Optimal transport for graph representation learning

Cédric Vincent-Cuaz

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

Transport Optimal pour l'apprentissage de représentation de graphes

Cédric Vincent-Cuaz

Laboratoire Jean Alexandre Dieudonné
Centre INRIA d'Université Côte d'Azur

Présentée en vue de l'obtention du grade de docteur en Sciences pour l'ingénieur
d'Université Côte d'Azur

Dirigée par : Rémi Flamary
Co-encadrée par : Marco Corneli
Soutenue le : 29/03/2023

Devant le jury, composé de :
Gabriel Peyré, DR, ENS Paris-Saclay
Florence d'Alché-Buc, PR, Télécom Paris
Laetitia Chapel, MCF, Université Bretagne Sud
Hongteng Xu, Associate Professor, Renmin University of China
Marco Gori, PR, University of Siena

3iA Côte d'Azur
Institut interdisciplinaire d'intelligence artificielle

FRANCE 2030

Optimal Transport for Graph Representation Learning

Jury :

Rapporteurs

Gabriel Peyré, Directeur de recherche, ENS Paris-Saclay
Florence d'Alché-Buc, Professeure, Télécom Paris

Examinateurs

Laetitia Chapel, Maître de conférence, Université Bretagne Sud
Hongteng Xu, Associate Professor, Renmin University of China
Marco Gori, Professor, University of Siena

Invité

Titouan Vayer, Chargé de recherche, ENS Lyon

Abstract

A key challenge in Machine Learning (ML) is to design models able to learn efficiently from *graphs*, characterized by *nodes* with attributes and a prescribed *structure* encoding their relationships. Graph Representation Learning (GRL) aims to encode these two sources of heterogeneity into a vectorial *graph embedding* easing downstream tasks. In this field, Optimal Transport (OT) has been successful in providing meaningful comparison between graphs seen as *discrete probability distributions*. This thesis focuses on GRL through the lens of OT, with both concepts introduced in dedicated chapters.

Modern supervised GRL mostly relies on Graph Neural Networks (GNN) which implicitly encode the graph topology via two main elements: node features embedding through message passing, and aggregation with a specialized form of pooling. We introduce in this thesis a novel point of view, which places distances to some *learnable graph templates* at the core of the graph representation. This distance embedding is constructed by means of an OT distance: the *Fused Gromov-Wasserstein* (FGW) distance, which simultaneously handles feature and structure dissimilarities by solving a soft graph-matching problem. We postulate that the vector of FGW distances to a set of template graphs, has a strong discriminative power, which is then fed to a non-linear classifier for final predictions. This distance embedding acts as a new pooling layer called TFGW, and can leverage on existing message passing techniques to promote sensible feature representations, learned in an end-to-end fashion. We empirically validate our claim on several graph classification tasks, where our method outperforms both kernels and GNN approaches in terms of expressivity and generalization abilities.

Another contribution of this thesis aims at making Dictionary Learning (DL) amenable to graphs dataset analysis, a key tool for *unsupervised* representation learning. DL explains vector data as a linear combination of a few basic elements, accessing the quality of learned representations via dissimilarities associated with a single ambient space. Since graphs depict their own spaces, we propose the first linear approach adapted to Graph Dictionary Learning (GDL), using (F)GW as the data fitting term. In our work, graphs are modeled as convex combination of graph atoms, estimated via an online stochastic algorithm. GDL is completed by a novel upper-bound that can be used as a fast approximation of FGW in the embedding space. We empirically show the interest of our approach for graphs clustering, classification, completion and for *online* graph subspace estimation and tracking.

Finally, the mass conservation at the core of OT, imposing a coupling between all the nodes from the two compared graphs, has specific implications in GRL. Learning structure and feature representations via FGW is considerably sensitive to the nodes relative importance induced by modeling graphs as probability distributions. Managing this extra degree of freedom, as we made possible, improves (F)GW-based models by adding minimal computational cost in TFGW but significant model complexity for GDL. Thus we propose to address the limits of mass conservation constraints in (F)GW, by introducing a novel OT-based discrepancy, called the *semi-relaxed (Fused) Gromov-Wasserstein divergence* (sr(F)GW). srFGW provides correspondences between two graphs, while searching for a reweighed subgraph in the target graph at a minimum (F)GW distance from the input. The latter can be estimated

more efficiently than (F)GW and competes with methods dedicated to graph partitioning while being more generic. Moreover, estimating a srFGW "barycenter" induces a novel DL, where graphs are embedded as reweighed subgraphs of a *single graph atom*. srFGW DL competes favorably with other DL-based competitors on various unsupervised tasks, while being considerably faster to compute.

Key words: Machine Learning, Graphs, Optimal Transport, Graph Representation Learning.

Résumé

Un défi majeur de l'apprentissage machine est de concevoir des modèles capables d'apprendre efficacement à partir de *graphes*, constitués de *noeuds* dotés d'attributs et d'une *structure* décrivant leurs relations. L'apprentissage de représentation de graphe (ARG) vise à synthétiser ces deux sources d'hétérogénéité dans un vecteur, afin de simplifier son traitement à postériori. Dans ce domaine, le Transport Optimal (TO) fournit des comparaisons pertinentes entre graphes, vus comme des *distributions discrètes de probabilité*. Cette thèse se concentre sur l'ARG à travers le prisme du TO, tous deux détaillés dans des chapitres dédiés.

L'ARG supervisé s'appuie essentiellement sur les réseaux de neurones graphiques (RNG), qui encodent implicitement la topologie du graphe par le raffinement des attributs nodaux via l'information issue de leur voisinage, et l'agrégation de cette information à l'échelle du graphe. Nous introduisons dans cette thèse un nouveau concept, qui considère comme représentation du graphe, des distances à certains *graphes templates* apprenables. A cette fin, nous exploitons la distance de *Fused Gromov-Wasserstein* (FGW) issue du TO, qui traite simultanément les dissimilarités entre nœuds et structures, en résolvant un problème de correspondance de nœuds entre les graphes. Ce vecteur de distances possède un fort pouvoir discriminant, qui est ensuite transmis à un classifieur gérant les prédictions finales. Ce vecteur agit telle une couche de RNG, appelée TFGW, et peut se superposer à leurs techniques de raffinement d'attributs nodaux, le tout étant appris simultanément. Nous validons empiriquement notre postulat sur de multiples tâches de classification de graphes, où TFGW surpasse les méthodes à RNG et à noyaux en termes d'expressivité et de capacité de généralisation.

Le chapitre suivant étend l'apprentissage de dictionnaire (AD), un outil clé pour l'apprentissage *non supervisé* de représentation, à l'analyse de graphes. l'AD représente les données vectorielles par des combinaisons linéaires de quelques éléments de base, accédant à la qualité de ces dernières via des dissimilarités associées à un espace ambiant unique. Ainsi, nous proposons la première approche linéaire adaptée à l'AD de graphes (ADG), en utilisant (F)GW comme mesure de qualité. Nous modélisons les graphes tels des combinaisons convexes d'atomes de graphes, estimés grâce à un algorithme stochastique. L'ADG est complété par une nouvelle approximation, facilement calculable, de FGW dans l'espace des représentations. Nous montrons empiriquement l'intérêt de notre approche pour le clustering, la classification, la complétion de graphes, ainsi que le suivi *en ligne* de sous-espaces de graphes.

Enfin, la conservation de la masse au cœur du TO, imposant un couplage entre tous les nœuds des deux graphes comparés, a des implications en ARG. L'apprentissage de structures et d'attributs nodaux via FGW est sensible à l'importance relative des nœuds induite par la modélisation des graphes tels des distributions de probabilité. La gestion de ces pondérations, comme nous l'avons rendu possible, améliore les modèles susmentionnés basés sur (F)GW à des coûts de calcul supplémentaires variables. Ainsi, nous modulons ce principe de conservation, via l'introduction de la *divergence de (Fused) Gromov-Wasserstein sémi-relachée* (sr(F)GW). srFGW fournit des correspondances entre deux graphes, tout en recherchant un sous-graphe re-pondéré dans le graphe cible à une distance (F)GW minimale de l'entrée. Cette dernière

rivalise, entre autres, avec les méthodes dédiées au partitionnement de graphes tout en étant plus générique. De plus, l'estimation d'un "barycentre" de srFGW induit un nouvel AD, où les graphes sont intégrés comme des sous-graphes re-pondérés d'un *unique atome de graphe*. L'AD srFGW rivalise favorablement avec d'autres concurrents basés sur l'AD dans diverses tâches non supervisées, tout en étant considérablement plus rapide à calculer.

Mots clés: Apprentissage statistique, Graphes, Transport Optimal, Apprentissage de répresentation de graphes.

Remerciements

Je profite de cette parenthèse pour exprimer, sur un ton plus léger, ma profonde gratitude envers les nombreuses personnes qui m'ont accompagné durant l'élaboration de cette thèse. Vous avez fait de ces trois dernières années, teintées de moult nuances de Forrest Gump et Will Hunting, une expérience humaine et scientifique inoubliable !

Mille mercis à mes encadrants Rémi et Marco. Votre enthousiasme, vos enseignements et la confiance que vous m'avez accordé, ont rendu cette aventure excitante, gratifiante et si enrichissante. Vous avez redéfini ma perception de la recherche et en marquerez ma pratique durant de nombreuses années. Je remercie également les officieux Nicolas et Titouan, cela a été un grand plaisir de collaborer à distance avec vous pendant ces trois dernières années. Je me souviendrai des discussions lors des périodes de soumission, ponctuées de belles idées, de débats à la française et de rires. A chaque fois, l'émulation qui en naissait était des plus inspirantes et a amplement contribué à l'aboutissement de cette thèse. J'espère revoir cette fine équipe réunie d'ici peu.

Merci également aux rapporteurs, Florence d'Alché-Buc et Gabriel Peyré, aux membres du jury, Laetitia Chapel, Marco Gori et Hongteng Xu, ainsi qu'aux membres du comité de suivi, Patricia Reynaud-Bouret, Julie Delon et Charles Bouveyron, chercheuses et chercheurs dont j'admire les travaux. Je vous remercie d'avoir pris le temps de vous intéresser aux miens.

Merci à l'équipe MAASAI et ses guerriers aux grands cœurs, avec qui cela a été une immense joie de voyager à travers mers et montagnes, ou, sentiers éclairés et sombres ruelles pavées. Rendez-vous au café des amis ! Merci au laboratoire Lagrange et ses jeunes érudits qui, pendant mon bref passage, m'ont rappelé de regarder les étoiles quand notre bas monde partait en flammes.

Je remercie infiniment ma famille pour m'avoir soutenu tout au long de ma vie et évidemment de cette thèse, j'espère avoir rendu un brin compréhensible ce qui occupait tant mon esprit ces derniers temps. Merci à ma mère et ma soeur, mes deux reines de leur royaume respectif, qui m'ont toujours tant apporté. J'espère réussir à vous le rendre. Merci à mes nièces de m'animer, quelle chance d'avoir deux si beaux soleils quand on aime lézarder en bon niçois. Merci à mon frère d'arme, toujours là pour me rappeler que je suis inépuisable. Enfin merci à tous mes amis: les anciens qui ne me connaissent que trop bien; les membres de la tribu dont je continue à porter l'étendard; les colocs qui sous leurs airs hippies m'ont fait garder les pieds sur terre; les niçois de naissance ou d'adoption avec qui la douceur de la Côte d'Azur n'est que démultipliée.

Contents

Notations

Linear Algebra

$[\![n]\!]$	the subset $\{1, ..., n\}$ of $\mathbb{N}$.
$\boldsymbol{x}$	All vectors in $\mathbb{R}^n$ are written in bold and lower case, for all $n > 1$. The i^{th} coordinate is written x_i for all $i \in [\![n]\!]$.
$\boldsymbol{X}$	All matrices in $\mathbb{R}^{n \times m}$ are written in bold, for any $n > 1$ and $m > 1$. The coordinate (i, j) will be written X_{ij}, and unless specified otherwise the i^{th} row (resp. the j^{th} column) will be denoted $\boldsymbol{X}_{i,:}$ (resp. $\boldsymbol{X}_{:,j}$).
$\boldsymbol{I}_n$	The identity matrix of $\mathbb{R}^{n \times n}$.
$\text{diag}(\boldsymbol{x})$	The operator that transforms a vector $\boldsymbol{x} \in \mathbb{R}^n$ into the diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ such that $\forall (i, j) \in [\![n]\!] \times [\![n]\!]$, $D_{ij} = \begin{cases} x_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$.
$\odot$	The Hadamard product between two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in $\mathbb{R}^{n \times n}$, such that $\boldsymbol{A} \odot \boldsymbol{B} = (A_{ij} B_{ij})_{ij} \subset \mathbb{R}^{n \times n}$. Also used as element-wise product between two vectors.
$\oslash$	The element-wise division of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in $\mathbb{R}^{*n \times n}$, such that $\boldsymbol{A} \oslash \boldsymbol{B} = (A_{ij}/B_{ij})_{ij} \subset \mathbb{R}^{n \times n}$. Also used as element-wise division between two vectors.
$\otimes_K$	The Kronecker product of two matrices $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{B} \in \mathbb{R}^{p \times q}$, such that $\boldsymbol{A} \otimes_K \boldsymbol{B} = (A_{ij}\boldsymbol{B})_{ij} \in \mathbb{R}^{np \times mq}$.
$\otimes$	The tensor-matrix multiplication, such that for a tensor $\boldsymbol{L} \in \mathbb{R}^{n \times m \times p \times q}$ and a matrix $\boldsymbol{X} \in \mathbb{R}^{p \times q}$, $\boldsymbol{L} \otimes \boldsymbol{X} = (\sum_{kl} L_{ijkl} A_{kl})_{ij} \in \mathbb{R}^{n \times m}$.
$\text{vec}(\boldsymbol{X})$	The vectorization operator that stacks columns of a matrix $\boldsymbol{X} \in R^{n \times m}$ into a vector in $\mathbb{R}^{nm}$.
Tr	The trace operator for a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, such that $\text{Tr}(\boldsymbol{X}) = \sum_{ij} X_{ij}$.
$\|\cdot\|, \langle \cdot, \cdot \rangle$	A norm and an inner product that depend on the context.
l_p	The standard $\|\cdot\|_p$ norm.
$\|\cdot\|_F$	The Frobenius norm of a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, $\|\boldsymbol{X}\|_F = \sqrt{\text{Tr}(\boldsymbol{X}^\top \boldsymbol{X})} = \sqrt{\sum_{ij} X_{ij}^2}$.
$\langle \cdot, \cdot \rangle_F$	The Frobenius inner product between two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in $\mathbb{R}^{n \times m}$, $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F = \text{Tr}(\boldsymbol{A}^\top \boldsymbol{B}) = \sum_{ij} A_{ij} B_{ij}$.
$\mathfrak{S}_n$	The set of all permutations of $[\![n]\!]$ for all $n \in \mathbb{N}^*$.
Π_n	The set of permutation matrices in $\mathbb{R}^{n \times n}$ for all $n\mathbb{N}^*$.

Σ_n	The set of probability vectors in $\mathbb{R}_+^n$ defined in equation (3.1).
$\alpha\mathbb{S}$	The set of elements $\{\boldsymbol{x} = \alpha\boldsymbol{y} \vert \forall \boldsymbol{y} \in \mathbb{S}\}$ induced by any set $\mathbb{S}$ for any scalar $\alpha \in \mathbb{R}^*$.

Measure Theory

$\delta_{\boldsymbol{x}}$	The dirac measure on $\boldsymbol{x} \in \mathbb{R}^d$, such that $\boldsymbol{y} \in \mathbb{R}^d$, $\delta_{\boldsymbol{x}}(\boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{y} \\ 0 & \text{otherwise} \end{cases}$.
$\text{supp}(\mu)$	The support of μ, see equation (3.2).
$\mu \otimes \nu$	The product measure of two probability measures μ and ν, such that for all measurable subsets A and B, $\mu \otimes \nu(A \times B) = \mu(A)\nu(B)$.
Bernoulli(p)	The Bernoulli distribution with probability p.
$\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$	The multivariate Gaussian distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma}$.
$\mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$	The set of admissible couplings between two discrete probability measures with respective probability vectors $\boldsymbol{h}$ and $\overline{\boldsymbol{h}}$, defined in equation (3.3).
$\mathcal{T}_c(\mu, \nu)$	The Kantorovitch cost between two probability measures μ and ν, defined in equation (3.4).
$\text{W}_p(\mu, \nu)$	The p-Wasserstein distance between two discrete probability measures μ and ν, defined in equation (3.8).
$\text{GW}_p(\mu, \nu)$	The Gromov-Wasserstein distance of order p between two discrete probability measures μ and ν, defined in equation (3.21).
$\text{FGW}_{p,\alpha}(\mu, \nu)$	The Fused Gromov-Wasserstein distance of order p with trade-off parameter α between two discrete probability measures μ and ν, defined in equation (3.54).

Acronyms

OT	Optimal Transport.
W, GW, FGW	Stands respectively for Wasserstein, Gromov-Wasserstein, Fused Gromov-Wasserstein.
LP, QP	Stands respectively for Linear Program and Quadratic Program.
RL	Representation Learning.
GRL	Graph Representation Learning.
GNN	Graph Neural Networks.
DL	Dictionary Learning.

Chapter 1

Introduction

Contents

1.1 Structured data in Machine Learning

1.1.1 Structured data

One of the main challenges in Machine Learning (ML) is the design of efficient algorithms that are able to learn from *structured data*. The latter comes with a generic notion of *structure* that encode relationships between *entities* composing an object, or in other words a data instance. The entities are elements, that can be endowed with attributes, such as $\mathbb{R}^d$ vectors highlighting the singularity of an element, or discrete values referring to their assignments to certain categories. The structure then captures any type of relationship between the entities. In their simplest form, these relations can be binary interactions between elements. In more complex scenarios, they can be encoded as real-valued vectors in $\mathbb{R}^{d'}$ ($d' \geq 1$), expressing interactions of different nature or strength between components.
Typical instances of structured data encountered in this thesis are *e.g.* molecules, composed of atoms as entities, which can be categorized via their position in the periodic table, or characterized by certain chemical or physical properties such as their molecular mass. The interactions taken into account to describe the structure of the molecule can be of increasing complexity, ranging for example from covalent bonds to Van der Waals forces.

The manipulation of structured data relates to *Relational Reasoning* as detailed in Battaglia et al. (2018), which points out that these structured representations require "structured computations" dealing with entities and their composition as a whole. This assumes that the global understanding of these objects is conditioned by a simultaneous processing of their *feature information* and *structure information*, namely the knowledge

contained in the entities and their interactions. To this end, the methods of reasoning vary according to the nature of the structure.

Explicit structured data. A first natural type of structured data, on which this thesis focuses, appears when the structure is given *explicitly*. The resulting object can be understood in a broad sense as a *graph* $\mathcal{G}$ (Wu et al., 2020). Namely, a tuple $\mathcal{G} = (V, E)$, composed of a discrete set of vertices or nodes V, and a set of edges E quantifying connections between vertices. Nodes are usually indexed in $[\![n]\!] \subset \mathbb{N}$, where n denotes the number of nodes in the graph and is often referred as the graph *order*. They can also be assigned to attributes or features in $\mathbb{R}^d$, which induces the distinction between *unattributed* graphs and *attributed* ones. The edges usually assign a discrete or continuous value to pairs of nodes, which can be *undirected* or *directed* from one vertex to another. If the edges are vectors, whose components describe different individual interactions, the graph is called a multi-edged graph or a *multigraph*. Finally, a temporal component can also be associated with a graph modifying its nodes or interactions over time, which leads to the notion of *dynamic graph* (Ditzler et al., 2015).

Notable instances of graph data are found in chemical compounds or molecules modeling (Debnath et al., 1991; Sutherland et al., 2003; Borgwardt & Kriegel, 2005; Krichene et al., 2015; Jumper et al., 2021), brain connectivity (Bullmore & Sporns, 2009; Ktena et al., 2017; Heitmann & Breakspear, 2018), particle physics (Shlomi et al., 2020; Thais et al., 2022), social networks (Banerjee et al., 2013; Yanardag & Vishwanathan, 2015; Yin et al., 2017) or resources exchange (Yang & Leskovec, 2015). This family of explicit structured data also encompasses trees (Day, 1985), whose structure consists in a depth-wise hierarchy on the nodes. More traditional data can also be included, such as images (Shapiro et al., 2001) seen as two-dimensional regular grids, or time series (Ralaivola & d'Alché Buc, 2005; Wei, 2013) and text data (Chowdhary, 2020) whose structures are inherited from their sequential nature.

