operatorname{Tr}\left(\boldsymbol{S_1}^\intercal \operatorname{diag}(\boldsymbol{m_1})\boldsymbol{S_1}\operatorname{diag}(\boldsymbol{m_1})\right),$$

in which the proof of the last equation is provided in a summary sheet by Minka (2000). We note $\boldsymbol{W}_1 := \operatorname{diag}(\boldsymbol{m_1})$ and $\boldsymbol{S}_1$ is constructed symmetric; combining the pieces above we have

$$\operatorname{Tr}\left(\boldsymbol{m}_1^\intercal f_1(\boldsymbol{S}_1)\boldsymbol{m}_1\right) = \operatorname{Tr}\left(\left(\boldsymbol{W}_1^{\frac{1}{2}}\boldsymbol{S_1}\boldsymbol{W}_1^{\frac{1}{2}}\right)\left(\boldsymbol{W}_1^{\frac{1}{2}}\boldsymbol{S_1}\boldsymbol{W}_1^{\frac{1}{2}}\right)\right),$$

and similarly we obtain $\operatorname{Tr}\left(\boldsymbol{m}_2^\intercal f_2(\boldsymbol{S}_2)\boldsymbol{m}_2\right) = \operatorname{Tr}\left(\left(\boldsymbol{W}_2^{\frac{1}{2}}\boldsymbol{S_2}\boldsymbol{W}_2^{\frac{1}{2}}\right)\left(\boldsymbol{W}_2^{\frac{1}{2}}\boldsymbol{S_2}\boldsymbol{W}_2^{\frac{1}{2}}\right)\right)$. The GW distance (14) can be therefore represented as

$$\underbrace{\operatorname{Tr}\left(\left(\boldsymbol{W}_1^{\frac{1}{2}}\boldsymbol{S_1}\boldsymbol{W}_1^{\frac{1}{2}}\right)\left(\boldsymbol{W}_1^{\frac{1}{2}}\boldsymbol{S_1}\boldsymbol{W}_1^{\frac{1}{2}}\right)\right)}_{=:I_1} + \underbrace{\operatorname{Tr}\left(\left(\boldsymbol{W}_2^{\frac{1}{2}}\boldsymbol{S_2}\boldsymbol{W}_2^{\frac{1}{2}}\right)\left(\boldsymbol{W}_2^{\frac{1}{2}}\boldsymbol{S_2}\boldsymbol{W}_2^{\frac{1}{2}}\right)\right)}_{=:I_2} - 2\underbrace{\operatorname{Tr}\left(\boldsymbol{S}_1\boldsymbol{T}^*\boldsymbol{S}_2\left(\boldsymbol{T}^*\right)^\intercal\right)}_{=:I_3}. \tag{15}$$

$\diamondsuit$

E.4.2. Proof of Lemma E.4

Proof. First, recall that $\boldsymbol{\Pi}_w = \boldsymbol{W}_1^{\frac{1}{2}}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}_w\boldsymbol{W}_1^{-\frac{1}{2}} = \boldsymbol{C}_w^\intercal\boldsymbol{C}_w$. So, we have

$$\begin{aligned}
I_1' &= \operatorname{Tr}\left(\left[\left(\boldsymbol{C}_p\boldsymbol{W}_1\boldsymbol{C}_p^\intercal\right)^{\frac{1}{2}}\left(\bar{\boldsymbol{C}}_w\boldsymbol{S}_1\bar{\boldsymbol{C}}_w^\intercal\right)\left(\boldsymbol{C}_w\boldsymbol{W}_1\boldsymbol{C}_w^\intercal\right)^{\frac{1}{2}}\right]^2\right)\\
&= \operatorname{Tr}\left[\left(\boldsymbol{C}_p\boldsymbol{W}_1\boldsymbol{C}_p^\intercal\right)\left(\bar{\boldsymbol{C}}_w\boldsymbol{S}_1\bar{\boldsymbol{C}}_w^\intercal\right)\left(\boldsymbol{C}_w\boldsymbol{W}_1\boldsymbol{C}_w^\intercal\right)\left(\bar{\boldsymbol{C}}_w\boldsymbol{S}_1\bar{\boldsymbol{C}}_w^\intercal\right)\right]\\
&= \operatorname{Tr}\left(\boldsymbol{W}_1^{\frac{1}{2}}\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}_w\boldsymbol{S}_1\bar{\boldsymbol{C}}_w^\intercal\boldsymbol{C}_w\boldsymbol{W}_1\boldsymbol{C}_p^\intercal\bar{\boldsymbol{C}}_w\boldsymbol{S}_1\bar{\boldsymbol{C}}_w^\intercal\boldsymbol{C}_p\boldsymbol{W}_1^{\frac{1}{2}}\right)\\
&= \operatorname{Tr}\left(\boldsymbol{\Pi}_w\boldsymbol{U}_1\boldsymbol{\Pi}_w^\intercal\boldsymbol{\Pi}_w\boldsymbol{U}_1\boldsymbol{\Pi}_w\right)\\
&= \operatorname{Tr}\left[\left(\boldsymbol{\Pi}_w\boldsymbol{U}_1\boldsymbol{\Pi}_w\right)^2\right].
\end{aligned} \tag{16}$$