Implicit structured data. In many ML tasks, the prescribed data structure does not adequately reflect the geometry of the problem, such as the regular grid (resp. line) naturally associated with an image (resp. univariate time series). In order to learn from these objects, one must first extract relevant representations from them. The latter can be seen as specific instances of structured data with a more subtle or even *implicit* structure (Battaglia et al., 2018). They may be initially formatted as a set of entities without relationships, resulting *e.g* from a learning step, such as a word embedding (Goldberg, 2017; Otter et al., 2020), or feature maps associated to an image (Santoro et al., 2017; Wang et al., 2018) or one of its segments (Harchaoui & Bach, 2007; Minaee et al., 2021).
The connections between these elements are *relational inductive biases*, resulting from structural priors on the object representations learned by a model. For instance, modern deep learning architectures (Goodfellow et al., 2016) that stack building blocks or *layers*, impose hierarchical relations between entities via their processing, by capturing increasingly long range interactions among information in the input signal. These layers have a built-in inductive bias chosen in accordance with the representation of their input object.
Typically, a layer in Convolutional Neural Networks (CNN, LeCun et al. (1995)) processes a pixel locally, assuming independence *w.r.t.* distant entities, in a translation invariant fashion reflecting that convolutions remain relevant across different localities in the image (Bronstein et al., 2017; Chen et al., 2018; Kondor & Trivedi, 2018). On the contrary, *Recurrent layers* are adopted for sequence or text data to favor temporal invariance (Elman, 1990; Salehinejad et al., 2017). More precisely, their inputs and hidden representations, forming the entities, are related through the dependence of one step's hidden state to both the current input and its previous hidden state. This conditioning rule, carrying a bias for locality in the sequence via its Markovian structure (Norris & Norris, 1998), is reused over each step across all input

representations, resulting into temporal invariance. Finally, relations between elements can be guided by additional dependencies to classes or logic rules (Hu et al., 2016). The latter constrain the learned object representations, so that the associated embedding space has better class separability, knowledge transferability or interpretability (Pan & Yang, 2010; Courty et al., 2017b; Wang & Deng, 2018; Došilović et al., 2018; Garcez et al., 2019; Beaudouin et al., 2020; Lamb et al., 2020; Ciravegna et al., 2023).

1.1.2 Learning structured data representations

A wide range of ML approaches involve learning representations of the data that encode useful information in the construction of predictors, such as classifiers. Representation learning (RL) has been shown to be effective in dealing with many types of unstructured data, such as tabular data (Bengio et al., 2013). RL became particularly successful when operating on images or sequences, whose respective grid or line structures are shared across samples, by constraining (or guiding) the learned representations of these objects via structural priors such as translation or temporal invariance.

This is in contrast to graphs whose structures (and orders) are specific to each of them. These data therefore generally present a high degree of structural heterogeneity, to which may be added a heterogeneity arising from the feature information. Consequently, graph-structured data requires specific modeling which has recently given rise to the paradigm of *Graph Representation Learning* (GRL) (Hamilton, 2020).
GRL aims to encode a graph into a vector representation called *graph embedding*, which eases comparisons between graphs or any downstream Graph Machine Learning tasks. This can be addressed by the design of effective (dis)similarity measures to compare graphs, such as Graph Kernel methods (Kriege et al., 2020). The latter most often model a graph implicitly through its pairwise relations to the other graphs composing the dataset. However, one limit of kernel methods is that the graph embedding is most often fixed and cannot be successfully adapted to certain complex datasets. On the other hand, Geometric deep learning approaches (Bronstein et al., 2017) attempt to learn structured data representation by means of deep learning (Scarselli et al., 2008; Perozzi et al., 2014; Niepert et al., 2016). For example, some Graph Neural Networks generalize the notion of convolution applied to images to the graph, by considering adjacent nodes on the basis of the graph structure, instead of a grid surrounding a pixel (Gilmer et al., 2017; Kipf & Welling, 2016; Wu et al., 2020).

However, both kernel methods and many deep learning based representations for graphs suffer from the fundamental *pre-image* problem, that prevents recovering actual graph objects from the graph embeddings.

1.1.3 Graphs, incomparable spaces and Optimal Transport

Many GRL methods mainly focus on designing node embeddings that encode both the structure and the feature information of a graph. This is legitimately motivated by the desire to return to the classical framework of Machine Learning operating on vector data, that associates a discrete probability measure μ_n to a collection of samples composing a dataset, or in our setting, a collection of node embeddings composing a graph. This discrete probability measure is expressed as $\mu_n = \sum_{i=1}^{n} h_i \delta_{\boldsymbol{x}_i}$, where $\boldsymbol{x}_i$ is a sample in $\mathbb{R}^d$ that composes the *support* of μ_n, and $h_i \geq 0$ weights its relative importance in the dataset, while satisfying $\sum_{i=1}^{n} h_i = 1$. Modeling graphs as such allows, for instance, to compare two graphs via well-known similarities between probability measures, assuming that their respective supports belong to a *common ambient space*. The latter assumption comes down to model graphs as clouds of node embeddings, living in this ambient space, which we hope encode the topology of their respective graphs well enough. Note that a more simplistic representation of the

graph is often obtained by averaging its node embeddings (*i.e.* computing the expectation of μ_n).

Optimal Transport within a common space. However, even if this intermediate modeling of graphs as point clouds allows the use of a multitude of classical ML tools, some choices remain more relevant than others. For instance, few similarity measures are able to compare distributions while taking into account the geometry of the underlying space, while also providing correspondences between samples across the representations. *Optimal Transport* (OT) is an elegant theory making both possible (Villani, 2009; Santambrogio, 2015; Peyré & Cuturi, 2019), by addressing the problem of moving certain masses from one place to another while minimizing the distance traveled.
OT has proved to be very useful for numerous ML tasks, such as image retrieval or analysis (Rubner et al., 1998; Thorpe et al., 2017), NLP (Kusner et al., 2015; Huang et al., 2016; Grave et al., 2019), unsupervised learning (Arjovsky et al., 2017; Schmitz et al., 2018; Genevay et al., 2018), signal processing (Kolouri et al., 2017), domain adaptation (Courty et al., 2014; 2017b). Finally, OT has also been used for the comparison of node embeddings in Graph Kernels (Nikolentzos et al., 2017; Togninalli et al., 2019; Kriege et al., 2020), or within Graph Neural Networks (Kolouri et al., 2021; Chen et al., 2020a).

Optimal Transport across incomparable spaces. Nonetheless, since graphs are naturally observed and characterized by pairwise interactions between their nodes, assuming the existence of a node ambient space shared by graphs, seen as point clouds, might omit their respective and heterogeneous topologies. This paradigm has motivated the extension of OT limited to a single ambient space to OT across incomparable spaces (Mémoli, 2011; Sturm, 2012). One of the key ingredients in this case is to rely on the so called Gromov-Wasserstein (GW) distance. GW results from an OT problem adapted to the scenario in which the respective supports of the compared probability distributions lie in different metric spaces, hence GW is particularly suited for comparing relational data (Peyré et al., 2016; Solomon et al., 2016). GW has been extended to weighted directed graphs in Chowdhury & Mémoli (2019), and to attributed graphs thanks to the Fused Gromov-Wasserstein (FGW) distance in Vayer et al. (2019a; 2020). In this setting, FGW, which will be detailed later, aims to find an optimal coupling by minimizing an OT cost which is a trade-off of a linear OT cost between the node features and a GW cost between the similarity matrices.

1.2 Manuscript outline and contributions

This thesis covers all the author's (published) work and focuses on a single line of research that is Graph Representation Learning by means of Optimal Transport across incomparable spaces. This section provides a brief description of the other chapters of the manuscript, and of the contributions published during this thesis.

Chapter 2: Introduction to Graph Representation Learning

This chapter aims to provide a short overview of *Graph Representation Learning* (GRL) at the instance-level (namely, a graph is an observation). Since GRL lies at the crossroads of different fields, we make an overview of GRL from two different perspectives, fitting the contributions presented in the next chapters.

First, we introduce methods that rely on the design of effective (dis)similarity measures to compare graphs (Section 2.1). The most common paradigm relies on Graph *kernels* (Kriege et al., 2020) whose theoretical foundations are reported in Section 2.1.1. As these methods might suffer from empirical limitations, we also briefly discuss a theory of learning with

similarity functions developed by Balcan et al. (2008), that can be seen as a relaxation of the former. Then we review typical kinds of Graph Kernels that we distinguish according to whether they focus on local information within graphs (Section 2.1.2) or directly on their global topologies (Section 2.1.3).

Second, we introduce in Section 2.2 a family of more flexible methods designed to learn end-to-end graph representations, relying on Geometric Deep Learning (Bronstein et al., 2021a) and Graph Neural Networks (Hamilton, 2020; Wu et al., 2020). Following an analog pattern than in Section 2.1, for clarity, we first introduce predominant methods that focus on *node embedding schemes* (Section 2.2.1), before discussing *graph pooling* approaches allowing to extract global graph knowledge (Section 2.1.3).

Chapter 3: Introduction to Optimal Transport for graphs

We present here the theoretical foundations of Optimal Transport (OT) from an application perspective related to Graph Machine Learning. Since graphs, seen as finite sets of nodes, can be considered as an instance of discrete probability distributions, a first objective of this chapter (Section 3.1) is to briefly present the discrete OT theory, both mathematically and numerically (Villani, 2009; Santambrogio, 2015; Peyré & Cuturi, 2019), to compare point clouds.

However, as graphs are naturally described by their own topologies, we introduce extensions of OT across incomparable spaces. The latter is addressed in Section 3.2, whereas Section 3.3 deals with a mix of previous notions, allowing to compare attributed graphs thanks to the Fused Gromov-Wasserstein (FGW) distance (Vayer et al., 2020).

OT problems across incomparable spaces will be complemented by our new contributions, that concern estimations of "complete" barycentric graph distributions, thanks to novel sub-gradient derivations with respect to the barycenter masses. These results were first published in the following paper, which will be discussed in detail in Chapter 5:

- (Vincent-Cuaz et al., 2021) Online Graph Dictionary Learning. Cédric Vincent-Cuaz, Titouan Vayer, Rémi Flamary, Marco Corneli, and Nicolas Courty. *In Proceedings of the 38th International Conference on Machine Learning, ICML 2021.*

Chapter 4: Optimal Transport distances for graphs meet Graph Neural Networks

We propose in this work a novel point of view for Graph Neural Networks (GNN), which places FGW distances to some graph templates, learned end-to-end, at the core of the graph representation. This embedding of FGW distances can be seen as a new global pooling layer (Section 2.2.2), and can leverage on existing message passing techniques to learn sensible feature representations (Section 4.2). We empirically demonstrate the relevance of our approach on several synthetic and real life graph classification datasets, where our method surpasses kernel and GNN state-of-the-art approaches, both in terms of expressiveness and generalization abilities (Section 4.3).

- (Vincent-Cuaz et al., 2022c) Template based graph neural network with optimal transport distances. Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. *In Advances in Neural Information Processing Systems, NeurIPS 2022.*

Chapter 5: Fused Gromov-Wasserstein Linear Dictionary of graphs

This chapter focuses on unsupervised GRL through the lens of Dictionary Learning (DL). DL consists into modeling the observed data as a linear combination of a few basic elements. We make this analysis amenable to graph learning, by using the FGW distance for the data fitting term. This novel Graph DL (GDL) models graphs as convex combination of graph atoms, *i.e.* weighted sums, acting as graph embeddings, of pairwise similarity matrices and corresponding feature matrices (Section 5.2).
We then provide (stochastic) solvers to estimate both the projection of an input graph onto the dictionary, and the dictionary itself. Interestingly, we prove that the (F)GW distance in this embedding is upper-bounded by a Mahalanobis distance over the space of *embedding* weights, which is actually a reliable and fast approximation of (F)GW (Section 5.2.3). This proxy can be efficiently used for clustering and classification of graphs datasets (Section 5.3.2). Furthermore, we empirically demonstrate the relevance of our approach for online subspace estimation, subspace tracking by designing streams of graphs (Section 5.3.5) and graphs completion (Section 5.2.6).

- (Vincent-Cuaz et al., 2021) Online Graph Dictionary Learning. Cédric Vincent-Cuaz, Titouan Vayer, Rémi Flamary, Marco Corneli, and Nicolas Courty. *In Proceedings of the 38th International Conference on Machine Learning, ICML 2021.*

Chapter 6: Relaxing the Optimal Transport paradigm for unsupervised Graph Representation Learning

We propose in this chapter a relaxation of the OT paradigm across incomparable spaces to better address challenges of unsupervised GRL. At the core of OT is the idea of *conservation of mass*, which imposes a coupling between all the nodes from the two compared graphs. We argue in this chapter that this property can be detrimental for certain Graph ML tasks, and introduce novel OT based divergences between (attributed) graphs, derived from the (F)GW distances. We call them respectively the *semi-relaxed Gromov-Wasserstein* (srGW) and *semi-relaxed Fused Gromov-Wasserstein* (srFGW) divergences.

After discussing srFGW properties and motivating its use in ML applications (Section 6.2.1), we propose efficient solvers for the corresponding optimization problem or regularized versions (Section 6.2.2), which can benefit from modern parallel programming. We empirically demonstrate the relevance of our divergence on diverse unsupervised tasks, by recovering state-of-the-art performances at a significantly lower computational cost compared to methods based on pure (F)GW.

- (Vincent-Cuaz et al., 2022a) Semi-relaxed gromov-wasserstein divergence and applications on graphs. Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. *In International Conference on Learning Representations, ICLR 2022. Also presented at Conférence sur l'Apprentissage automatique, CAp.*

- (Vincent-Cuaz et al., 2022b) Semi-relaxed gromov-wasserstein divergence for graphs classification. Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer and Nicolas Courty. *In XXVIIIème Colloque Francophone de Traitement du Signal et des Images, GRETSI 2022.*

Chapter 7: Conclusion

This last Chapter is a short conclusion to the manuscript. It summarizes the contributions presented in this thesis, and discusses issues that I believe are important for future research in the field.

Chapter 2

Introduction to Graph Representation Learning

Contents

This chapter aims to provide a concise overview of *Representation Learning* (RL) based approaches dedicated to graph-structured data, to operate at the *instance-level*. RL consists in learning representations of the data that encode useful information when building classifiers or other predictors. To this end, many approaches have been proposed over the last decades and have proven to be effective in dealing with many types of problems related with vector data, images or texts (Bengio et al., 2013). Such data can be considered as discrete probability distributions belonging to a common ambient topological space, *unlike* graphs, which are intrinsically characterized by their own topologies. Consequently, graph-structured data requires specific modeling which has recently given rise to the paradigm of *Graph Representation Learning* (GRL).

In a nutshell, GRL aims to encode a graph, potentially endowed with node features, into a vector representation called *graph embedding*, which eases downstream Graph Machine Learning tasks. A flourishing body of research addresses this challinge drawing on various graph-related fields, such as Graph Theory (Bollobás, 1998; West et al., 2001; Chung, 1997) and non-Euclidean Geometry (Greenberg, 1993). These works relate to diverse types of approaches, such as Signal Processing (Ortega et al., 2018), Bayesian Statistics (Stephenson, 2000), Kernel methods (Kriege et al., 2020) or Geometric Deep Learning (Bronstein et al., 2017). They offer representations that are either *implicit* or *explicit*, and may focus on *node-level* or *instance-level* knowledge. Thus, since GRL lies at the crossroads of different fields, it is a hard task to give a systematic description of it. In the following we make an overview of GRL from two different perspectives, well fitting the contributions of this thesis.

First, we introduce methods that rely on the design of effective (dis)similarity measures to compare graphs (Section 2.1). Typical instances are *Graph Kernel methods* that refer to ML algorithms learning from the comparison of data points (here graphs) using specific pairwise similarities, *i.e* kernels (Kriege et al., 2020). Most often these approaches model graphs *implicitly* through their pairwise relations, relying on the kernel trick (Theodoridis & Koutroumbas, 2006, Section 4.19). However, the graph representations can also be expressed

explicitly, in a specific Hilbert or Krein space, provided that the related feature maps are tractable. Kernel-based methods lead to deterministic or probabilistic representations. Pioneering work on learnable graph representations involves combining kernels in a convex fashion (Rakotomamonjy et al., 2008). However, the resulting kernels are subject to overfitting when the information contained in the composite kernels overlaps, making complex the choice of the latter.

Second, we introduce in Section 2.2 a family of more flexible methods designed to learn end-to-end graph representations. Such modern approaches relate to Geometric Deep Learning (Bronstein et al., 2021a) and Graph Neural Networks (Hamilton, 2020; Wu et al., 2020), that aim at extending the success of Deep Learning to graph data (Goodfellow et al., 2016; Pouyanfar et al., 2018).

2.1 Similarity based approaches

We briefly summarize here various similarity based approaches operating on graphs. The most common paradigm relies on *kernels* whose theoretical foundations are reported in Section 2.1.1. Then we review typical kinds of Graph Kernels that we distinguish according to whether they focus on local information within graphs (Section 2.1.2) or directly on their global topologies (Section 2.1.3).

2.1.1 On the theoretical foundations of kernel methods

General concepts. In the heart of (Graph) Kernel methods lies a *kernel* function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is continuous on the topological product space $\mathcal{X} \times \mathcal{X}$[1]. If this kernel is *symmetric positive definite* (PD) on $\mathcal{X}^2$[2], then the Moore-Aronszajn (Aronszajn, 1950) and Riesz representation (Riesz, 1909) theorems allow to assign it a unique *Reproducing Kernel Hilbert space* $\mathcal{H}$, such that for any $(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{X} \times \mathcal{X}$, $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$ where $\phi : \mathcal{X} \to \mathcal{H}$ is called a *feature map*. So $\mathcal{H}$ is understood as the *feature space*. This relation enables Kernel methods to operate in a high-dimensional (potentially infinite) feature space, which can be let *implicit* to avoid untractable computation of representations $\phi(\boldsymbol{x})$, by simply computing the inner products in $\mathcal{H}$ through k. This operation is known as the *kernel trick* (Mohri et al., 2018, Chapters 5-6) and allows one to model complex data only through a pairwise kernel similarity matrix.

Kernel methods have widely been adopted in Machine Learning. For instance, Support Vector Machines (Cortes & Vapnik, 1995, SVM) will often be used for *graph classification* in this manuscript. In binary classification, SVM look for a maximum-margin hyperplane that linearly separate data points belonging to distinct classes. As data may often exhibit a poor linear separability between classes in the input space $\mathcal{X}$, SVM use implicit embedding into the high-dimensional feature space $\mathcal{H}$ to promote it. Estimating a SVM comes down to solve a constraint Quadratic Program whose Hessian depends on the Gram Matrix $\boldsymbol{K} = (k(\boldsymbol{x}_i, \boldsymbol{x}_j))_{ij \in [\![N]\!]}$ (Theodoridis & Koutroumbas, 2006, Section 4.18). Variants of these kernel-based methods have later been derived for regression (Drucker et al., 1996) and clustering (Ben-Hur et al., 2001). Finally, kernels have been studied in different frameworks than SVM, *e.g* Gaussian processes (Rasmussen, 2003), Kernel PCA or K-means for unsupervised learning (Schölkopf et al., 1997), and Kernel Density Estimation (Silverman, 2018).

Empirical limitations. In practice the design of PD kernels can be complex, especially for data characterized by various local or global invariants like images, or graphs (Feragen et al.,

[1] Or equivalently, each component of the kernel defined on $\mathcal{X}$ is continuous.

[2] k is PD if for any $n \in \mathbb{N}$, any elements $\{\boldsymbol{x}_i\}_{i \in [\![n]\!]} \subset \mathcal{X}$ and any scalars $\{c_i\}_{i \in [\![n]\!]} \subset \mathbb{R}$, we have $\sum_{i,j \leq n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

2015). To ease a such design, research partly focused on finding specific kernel that could be easily transformed into PD kernel. For instance, if a kernel ψ is conditionally negative definite (CND) [3], then the kernel $k = e^{-\lambda\psi}$ is PD for all $\lambda > 0$ (Schoenberg, 1938; Feragen et al., 2015, Theorem 5).

Many works also aimed at allowing the use of indefinite kernels. First, by proposing specific QP solvers that aim at approximating solutions while seeing the indefinite kernel as a noisy observation from an underlying positive definite signal (Luss & d'Aspremont, 2007; Chang & Lin, 2011). Second, by studying the theoretical foundations behind indefinite kernels, that boils down to an extension of Reproducing Kernels to Krein Spaces (Loosli et al., 2015, and references therein).

Although these inherent constraints on kernel methods may hinder their expressiveness, they can in some ways be relaxed by the theory of learning with similarity functions developed by Balcan et al. (2008). Authors proposed a natural general notion of a "good" similarity functions, including many indefinite kernels, and prove that they are sufficient for learning in classification problems. Specifically, they propose a two-stage approach where: i) a subset (still potentially large) of data points called *landmarks or templates* are sampled; ii) each data point is embedded into its similarities to the landmarks, forming a similarity vector. Standard methods such as Logistic Regression or SVM can then be used to classify the embeddings. Such approaches have been studied for graph geometric embeddings developed in Section 2.1.3 (Johansson & Dubhashi, 2015), and motivated some contributions of this manuscript detailed in Chapter 4.

Focus on graph data. Before further detailing how kernels dedicated to graphs can be built, we underline the omnipresence of invariants and symmetries in these data. Usually a graph is modeled as a tuple $\mathcal{G} = (V, E)$, composed of a set of vertices V indexed in $[\![n]\!] \subset \mathbb{N}$ (potentially assigned to attributes in $\mathbb{R}^d$), and a set of edges E quantifying connections between vertices.

Since the node indexing is arbitrary, models operating on graphs must be invariant to the permutation of nodes. Graph comparison methods, such as kernels, must therefore themselves be permutation invariant. Furthermore, these data may be characterized by specific substructures whose redundancy and/or variability must be taken into account in some way when comparing them. With these remarks in mind, we present below various graph kernels that rely on the identification and comparison of substructures with a specific granularity (Section 2.1.2), before detailing kernels that rely instead on global patterns (Section 2.1.3).

2.1.2 Substructure based Graph Kernels

Divide and Conquer strategies. A majority of graph kernels are instances of the so-called *convolution kernels*. Building upon Haussler's Convolution Framework (Haussler et al., 1999), the methods compare graphs by first dividing them into substructures of various granularity, *e.g.* vertices or subgraphs, and then evaluating *base kernels* between each pair of such sub-instances. More formally, let us consider a space of K components $\mathcal{C} = \mathcal{C}_1 \times ... \times \mathcal{C}_K$ such as any graph $\mathcal{G}$, *e.g.* included in a dataset of graphs $\mathcal{D}$, can be decomposed into elements of $\mathcal{C}$. Then consider a mapping $R : \mathcal{C} \to \mathcal{D}$ from components of $\mathcal{C}$ to graphs in $\mathcal{D}$ such as $R(c) = \mathcal{G}$ if and only if the components $c \in \mathcal{C}$ build back up the graph $\mathcal{G}$. Then the reciprocal image of R, denoted $R^{-1}(\mathcal{G}) = \{c \in \mathcal{C} | R(c) = \mathcal{G}\}$, consists in the set of all components of a graph $\mathcal{G}$ that we wish to compare. The $R-$*convolution kernel* between two graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ is

[3] A symmetric kernel ψ is CND if: i) $\forall \boldsymbol{x} \in \mathcal{X}, \psi(x, x) = 0$; ii) For any $n \in \mathbb{N}, \forall \{\boldsymbol{x}_i\}_{i\in[\![n]\!]} \subset \mathcal{X}, \forall \{c_i\}_{i\in[\![n]\!]} \subset \mathbb{R}$ s.t $\sum_i c_i = 0$, then $\sum_{ij} c_i c_j \psi(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 0$

computed as follows:

$$k_{CV}(\mathcal{G},\overline{\mathcal{G}}) = \sum_{c\in R^{-1}(\mathcal{G})}\sum_{\overline{c}\in R^{-1}(\overline{\mathcal{G}})}\prod_{i=1}^{K} k_i(c_i,\overline{c}_i) = \sum_{c\in R^{-1}(\mathcal{G})}\sum_{\overline{c}\in R^{-1}(\overline{\mathcal{G}})} k(c,\overline{c}) \tag{2.1}$$

where k_i is a kernel on the i^{th} component space $\mathcal{R}_i$.

The *vertex label kernel* k_{VL} (Kriege et al., 2020) is a simple instance of R-convolution for which the mapping R takes the attributes $x_u \in \mathcal{R}$ of each vertex $u \in V \cup \overline{V}$ composing both compared graphs $\mathcal{G} = (V, E)$ and $\overline{\mathcal{G}} = (\overline{V}, \overline{E})$, and maps them to the graph that u is a member of. This kernel comes down to comparing graphs at a node-level such that $k_{VL}(\mathcal{G},\overline{\mathcal{G}}) = \sum_{x_u\in\{x_u|u\in V\}}\sum_{x_v\in\{x_v|v\in V\}} k(x_u,x_v)$, where k is a base kernel.

Interestingly, if the kernels on substructures are invariant to orderings of nodes or edges, so is the graph kernel. However, as such approaches rely on sums over all pairs of components, they might suffer from several computational and expressiveness limitations. The latter relates to the so called "diagonal dominance problem", that arises when sub-instances become too specific, so each graph becomes increasingly similar to itself but no longer to any other graph (Greene & Cunningham, 2006). First methods to tackle this problem mostly consists in adding importance weights to the components (Yanardag & Vishwanathan, 2015; Aiolli et al., 2015) or reducing the sum to predefined subset of pairs (Shin & Kuboyama, 2008).

Another paradigm aims at finding optimal assignments between selected substructures leading to *Optimal Assignment (OA) kernels* (Fröhlich et al., 2005). For example, in the first OA kernel, each node of both compared graphs is assigned to a label, acting as node representation, which is associated to a base kernel. Then, the similarity between the two graphs is computed based on a correspondence of their node representations, which maximizes their pairwise similarities *w.r.t* the base kernel. More formally, assume that graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ are respectively decomposed as sets $c = \{c_1, ..., c_n\}$ and $\overline{c} = \{\overline{c}_1, ..., \overline{c}_n\}$, then for a given base kernel k, the OA kernel between both graphs reads as

$$k_{OA}(\mathcal{G},\overline{\mathcal{G}}) = \max_{\sigma\in\mathfrak{S}_n}\sum_{i=1}^{n} k(c_i,\overline{c}_{\sigma(i)}) \tag{2.2}$$

These kernels have few drawbacks. First they usually do not lead to PSD kernels (Vert, 2008; Vishwanathan et al., 2010), except for certain types of "strong" base kernels (Kriege et al., 2016). Second they impose on the compared graphs to be decomposed into the same number of components, *e.g* to have the same number of nodes if the division is node-wise. So they require the use of padding operations to handle graphs of various orders. Finally, the OA kernel can be extended by performing *soft matching* between any component sets, modeled as probability distributions of n and $\overline{n}$ bins using Optimal Transport (Nikolentzos et al., 2017). This latter paradigm, that will be further developed in Chapter 3, generally leads to an indefinite kernel (Naor & Schechtman, 2007; Nikolentzos et al., 2017).

We know provide an overview of ways to decompose graphs into substructures of various granularity.

Node embeddings via neighborhood aggregation. The most natural way to partition a graph consists in considering the nodes that compose it. As node features are most often not representative of the graph topology, some kernels build upon node embeddings or labels that summarize information from the nodes neighborhood. One representative of such approaches, for graphs potentially endowed with discrete node features, is the *Weisfeiler-Lehman (WL) subtree kernel* (Shervashidze et al., 2011), based on the 1-dimensional WL test of graph isomorphism or color refinement algorithm (Weisfeiler & Leman, 1968). Let us consider two attributed graphs $\mathcal{G} = (V, E)$ and $\overline{G} = (\overline{V}, \overline{E})$, plus a mapping $l_0 : V \cup \overline{V} \to Y_0$ from

their vertices to their node labels (or colors). At each iteration $t \in [0, ..., T] \subset \mathbb{N}$, the 1-WL algorithm computes new label functions $l_t : V \cup \overline{V} \to Y_t$, such that for any node v,

$$l_t(v) = \text{relabel}\left([l_{t-1}(v), \text{sort}(\{\{l_{t-1}(u) | u \in N(v)\}\})]\right) \tag{2.3}$$

where $N(v)$ is the neighborhood of v. sort($\mathbb{S}$) orders the multiset $\mathbb{S}$ and relabel maps the pair of the previous node label $l_{t-1}(v)$ and multiset of its neighbors $\{\{l_{t-1}(u) | u \in N(v)\}\}$ to a *unique* new label in $Y_t \subset \mathbb{N} / \cup_{i=0}^{t-1} Y_i$. The algorithm might be stopped if $t = T$ or if the respective images of l_{t-1} and l_t have the same cardinality. Finally for $\mathcal{G}$ (resp. $\overline{\mathcal{G}}$), the feature vector $\phi(\mathcal{G})$ (resp. $\phi(\overline{\mathcal{G}})$) is the concatenation over t of counting vectors summarizing occurrences of labels in Y_t. Then this convolution-like graph kernel reads as

$$k_{WL}(\mathcal{G}, \overline{\mathcal{G}}) = \sum_{i \in \text{card}(\cup_t Y_t)} \mathbf{1}_{\phi(\mathcal{G})_i = \phi(\overline{\mathcal{G}})_i} \tag{2.4}$$

Originally, the 1-WL test can detect that two graphs are not isomorphic at any iteration t if color counts $\phi_{|Y_t}(\mathcal{G})$ and $\phi_{|Y_t}(\overline{\mathcal{G}})$ are different. However, it can only identify whether graphs are *potentially isomorphic* (see *e.g.* Sato (2020), Section 3.1, and references therein). Informally, this is mostly due to the lack of memory over past iterations while iteratively looking at multisets of colors at iteration t which *implicitly* merges knowledge of t-hop neighborhoods. This process might suffer from overlaps when graphs are regular, or contain substructures like cliques or cycles. Shervashidze et al. (2011) also proposed various ways to iteratively explore or define node neighborhood, *e.g* with the WL edge or shortest-path kernels. These 1-WL approaches can be extended to k-WL tests where the coloring is achieved on each tuple of k nodes (Morris et al., 2017). Other methods rather focus on *explicit depth-based propagation* to iteratively get node representations, often relying on the Shannon entropy of explored subgraphs of depth t at iteration t (Bai et al., 2014; 2015). Finally different base kernels and coloring schemes have been introduced to handle continuous node features (Morris et al., 2016; Neumann et al., 2016).

Simple aggregation schemes as the one described in equation (2.4) have been improved using optimal assignments or soft matchings (see *e.g.* Kriege et al. (2016); Togninalli et al. (2019)).

Subgraphs based kernels. Another partition strategy is to consider graphs as sets of subgraphs instead of sets of nodes or edges. For instance, Shervashidze et al. (2009) proposed to count occurrences of subgraph patterns of a fixed size called *graphlets*. These refer to canonical subgraph representations of a given order m, *e.g* graphlets of order $m = 3$ are all possible equivalence classes composed of unattributed graphs with 3 nodes up to node permutations.

Subgraphs detection allows to leverage direct encoding of a graph topology, but counting graphlets is not trivial and their number grows exponentially with their order m. Efficient heuristics have been proposed to detect those with small orders (Shervashidze et al., 2009; Ribeiro et al., 2021; Hočevar & Demšar, 2014) or by relaxing the subgraph isomorphism condition thanks to expressive but non-injective encoding schemes (Costa & De Grave, 2010). Two main aspects have been studied to address the computational complexity of these approaches. The first consists in estimating graphlet counts by diverse subgraph sampling strategies (Ahmed et al., 2016; Chen & Lui, 2018; Bressan et al., 2017), whereas the second aims at restraining the scope of searched graphlets, *e.g.* to cycles or trees (Horváth et al., 2004; Ramon & Gärtner, 2003; Mahé & Vert, 2009).

As these kernels got more computationally tractable, some have been adapted to graphs with discrete attributes just by endowing the subset of graphlets with all possible node labels.

Naturally, graphs with continuous attributes require these kernels to be coupled with custom base kernels handling vertices or subgraphs comparison (Kriege & Mutzel, 2012).

2.1.3 Global structure based kernels

Paths and walks. To circumvent the dependence of subgraph-based kernels on specific sets of patterns, several works have proposed to compare graphs through sequences formed by traversing their nodes. The two most common traversal schemes are based on shortest paths (SP) or random walks (RW).

Borgwardt & Kriegel (2005) proposed the SP kernel, whose similarity between two graphs $\mathcal{G} = (V, E)$ and $\overline{\mathcal{G}} = (\overline{V}, \overline{E})$ is computed by comparing attributes and shortest path lengths between all pairs of nodes respectively composing them. The base kernel defined on $V^2 \times \overline{V}^2$ fuses feature and topology knowledge, by multiplying outputs of a kernel on $V \times \overline{V}$ comparing node features, and another kernel defined on $V^2 \times \overline{V}^2$ comparing shortest path lengths. Interestingly, feature maps associated to this kernel can be computed explicitly.

A consequent part of the graph kernel literature considers random walks, *i.e* sequences $[u_0, ..., u_t, u_{t+1}, ..., u_T]$ of nodes along with their labels, potentially with repetitions. These are generated by iteratively traveling from a node u_t to another node u_{t+1} randomly sampled in its neighborhood $N(u_t)$. As such, graphs can be considered as Markov chains (Gärtner et al., 2003; Kashima et al., 2003; Vishwanathan et al., 2010; Mahé et al., 2004). This is supported by a probabilistic view of kernels related to the notion of *marginalized kernels*. The feature space of the kernel derives from the infinite set of unbounded and possible sequences produced by walks. Interestingly, these kernels often admit recursive formulations which at the end of the day boils down to finding the stationary state of a Markov Chain, either given by closed-form formula computed by matrix inversion, or estimated by fixed-point solvers (Vogel, 2002).

Global graph embeddings and properties. Other approaches directly rely on specific graph matrix representations to capture global knowledge of the graph topology. Famous instances are variants of the Laplacian (PD) matrix, defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{2.5}$$

where $\boldsymbol{D}$ is the diagonal matrix of node degrees and $\boldsymbol{A}$ the graph adjacency matrix.
These methods leverage a probabilistic interpretation of graphs, to model them as a discrete probability distributions, easing their comparison (Kondor & Pan, 2016). Indeed one can construct a Gaussian Markov random field over the nodes of the graph seen as random variables (Rue & Held, 2005), such that two graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ are respectively assigned to distributions $\mu = \mathcal{N}(\mathbf{0}, \boldsymbol{L}^{-1})$ and $\overline{\mu} = \mathcal{N}(\mathbf{0}, \overline{\boldsymbol{L}}^{-1})$. Then both distributions can be compared using for instance the Bhattacharyya kernel $k(\mu, \overline{\mu}) = \int \sqrt{\mu(x)\overline{\mu}(x)}dx$ that can be computed in closed form for Gaussian distributions (Jebara & Kondor, 2003). Kondor & Pan (2016) also studied the feature space associated to this Laplacian graph kernel, to derive multiscale Laplacian kernels capturing from the relationships between their vertices but also between subgraphs.

Related approaches aim at obtaining *geometric embeddings* of the nodes *e.g.* through matrix decomposition of the Laplacian based matrices or the incidence matrix (Johansson & Dubhashi, 2015). For instance, Johansson & Dubhashi (2015) proposed to use Cholesky decompositions of the Laplacian or pseudo-inverse Laplacian to get such embeddings (Gentle, 1998). Authors also studied embeddings using orthonormal basis inherent to the computation of the Lovász number (Johansson et al., 2014; Lovász, 1979, Definition 1) which is a graph global statistic.

Finally, global information about graph topologies and features can be obtained by aggregating knowledge from various kernels, operating at any granularity, preferably if the information they contain is not too redundant to avoid overfitting. Indeed, since positively weighted sums of PD kernels still is a PD kernel, several works proposed to jointly learn optimal kernel weights and support vectors (*e.g.* Rakotomamonjy et al. (2008)) to perform graph classification (Kriege, 2019; Salim et al., 2020).

2.2 Graph Neural Networks

As discussed in Section 2.1, all kernel methods that historically constitute the foundations of GRL, lead to representations which are either deterministic, or probabilistic, if a random process underlies the kernel design. The multiple kernel learning paradigm that learns linear combinations of kernels to benefit from their individual specificity can be considered as a first axis of learnable graph representations. But one might argue that these methods lack of flexibility and might rather seek for fully learnable graph representations. Nowadays, such a goal is what is often understood as Graph Representation Learning (Hamilton, 2020) or, in a slightly wider meaning, as Geometric Deep Learning (Bronstein et al., 2017). These refer to emerging techniques attempting to extend Deep Learning models (Goodfellow et al., 2016; Pouyanfar et al., 2018) to non-Euclidean domains, such as graphs, mainly relying on Graph Neural Networks (GNN).

We provide next a short overview on this topic and refer the reader interested in more details to the recent books (Hamilton, 2020; Bronstein et al., 2021a) and surveys (Wu et al., 2020; Zhou et al., 2020; Kazemi et al., 2020; Zheng et al., 2022). We first introduce predominant methods that focus on *node embedding schemes* (Section 2.2.1) in the vein of Section 2.1.2, before introducing *graph pooling* approaches that focus on more global graph knowledge (Section 2.1.3).

Notice that such approaches mostly focus on encoding the graph topology and node features to perform supervised or semi-supervised down-stream task, such as node or graph classification, usually in an end-to-end fashion. However, they can also be directly plugged as the encoder in a Deep Autoencoder architecture (Hinton & Salakhutdinov, 2006) for graph reconstruction or generation purposes, using simple heuristics directly over the latent space as decoder (Wu et al., 2020, Section 6).

2.2.1 Node embeddings in GNN

GNN based methods aim at generating node representations that contain knowledge from both the graph topology and features. Most of them do so thanks to a form of *neural message passing* in which vector messages are exchanged between nodes and updated using neural networks (Gilmer et al., 2017).

Message Passing Neural Networks (MPNN). Assume that we observe an input graph $\mathcal{G} = (V, E)$, of order n, whose node features in $\mathbb{R}^d$, are collected as rows in the matrix $\boldsymbol{F} = (\boldsymbol{f}_u)_{u \in [\![n]\!]} \in \mathbb{R}^{n \times d}$. For all nodes $u \in V$, the common message-passing (MP) at the l^{th} iteration in a GNN, consists in updating an hidden node embedding $\boldsymbol{h}_u^{(l)}$ by aggregating information from the node neighborhood $N(u)$ encoded in $\{\boldsymbol{h}_v^{(l-1)} | v \in N(u)\}$. For one GNN layer, this process can be expressed as follows (considering $\boldsymbol{h}_u^{(0)} = \boldsymbol{f}_u$):

$$\boldsymbol{h}_u^{(l)} = \text{update}^{(l-1)} \left(\boldsymbol{h}_u^{(l-1)}, \text{aggregate}^{(l-1)}(\left\{ \boldsymbol{h}_v^{(l-1)} | \forall v \in N(u) \right\}) \right) \tag{2.6}$$

where $\forall l \in [1, L]$, $\text{update}^{(l)}$ and $\text{aggregate}^{(l)}$ are arbitrary differentiable functions, typically neural networks. The aggregation function takes as input the multiset of neighbor embeddings.

FIGURE 2.1: Illustration of the mechanism of a MPNN. The update and aggregate functions are summarized by an application $\phi_{\boldsymbol{u}}$ parameterized by $\boldsymbol{u}$ that can be decomposed into L components or layers.

As such, after L iterations or layers, the embeddings $\{\boldsymbol{h}_u^{(l)}\}$ encode knowledge from the L-hop neighbor structures and features, and can be used as such to perform *e.g.* node classification. The mechanism of MPNN is illustrated in Figure 2.1.

Initially, only the node embeddings from the last layer $\{\boldsymbol{h}_u^{(L)}\}_{u\in V}$ were considered as node representations. However a common practice first proposed by Xu et al. (2018) and inspired from *skip-connections* (Srivastava et al., 2015), consists in concatenating those layer-wise node embeddings to form final node representations, to benefit from the encoding of various neighborhood lengths. Overall this procedure clearly resembles the neighborhood aggregation scheme used for the WL kernels (equation 2.3), except that differentiable operations are applied. The diversity of these operations is what mostly forms the taxonomy of GNN whose brief synthesis is provided in the following paragraphs

Spatial GNN. Methods that mostly rely on the adjacency of nodes are often referred as Spatial GNN. Naturally they relate to the convolution operation of Convolutional Neural Networks performed on grids such as images (LeCun et al., 1995). These methods then differ depending on the chosen "update" and "aggregate" functions in equation (2.6).

A historically famous and efficient way to aggregate the neighborhood message is by summation, *i.e* $\sum_{v\in N(u)} \boldsymbol{h}_v^{(l)}$. For instance, one first GNN proposed by Scarselli et al. (2008) considers the update:

$$\boldsymbol{h}_u^{(l)} = \sigma\left(\boldsymbol{W}_{self}^{(l)}\boldsymbol{h}_u^{(l-1)} + \boldsymbol{W}_{neigh}^{(l)}\sum_{v\in\mathcal{N}(u)}\boldsymbol{h}_v^{(l-1)} + \boldsymbol{b}^{(l)}\right) \tag{2.7}$$

where $(\boldsymbol{W}_{self}^{(l)}, \boldsymbol{W}_{neigh}^{(l)})$ and $\boldsymbol{b}^{(l)}$ are respectively trainable parameter matrices and bias, and σ is a non-linear activation function (element-wise). A recent state-of-the-art method from Xu et al. (2019c), the Graph Isomorphism Networks (GIN), considers the update

$$\boldsymbol{h}_u^{(l)} = \text{MLP}^{(l)}\left((1+\varepsilon^{(l)})\boldsymbol{h}_u^{(l-1)} + \sum_{v\in N(u)}\boldsymbol{h}_v^{(l-1)}\right) \tag{2.8}$$

where $\varepsilon^{(l)} \geq 0$, and a MLP with two layers and non-linear activations is used for the embedding update. We refer to Hamilton (2020), Section 5.1, for explanations regarding the asymmetric processing of node self-embedding and neighbor embeddings.

A more involved concept incorporates an attention weight (Bahdanau et al., 2014) or importance to each neighbor, using $\sum_{v\in N(u)} \alpha_{u,v}^{(l)}\boldsymbol{h}_v^{(l)}$ as aggregation function. A first instance is the Graph Attention Network (Veličković et al., 2018, GAT) that defines the attention weights as

$$\alpha_{u,v}^{(l)} = \frac{\exp\{\boldsymbol{a}^\top\boldsymbol{W}^{(l)}\left[\boldsymbol{h}_u^{(l-1)}, \boldsymbol{h}_v^{(l-1)}\right]\}}{\sum_{v'\in\mathcal{N}(u)}\exp\{\boldsymbol{a}^\top\boldsymbol{W}^{(l)}\left[\boldsymbol{h}_u^{(l-1)}, \boldsymbol{h}_{v'}^{(l-1)}\right]\}} \tag{2.9}$$

where for any $(u,v)\in V\times V$, $\left[\boldsymbol{h}_u^{(l-1)}, \boldsymbol{h}_v^{(l-1)}\right]$ is the concatenation of both node embeddings at layer $(l-1)$. Other variants consider bilinear inner operations or more involved mechanisms (see *e.g.* Zhu et al. (2021) and references therein), potentially with multi-head attentions in the vein of Transformers (Vaswani et al., 2017).