This implies $I_1 - I_1' = \operatorname{Tr}\left[\boldsymbol{U}^2 - \left(\boldsymbol{\Pi}_w\boldsymbol{U}\boldsymbol{\Pi}_w\right)^2\right]$. Applying Lemma E.9 yields $I_1 - I_1' \geq 0$.

To bound the other direction, we have

$$\begin{aligned}
&I_1 - I_1' = \operatorname{Tr}\left[\boldsymbol{U}^2 - \left(\boldsymbol{\Pi}_w\boldsymbol{U}\boldsymbol{\Pi}_w\right)^2\right]\\
&= \sum_{i=1}^{N_1}\lambda_i^2 - \sum_{i=1}^{n_1}\left(\lambda_i^{(c)}\right)^2 \overset{\text{(i)}}{\leq} \sum_{i=1}^{N_i}\lambda_i^2 - \sum_{i=1}^{n_i}\lambda_i^{(c)}\lambda_{N-n+i}\\
&= \sum_{i=1}^{n_1}\left(\lambda_i - \lambda_i^{(c)}\right)\lambda_{N-n+i} + \sum_{i=1}^{n_1}\lambda_i(\lambda_i - \lambda_{N-n+i}) + \sum_{i=n_1+1}^{N_1}\lambda_i^2\\
&= \sum_{i=1}^{n_1}\left(\lambda_i - \lambda_i^{(c)}\right)\lambda_{N_1-n_1+i} + C_{\boldsymbol{U},n_1}\\
&\overset{\text{(ii)}}{\leq} \lambda_{N_1-n_1+1}\sum_{i=1}^{n_1}\left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},n_1},
\end{aligned} \tag{17}$$

Here, both (i) and (ii) are by Theorem D.1. $\diamondsuit$

E.4.3. Proof of Lemma E.5

Proof. By applying Lemma E.7, we have

$$0 \le \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal - \boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_2 \left(\boldsymbol{T}_{co}^*\right)^\intercal\right) \le \operatorname{Tr}\left[\left(\boldsymbol{U} - \boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w\right) \boldsymbol{V}\right]$$
$$= \operatorname{Tr}\left(\boldsymbol{U}\boldsymbol{V}\right) - \operatorname{Tr}\left(\boldsymbol{C}_w \boldsymbol{U} \boldsymbol{C}_w^\intercal \boldsymbol{C}_w \boldsymbol{V} \boldsymbol{C}_w^\intercal\right).$$

Recall that $\left\{\lambda_i^{(c)}\right\}_{i=1}^{n_1}$ are eigenvalues of $\boldsymbol{C}_w \boldsymbol{U} \boldsymbol{C}_w^\intercal$, and let $\nu_1 \ge \nu_2 \ge \cdots \ge \nu_{n_1}$ be eigenvalues of $\boldsymbol{C}_w \boldsymbol{V} \boldsymbol{C}_w^\intercal$. Applying Ruhe's trace inequality (c.f. Appendix D.2), we further have