Many other principles exist, such as *set aggregation* that first applies a (non-linear) MLP on neighbor embeddings before summation and provides a universal set function approximator (Zaheer et al., 2017; Hamilton et al., 2017). Other methods directly see the neighbor embeddings as sequences which are then processed and aggregated with Recurrent Neural Networks (Li et al., 2015; Wu et al., 2020, Section 4).

Spectral GNN. A thorough analysis of the convolution operations from Euclidean to Non-Euclidean domains, detailed in Bronstein et al. (2017; 2021a), motivated the use of graph signal filters derived from the (normalized) Laplacian matrix (resp. operator) to operate on graphs (resp. manifolds).
Since $\boldsymbol{L}$ is a PSD matrix it can be diagonalized in an orthonormal basis of eigenvectors $\boldsymbol{U} = (\boldsymbol{u}_i)_{i\in[\![n]\!]} \subset \mathbb{R}^{n\times n}$, called the Graph Fourier Transform operator, such that $\boldsymbol{L} = \boldsymbol{U}\Lambda\boldsymbol{U}^\top$. Then in Graph Signal Processing (Ortega et al., 2018), a graph signal $\boldsymbol{x} \in \mathbb{R}^n$ is a feature vector of all nodes of a graph. Using the graph Fourier transform $\boldsymbol{x} \to \boldsymbol{U}^\top\boldsymbol{x}$ and its inverse $\boldsymbol{y} \to \boldsymbol{U}\boldsymbol{y}$, one can express the graph convolution of the input signal $\boldsymbol{x}$ with a parameterized filter $\boldsymbol{g}_\theta \in \mathbb{R}^n$ as $\boldsymbol{x} \star_\mathcal{G} \boldsymbol{g}_\theta = \boldsymbol{U}(\boldsymbol{U}^\top\boldsymbol{x} \odot \boldsymbol{U}^\top\boldsymbol{g}_\theta)$. Spectral CNN (Bruna et al., 2013) considered $\boldsymbol{f}_\theta = \mathrm{diag}(\boldsymbol{U}^\top g_\theta)$ to factor the convolution as $\boldsymbol{x} \star_\mathcal{G} \boldsymbol{f}_\theta = \boldsymbol{U}\boldsymbol{f}_\theta\boldsymbol{U}^\top\boldsymbol{x}$, which can then be seen as a message passing operation in a spectral domain.

Due to the eigen-decomposition of the Laplacian matrix, Spectral CNN suffers from sensitivity issues, as any graph perturbation induces an eigenbasis change, and the learned filters cannot be applied to a different graph structure. Moreover, the decomposition requires $O(n^3)$ operations that prevents its application to large graphs. Several improvements were made to overcome these limitations by using approximations and simplifications, *e.g* using K-order polynomial functions (Defferrard et al., 2016, Chebnet) or first-order Taylor expansions (Kipf & Welling, 2016, GCN). The latter methods respectively boil down to the aggregation of K-hop and 1-hop neighborhood knowledge. As such, GCN is equivalent to a Spatial GNN with non-parametric reweighed messages given by the following update rule:

$$\boldsymbol{h}_u^{(l)} = \sigma\left(\boldsymbol{W}^{(l)} \sum_{v\in\mathcal{N}(u)\cup\{u\}} \frac{\boldsymbol{h}_v^{(l-1)}}{\sqrt{|N(v)||N(u)|}}\right) \tag{2.10}$$

The expressive power of GNN. The most recent research on GNN has focused on the expressiveness of the models, *e.g.* it has been shown that GIN is as expressive as a 1-WL test (Xu et al., 2019c). These considerations deepen the roots of GNN methods in the literature on Graph Kernels, Graph Theory and distributed local algorithmic. For instance, new GNN models proposed to extend the neural message passing to higher-order neighborhoods modeled as tuples, inspired from the k-WL tests or kernels (Morris et al., 2019; Maron et al., 2019a; Chen et al., 2020c). Note that even if clear progress is observed on GNN expressiveness, it is not necessarily coupled with improvements in terms of generalization capabilities, so both aspects remain the biggest challenges of supervised Graph Machine Learning. We refer the reader to the survey of Sato (2020) for a comprehensive overview of studies on GNN expressiveness.

2.2.2 Graph pooling operations

While many methods for node embeddings have been studied (Section 2.2.1), less attention has been devoted to Graph pooling layers (Xu et al., 2019c; Mesquita et al., 2020; Cheung et al., 2020). These can be divided into two categories, namely *global and hierarchical pooling* operators. Briefly, the first paradigm consists in factoring a graph as a vector to ease its processing for down-stream tasks. Whereas, the second aims at reducing the size of a graph, *while preserving a graph structure*, instead of reducing it to a simple vector.

Notice that global and hierarchical pooling operators have different roles and both can be part of the same graph-level GNN. The former compute graph embeddings to interface with traditional Deep Learning layers, while the latter provide a multi-resolution representation of the graph from which the GNN can gradually distill high-level properties.

Taxonomy. Grattarola et al. (2022) recently proposed the first taxonomy of Graph Pooling operators, as combinations of three functions:

- The *selection function* (S) groups the nodes $\{\boldsymbol{v}_i\}_{i\in[\![n]\!]}$ of the input graph $\mathcal{G}$ into subsets $\{S_k\}_{k\in[\![K]\!]}$, called supernodes, with $K < n$, such that $S_k = \{(\boldsymbol{v}_i, s_i)|s_i > 0\}$ where the value s_i scores how much node $\boldsymbol{v}_i$ contributes to S_k.
- The *reduction function* (R) computes embedded (super)node attributes $\{\boldsymbol{v}_k^{(1)}\}_{k\in[\![K]\!]}$ by aggregating input node attributes selected in each supernode S_k.
- The *connection function* (C) links the embedded and reduced nodes $\{\boldsymbol{v}_k^{(1)}\}_{k\in[\![K]\!]}$ with (possibly attributed) edges and outputs the pooled graph $\mathcal{G}^{(1)} = (V^{(1)}, E^{(1)})$, with embedded edges $E^{(1)}$ and node features $V^{(1)} = \{\boldsymbol{v}_k^{(1)}\}_{k\in[\![K]\!]}$.

In this framework, a global pooling operation is a degenerate case where $K = 1$.

Global pooling operators. Usually for graph-level tasks, a first step consists in computing node representation $\{\boldsymbol{h}_v\}_{v\in[\![n]\!]}$ of a given graph $\mathcal{G}$ (see Section 2.2.1), then a *readout* function provides the graph embedding:

$$\boldsymbol{h}_{\mathcal{G}} = \text{readout}\left(\{\boldsymbol{h}_v\}_{v\in[\![n]\!]}\right) \tag{2.11}$$

This representation $\boldsymbol{h}_{\mathcal{G}}$ is then fed *e.g* to a Multi-Layer Perceptron (MLP) for graph classification. The readout function, also called the *global pooling operator*, is usually a non-parametric (un)weighted sum operation taken across nodes (Li et al., 2015; Zhang et al., 2018; Wu et al., 2019; Atwood & Towsley, 2016; Bai et al., 2019), but can be more advanced and preferably injective (Navarin et al., 2019; Corcoran, 2020). A parametric global pooling operation using Optimal Transport (Chapter 3) distances constitute the main contribution of Chapter 4.

Hierarchical pooling operators. Grattarola et al. (2022) proposed to distinguish hierarchical pooling methods based on the following criterion: trainability, computational complexity (linear or quadratic), adaptivity of the pooled graph size.

These pooling methods, partly mentioned in Bruna et al. (2013), are popular graph clustering algorithms that previously found applications in Computer Vision, like variants of Independent Component Analysis (Coates & Ng, 2011; Hyvärinen & Oja, 2000), Spectral Clustering (Von Luxburg, 2007; Bravo Hermsdorff & Gunderson, 2019; Ma et al., 2019; Bianchi et al., 2020b), weighted or normalized graph cuts (Dhillon et al., 2007; Monti et al., 2017; Fey et al., 2018; Levie et al., 2018), or other graph decomposition techniques (Luzhnica et al., 2019; Noutahi et al., 2019; Xie et al., 2020a).

The state-of-the-art approaches consists in learnable operators that can dynamically adapt to a particular task to compute optimal pooling. Some well-known are based on (potentially hierarchical) clustering (Ying et al., 2018; Bianchi et al., 2020a) or other node selection mechanisms (Gao & Ji, 2019; Lee et al., 2019; Ranjan et al., 2020).

2.3 Conclusion

This chapter introduced diverse methods to obtain representations of (attributed) graphs useful for down-stream tasks such as node or graph classification. First, we detailed similarity based

approaches (Section 2.1) that mostly relate to the design of Graph Kernels, or relaxations such as Balcan's theory. These methods commonly lead to deterministic graph representations that might lack of flexibility and scalability. To circumvent such limitations we then introduced in Section 2.2 some concepts from the modern and flourishing Geometric Deep Learning literature to learn graph representations end-to-end. Note that Graph Kernels remain state-of-the-art methods on some datasets suggesting that GNN have not yet reached their full potential.

Next chapter (Chapter 3) will introduce a major focus of this thesis, namely Optimal Transport (OT), that can provide distances between node embeddings or graphs, hence directly linked to Section 2.1.

Chapter 3

Introduction to Optimal Transport for graphs

Contents

This chapter presents the theoretical foundations of Optimal Transport (OT) from an application perspective related to Graph Machine Learning. The OT theory originated in the 18th century to solve an everyday problem of moving one finitely divisible object to another according to the *least effort principle*. It resulted in an elegant way of comparing probability distributions while taking into account the geometry of the underlying space. Since graphs, seen as finite sets of nodes, are an instance of discrete probability distributions, a first objective of this chapter (Section 3.1) is to briefly present the main results of discrete OT theory, both mathematically and numerically. We refer the reader interested in complete overviews of these two aspects to the books Villani (2009); Santambrogio (2015) and Peyré & Cuturi (2019) respectively.

However, as graphs are naturally observed and characterised by pairwise interactions between their nodes, the aforementioned modeling (assuming the existence of a node ambient space shared by graphs) might omit their respective and diverse topologies. This paradigm has motivated the extension of OT limited to a single ambient space to OT across incomparable spaces (Mémoli, 2011; Sturm, 2012). The latter is addressed in Section 3.2 of this chapter,

whereas Section 3.3 deals with a mix of both concepts (Vayer et al., 2020), more suitable for graphs endowed with node features.

The introduction of the latter two OT problems across incomparable spaces will be complemented by some of our new contributions to graph distribution learning (Vincent-Cuaz et al., 2021), that will be used in the following Chapters. More specifically, we extend the seminal Graph Representation Learning approaches based on the estimation of graph barycenter using OT distances thanks to novel sub-gradient derivations provided in Theorems 5 and 7 (Sections 3.2.5 and 3.3.4).

3.1 Optimal Transport within a common space

In this section, we introduce the formulation of Optimal Transport between discrete measures that is ubiquitous in this manuscript, namely the Kantorovich optimal transport problem (Section 3.1.1). Then we detail how this formulation naturally leads to the definition of the so-called Wasserstein distances (Section 3.1.2). After providing insights on the geometric and statistical properties of these distances, we present several solvers for the optimization problem inherent to their computation, with a focus on those used in the following Chapters (Section 3.1.3). Finally, we develop the notion of barycenter with respect to the Wasserstein distances, before discussing some unsupervised representation learning approaches based on OT (Section 3.1.4).

3.1.1 Problem statements

The first mathematical formulation of the OT problem has been proposed in a Mémoire by Gaspard Monge (Monge, 1781). It was a question of finding the most efficient way to move a pile of sand, *de facto* subdivided into grains of limited granularity, from one place (*déblais*) to another (*remblais*).

Discrete measures. A natural way to formally represent such amorphous objects is to use discrete probability measures defined as follow. First, remark that we will use interchangeably the terms probability vector and histogram for any element $\boldsymbol{h} \in \Sigma_n$ that belongs to the probability simplex with n components:

$$\Sigma_n := \{\boldsymbol{h} \in \mathbb{R}_+^n | \sum_i h_i = 1\} \tag{3.1}$$

A discrete measure with masses $\{h_i\}_{i\in[\![n]\!]} \subset \mathbb{R}_+$ encoded in vector form as $\boldsymbol{h} \in \mathbb{R}_+^n$, and with corresponding *finite* locations $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\} \subset \mathcal{X}$, reads as

$$\mu = \sum_i h_i \delta_{\boldsymbol{x}_i} \tag{3.2}$$

where for all $i \in [\![n]\!]$, $\delta_{\boldsymbol{x}_i}$ is the Dirac measure at position $\boldsymbol{x}_i$, such that for any $\boldsymbol{x} \in \mathcal{X}$, $\delta_{\boldsymbol{x}_i}(\boldsymbol{x}) = \mathbb{1}_{\{\boldsymbol{x}_i = \boldsymbol{x}\}}$. The set of points $\{\boldsymbol{x}_i\}$ composing μ is referred as its support $\text{supp}(\mu)$. Such a measure μ describes a *discrete probability measure* as $\boldsymbol{h} \in \Sigma_n$. Notice that it is a specific case of an *discrete positive measure* whose total mass is constrained to be 1, instead of any real positive value.[1]

From Monge to Kantorovich OT problems. Under such formalism, the original problem can be cast as follows: given two probability distributions $\mu = \sum_{i\in[\![n]\!]} h_i \delta_{\boldsymbol{x}_i}$ and

[1] We refer the reader to Peyré & Cuturi (2019), Remark 2.2, for a generic definition of measures to which OT theory extends.

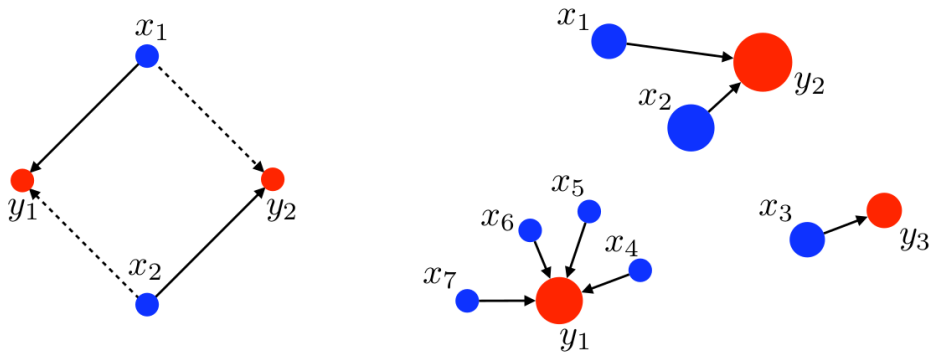

FIGURE 3.1: Left: blue dots for measure μ and red dots from measure $\overline{\mu}$ are pairwise equidistant. Hence, either matching $\sigma = (1, 2)$(full line) or $\sigma = (2, 1)$ (dotted line) is optimal. Right: a Monge map can associate the blue measure μ to the red measure $\overline{\mu}$. The weights h_i are displayed proportionally to the area of the disk marked at each location. The mapping here is such that $T(\boldsymbol{x}_1) = T(\boldsymbol{x}_2) = \boldsymbol{y}_2$, $T(\boldsymbol{x}_3) = y_3$, whereas for $i \in \{4, 5, 6, 7\}, T(\boldsymbol{x}_i) = \boldsymbol{y}_1$. Inspired from Peyré & Cuturi (2019).

$\overline{\mu} = \sum_{j \in [\![\overline{n}]\!]} \overline{h}_j \delta_{\boldsymbol{y}_j}$ of respectively n and $\overline{n}$ points, with potentially $n \neq \overline{n}$, how do we *transfer* all the mass of μ onto $\overline{\mu}$ so that the overall *effort* of transferring this mass is minimized?

Henceforth the notions of transfer and effort are to be expressed. Several assumptions were made to formalize such notions that led to various concepts of Optimal Transport. At first, the aforementioned transfer relates to a *map* T (also called a *Monge map*) that associates to each point $\boldsymbol{x}_i$ a single point $\boldsymbol{y}_j$ which must push the mass of μ towards $\overline{\mu}$'s one.
Then for this map to satisfy the *least effort principle*, it should minimize some transportation cost from $\text{supp}(\mu)$ to $\text{supp}(\overline{\mu})$ according to a certain cost function $c : (\boldsymbol{x}, \boldsymbol{y}) \in \text{supp}(\mu) \times \text{supp}(\overline{\mu}) \to \mathbb{R}$.

Seeing the notion of transfer as such leads to an oriented transportation problem, since it might be impossible to find a Monge map satisfying the *mass conservation constraints*, without allowing the probability mass h_i at location $\boldsymbol{x}_i$ to be sent to *various target locations* $\{\boldsymbol{y}_j\}_j$. We provide two instances of the described Monge's OT problems in Figure 3.1. Remark that assuming $\boldsymbol{h} = \overline{\boldsymbol{h}} = \frac{1}{n}\mathbf{1}_n$ (thus $n = \overline{n}$), as in the right plot with $n = 2$, the conservation of mass implies that the Monge map defines a bijection on $[\![n]\!]$. This emphasizes the complexity of such an OT problem because, even in this simplified scenario with uniform distributions of the same size, it seeks for an optimal solution among the non-convex and large set of permutations $\mathfrak{S}_n$, satisfying $\text{card}(\mathfrak{S}_n) = n!$. Finally, the infeasibility of Monge problem can be foreseen from the right plot of Figure 3.1 showing an optimal map from μ to $\overline{\mu}$ where no Monge map can be defined in the opposite direction. For instance, $\boldsymbol{y}_2$ with mass $\overline{h}_2$ can not be assigned to any location $\{\boldsymbol{x}_i\}_{i \in [\![7]\!]}$ while prohibiting mass splitting, as for all $i \in [\![7]\!], h_i < \overline{h}_2$.

Since Monge maps may not exist nor be unique, such a transportation paradigm remained an open mathematical problem for over two centuries. Seminal results on discrete or continuous Monge matchings were discovered by Appell (1887) but only for specific cases, until the celebrated Brenier's theorem (Brenier, 1991) which revealed an unexpected link between Optimal Transport and Fluid Mechanics. This theorem provided the existence and the unicity of Monge maps between distributions, with finite second moments and admitting density (in source), while using the squared euclidean distance $c(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ as ground cost.

Kantorovich primal problem. Still, the *transfer* notion of Monge imposes a deterministic nature to the transportation problem which might be detrimental *e.g.* to perform more general allocation problems. Kantorovich (1942) then proposed to relax Monge's transportation problem by allowing the mass at any source point to be potentially split across several target

points. In a figurative way, the transportation problem involving *finitely* subdivisible objects initially, now concerns *infinitely* splittable ones. Formally, the matching problem switches from deterministic to probabilistic, as it now seeks for a mapping modeling a joint distribution, between the source and target distributions, that defines how the mass is transported.
The mapping is then encoded as a coupling matrix $\boldsymbol{T} \in \mathbb{R}_+^{n \times \overline{n}}$, whose entry T_{ij} depicts the amount of mass transported from $\boldsymbol{x}_i$ to $\boldsymbol{y}_j$. To satisfy the mass conservation paradigm, an *admissible coupling* must belong to the set

$$\begin{aligned} \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}}) &:= \{\boldsymbol{T} \in \mathbb{R}_+^{n \times \overline{n}} | \boldsymbol{T}\mathbf{1}_{\overline{n}} = \boldsymbol{h}, \boldsymbol{T}^\top \mathbf{1}_n = \overline{\boldsymbol{h}}\} \\ &= \{\boldsymbol{T} \in \mathbb{R}_+^{n \times \overline{n}} | \forall (i,j) \in [\![n]\!] \times [\![\overline{n}]\!], \sum_k T_{ik} = h_i, \sum_k T_{kj} = \overline{h}_j\} \end{aligned} \tag{3.3}$$

Interestingly, the set of admissible coupling matrices $\mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$ defines a compact and convex set, as a convex polytope resulting from $n + \overline{n}$ mass equality constraints.[2] Hence the OT problem in its relaxed form reads as

$$\mathcal{T}_c(\mu, \overline{\mu}) = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{C}, \boldsymbol{T} \rangle_F \tag{3.4}$$

where $\boldsymbol{C} = (c(\boldsymbol{x}_i, \boldsymbol{y}_j))_{i,j \in [\![n]\!] \times [\![\overline{n}]\!]}$. So OT now relates to a convex linear problem constrained to the convex set $\mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$. The resulting cost $\mathcal{T}_c(\mu, \overline{\mu})$ corresponds to the minimal total cost of moving μ towards $\overline{\mu}$ by potentially splitting their masses and transporting forward the pieces according to the optimal coupling matrix $\boldsymbol{T}^*$. This linear OT formulation is also ubiquitous in everyday resource allocation problems, as illustrated by the famous examples of supplying raw materials from warehouses to factories (Peyré & Cuturi, 2019, Remark 2.10)(Kantorovich, 1942; Hitchcock, 1941), or distributing bread from bakeries to cafes in Manhattan (Villani, 2021).

Moreover, one can observe that the Birkhoff polytope $\mathcal{U}(\frac{1}{n}\mathbf{1}_n, \frac{1}{n}\mathbf{1}_n)$ (Birkhoff, 1946) contains the set of permutation matrices $\mathrm{Perm}(n)$ scaled by a factor $1/n$,

$$\frac{1}{n}\mathrm{Perm}(n) := \{\boldsymbol{P}_\sigma \in \mathbb{R}^{n \times n} | \forall \sigma \in \mathfrak{S}_n, \forall (i,j) \in [\![n]\!]^2, (\boldsymbol{P}_\sigma)_{ij} = \frac{1}{n}\mathbb{1}_{\{j = \sigma(i)\}}\} \tag{3.5}$$

which is in bijection with $\mathfrak{S}_n$. Thus the OT cost resulting from Kantorovich's formulation (3.4) necessarily bounds by below the one from Monge's problem, or equivalently the optimal assignment problem in this scenario. Actually, these two problems result in the same optimal cost in such settings, *i.e* Kantorovich's problem always admits a permutation matrix (up to the scaling $\frac{1}{n}$) as optimal coupling (Peyré & Cuturi, 2019, Proposition 2.1).

Kantorovich dual problem. A last observation on the primal Kantorovich's problem (3.4) lies in its relation to its dual problem. Indeed, as the primal problem is convex, strong duality for Linear Programs holds (Bertsimas & Tsitsiklis, 1997, Theorem 4.4). So we have the equality relation stated in the following Proposition 1:

Proposition 1 (Proposition 2.4, Peyré & Cuturi (2019)) *The primal problem 3.4 admits the following dual formulation*

$$\mathcal{T}_c(\mu, \overline{\mu}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}(\boldsymbol{C})} \langle \boldsymbol{\alpha}, \boldsymbol{h} \rangle + \langle \boldsymbol{\beta}, \overline{\boldsymbol{h}} \rangle \tag{3.6}$$

[2] as the convex hull of a finite set of matrices (Brualdi, 2006).

where the set of admissible "Kantorovich's dual potentials" is

$$\mathcal{R}(\boldsymbol{C}) := \{(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbb{R}^n \times \mathbb{R}^{\overline{n}} | \forall (i,j) \in [\![n]\!] \times [\![\overline{n}]\!], \alpha_i + \beta_j \leq C_{ij}\} \tag{3.7}$$

This dual formulation of the optimal transport cost induces an elegant switch in interpretation, from the aforementioned resource allocation problems, to profit maximization problems if the allocations were to be handled by a third-party deliverer (see Peyré & Cuturi (2019), Remark 2.21). Moreover in a Machine Learning perspective, this dual formulation could be used to derive subgradients *w.r.t* probability vectors $\boldsymbol{h}$ or $\overline{\boldsymbol{h}}$ in order to learn reweighed representations, as the primal problem is not differentiable w.r.t these variables.

3.1.2 Wasserstein distances: definitions and properties

p-Wasserstein distances. The most notable scenario occurring in OT applications is when the respective support of compared measures belongs to a Polish space[3], denoted $(\mathcal{X}, d)$, like an Euclidean space. In such cases, the ground cost c of the OT problem is naturally defined from the distance d depicting the space topology. These considerations lead to the definition of the so-called *p-Wasserstein distances* reading, for any $p \in [1, \infty]$ as a Kantorovich problem (3.4) whose ground cost matrix $\boldsymbol{C} = \boldsymbol{D}^p = (d^p(\boldsymbol{x}_i, \boldsymbol{y}_j))_{(i,j) \in [\![n]\!] \times [\![m]\!]}$:

$$\mathrm{W}_p^p(\mu, \nu) := \mathcal{T}_{d^p}(\mu, \nu) = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{D}^p, \boldsymbol{T} \rangle_F \tag{3.8}$$

This function satisfies all the axioms of a distance on the space of (discrete) probability distributions with bounded p-moments as detailed in the next theorem, which can naturally be adapted to continuous or semi-continuous settings:

Theorem 1 (Theorem 7.3, Villani (2021)) *Let $(\mathcal{X}, d)$ be a Polish space, $p \in [1, +\infty]$ and any discrete probability distributions μ and ν on $\mathcal{X}$ with finite p-moments. Then $0 \leq \mathrm{W}_p(\mu, \nu) < +\infty$ and*

i) $\mathrm{W}_p(\mu, \nu) = \mathrm{W}_p(\mu, \nu)$ *(symmetry)*

ii) $\mathrm{W}_p(\mu, \nu) = 0 \Leftrightarrow \mu = \nu$ *(separation)*

iii) $\mathrm{W}_p(\mu, \nu) \leq \mathrm{W}_p(\mu, \xi) + \mathrm{W}_p(\xi, \nu)$ *(triangle inequality), with ξ a discrete probability distribution on $\mathcal{X}$ with finite p-moments.*

Proof sketch of Theorem 1. We refer the reader to Villani (2021), Theorem 7.3, for a detailed proof of Theorem 1. We emphasize that the triangle inequality partly results from the Gluing Lemma (Villani, 2021, Lemma 7.6) who provides an admissible and sub-optimal coupling from μ to ν, composed of optimal couplings between both other matching problems. □

Insights on the Wasserstein distance properties. An appealing property of Wasserstein distances lie in their abilities to compare probability measures while taking into account the geometry of the underlying space, governed by the ground cost c. This contrasts with the most commonly used f-divergences (Csiszár, 1967), like the Total Variation (TV) norm or the Kullback-Leibler (KL) divergence, which compare these measures in a point-wise manner and

[3] A Polish space is a separable completely metrizable topological space. Namely, it contains a countable, dense subset *i.e* there exists a countable sequence of elements of the space such that every nonempty open subset of the space contains at least one element of the sequence. Moreover, there exists at least one metric d on $\mathcal{X}$ such that (X, d) is a complete metric space and d induces the space topology.

fail to capture the geometric nature of the problem.
As a result, the latter divergences do not metrize the weak convergence (or convergence in law) and are unstable *w.r.t* deformations of the distributions' supports. We recall that a sequence $(\mu_n)_{n\in\mathbb{N}}$ of probability measures on $\mathcal{X}$ a Polish space, is said to *converge weakly* to μ in $\mathcal{X}$ if for all continuous and bounded functions $f : \mathcal{X} \to \mathbb{R}$ we have

$$\int_{\mathcal{X}} f d\mu_n \quad \xrightarrow[n\to+\infty]{} \quad \int_{\mathcal{X}} f d\mu \tag{3.9}$$

However, the Wasserstein distance *metrizes the weak convergence* of probability measures, meaning that (μ_n) converges weakly to μ if and only if $\mathrm{W}_p(\mu_n, \mu) \to 0$ (Villani, 2021, Theorem 6.9).
Note that other divergences metrize the weak convergence of probability measures, *e.g* the Maximum Mean Discrepancies (MMD) resulting from the integration of kernels (Gretton et al., 2006), but they do not necessarily induce the same geometry and statistical behavior on the space of probability measures. Notably, MMD attaches "flat" geometry to this space whereas W_p provides a geodesic one, which is important in order to derive dynamic formulations and gradient flows (Ambrosio et al., 2005; Chizat & Bach, 2018).

However, this refined structure modeling generally comes with higher computational cost for the distance estimation (see Section 3.1.3), and higher *sample complexity* or approximation error when sampling from a distribution. To formalize this latter point, let us consider any probability distribution μ on $\mathbb{R}^d$ and an empirical distribution $\mu_n = \frac{1}{n}\sum_{i\in[\![n]\!]} \delta_{\boldsymbol{x}_i}$ composed of the samples $\boldsymbol{x}_i \sim_{iid} \mu$, that is expected to be a good proxy for μ. In this case, the sample complexity of the estimation of the Wasserstein distance is exponential in the ambient space:

$$\mathbb{E}\{\mathrm{W}_p(\mu_n, \mu)\} = O(n^{-1/d}) \tag{3.10}$$

So Wasserstein distances suffer from the curse of dimensionality (Dudley, 1969; Weed & Bach, 2019). Weed & Bach (2019) also showed a potential refinement of this result in $O(n^{-1/p})$ where p is the intrinsic dimension of the distribution. Still, these results highlight a major bottleneck for the use of OT in high dimensions. We refer the reader interested in more details on statistical aspects of OT to the thesis of Weed (2019).

Special cases. In some specific scenarios, the Wasserstein distances admit closed-form solutions. One first relates to OT between empirical distributions whose supports lie on the real line. Interestingly, the p-Wasserstein distance between both measures is given by the l^p norm between two vectors formed by sorting the support of each measure, which can be computed in $O(n\log(n))$ operations using *e.g* a merge sort (Knuth, 1973). This relation is actually a simple consequence of the generic result detailed in Peyré & Cuturi (2019), Remark 2.30. Namely, the OT plan between two measures μ and $\overline{\mu}$ supported on $\mathbb{R}$, respects the ordering of the elements given by the monotone rearrangement of both measures, given by the distance induced by the $l^p([0,1])$ norm between their respective quantile functions. We refer the reader to Santambrogio (2015), Chapter 2, for a detailed survey on OT over the real line.

These easily tractable closed-form solutions motivated the definition of sliced OT distances such as the Sliced Radon Wasserstein (Bonneel et al., 2015) that computes the expected value of the Wasserstein distance over 1D linear projections integrated on the unit sphere $\mathbb{S}^{d-1}$ (often approximated from finite random directions). This sliced distance has been used as a fitting term for generative networks (Kolouri et al., 2018; Deshpande et al., 2018; Liutkus et al., 2019) or *e.g* as a kernel for persistence diagrams (Carriere et al., 2017). The Sliced Radon Wasserstein (SW) distances are intimately linked to the Radon transform (Radon,

1986), whose generalization to non-linear 1D projections (Beylkin, 1984) leads to Generalized SW distances (Kolouri et al., 2019).

A second scenario where OT admits closed-form solutions is when probability measures respectively depicts Gaussian distributions of $\mathbb{R}^d$, such as $\mu = \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ and $\overline{\mu} = \mathcal{N}(\overline{\boldsymbol{m}}, \overline{\boldsymbol{\Sigma}})$, and c is the Euclidean norm. The 2-Wasserstein distance between such distributions can be expressed using the Bures-Wasserstein metric between positive definite matrices, which can be computed from simple matrix operations (Bures, 1969). We refer the reader to Peyré & Cuturi (2019), Remark 2.31, for more details on the Bures-Wasserstein metric, and an explicit derivation of the OT map in this setting. Interestingly, when only empirical estimates of the means and covariances are available for both distributions, the formula applied in these estimates approximates $\mathrm{W}_2^2(\mu, \overline{\mu})$ with a sample complexity of $O(n^{-1/2})$ as proven in Flamary et al. (2019). Note that an analog close form solution to the 2-Wasserstein problem between elliptical distributions which generalizes Gaussian ones can be found in Muzellec & Cuturi (2018).

3.1.3 Solving for linear OT

In the following, we address the estimation of an optimal solution to the primal (equation (3.4)), and/or dual (equation (3.6), Proposition 1), Kantorovich's transport problems between discrete probability distributions $\mu = \sum_{i \in [\![n]\!]} h_i \delta_{\boldsymbol{x}_i}$ and $\overline{\mu} = \sum_{j \in [\![\overline{n}]\!]} \overline{h}_j \delta_{\boldsymbol{y}_j}$. The problem can be solved in many ways, so we essentially detail the approaches used in the remaining chapters of this manuscript. Namely, the Network Flow algorithm seeking for exact solutions, the Sinkhorn algorithm solving for an entropically regularized OT problem and the Proximal Point algorithm acting as its exact counterpart.

Minimum-cost Network Flow algorithm. The Minimum-cost Network Flow problem is a common optimization and decision process, to find the cheapest possible way of sending a certain amount of mass through a flow network. The latter is simply a directed graph, whose nodes correspond to the source and target locations, and whose edges are defined as the costs of moving a certain amount of mass along a given pair of source and target nodes. This problem exactly translates to OT and can be solved efficiently using the Network Simplex algorithm.

Before briefly detailing the Network Flow algorithm, we emphasize that its motivation is to couple updates of the primal and dual respective solution estimates, to reach optimality on both problems simultaneously. To this end, we highlight certain relations between both problems being *a priori* independent.

First, from the complementary slackness of the Karush–Kuhn–Tucker (KKT) conditions, one can deduce that any optimal solutions $\boldsymbol{T}^*$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ to their respective problem must satisfy for all $(i, j) \in [\![n]\!] \times [\![\overline{n}]\!]$,

$$T^*_{ij}(C_{ij} - \alpha^*_i - \beta^*_j) = 0 \tag{3.11}$$

Therefore, the dual feasibility (equation (3.7), Proposition 1) implies that if $T^*_{ij} > 0$ then $\alpha^*_i + \beta^*_j = C_{ij}$, otherwise if $T^*_{ij} = 0$ necessarily $\alpha^*_i + \beta^*_j < C_{ij}$ (Peyré & Cuturi, 2019, Proposition 3.2). Then, we introduce the idea that *variables* $\boldsymbol{T}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, respectively related to both problems, are complementary w.r.t $\boldsymbol{C}$ if $\forall (i, j) \in [\![n]\!] \times [\![\overline{n}]\!]$ such that $T_{ij} > 0$ then $C_{ij} = \alpha_i + \beta_j$ (Peyré & Cuturi, 2019, Definition 3.1). This way, if a pair of variable $\boldsymbol{T}$ and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are *feasible and complementary* then they are optimal (Peyré & Cuturi, 2019, Proposition 3.3).

The Network Flow algorithm thus relies on two simple principles: to each *feasible* primal solution $\boldsymbol{T}$ one can associate a complementary pair $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. If that pair is feasible, then optimality is reached. Otherwise, one can consider a modification of $\boldsymbol{T}$ that remains feasible

and whose complementary pair $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is modified so that it becomes closer to feasibility. The first principle imposes to provide an *extremal* primal solutions $\boldsymbol{T}$ as starting point, which can be carried out using the North-West Corner Rule (Peyré & Cuturi, 2019, 3.4.2). Then the association of a complementary pair of dual variables and the potential next update of $\boldsymbol{T}$ are achieved thanks to heuristics requiring cycle detection and tree traversals within a bipartite graph modeling the mass flow induced by $\boldsymbol{T}$ (Bertsekas & Eckstein, 1988). We refer the reader to Peyré & Cuturi (2019), Section 3.5, for further details on each algorithm step.

Orlin (1997) provided the first algorithmic solution with proven polynomial worst-case complexity of $O\left((n+m)^2 nm \log((n+m)\|\boldsymbol{C}\|_\infty)\right)$, which was later improved by Tarjan (1997) to $O\left((n+m)nm \log(n+m) \min\{\log((n+m)\|\boldsymbol{C}\|_\infty), nm \log(n+m)\}\right)$ thanks to the use of dynamic trees. Numerous variants of this algorithm exist with various update rules and modeling of the spanning tree structure (Kovács, 2015). In the following, implementations of various OT methods rely on the Network Flow solver from POT (Flamary et al., 2021), based on an efficient C++ implementation operating on CPU of the network simplex algorithm with a block search pivoting strategy (Bonneel et al., 2011; Kelly & ONeill, 1991). Recent works (Ploskas & Samaras, 2014) focused on accelerating Network Flow algorithms on GPU, which require advanced algorithmic considerations to leverage the parallelization potential of these units. So an integration of such solver within common GPU accelerated Machine Learning Library, *e.g* Pytorch (Paszke et al., 2019), is still to come and can be (hopefully temporarily) a bottleneck for OT based end-to-end models (see Chapter 4)

Entropic regularization of OT. One practice to approximate solutions of (3.4), made popular by Cuturi (2013), consists in regularizing the OT problem thanks to the discrete entropy of a coupling:

$$\mathrm{H}(\boldsymbol{T}) = -\sum_{ij} T_{ij} \left(\log(T_{ij}) - 1\right) = -\langle \boldsymbol{T}, \log \boldsymbol{T} - \mathbf{1}_n \mathbf{1}_m^\top \rangle_F \tag{3.12}$$

for any $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$. H is 1-strongly concave on the set of admissible couplings, so the regularized OT problem reading, for any $\varepsilon > 0$, as

$$\mathcal{T}_c^\varepsilon(\mu, \overline{\mu}) = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{T}, \boldsymbol{C} \rangle_F - \varepsilon \, \mathrm{H}(\boldsymbol{T}) \tag{3.13}$$

becomes ε-strongly convex, so admits a unique optimal solution. As proven in Peyré & Cuturi (2019), Proposition 4.1, one can recover the unregularized OT cost (3.4) as $\varepsilon \to 0$, *i.e* $\lim_{\varepsilon \to 0} \mathcal{T}_c^\varepsilon(\mu, \overline{\mu}) = \mathcal{T}_c(\mu, \overline{\mu})$. In contrast, for large regularization coefficient, the solution to (3.13) denoted $\boldsymbol{T}_\varepsilon$ converges to the coupling with maximal entropy between both marginals, being the joint probability between independent random variables respectively distributed as $\boldsymbol{h}$ and $\overline{\boldsymbol{h}}$, *i.e* $\lim_{\varepsilon \to \infty} \boldsymbol{T}_c^\varepsilon(\mu, \overline{\mu}) = \mu \otimes \overline{\mu} = \boldsymbol{h}\overline{\boldsymbol{h}}^\top$. This regularization leads to an OT problem less sensitive to small variations in the distributions, proportionally to the regularization strength, at the cost of blurring the OT matrix that cannot be sparse for $\varepsilon > 0$, which negatively impacts its interpretability. However, this behavior can improve the modeling of certain transport problems, for example when studying traffic patterns, as human decision making tends to produce diffuse paths for a given route (Wilson, 1969).

A simple analytic solution to (3.13) shown thanks to Lagrangian duality (Peyré & Cuturi, 2019, Proposition 4.3) takes the form $\boldsymbol{T}^* = \mathrm{diag}(\boldsymbol{u}) \boldsymbol{K} \mathrm{diag}(\boldsymbol{v})$ where $\boldsymbol{K} = e^{-\boldsymbol{C}/\varepsilon}$ (applied element-wise) and $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}_+^n \times \mathbb{R}_+^{\overline{n}}$. The marginal constraints on $\boldsymbol{T}^*$ exhibit a well-known matrix scaling problem parameterized by $\boldsymbol{u}$ and $\boldsymbol{v}$ (Nemirovski & Rothblum, 1999). The latter can be solved using the iterative Sinkhorn algorithm (Sinkhorn & Knopp, 1967), detailed in Algorithm 1.

Algorithm 1 Sinkhorn algorithm solver for regularized OT problem (3.13)

Inputs: $\boldsymbol{h}, \overline{\boldsymbol{h}}, \boldsymbol{K} = e^{-\boldsymbol{C}/\varepsilon}, \boldsymbol{v} = \mathbf{1}_{\overline{n}}$.
repeat
  $\boldsymbol{u} \leftarrow \boldsymbol{h} \oslash \boldsymbol{K}^\top \boldsymbol{v}$ (Update left scaling)
  $\boldsymbol{v} \leftarrow \overline{\boldsymbol{h}} \oslash \boldsymbol{K}\boldsymbol{u}$ (Update right scaling)
until convergence
Return: $\boldsymbol{T}^* = \mathrm{diag}(\boldsymbol{u})\boldsymbol{K}\mathrm{diag}(\boldsymbol{v})$.

Interestingly, the problem (3.13) is equivalent to searching for an OT matrix as close as possible to a kernel $\boldsymbol{K}$, in the sense of the Kullback-Leibler (KL) geometry:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} D_{\mathrm{KL}}(\boldsymbol{T}|\boldsymbol{K}) \tag{3.14}$$

where $\boldsymbol{K} = e^{-\boldsymbol{C}/\varepsilon}$ and $D_{\mathrm{KL}}(\boldsymbol{T}|\boldsymbol{K}) = \sum_{ij} T_{ij} \log \frac{T_{ij}}{K_{ij}} - T_{ij} + K_{ij}$. This reformulation allows the use of iterative Bregman projections (Bregman, 1967) to solve this problem, and to analyze algorithms' convergence (Benamou et al., 2015) as the resulting updates are actually equivalent to those of Algorithm 1 (Peyré & Cuturi, 2019, Remark 4.8).

As the entropically regularized problem (3.13) ($\varepsilon > 0$) explicitly does not solve for the unregularized OT problem ($\varepsilon = 0$), one can approximate solution of the latter using a Proximal Point algorithm (PPA) with the KL metric while benefiting from the computational simplicity of Algorithm 1 (Peyré, 2015; Schmitzer, 2019). The PPA comes down to solve at *each iteration* t, a quite similar sub-problem than (3.13) using Sinkorn's Algorithm 1, up to a change of kernel $\boldsymbol{K}^{(t-1)} = e^{-\boldsymbol{C}/\varepsilon} \odot \boldsymbol{T}^{(t-1)}$. The update of the OT matrix then reads $\boldsymbol{T}^{(t)} = \mathrm{diag}(\boldsymbol{u}^{(t)})\boldsymbol{K}^{(t-1)}\mathrm{diag}(\boldsymbol{v}^{(t)})$. Moreover developing recursively this update rule from the first to t^{th} iterations exhibits the kernel $e^{-\boldsymbol{C}/(\varepsilon/t)}$, so PPA can be seen as a Sinkhorn's algorithm with a decaying regularization strength ε/t.

From an application perspective, the entropic regularization of OT, either explicit or implicit, has several computational advantages: the solvers only depend on matrix-vector products, hence are particularly suited to modern GPU programming, in contrast to the exact Network Flow solver. Moreover their theoretical computational complexity is lower. However, these algorithms can suffer from stability issues when $\varepsilon \to 0$ (either fixed or adapted) as the kernel $\boldsymbol{K}$ vanishes rapidly leading to divisions by 0 during the algorithms' iterations, due to the limited numerical precision. There are several ways to get around this problem, but often at the cost of a slight loss in computation speed (Schmitzer, 2019; Feydy et al., 2019).