$$\begin{aligned}
&\operatorname{Tr}\left(\boldsymbol{U}\boldsymbol{V}\right) - \operatorname{Tr}\left(\boldsymbol{C}_w \boldsymbol{U} \boldsymbol{C}_w^\intercal \boldsymbol{C}_w \boldsymbol{V} \boldsymbol{C}_w^\intercal\right) \overset{\text{(i)}}{\le} \sum_{i=1}^{N_1} \lambda_i \nu_i - \sum_{i=1}^{n_1} \lambda_i^{(c)} \nu_{n_1-i+1}^{(c)} \overset{\text{(ii)}}{\le} \sum_{i=1}^{N_1} \lambda_i \nu_i - \sum_{i=1}^{n_1} \lambda_i^{(c)} \nu_{N_1-i+1} \\
&= \sum_{i=1}^{n_1} \left(\lambda_i - \lambda_i^{(c)}\right) \nu_{N_1-i+1} + \sum_{i=1}^{n_1} \lambda_i \left(\nu_i - \nu_{N_1-i+1}\right) + \sum_{i=n_1+1}^{N_1} \lambda_i \nu_i = \sum_{i=1}^{n_1} \left(\lambda_i - \lambda_i^{(c)}\right) \nu_{N_1-i+1} + C_{\boldsymbol{U},\boldsymbol{V},n_1} \\
&\overset{\text{(iii)}}{\le} \nu_{N_1-n_1+1} \sum_{i=1}^{n_1} \left(\lambda_i - \lambda_i^{(c)}\right) + C_{\boldsymbol{U},\boldsymbol{V},n_1},
\end{aligned} \tag{18}$$

(i) is by Ruhe's trace inequality (Lemma D.2), since both $\boldsymbol{U}$ and $\boldsymbol{V}$ are PSD, and both (ii) and (iii) are by Poincaré separation theorem (Theorem D.1). ♢

E.5. Other technical lemmas

Lemma E.7. *We have*

$$0 \le \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal - \boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_2 \left(\boldsymbol{T}_{co}^*\right)^\intercal\right) \le \operatorname{Tr}\left[\left(\boldsymbol{U} - \boldsymbol{\Pi}_w \boldsymbol{U} \boldsymbol{\Pi}_w\right) \boldsymbol{V}\right].$$

Proof. The proof is based on the optimality of $\boldsymbol{T}_{co}^*$, i.e. the GW distance induced by $\boldsymbol{T}_{co}^*$ must be upper bounded by any other transport matrix. Intuitively, we can imagine the mass of a cluster center is transported to the same target nodes in G_2 as the original source nodes within this cluster, which corresponds to the transport matrix $\widetilde{\boldsymbol{T}}_{co} := \boldsymbol{C}_{1,p} \boldsymbol{T}^*$. We can verify $\widetilde{\boldsymbol{T}}_c \in \Pi(\mu_1^{(c)}, \mu_2)$ is feasible, since $(\boldsymbol{C}_{1,p}\boldsymbol{T}^*)\mathbf{1}_{N_2} = \boldsymbol{C}_{1,p}\boldsymbol{m}_1 = \boldsymbol{m}_1^{(c)}$ and $\left(\boldsymbol{C}_{1,p}\boldsymbol{T}^*\right)^\intercal \mathbf{1}_{n_1} = (\boldsymbol{T}^*)^\intercal \mathbf{1}_{N_1} = \boldsymbol{m}_2$.

To derive the upper bound, applying the optimality of $\boldsymbol{T}_{co}^*$ yields

$$\operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal - \boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_2 \left(\boldsymbol{T}_{co}^*\right)^\intercal\right) \le \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \widetilde{\boldsymbol{T}}_{co} \boldsymbol{S}_2 \widetilde{\boldsymbol{T}}_{co}^\intercal\right) \tag{19}$$
$$= \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal\right) - \operatorname{Tr}\left(\left(\bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w^\intercal\right)\left(\boldsymbol{C}_p \boldsymbol{T}^*\right) \boldsymbol{S}_2 \left(\boldsymbol{C}_p \boldsymbol{T}^*\right)^\intercal\right) \tag{20}$$
$$= \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 (\boldsymbol{T}^*)^\intercal\right) - \operatorname{Tr}\left(\bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w^\intercal \boldsymbol{C}_p \boldsymbol{T}^* \boldsymbol{S}_2 (\boldsymbol{T}^*)^\intercal \boldsymbol{C}_p^\intercal\right) \tag{21}$$
$$= \operatorname{Tr}\left(\boldsymbol{S}_1 \left(\boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}}\right) \boldsymbol{S}_2 \left(\boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}}\right)^\intercal\right) - \operatorname{Tr}\left(\bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w^\intercal \boldsymbol{C}_p \left(\boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}}\right) \boldsymbol{S}_2 \left(\boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal \boldsymbol{W}_1^{\frac{1}{2}}\right) \boldsymbol{C}_p^\intercal\right)$$
$$= \operatorname{Tr}\left(\boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal \boldsymbol{W}_1^{\frac{1}{2}}\right) - \operatorname{Tr}\left(\bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w^\intercal \boldsymbol{C}_p \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{C}_p^\intercal\right)$$
$$= \operatorname{Tr}\left[\left(\underbrace{\boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{S}_1 \boldsymbol{W}_1^{\frac{1}{2}} - \boldsymbol{W}_1^{\frac{1}{2}} \boldsymbol{C}_p^\intercal \bar{\boldsymbol{C}}_w \boldsymbol{S}_1 \bar{\boldsymbol{C}}_w^\intercal \boldsymbol{C}_p \boldsymbol{W}_1^{\frac{1}{2}}}_{\boldsymbol{D}_1}\right) \boldsymbol{P} \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{S}_2 \boldsymbol{W}_2^{\frac{1}{2}} \boldsymbol{P}^\intercal\right]. \tag{22}$$

The treatment is similar for the lower bound. We replace $\boldsymbol{T}^*$ with $\widetilde{\boldsymbol{T}} := \bar{\boldsymbol{C}}_w^\intercal \boldsymbol{T}_{co}^* \in \Pi(\mu_1, \mu_2)$. Then again due to the optimality of $\boldsymbol{T}^*$:

$$\operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_2 \left(\boldsymbol{T}_{co}^*\right)^\intercal - \boldsymbol{S}_1 \boldsymbol{T}^* \boldsymbol{S}_2 \left(\boldsymbol{T}^*\right)^\intercal\right) \le \operatorname{Tr}\left(\boldsymbol{S}_1^{(c)} \boldsymbol{T}_{co}^* \boldsymbol{S}_2 (\boldsymbol{T}_{co}^*)^\intercal\right) - \operatorname{Tr}\left(\boldsymbol{S}_1 \widetilde{\boldsymbol{T}} \boldsymbol{S}_2 \widetilde{\boldsymbol{T}}^\intercal\right) = 0$$

given our definition. ♢

Remark. An intuitive scheme to control the upper bound (22) is to upper bound the trace of the G_1-related matrix difference $\boldsymbol{D}_1$, since in coarsening G_1 we have no information about G_2; we will shortly showcase the term is the key to bound the whole GW distance difference $|(15) - (12)|$ as well.

Lemma E.8. *Consider a non-negative matrix $\boldsymbol{T} \in \mathbb{R}^{N_1 \times N_2}$ in which all the elements $t_{ij} \geq 0$. We keep to denote $\boldsymbol{W}_1 = \operatorname{diag}(\boldsymbol{T}\mathbf{1}_{N_2}), \boldsymbol{W}_2 = \operatorname{diag}(\boldsymbol{T}^\intercal \mathbf{1}_{N_1})$, and $\boldsymbol{P} = \boldsymbol{W}_1^{-\frac{1}{2}} \boldsymbol{T} \boldsymbol{W}_2^{-\frac{1}{2}}$. Then we have $\|\boldsymbol{P}\| \leq 1$.*

Proof. Motivated by the regular proof for bounding the eigenvalues of normalized Laplacian matrices, with two arbitrary vectors $\boldsymbol{u} \in \mathbb{R}^{N_1}, \boldsymbol{v} \in \mathbb{R}^{N_2}$ on the unit spheres we can recast the target statement as

$$1 - \boldsymbol{u}^\intercal \boldsymbol{P} \boldsymbol{v} \geq 0, \quad \forall\, \boldsymbol{u}, \boldsymbol{v} \text{ s.t. } \|\boldsymbol{u}\| = 1, \|\boldsymbol{v}\| = 1. \tag{23}$$

We denote the diagonals for $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ respectively as $\boldsymbol{m}_1 = [m_{1,1}, m_{1,2}, \ldots, m_{1,N_1}]^\intercal$ and $\boldsymbol{m}_2 = [m_{2,1}, m_{2,2}, \ldots, m_{2,N_2}]^\intercal$; then we can rewrite the right-hand-side quantity in Inequality (23) as

$$\begin{aligned} 1 - \boldsymbol{u}^\intercal \boldsymbol{P} \boldsymbol{v} &= \frac{1}{2}\left(\|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2\right) - \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} \frac{t_{ij}\boldsymbol{u}_i \boldsymbol{v}_j}{\sqrt{m_{1,i} m_{2,j}}} \\ &= \frac{1}{2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} \frac{t_{ij}\boldsymbol{u}_i^2}{m_{1,i}} + \frac{1}{2}\sum_{j=1}^{N_2}\sum_{i=1}^{N_1} \frac{t_{ij}\boldsymbol{v}_j^2}{m_{2,j}} - \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} \frac{t_{ij}\boldsymbol{u}_i \boldsymbol{v}_j}{\sqrt{m_{1,i} m_{2,j}}} \\ &= \frac{1}{2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} t_{ij}\left(\frac{\boldsymbol{u}_i}{\sqrt{m_{1,i}}} - \frac{\boldsymbol{v}_j}{\sqrt{m_{2,j}}}\right)^2 \geq 0, \end{aligned}$$