Moreover, this entropic regularization induces biases as $\mathcal{T}_c^\varepsilon(\mu, \mu) \neq 0$ (idem for $\overline{\mu}$) which can be problematic for learning from $\mathcal{T}_c^\varepsilon(\mu, \overline{\mu})$. Genevay et al. (2018) introduced the Sinkhorn divergences resulting from correcting these biases, which enjoy appealing properties. They define symmetric positive definite smooth functions on the space of probability measures, which are convex in its inputs, and metrize the weak convergence (3.9) (Feydy et al., 2019). They also act as an interpolation between Wasserstein distance and MMD (Gretton et al., 2006), resulting in a lower sample complexity than Wasserstein's one, *i.e* $O(\varepsilon^{-d/2} n^{-1/2})$ compared to $O(n^{-1/d})$ (Genevay et al., 2019).

Other OT solvers. Other regularizations in the vein of the entropic one have been explored for OT, notably smooth strictly convex ones (Dessein et al., 2018), so the resulting optimization problem becomes strictly convex. In general, one also needs to project onto the positive orthant so a more general Dikstra algorithm has to be used (Bauschke et al., 2011, Chapter 29). Regularization has also been used to encode additional information as

group-structure observed in data. To this end, Courty et al. (2014); Flamary et al. (2016a) proposed a concave group-lasso like regularization to the entropic OT problem which can be tackled using a Majoration Minimization algorithm (Hunter & Lange, 2004).
One generic way to solve for regularized OT, including non-convex regularizations, is Conditional Gradient (CG) descent (Frank & Wolfe, 1956). This approach relies on optimizing at each iteration a linearization of the objective function using non-regularized OT solvers (*e.g* the Network Flow algorithm.) CG is highly flexible while being ensured to converge even for non-convex objectives (Lacoste-Julien, 2016), but comes at the price of solving non-regularized OT problems at each iteration which can be computationally intensive. A trade-off between CG and Sinkhorn's algorithm, leveraging entropy as a complementary regularization, can be achieved using the Generalized Conditional Gradient algorithm (Bredies et al., 2009; Rakotomamonjy et al., 2015).

A last notable line of works aims at coupling Stochastic optimization with known or novel OT solvers to estimate OT plans between semi-discrete distributions, from sub-samples of these distributions. These approaches are mostly dedicated to solve OT for large-scale problems. As our studies reported in the remaining chapters of this manuscript focus on discrete measures, we refer the reader seeking for details on these stochastic approaches to *e.g.* Peyré & Cuturi (2019), Chapter 5; Genevay et al. (2016); Staib et al. (2017).

3.1.4 OT for unsupervised representation learning

Many ML applications focus on learning the support or the probability masses of a measure involved in a Wasserstein problem. In this scenario, we model distributions, *e.g* $\mu = \sum_{i\in[\![n]\!]} h_i \delta_{\boldsymbol{x}_i}$ and $\overline{\mu} = \sum_{j\in[\![\overline{n}]\!]} \overline{h}_j \delta_{\overline{\boldsymbol{x}}_j}$, as tuples $(\boldsymbol{X}, \boldsymbol{h})$ and $(\overline{\boldsymbol{X}}, \overline{\boldsymbol{h}})$, where first $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]^\top \in \mathbb{R}^{n\times d}$ and $\overline{\boldsymbol{X}} = [\overline{\boldsymbol{x}}_1, ..., \overline{\boldsymbol{x}}_{\overline{n}}]^\top \in \mathbb{R}^{\overline{n}\times d}$ gather locations of their respective measure in matrix forms. While $\boldsymbol{h} \in \Sigma_n$ and $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$ are the corresponding probability vectors.

Minimizing for instance the transportation cost (3.8) *w.r.t* $\overline{\boldsymbol{X}}$, for a fixed OT matrix, requires to differentiate through the pairwise cost matrix $\boldsymbol{D}^p(\boldsymbol{X}, \overline{\boldsymbol{X}})$. A classic choice is to consider $p = 2$. In this way, the cost matrix can be computed with simple vector-matrix operations, *i.e* $\boldsymbol{D}^2(\boldsymbol{X}, \overline{\boldsymbol{X}}) = \boldsymbol{X}^2 \mathbf{1}_d \mathbf{1}_{\overline{n}}^\top + \mathbf{1}_n \mathbf{1}_d^\top \overline{\boldsymbol{X}}^2 - 2\boldsymbol{X}\overline{\boldsymbol{X}}^\top$ (power operations are taken element-wise). And the minimization of the objective function *w.r.t* $\overline{\boldsymbol{X}}$, *for a fixed transport plan*, can be expressed as a convex Quadratic Program with Hessian equals to $\mathrm{diag}(\overline{\boldsymbol{h}}) \otimes_K \boldsymbol{I}_d$.[4]

Since manipulations of the 2-Wasserstein cost function, like the one described above, will be ubiquitous in this manuscript, we introduce the following notations:

$$\mathrm{W}_2^2(\mu, \overline{\mu}) := \mathrm{W}_2^2(\boldsymbol{X}, \boldsymbol{h}, \overline{\boldsymbol{X}}, \overline{\boldsymbol{h}}) := \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{D}^2(\boldsymbol{X}, \overline{\boldsymbol{X}}), \boldsymbol{T}\rangle := \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \mathcal{E}^W(\boldsymbol{X}, \overline{\boldsymbol{X}}, \boldsymbol{T}) \quad (3.15)$$

With $\mathcal{E}^W$ the 2-Wasserstein objective function for a given admissible transport plan $\boldsymbol{T}$.

Wasserstein barycenters. A consequence of the geometry induced by the Wasserstein distances is the definition of *Wasserstein barycenters*, that relates to specific *weighted mean* of a set of probability distributions (McCann, 1997; Agueh & Carlier, 2011). The typical Euclidean barycenter can be generalized to any metric space $(\mathcal{X}, d)$, then referred as Fréchet (or Karcher) mean (Fréchet, 1948; Karcher, 2014). In the space of probability distribution endowed with the Wasserstein geometry, a Fréchet mean $\overline{\mu}$ of $I \in \mathbb{N}$ measures $\{\mu_i\}_{i\in[\![I]\!]}$ results from the following optimization problem:

$$\min_{\overline{\mu}} \sum_{i\in[\![I]\!]} \lambda_i \, \mathrm{W}_p^p(\mu_i, \overline{\mu}) \quad (3.16)$$

[4] As for any $(i, j) \in [\![\overline{n}]\!]^2$ and $(e, e') \in [\![d]\!]^2$, $\frac{\partial^2(\cdot)}{\partial \overline{x}_{ie} \partial \overline{x}_{je'}} = 1_{(i,e)=(j,e')} \overline{h}_i$.

Where $\overline{\mu}$ is the discrete measure arbitrarily formed of $\overline{n}$ locations, and $\{\lambda_i\}$ are the barycentric coordinates assigned to each input measure, such that $\forall i \in [\![I]\!],\ \lambda_i > 0,\ \sum_i \lambda_i = 1$.
We detail now this problem formulation for W_2^2 according to our operative notations in equation (3.15). First, we model each input measure μ_i as a tuple $(\boldsymbol{X}_i, \boldsymbol{h}_i) \in \mathbb{R}^{n_i \times d} \times \Sigma_{n_i}$. Then (3.16) seeks for a measure $\overline{\mu}$, composed of $\overline{n}$ points, encoded as $(\overline{\boldsymbol{X}}, \overline{\boldsymbol{h}}) \in \mathbb{R}^{\overline{n} \times d} \times \Sigma_{\overline{n}}$ such that

$$\begin{aligned}(\overline{\boldsymbol{X}}, \overline{\boldsymbol{h}}) &= \underset{\overline{\boldsymbol{X}}, \overline{\boldsymbol{h}}}{\arg\min} \sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{W}_2^2(\boldsymbol{X}_i, \boldsymbol{h}_i, \overline{\boldsymbol{X}}, \overline{\boldsymbol{h}}) \\ &= \underset{\overline{\boldsymbol{X}}, \overline{\boldsymbol{h}}}{\arg\min} \sum_{i \in [\![I]\!]} \lambda_i \min_{\boldsymbol{T}_i \in \mathcal{U}(\boldsymbol{h}_i, \overline{\boldsymbol{h}})} \mathcal{E}^W(\boldsymbol{X}_i, \overline{\boldsymbol{X}}, \boldsymbol{T}_i)\end{aligned} \tag{3.17}$$

Interestingly if $\overline{\boldsymbol{X}}$ is *fixed*, for instance when distribution learning is performed on a grid, the minimization problem *w.r.t* $\overline{\boldsymbol{h}}$ is a Linear Program (Agueh & Carlier, 2011). Elegant approaches exist to approximate this problem, such as the Bregman Projection algorithm (Benamou et al., 2015), or convolution based algorithm when operating on regular grids (Solomon et al., 2015). Even though there exist special easier cases (Peyré & Cuturi, 2019, Section 9.2), finding a solution to the generic problem is complex. Cuturi & Doucet (2014) proposed to solve (3.17) using a Block Coordinate Descent (BCD) algorithm which iterates, until convergence, over the following steps:

i) Estimate OT matrices $\{\boldsymbol{T}_i\}$, jointly with its corresponding dual potentials $\{(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)\}$ (Proposition 1), for the I independent OT problems between $(\boldsymbol{X}_i, \boldsymbol{h}_i)$ and fixed $(\overline{\boldsymbol{X}}, \overline{\boldsymbol{h}})$ following Section 3.1.3.

ii) Update $\overline{\boldsymbol{X}}$ at fixed $\boldsymbol{T}_i$ using the following closed-form (Cuturi & Doucet, 2014, Equation 8):

$$\overline{\boldsymbol{X}} = \mathrm{diag}(\mathbf{1}_{\overline{n}} \oslash \overline{\boldsymbol{h}}) \left(\sum_{i \in [\![I]\!]} \lambda_i \boldsymbol{T}_i^\top \boldsymbol{X}_i \right) \tag{3.18}$$

iii) Update $\overline{\boldsymbol{h}}$ according to Projected subgradient descent using KL geometry (Beck & Teboulle, 2003), and proposition 1, as detailed in Cuturi & Doucet (2014), Algorithm 1. This update rule prevents sparsity within $\overline{\boldsymbol{h}}$ so equation (3.18) is well-defined. Another possible way is to perform an Euclidean projection leading to sparse solution $\overline{\boldsymbol{h}}$, hence reducing iteratively $\overline{\boldsymbol{X}}$ to the activated components such as $\overline{\boldsymbol{h}}_i > 0$.

If the set of input discrete measures $\{(\boldsymbol{X}_i, \boldsymbol{h}_i)\}_i$ actually relates to a single measure supported on a discrete subset of $\mathbb{R}^d$, learning a barycenter with at most k points is equivalent to the k-means problem (Canas & Rosasco, 2012). Indeed, the support of the barycenter $\overline{\boldsymbol{X}}$ is composed of the centroids output by k-means, and their masses $\overline{\boldsymbol{h}}$ correspond to the fraction of points assigned to each centroid. If input measures are rather samples drawn independently from a single measure, the barycenter problem with uniform weights $\boldsymbol{\lambda}$ can be seen as an empirical mean, that also asymptotically leads to the aforementioned equivalence. We refer the reader to Peyré & Cuturi (2019), Remarks 9.2-9.3, and references therein for further details on these matters. Remark that these considerations resonate with the method, dedicated to graphs, that we will develop later in Chapter 6.

Barycenter estimations can be used for *unsupervised* Representation Learning, for example by performing a k-means algorithm with respect to the Wasserstein geometry, considering k Wasserstein barycenters. Overall, a such approach is rather limited regarding the learned representations as a large amount of measures can be assigned to a single barycenter. On one hand, it can induce interesting denoising properties, on the other hand, representing samples with a unique centroid can result in a loss of intra-cluster variability potentially detrimental to downstream tasks.

OT based Dictionary Learning. One generic paradigm to factor a set of data points, while preserving an interpretable description of the induced discrete manifold, is *Dictionary Learning* (DL). DL aims at representing the input data as linear combination (or *embedding*) of basic dictionary elements, called *atoms*, as faithfully as possible. This problem is usually formulated as seeking for a dictionary of K atoms $\boldsymbol{D} \in \mathbb{R}^{d \times K}$ and embeddings $\boldsymbol{W} = (\boldsymbol{w}_i)_{i \in [\![I]\!]} \in \mathbb{R}^{K \times I}$ for the I data points composing $\boldsymbol{X} = (\boldsymbol{x}_i)_{i \in [\![I]\!]} \in \mathbb{R}^{d \times I}$, such that $\boldsymbol{X} \approx \boldsymbol{DW}$. Notable special cases of *linear* DL are the Principal Component Analysis (Hotelling, 1933, PCA) and Non-negative Matrix Factorization (Lee & Seung, 1999; 2000, NMF)

At first, most DL approaches using linear OT focused on *histograms*, *i.e* considered measure supports fixed on a regular grid. In this scenario, the data is expressed as $\boldsymbol{A} = (\boldsymbol{a}_i)_{i \in [\![n]\!]} \in \Sigma_d^n$, and DL seeks for embeddings $\boldsymbol{W} \in \Sigma_K^n$ and atoms $\boldsymbol{D} \in \Sigma_d^K$ modeled as histograms, such that $\boldsymbol{A} \approx \boldsymbol{DW}$. Hence divergences between histograms better suit this problem, *e.g* the KL (Lee & Seung, 2000) or the Itakura Saito (Févotte et al., 2009) divergences, or finally the Wasserstein distance (Sandler & Lindenbaum, 2011) which leads to the following optimization problem:

$$\min_{\{\boldsymbol{w}_i\} \subset \Sigma_K, \boldsymbol{D} \in \Sigma_d^K} \sum_{i \in [\![I]\!]} \mathrm{W}_c(\boldsymbol{a}_i, \boldsymbol{D}\boldsymbol{w}_i) \tag{3.19}$$

where the ground cost c encodes geometrical relationship between the histograms' components and can be fixed to any regular enough function, or learned (Cuturi & Avis, 2014). Problem (3.19) has also been studied with entropic OT (equation (3.13)) to obtain smoother estimations while speeding up the learning process (Rolet et al., 2016, and Section 3.1.3).

A more complex *non-linear* counter-part to (3.19), embedding data as weights within Wasserstein barycenters was proposed by Schmitz et al. (2018). Observing that a linear model $\boldsymbol{Dw}$ can be seen as an Euclidean barycenter weighted by $\boldsymbol{w} \in \Sigma_K$, the aforementioned extension to Wasserstein geometry leads to the problem:

$$\min_{\{\boldsymbol{w}_i\} \subset \Sigma_K, \boldsymbol{D} \in \Sigma_d^K} \sum_i \mathrm{W}_c(\boldsymbol{a}_i, B(\boldsymbol{w}_i, \boldsymbol{D})) \quad \text{with} \quad B(\boldsymbol{w}, \boldsymbol{D}) = \arg\min_{\boldsymbol{b} \in \Sigma_K} \sum_k w_k \, \mathrm{W}_c(\boldsymbol{b}, \boldsymbol{d}_k) \tag{3.20}$$

This extension to nonlinear embedding on the Wasserstein simplex suggests diverse extensions of non-supervised methods among which the Principal Component Analysis.

PCA seeks for directions that maximize the variance in the data and by definition, the variance relates to the Euclidean geometry. The extension to geodesic spaces (as the Wasserstein one) requires defining the notion of geodesic variance and projection as studied in Fletcher et al. (2004), leading to the Principal Geodesic Analysis (PGA). PGA is complex to achieve in practice as there is no analytical expression for the exponential or logarithmic maps allowing one to travel to and from the corresponding Wasserstein tangent space. Efficient solvers have only been proposed recently (Seguy & Cuturi, 2015; Cazelles et al., 2018; Courty et al., 2017a).

As mentioned these methods are dedicated to histograms hence can not operate on generic discrete measures, such as graphs seen as point clouds of their node embeddings (Chapter 2), whose number of points can differ within a graphs' dataset. Up to our knowledge, such DL have first been envisioned as specific instances of Fused Gromov-Wasserstein (Vayer et al., 2020) based DL that will be developed in Section 3.3 and Chapters 5-6.

3.2 Optimal Transport across incomparable spaces

Next we introduce another transport paradigm that extends the linear OT problem (Section 3.1) dedicated to a single ambient space, to OT across incomparable spaces. This allows the comparison of graphs, seen as discrete probability measures, via Gromov-Wasserstein distances (Section 3.2.1) whose properties are detailed in Section 3.2.3. We then present

solvers for estimating these distances (Section 3.2.4) and approaches for learning GW-based graph representations, such as GW barycenters (Section 3.2.5).

3.2.1 Problem statement

The linear OT problem assumes the existence of a ground metric space containing the supports of compared measures, which may be a limitation. For instance, when the supports are respectively in $\mathbb{R}^3$ and $\mathbb{R}^2$, a meaningful ground cost $c : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}$ is hard to define, and it can not be a metric. Hence the linear OT problem endowed with a such ground cost can no longer induce an OT *distance* between measures over distinct spaces. Besides, even if measures lie in the same space, the Wasserstein distance is sensitive to a large class of invariants such as isometries, which prevents its application to certain challenges like shape matching.

The Gromov-Wasserstein problem. The Gromov-Wasserstein (GW) problem first introduced in Mémoli (2007) overcomes the aforementioned limitations of linear OT. It was first devoted to the matching of *measurable metric spaces*, *i.e.* when the respective geometry of the compared spaces is induced by a metric (Mémoli, 2011). Then the GW problem has been extended to *gauged measure spaces* (Sturm, 2012) where the inner geometries do not necessarily satisfy the triangle inequality, to be adapted finally to the space of measurable networks, where inner (dis)similarities between points can be asymmetric (Chowdhury & Mémoli, 2019).[5]

The Gromov-Wasserstein problem is built upon a *quadratic Optimal Transport* problem, instead of a linear one, which informally aims at quantifying the (dis)similarity distortion when transporting points from one space to another. In the following we explicit the GW problem between *discrete* measurable spaces and refer the reader to Mémoli (2011); Sturm (2012); Chowdhury & Mémoli (2019) for a complete overview of the problem in any measurable spaces.

In its generic form, the GW problem considers any pair of measurable networks $(\mathcal{X}, c, \mu)$ and $(\overline{\mathcal{X}}, \overline{c}, \overline{\mu})$. Where $(\mathcal{X}, d_{\mathcal{X}})$ and $(\overline{\mathcal{X}}, d_{\overline{\mathcal{X}}})$ are polish spaces. Further endowed with continuous measurable functions $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $\overline{c} : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \mathbb{R}$, and probability measures μ and $\overline{\mu}$, respectively supported on $\mathcal{X}$ and $\overline{\mathcal{X}}$. Then in a discrete scenario, each measurable network can be encoded as a graph $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{h})$ composed of $n = \text{card}(\mathcal{X})$ nodes, where $\boldsymbol{C} = (c(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j \in [\![n]\!]} \in \mathbb{R}^{n \times n}$ is a matrix encoding the connectivity induced by c between nodes, and $\boldsymbol{h} \in \Sigma_n$ is a probability vector resulting from μ modeling the relative importance of the nodes within the graph. In the same way, we summarize $(\overline{\mathcal{X}}, \overline{c}, \overline{\mu})$ as the graph $\overline{\mathcal{G}} = (\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ composed of $\overline{n}$ nodes, with potentially $n \neq \overline{n}$.

Notice that the conventional modeling of a graph $\mathcal{G}$ considers a tuple (V, E), composed of a set of vertices or nodes $V = \{v_i\}_{i \in [\![n]\!]}$ and a set of edges E, *i.e.* connectivity relations between vertices. Fitting this modeling to the one of the GW problem, comes down to assuming the existence of an application $s : V \to \mathcal{X}$ mapping a vertex $v_i \in V$ from the graph to its structure representation $\boldsymbol{x}_i = s(v_i)$ in the structure space $(\mathcal{X}, c)$. So the graph is entirely described by a fully supported probability measure over $(\mathcal{X}, c)$, denoted $\mu = \sum_{i \in [\![n]\!]} h_i \delta_{\boldsymbol{x}_i}$.

The GW problem between two graphs seeks for a matching between their nodes, being as close as possible to an isometry. It does so by finding an optimal *soft* assignment matrix between the nodes, solving

$$\text{GW}_p^p(\mu, \overline{\mu}) = \text{GW}_p^p(\boldsymbol{C}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) := \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \sum_{ijkl} |C_{ij} - \overline{C}_{kl}|^p T_{ik} T_{jl} \tag{3.21}$$

[5] Note that in both these last settings, compared spaces are still assumed to be Polish spaces, except that the pairwise (dis)similarity between points (of the respective space) is decoupled from the theoretical topology of the space.

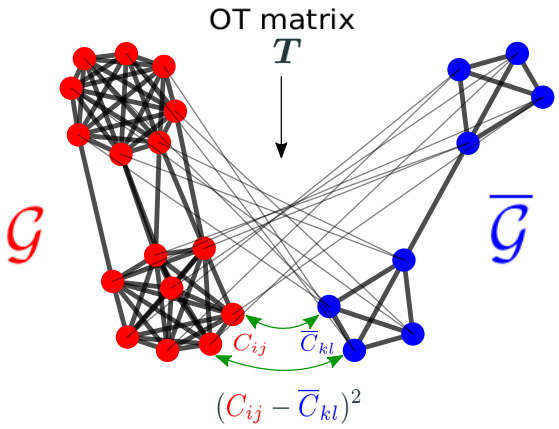

FIGURE 3.2: Illustration of the matching resulting from the GW_2 problem (3.21) between two graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ depicting their respective measure networks, where points or nodes respectively colored in red and blue are assigned to uniform probability masses $\boldsymbol{h}$ and $\overline{\boldsymbol{h}}$. The connections between nodes in each space are colored with bold dark lines, whereas the node assignment resulting from the OT matrix $\boldsymbol{T}$ are colored with finer dark lines.

An optimal coupling $\boldsymbol{T}^* \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$ (see Equation (3.3)) (if it exists) acts as a probabilistic matching of nodes which tends to associate pairs of nodes that have similar pairwise relations in $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ respectively, while preserving masses $\boldsymbol{h}$ and $\overline{\boldsymbol{h}}$ through its marginals constraints. An instance of such a GW matching can be found in Figure 3.2, where two graphs described by their respective binary adjacency matrices are compared. These graphs respectively contain 2 clusters or communities, which are then matched by the OT matrix $\boldsymbol{T}$ resulting in a minimal structure distortion of both graphs, hence a low GW cost.

Existence of solutions. In our discrete setting, a simple condition to avoid definition issues of the GW problem (3.21) is to assume respective inner connectivity matrices $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ to be bounded. As such the function $\boldsymbol{T} \to \sum_{ijkl} |C_{ij} - \overline{C}_{kl}|^p T_{ik} T_{jl}$ is lower-semi continuous, so the Weierstrass theorem (Santambrogio, 2015, remark 1.1) ensures the existence of an OT matrix, as $\mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$ is compact (*i.e* closed and bounded). We refer to Vayer (2020), Section 2.2, for a complete discussion on the existence of solutions to the GW problem (3.21).

On the regularity of OT maps. In the vein of Brenier's theorem (Brenier, 1991), stating that linear OT with euclidean ground cost admits OT maps given by the gradient of convex functions, one may ask whether there are special cases of GW problems where the OT plans also exhibit prescribed structures, or equivalently, regularities. Sturm first raised this question of regularity as such: *"Are there 'nice' spaces in which we are able to prove Brenier's results for GW?"* (Sturm, 2012, Challenge 3.6)

The first identified spaces of various dimensions, inducing regularity in their GW matching, are those described by *scalar products* (Vayer, 2020, Theorem 4.2.3). Under mild regularity conditions on the source probability distribution, author identified sufficient conditions to the existence of a deterministic map resulting from the composition of a convex function and a linear application acting as a global transformation "realigning" the probability measures in the same space (Vayer, 2020, Theorem 4.2.3).
Another more challenging scenario is to compare spaces equipped with *Euclidean norms* as inner costs. Seminal results with strong sufficient conditions were given in Sturm (2012), Theorem 9.21, and Vayer (2020), Proposition 4.2.4. Only very recently, Dumont et al. (2022) provided complete characterizations of OT maps for the GW problem between spaces described by scalar products (Dumont et al., 2022, Theorem 4) and Euclidean norms (Dumont et al., 2022, Theorem 5).

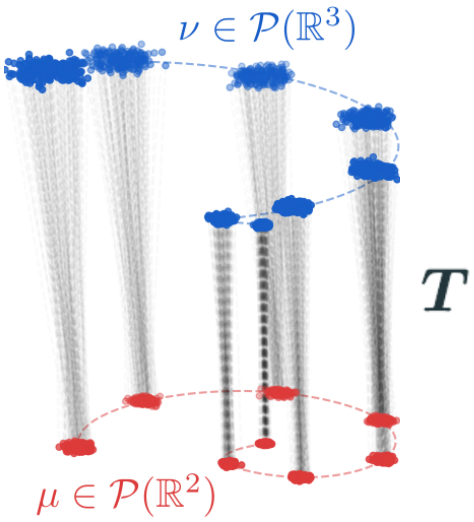

FIGURE 3.3: Illustration of the matching resulting from the GW_2 problem (3.21) between two discrete distributions $\mu \in \mathcal{P}(\mathbb{R}^3)$ and $\nu \in \mathcal{P}(\mathbb{R}^2)$ whose graphs are derived from the euclidean norm between locations of the respective distribution. The OT matrix $\boldsymbol{T}$ is drawn with dashed lines. Inspired from Vayer (2020).

These results are appealing for instance to compare point clouds, seen as discrete probability distributions, that respectively belong to $\mathbb{R}^p$ and $\mathbb{R}^q$ ($p \neq q$). The most natural topology to link samples within each space is its euclidean norm, forming then graphs whose node pairwise connectivities are euclidean distance matrices. To illustrate such a scenario, Vayer (2020) considered discrete probability distributions μ and ν, forming spirals whose modes are generated thanks to Gaussian mixtures, respectively in $\mathbb{R}^2$ and $\mathbb{R}^3$. The estimated GW matching of these two distributions is illustrated in Figure 3.3. We can see that a GW matching between both distributions highlights the instinctive minimum transport distortion of one measure over the other, since their respective modes are matched.
We emphasize that these theoretical results on the GW matching of euclidean distance matrices can be relevant for graph matching, since euclidean representations can be derived from a graph (potentially weighted) adjacency matrix, using for instance its *sif distance* matrix (Bavaud, 2010).

3.2.2 Gromov-Wasserstein barycenter

Since the GW problem induces a distance, one can also define a notion of barycenter, in the same vein as the Wasserstein barycenters (Section 3.1.4). This GW barycenter estimation problem was first investigated in Peyré et al. (2016) for discrete probability measures, defined over various spaces endowed with symmetric inner costs such as *undirected graphs*.

Learning the barycenter structure. Let us consider a dataset of graphs $\{\mathcal{G}_i = (\boldsymbol{C}_i, \boldsymbol{h}_i)\}_{i \in [\![I]\!]}$ seen as discrete measurable networks (Chowdhury & Mémoli, 2019), where for a given graph, $\boldsymbol{C}_i$ is an arbitrary node pairwise connectivity matrix, and $\boldsymbol{h}_i$ a probability vector modeling the relative significance of the graph nodes. When the weighting of the barycenter is fixed and given by $\boldsymbol{\lambda} \in \Sigma_I$, the GW barycenter problem in its first formulation seeks for a target structure $\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n} \times \overline{n}}$, composed on $\overline{n}$ nodes whose relative importance is fixed to $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$, solving

$$\min_{\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n} \times \overline{n}}} \sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) = \min_{\overline{\boldsymbol{C}}} \min_{\{\boldsymbol{T}_i \in \mathcal{U}(\boldsymbol{h}_i, \overline{\boldsymbol{h}})\}_i} \sum_{i \in [\![I]\!]} \lambda_i \mathcal{E}^{GW}(\boldsymbol{C}_i, \overline{\boldsymbol{C}}, \boldsymbol{T}_i) \tag{3.22}$$

This problem is non-convex in general, however the minimization sub-problem *w.r.t* $\overline{\boldsymbol{C}}$, with fixed OT plans, is convex while considering $p = 2$ in GW. Peyré et al. (2016) proposed to solve for problem (3.44) using a BCD procedure which alternates between two steps:

i) Solving I independent GW problems, with respective estimated OT matrices $\boldsymbol{T}_i$, from inputs $(\boldsymbol{C}_i, \boldsymbol{h}_i)$, respectively to the fixed and common target structure $\overline{\boldsymbol{C}}$.

ii) Updating $\overline{\boldsymbol{C}}$ for fixed $\boldsymbol{T}_i$ using a closed form formula for this minimization sub-problem.

Indeed, the gradient *w.r.t* to $\overline{\boldsymbol{C}}$ of the objective function in (3.44) reads as

$$\nabla_{\overline{\boldsymbol{C}}}(\cdot) = 2 \sum_{i \in [\![I]\!]} \lambda_i \{\overline{\boldsymbol{C}} \odot \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top - \boldsymbol{T}_i^\top \boldsymbol{C}_i \boldsymbol{T}_i\} \tag{3.23}$$

so the first optimality condition leads to the update

$$\overline{\boldsymbol{C}} \leftarrow \left(\sum_i \lambda_i \boldsymbol{T}_i^\top \boldsymbol{C}_i \boldsymbol{T}_i \right) \oslash \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \tag{3.24}$$

assuming that the entry-wise division by $\overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top$ is well-defined, *i.e* that $\overline{\boldsymbol{h}}$ does not have a null entry. Otherwise one can simply update the rows and columns of $\overline{\boldsymbol{C}}$ whose corresponding nodes have non-null probability mass. Note that any solver detailed in Section 3.2.4 can be used to solve the inherent OT problems. This GW barycenter problem has also been studied by leveraging the Riemannian geometry of the Gromov-Wasserstein space (Chowdhury & Needham, 2020). The estimation of GW barycenters has recently found many applications for instance for shape interpolation, image clustering (Peyré et al., 2016), graph clustering and partitioning (Xu, 2020), graphon estimation (Xu et al., 2021a), large-scale GW estimation (Xu et al., 2019a) and structured graph prediction (Brogat-Motte et al., 2022).

Fully learning the barycentric measure. This first formulation of the GW barycenter problem (3.44) enforces a *fixed* probability vector $\overline{\boldsymbol{h}}$ to the barycentric measure, despite being unknown in practice, which can be detrimental to certain applications, for instance those relying on partitioning of the input structures such as in Xu et al. (2019a). In that paper, authors use a barycenter of small order as intermediate matching, to estimate the GW matching between large input graphs without comparing them directly (divide and conquer strategy). In this case, the mass conservation constraint impose to get partitions of the input structures whose proportions suit entries of $\overline{\boldsymbol{h}}$.
As such, Xu et al. (2019a) proposed heuristics to estimate these proportions directly from degree distributions of the input measures. However the resulting estimated $\overline{\boldsymbol{h}}$ is most likely limited in many applications. For instance, it is easy to generate synthetic graphs from well-known Stochastic Block Models whose degrees would be uncorrelated to cluster proportions. These considerations advocate for also learning the barycenter probability vector $\overline{\boldsymbol{h}}$, solving for the following optimization problem:

$$\min_{\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n} \times \overline{n}}, \overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}} \quad \sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \tag{3.25}$$

However, any first-order optimization procedure would require to compute a subgradient of the GW distance *w.r.t* this parameter. To the best of our knowledge no theoretical results existed in the literature for finding such subgradients. To this end, we provided in our paper Vincent-Cuaz et al. (2021), fully reported in Chapter 5, a simple way to circumvent to this limitation. Interestingly, subgradients with respect to the probability vectors involved in the GW matching can be computed from subgradients of the well-known Wasserstein distance:

Theorem 2 (subgradient w.r.t masses of GW problem) *Let $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ be two graphs. Let $\boldsymbol{T}^*$ be an optimal coupling of the GW problem between $(\boldsymbol{C}, \boldsymbol{h}), (\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. We define the following cost matrix $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* = \left(\sum_{kl}(C_{ik} - \overline{C}_{jl})^2 T^*_{kl}\right)_{ij}$. Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the following linear OT problem:*

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T} \rangle_F \tag{3.26}$$

Then $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ (resp. $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$) is a subgradient of the function $\mathrm{GW}_2^2(\boldsymbol{C}, \cdot, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ (resp. $\mathrm{GW}_2^2(\boldsymbol{C}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \cdot)$).

Sketch of the proof of Theorem 5. The detailed proof can be found in Annex (8.1.1). A first step to prove Theorem 5 consists in relating a solution $\boldsymbol{T}^*$ of the GW problem to a solution of the Linear Program (LP) given in Equation (3.48) using (Murty, 1988, Theorem 1.12), reported in Theorem 12. Denoting $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ and $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ an optimal solution to the dual problem of (3.48), we have by strong duality

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T} \rangle_F = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h} \rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}} \rangle = \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}^* \rangle_F = \mathrm{GW}(\boldsymbol{C}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \tag{3.27}$$

Then the objective is to show that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \boldsymbol{q})$ (by symmetry the result will be true for $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$). We will do so by leveraging the weak-duality of the GW problem as described in the next lemma:

Lemma 1 *For any vectors $\boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\nu} \in \mathbb{R}^m$ we define:*

$$\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\nu}) := \min_{\boldsymbol{T} \geq 0} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} - \boldsymbol{\mu} \mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^\top, \boldsymbol{T} \rangle$$

Let $\boldsymbol{T}^*$ be an optimal solution of the GW problem. Consider:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T} \rangle_F \tag{3.28}$$

where $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*$. Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the problem in (3.48). If $\mathcal{G}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ then $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{GW}_2(\boldsymbol{C}, \boldsymbol{C}, \boldsymbol{h}, \boldsymbol{q})$

After proving Lemma 2, we conclude the proof of Theorem 5 by showing that $\mathcal{G}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ thanks to equation (3.49), where $\mathcal{G}$ is defined in Lemma's equation (2) □

The proposition above shows that the subgradient of GW *w.r.t.* the probability vectors can be found by solving a linear OT problem which relates to a Wasserstein distance. The ground cost $\boldsymbol{M}(\boldsymbol{T}^*)$ of this Wasserstein problem is moreover the gradient *w.r.t.* the couplings of the optimal GW loss (up to a factor 2) when structure matrices are assumed symmetric (see equation (3.39)). Theorem 5 can be equivalently written using $\boldsymbol{M}(\boldsymbol{T}^*) = \frac{1}{2}\{\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + \mathcal{L}(\boldsymbol{C}^\top, \overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T}^*\}$, to highlight the gradient of the GW loss in the general case to include asymmetric structure matrices.
In practice, when the GW problem is solved with a Conditional Gradient algorithm (Section 3.2.4), the latter already requires to solve this linear OT problem at each iteration. Thus a subgradient *w.r.t.* the weights can be extracted for free from the last iteration of the CG algorithm.

Algorithm 2 Projected Gradient Descent solver for extended GW barycenter problem (3.47).

1: Inputs: Graphs $\{(\boldsymbol{C}_i, \boldsymbol{h}_i)\}$, learning rates η_C and η_h.
2: **repeat**
3: Compute OT matrices with corresponding dual potentials $(\boldsymbol{T}_i, \boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ of independent GW matching of $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$.
4: Perform the following updates of $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{h}}$ using gradients respectively given in equations (3.45) and (3.52):

$$\overline{\boldsymbol{C}} \leftarrow \overline{\boldsymbol{C}} - \eta_C \nabla_{\overline{\boldsymbol{C}}} \quad \text{and} \quad \overline{\boldsymbol{h}} \leftarrow Proj_{\Sigma_{\overline{n}}}(\overline{\boldsymbol{h}} - \eta_h \nabla_{\overline{\boldsymbol{h}}}) \tag{3.31}$$

5: **until** convergence.

Therefore using Theorem 5, the computation of a subgradient *w.r.t* $\overline{\boldsymbol{h}}$ of the objective function of the extended GW barycenter problem (3.47) can be achieved, for a fixed $\overline{\boldsymbol{C}}$, using the following primal-dual like relation:

$$\sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) = \sum_{i \in [\![I]\!]} \lambda_i \left(\langle \boldsymbol{\mu}_i, \boldsymbol{h}_i \rangle + \langle \boldsymbol{\nu}_i, \overline{\boldsymbol{h}} \rangle \right) \tag{3.29}$$

where $(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ are the dual potentials associated to the linear program given in equation (3.48), for each independent GW matching between $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. Note that any dual optimum $(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ is determined up to an additive constant, since for any real value $x \in \mathbb{R}$ the pair $(\boldsymbol{\mu}_i + x\mathbf{1}_{n_i}, \boldsymbol{\nu}_i - x\mathbf{1}_{\overline{n}})$ is also a feasible solution to the underlying linear problem (Cuturi & Avis, 2014). As such, we consider a normalized version of $\boldsymbol{\nu}_i$ which sums to zero, denoted $\widetilde{\boldsymbol{\nu}}_i$, so that the subgradient *w.r.t* $\overline{\boldsymbol{h}}$ reads as

$$\nabla_{\overline{\boldsymbol{h}}} \left(\sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \right) = \sum_{i \in [\![I]\!]} \lambda_i \widetilde{\boldsymbol{\nu}}_i \tag{3.30}$$

We then propose to solve for (3.47) using a projected (sub)gradient descent algorithm, summarized in Algorithm 3. At each iteration, we first find an OT matrix $\boldsymbol{T}_i$ with corresponding dual potential $\widetilde{\boldsymbol{\nu}}_i$ for each GW matching from the input graph $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ to $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. Then we perform a projected gradient update of $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{h}}$, whose respective gradients are evaluated at fixed pairs $\{(\boldsymbol{T}_i, \widetilde{\boldsymbol{\nu}}_i)\}$ by using the Envelope Theorem (Bonnans & Shapiro, 2000). In practice, we suggest to use accelerated projected gradient descent using *e.g.* Adam optimizer (Kingma & Ba, 2015).

Experiments on graph multi-partitioning. We briefly compare now both approaches, namely the vanilla GW barycenter problem (3.44) and its extended version (3.47), on a simple task of graph multi-partitioning. To this end, we design synthetic datasets of 10 homogeneous graphs with orders varying in $\{20, 22, ..., 38\}$, generated via Stochastic Block Model (SBM, Holland et al. (1983)). Each graph is composed of two clusters in imbalanced proportions $[30\%; 70\%]$, both with intra-cluster and inter-cluster connection probabilities of 0.8 and 0.2, respectively.

To perform the simultaneous partitioning of the generated graphs, we first estimate a GW barycenter $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$, either fixing $\overline{\boldsymbol{h}}$ to uniform (equation (3.44)) or learning it (equation (3.47)), with as many nodes as the true number of clusters (*i.e.* $\overline{n} = 2$). Then we recover the OT matrix $\boldsymbol{T}_i$ from each GW problem between an input graph $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ and the estimated barycenter $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$, already computed during the barycenter computation. Finally, the cluster assignment of a node j for a given graph $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ is taken as the *argmax* over the i^{th} row $\boldsymbol{T}_i$

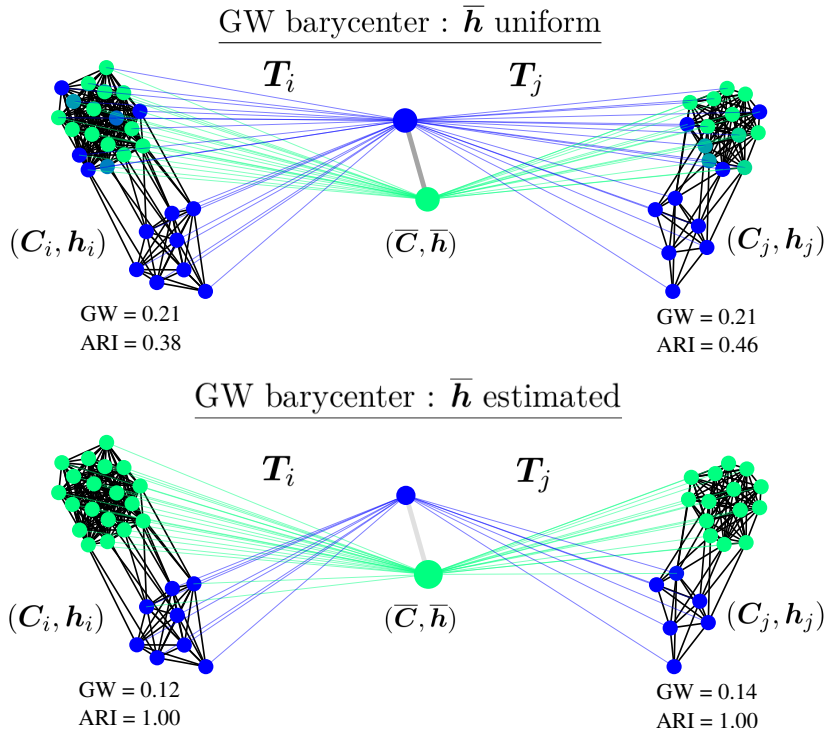

FIGURE 3.4: Illustration of the multi-partitioning tasks performed by learning barycenters of the simulated graphs, either fixing $\overline{\boldsymbol{h}}$ to uniform (top) or estimating it (bottom). Colors are given to nodes of the barycenter, then node colors of the input graphs are assigned by transporting colors of the barycenter nodes according to $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ respectively.

of the OT matrix. The quality of the multi-partitioning of the graphs is then measured by means of Adjusted Rand Index (ARI), between the predicted cluster assignments and the *known* ground truth, and taken on average over all graphs. For both barycenter problems, we run these experiments on 5 simulated datasets, then the respective averaged ARIs and estimated optimal loss values are reported in Table 3.2. Moreover, we provide an illustration of the GW matchings of two simulated samples to the learned barycenters in Figure 3.6.

First, we can conclude that estimating the barycenter masses $\overline{\boldsymbol{h}}$ is crucial to simultaneously partition these graphs perfectly. As expected this approach also leads to a better overall averaged GW reconstruction of input graphs, hence the learned representation (*i.e* the barycenter) relates to a better understanding of the structures. Interestingly, the loss while estimating weights exhibits a larger variance than while considering uniform weights. As input graphs have a few nodes sampled from SBM, they naturally have more diversity in their actual structures, which might explain this variance.

We postulate that our approach can also lead to improvement for the simultaneous partitioning of *heterogeneous* graphs, as Xu et al. (2019a) studied for barycenters endowed of fixed probability weights. In such scenario, the partitioning has to be performed recursively, by iteratively learning a barycenter of graphs before splitting them into subgraphs.

However considering the broader paradigm of Graph Representation Learning (Chapter 2), representing graphs as a single structure like their GW barycenter might be too limiting to perform well on down-stream tasks. As such, we proposed novel GW based Representation Learning methods that will be developed in Chapters 4, 5 and 6.