where the second equation holds due to the conditions $\sum_{j=1}^{N_2} t_{ij} = m_{1,i}$ and $\sum_{i=1}^{N_1} t_{ij} = m_{2,j}$, and the last inequality holds since $t_{ij} \geq 0, \forall i \in [N_1], j \in [N_2]$. ♢

Lemma E.9. *Consider a positive semi-definite (PSD) similarity matrix $S \in \mathbb{R}^{N \times N}$ along with the probability mass vector $\boldsymbol{m}$ and the diagonal matrix $\boldsymbol{W} := \operatorname{diag}(\boldsymbol{m})$. For any non-overlapping partition $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$, we denote the corresponding coarsening matrices $\boldsymbol{C}_p, \bar{\boldsymbol{C}}_w$ and the projection matrix $\boldsymbol{\Pi}_w = \boldsymbol{C}_w^\intercal \boldsymbol{C}_w$. Let $\boldsymbol{A} := \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{S} \boldsymbol{W}^{\frac{1}{2}}$. Then we have*

$$\operatorname{Tr}\left(\boldsymbol{A}^2\right) \geq \operatorname{Tr}\left[\left(\boldsymbol{\Pi}_w \boldsymbol{A} \boldsymbol{\Pi}_w\right)^2\right]. \tag{24}$$

Proof. We first transform $\operatorname{Tr}\left[\left(\boldsymbol{\Pi}_w \boldsymbol{A} \boldsymbol{\Pi}_w\right)^2\right]$ as follows:

$$\operatorname{Tr}\left[\left(\boldsymbol{\Pi}_w \boldsymbol{A} \boldsymbol{\Pi}_w\right)^2\right] = \operatorname{Tr}\left(\boldsymbol{\Pi}_w \boldsymbol{A} \boldsymbol{\Pi}_w \boldsymbol{\Pi}_w \boldsymbol{A} \boldsymbol{\Pi}_w\right) = \operatorname{Tr}\left(\boldsymbol{C}_w^\intercal \boldsymbol{C}_w \boldsymbol{A} \boldsymbol{C}_w^\intercal \boldsymbol{C}_w \boldsymbol{A} \boldsymbol{C}_w^\intercal \boldsymbol{C}_w\right) = \operatorname{Tr}\left(\boldsymbol{C}_w \boldsymbol{A} \boldsymbol{C}_w^\intercal \boldsymbol{C}_w \boldsymbol{A} \boldsymbol{C}_w^\intercal\right),$$

and the last two equations hold due to $\boldsymbol{C}_w \boldsymbol{C}_w^\intercal = \boldsymbol{I}_n$.

We notice $\boldsymbol{A} := \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{S} \boldsymbol{W}^{\frac{1}{2}}$ is symmetric and we can apply Poincaré separation theorem (c.f. Appendix D.1 for the complete statement) to control the eigenvalues of $\boldsymbol{C}_w \boldsymbol{A} \boldsymbol{C}_w^\intercal$. Specifically, let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ be the eigenvalues of $\boldsymbol{A}$, and let $\lambda_1^{(c)} \geq \cdots \geq \lambda_n^{(c)}$ be the eigenvalues of $\boldsymbol{C}_w \boldsymbol{A} \boldsymbol{C}_w^\intercal$; for all $i \in [n]$ we have $\lambda_i \geq \lambda_i^{(c)} \geq 0$ (being non-negative due to the PSD-ness of $\boldsymbol{A}$); and a further conclusion $\lambda_i^2 \geq \left(\lambda_i^{(c)}\right)^2, \forall i \in [n]$. We can therefore complete the proof with $\operatorname{Tr}\left(\boldsymbol{A}^2\right) = \sum_{i=1}^N \lambda_i^2 \geq \sum_{i=1}^n \left(\lambda_i^{(c)}\right)^2 = \operatorname{Tr}\left[\left(\boldsymbol{\Pi}_w \boldsymbol{A} \boldsymbol{\Pi}_w\right)^2\right]$. ♢