3.2.3 Gromov-Wasserstein properties

We now develop the properties of the distortion cost resulting from the GW problem (3.21). First, we detail invariants of the problem for any inner costs for which the matching problem

TABLE 3.1: Graph Multi-partitioning Performances of both GW barycenter problems, namely with fixed uniform weights (unif.) and estimated ones (est.).

GW bary.	ARI	loss
$\overline{\boldsymbol{h}}$ unif	0.35(0.03)	2.07(0.01)
$\overline{\boldsymbol{h}}$ est.	**1.00(0.00)**	**1.39(0.09)**

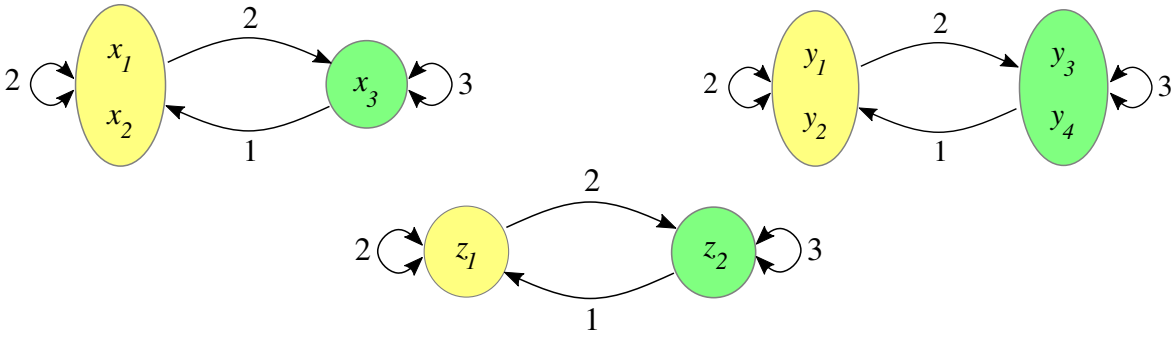

FIGURE 3.5: Illustration of 3 weakly isomorphic (asymmetric) networks, respectively supported on $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$. Inspired from Chowdhury & Mémoli (2019).

is well-defined, before formalizing its resulting metric properties.

Invariants of the GW problem. The GW problem (3.21) allows to compare probability distributions whose supports lie in different, potentially non-related, spaces by defining a notion of equivalence between both compared measures.
Following the generic setup developed in Chowdhury & Mémoli (2019), a first query is that the information contained in a graph (or network) should be unchanged under relabeling of its nodes. When operating on graphs modeled as probability distributions (Section 3.2.1), this invariance relates to the concept of *strong isomorphism* formalized in general as follows

Definition 1 (Strong isomorphism) *Two measurable networks $(\mathcal{X}, c, \mu)$ and $(\overline{\mathcal{X}}, \overline{c}, \overline{\mu})$ are strongly isomorphic if and only if there exists a Borel measurable bijection $\phi : \mathcal{X} \to \overline{\mathcal{X}}$ satisfying both:*

i) $c(x, x') = \overline{c}(\phi(x), \phi(x'))$ *for all* $x, x' \in \mathcal{X}$.

ii) $\phi_{\#}\mu = \overline{\mu}$.

Moreover if both $\mathcal{X}$ and $\overline{\mathcal{X}}$ are finite of cardinality n and $\overline{n}$, respectively, then the corresponding graphs $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ are strongly isomorphic if and only if $n = \overline{n}$ and there exists a permutation matrix $\boldsymbol{P} \in \{0,1\}^{n \times n}$ such that (i) $\overline{\boldsymbol{C}} = \boldsymbol{P}\boldsymbol{C}\boldsymbol{P}^\top$ and (ii) $\boldsymbol{P}\boldsymbol{h} = \overline{\boldsymbol{h}}$.

Therefore the notion of strong isomorphism corresponds to the existence of a measure preserving map between both matched graphs. Moreover, if $(\mathcal{X}, c)$ and $(\overline{\mathcal{X}}, \overline{c})$ are metric spaces, the structure preservation constraint (i) is equivalent to the existence of an *isometry* between both spaces, *i.e* a surjective map $\phi : \mathcal{X} \to \overline{\mathcal{X}}$ that preserves the distances, which is *de facto* bijective on these metric spaces, since for $\phi(x) = \phi(x')$ we have $\overline{c}(\phi(x), \phi(x')) = 0 = c(x, x')$ that implies $x = x'$.

The relaxation to non-metric inner costs also allows multiple nodes to have the same internal and external perceptions, *i.e* they have the same incoming and outgoing edge weights. Thus these nodes can be merged or split without information loss. This idea is formalized with the notion of *weak isomorphism*:

Definition 2 (Weak isormorphism) *Two measurable networks* $(\mathcal{X}, c_\mathcal{X}, \mu_\mathcal{X})$ *and* $(\mathcal{Y}, c_\mathcal{Y}, \mu_\mathcal{Y})$ *are weakly isomorphic if there exists* $(\mathcal{Z}, c_\mathcal{Z}, \mu_\mathcal{Z})$ *with* $\text{supp}(\mu_\mathcal{Z}) = \mathcal{Z}$ *and measurable maps* $\phi : \mathcal{Z} \to \mathcal{X}$ *and* $\psi : \mathcal{Z} \to \mathcal{Y}$ *such that*

i) $c_\mathcal{Z}(z, z') = c_\mathcal{X}(\phi(z), \phi(z')) = c_\mathcal{Y}(\psi(z), \psi(z'))$ *for all* $z, z' \in \mathcal{Z}$.

ii) $\phi_\# \mu_\mathcal{Z} = \mu_\mathcal{X}$ *and* $\psi_\# \mu_\mathcal{Z} = \mu_\mathcal{Y}$.

The weak isomorphism highlights a "tripod structure" in which the equivalence between spaces is defined through a third space $\mathcal{Z}$. The latter can be seen as an irreducible factorization of both matched graphs whose weakly isomorphic nodes are aggregated as a unique node. An illustration of isomorphic graphs is provided in Figure 3.5, where the two first networks on the first row of plots are weakly isomorphic, and admit as irreducible network the one illustrated on the second row. This notion of isomorphism is mostly relevant for *non-metric* inner costs, as for metric ones strong and weak isomorphisms are equivalent (Sturm, 2012, Lemma 1.10).

Metric properties of GW. In the vein of the Wasserstein distances (Section 3.1.2) with p-norms as ground cost, the Gromov-Wasserstein problems endowed with $(x, y) \in \mathbb{R} \times \mathbb{R} \to |x - y|^p$ as ground cost on the real-line defines a (pseudo) metric *w.r.t* the aforementioned notions of isomorphism. The following theorem aims at unifying the metric properties of GW respectively addressed in Sturm (2012); Chowdhury & Mémoli (2019):

Theorem 3 (Metric properties of GW) *Let* $(\mathcal{X}, c_\mathcal{X}, \mu_\mathcal{X})$ *and* $(\mathcal{Y}, c_\mathcal{Y}, \mu_\mathcal{Y})$ *two measurable networks and any* $p \in \mathbb{N}^*$,

i) GW_p *is symmetric, positive and satisfies the triangle inequality. So for any measurable network* $(\mathcal{Z}, c_\mathcal{Z}, \mu_\mathcal{Z})$ *we have:*

$$\text{GW}_p(\mu_\mathcal{X}, \mu_\mathcal{Y}) \leq \text{GW}_p(\mu_\mathcal{X}, \mu_\mathcal{Z}) + \text{GW}_p(\mu_\mathcal{Z}, \mu_\mathcal{Y}) \tag{3.32}$$

ii) $\text{GW}_p(\mu_\mathcal{X}, \mu_\mathcal{Y}) = 0$ *if and only if* $(\mathcal{X}, c_\mathcal{X}, \mu_\mathcal{X})$ *and* $(\mathcal{Y}, c_\mathcal{Y}, \mu_\mathcal{Y})$ *are weakly isomorphic.*

iii) if $c_\mathcal{X}$ *and* $c_\mathcal{Y}$ *are metrics then for any* $q \geq 1$, $\text{GW}_p(\mu_\mathcal{X}, \mu_\mathcal{Y}) = 0$ *if and only if* $(\mathcal{X}, c_\mathcal{X}^q, \mu_\mathcal{X})$ *and* $\left(\mathcal{Y}, c_\mathcal{Y}^q, \mu_\mathcal{Y}\right)$ *are strongly isomorphic.*

Proof of Theorem 3. The positiveness in (i) is straightforward. We refer the reader to the proof of Chowdhury & Mémoli (2019), Theorem 16, for the symmetry, and the triangle inequality partly proven thanks to the Gluing Lemma (Villani, 2021, Lemma 7.6) applied across spaces. (ii) and (iii) are respectively proven in Theorem 18 of Chowdhury & Mémoli (2019) and Lemma 9.2 of Sturm (2012). Interestingly, the proof for (ii) is also valid even if $c_\mathcal{X}$ and $c_\mathcal{Y}$ are not p-integrable. □

Theorem 3 has a lot of implications. First it can endow the space of all measurable networks or the space of all measurable metric spaces with a topology, a geometric structure, induced by Gromov-Wasserstein distances. However, this assertion holds if all pairwise GW distances are finite. This requirement is handled by considering the intersection of both aforementioned spaces to the space of measurable networks with finite L^p-size, defined for any $(\mathcal{X}, c, \mu)$ as

$$\text{size}_p(\mathcal{X}, c, \mu) = \left(\int_{\mathcal{X} \times \mathcal{X}} |c(x, x')|^p d\mu(x) d\mu(x') \right)^{1/p} \tag{3.33}$$

As such the main theorem on the metric properties of GW can be stated as:

Theorem 4 (GW is a distance) *For any $p \in \mathbb{N}^*$,*

i) GW_p *is a distance on the space of all metric measure spaces with finite* L^p*-size quotiented by the strong isomorphisms.*

ii) GW_p *is a distance on the space of all measurable networks with finite* L^p*-size quotiented by the weak isomorphisms.*

These theorems emphasize that GW is well suited for comparing objects while seeking for a large class of invariants such as rotations, translations or permutations. Instances of related applications are then shape matching where the orientation of a shape does not define its nature or for graphs where node ordering is arbitrary.

Interestingly, the space of all measurable metric spaces endowed with the topology induced by the GW distance has a nice geodesic structure (Ambrosio et al., 2005; Sturm, 2012), in resonance with the geodesic structure of the space of probability measures equipped with the Wasserstein distance (Subsection 3.1.2).

GW relations to other OT problems. Gromov-Wasserstein was first introduced in Mémoli (2011) as a probabilistic relaxation of the Gromov-Haussdorf distance (Mémoli & Sapiro, 2004; 2005) quantifying how two metric spaces are far from being isometric, and acts as a "smoothing" of the latter that promotes computational tractability.

On one hand, The recent success and flexibility of linear or quadratic OT paradigms for graph data processing has led to an interest in unifying various existing approaches, or even theories. For instance, the growing appeal of Weisfeiler-Lehman (WL) isomorphism tests and Gromov-Wasserstein distances has prompted research into the relationship between these two concepts. WL tests stand by their ability to *potentially* detect isomorphism in linear time, thus inspiring, for instance, most recent designs of Graph Neural Networks (GNN, Section 2.2, Chapter 4). In the meanwhile, GW shines by its ability to quantify how far two graphs are from being isomorphic, while providing interpretable soft assignments between nodes of the graphs. Chen et al. (2022) recently proposed the WL distance between *measurable Markov Chains* encompassing (attributed) graphs quantifying how far two such objects are far from a dedicated notion of isomorphism. Then, authors identified a variant of GW suiting their Markov Chain based formalism, that upper bounds the WL distance, as excepted from their respective ability to detect isomorphism.

On the other hand, the broad class of invariants inherent to the GW problem may not suit all matching problems. For instance, in unsupervised word translation resulting from the alignment of embedded words known to belong to spaces invariant to rotations. In this scenario, an explicit encoding of the desired invariants instead of treating all isomorphic transformations, might lead to a better matching, *e.g* using Wasserstein Procrustes (Grave et al., 2019) or other rotation invariant OT problems (Alvarez-Melis et al., 2019).

3.2.4 Solving Gromov-Wasserstein problems

We further detail the optimization problem inherent to the GW distance (3.21) between measurable networks of finite support, before describing solvers used to estimate solutions of the problem. We respectively encode these networks or graphs, as tuples $(\boldsymbol{C}, \boldsymbol{h}) \in \mathbb{R}^{n\times n} \times \Sigma_n$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \in \mathbb{R}^{\overline{n}\times\overline{n}} \times \Sigma_{\overline{n}}$. Then the GW matching problem between both graphs aims at solving :

$$\mathrm{GW}_p^p(\boldsymbol{C}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) = \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \sum_{ijkl} |C_{ij} - \overline{C}_{kl}|^p T_{ik} T_{jl} = \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}})^p \otimes \boldsymbol{T}, \boldsymbol{T}\rangle \quad (3.34)$$

where we define $\boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}})$ as the tensor $\boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) = \left(|C_{ik} - \overline{C}_{jl}|\right)_{i,j,k,l} = (L_{ijkl})$ and $\otimes$ is the tensor-matrix multiplication, such that $\boldsymbol{L} \otimes \boldsymbol{T} = \left(\sum_{kl} L_{ijkl} T_{kl}\right)_{ij}$. In general, the optimization problem (3.34) is a non-convex quadratic program which belongs to the class of NP-hard problems (Loiola et al., 2007). It is therefore expected that its approximation is costly. *De facto* an evaluation of the objective function depends on all comparisons of connections within each graph inducing $O(n^2\overline{n}^2)$ operations, as the computation of $\boldsymbol{L} \otimes \boldsymbol{T}$.

The specific case of $p = 2$ attracted attention, partly because the tensor-matrix multiplication can then be factored as

$$\boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} = \boldsymbol{c}_{\boldsymbol{C},\overline{\boldsymbol{C}}} - 2\boldsymbol{C}\boldsymbol{T}\overline{\boldsymbol{C}}^\top \tag{3.35}$$

where $\boldsymbol{c}_{\boldsymbol{C},\overline{\boldsymbol{C}}} = \boldsymbol{C}^2\boldsymbol{h}\boldsymbol{1}_{\overline{n}}^\top + \boldsymbol{1}_n\overline{\boldsymbol{h}}^\top\overline{\boldsymbol{C}}^2$, which can be computed within $O(n^2\overline{n} + \overline{n}^2 n)$ operations (Peyré et al., 2016, proposition 1). Then the GW problem can be expressed as

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{c}_{\boldsymbol{C},\overline{\boldsymbol{C}}}, \boldsymbol{T} \rangle - 2\langle \boldsymbol{C}\boldsymbol{T}\overline{\boldsymbol{C}}^\top, \boldsymbol{T} \rangle \tag{3.36}$$

which admits as equivalent classical QP formulation:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \frac{1}{2}\mathrm{vec}(\boldsymbol{T})^\top \boldsymbol{Q}\mathrm{vec}(\boldsymbol{T}) + \mathrm{vec}(\boldsymbol{c}_{\boldsymbol{C},\overline{\boldsymbol{C}}})^\top \mathrm{vec}(\boldsymbol{T}) \tag{3.37}$$

where vec is the column-stacking matrix operator and $\boldsymbol{Q} = -4\boldsymbol{C} \otimes_K \overline{\boldsymbol{C}}$ with $\otimes_K$ the Kronecker product. Problem (3.37) is in general a non-convex QP, as its Hessian $\boldsymbol{C} \otimes \overline{\boldsymbol{C}}$ admits for eigenvalues the product of the eigenvalues of $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$.

GW as a generalized soft graph matching problem. Interestingly, the quadratic term in the formulation (3.36) emphasizes the close intimacy between the GW problem and the original *Graph Matching problem* which itself belongs to the class of *Quadratic Assignment Problems* (QAP) (Berg et al., 2005; Loiola et al., 2007; Lyzinski et al., 2015; Maron & Lipman, 2018). Indeed, Graph matching problems aim at matching two graphs of the *same order*, modeled as connectivy matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ in $\mathbb{R}^{n \times n}$. These types of problem are often addressed by solving:

$$\min_{\boldsymbol{P} \in \Pi_n} \|\boldsymbol{C}_1\boldsymbol{P} - \boldsymbol{P}\boldsymbol{C}_2\|^2 \Leftrightarrow \min_{\boldsymbol{P} \in \Pi_n} -2\langle \boldsymbol{C}_1\boldsymbol{P}\boldsymbol{C}_2^\top, \boldsymbol{P} \rangle \tag{3.38}$$

where Π_n is the set of permutation matrices. One way to approximate solution of (3.38) is to relax the combinatorial nature of the problem, by expanding the constraint set to the convex-hull of Π_n, namely the set of doubly stochastic matrices $DS = \{\boldsymbol{X} \in \mathbb{R}^{n \times n} | \boldsymbol{X}\boldsymbol{1}_n = \boldsymbol{1}_n, \boldsymbol{X}^\top\boldsymbol{1}_n = \boldsymbol{1}_n, \boldsymbol{X} \geq \boldsymbol{0}_{n \times n}\}$ (Aflalo et al., 2015; Kezurer et al., 2015; Dym et al., 2017; Bernard et al., 2018). As such, the GW problem between two graphs of the same order, with uniform probability measures such that $\boldsymbol{h} = \overline{\boldsymbol{h}} = \boldsymbol{1}_n/n$, is equivalent to the convex relaxation of the original Graph Matching problem. Therefore, GW can be seen as a generalization to graphs of any order n and $\overline{n}$, which may differ from each other, while additionally inducing a notion of relative importance between nodes through $\boldsymbol{h} \in \Sigma_n$ and $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$.

Conditional Gradient algorithm. The GW problem expressed in its classical QP form (3.37) might be treated as a Wasserstein problem (3.4) with a non-convex quadratic regularization (Ferradans et al., 2014; Flamary et al., 2014). As mentioned in Section 3.1.3, the non-convexity of the regularization motivates the use of a generic Conditional Gradient (CG) algorithm, *a.k.a* Frank-Wolfe algorithm (Frank & Wolfe, 1956). The algorithm then iterates over the 3 following steps (Vayer et al., 2019a):

i) Compute the first-order Taylor approximation of the GW cost evaluated at current estimate $\boldsymbol{T}$, given by its gradient:

$$\boldsymbol{G}(\boldsymbol{T}) = \boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + \boldsymbol{L}(\boldsymbol{C}^\top, \overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T} \tag{3.39}$$

ii) Find a descent-direction by minimizing the computed linearization of the problem (3.39):

$$\boldsymbol{X}^* \in \underset{\boldsymbol{X} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})}{\arg\min} \langle \boldsymbol{G}(\boldsymbol{T}), \boldsymbol{X} \rangle \tag{3.40}$$

which is a classical linear OT problem, with $\boldsymbol{G}(\boldsymbol{T})$ as cost matrix, efficiently estimated *e.g* using a Network Flow algorithm (see Section 3.1.3).

iii) Update the coupling estimate with a line-search between $\boldsymbol{T}$ and $\boldsymbol{X}^*$, which comes down to a constraint minimization of a second degree polynomial function admitting closed form solution. We refer the reader to Section 3.3 where this step is detailed in a more generic scenario.

While the problem is non-convex, CG is known to converge to a local stationary point with $O(\frac{1}{\sqrt{t}})$ rate, where t denotes the number of performed iterates (Lacoste-Julien, 2016). Steps i) and iii) have numerical complexity $O(n^2\overline{n} + n\overline{n}^2)$ using the factorization of $\boldsymbol{L}$ given in equation (3.35). Whereas solving linear OT problem using the Network Flow algorithm consists in $O\left(VE \log V \min\{\log(V\|\boldsymbol{G}\|_\infty), E \log V\}\right)$ operations at each iteration (Tarjan, 1997), where $V = n + \overline{n}$ is the number of nodes and $E = O(n\overline{n})$ is the number of edges. So the latter is the theoretical bottleneck of the method, even if an efficient C++ implementation exist (Bonneel et al., 2011; Flamary et al., 2021).

Entropic Regularization. Adapting the well-known entropic regularization of the linear OT problem (3.13) to the GW problem may also lead to computational benefits. To this end, Peyré et al. (2016); Solomon et al. (2016) proposed to estimate solutions to (3.36) thanks to the following optimization problem:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) - \varepsilon\, \mathrm{H}(\boldsymbol{T}) := \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \mathcal{E}^{GW}_\varepsilon(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) \tag{3.41}$$

This is a non-convex problem which was tackled using projected gradient descent using the KL geometry for both gradient and projection steps (Peyré et al., 2016). More precisely, this algorithm consists in iteratively updating the estimate $\boldsymbol{T}$ following

$$\boldsymbol{T} \leftarrow \mathrm{Proj}^{D_{\mathrm{KL}}}_{\mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \left(\boldsymbol{T} \odot \exp(-\tau \nabla \mathcal{E}^{GW}_\varepsilon(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T})) \right) \tag{3.42}$$

where first, the projection operator relates to the equation (3.14). Then $\tau > 0$ is the step size of descent, and the gradient satisfies $\nabla \mathcal{E}^{GW}_\varepsilon(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) = \boldsymbol{G}(\boldsymbol{T}) + \varepsilon \log(\boldsymbol{T})$ with $\boldsymbol{G}(\boldsymbol{T})$ given in equation (3.39). Peyré et al. (2016) suggested to set $\tau = \frac{1}{\varepsilon}$ so the algorithm iterates (3.42) are equivalent to solve for an entropically regularized linear OT problem with ground cost $\boldsymbol{G}(\boldsymbol{T})$, whose steps consist in

i) Computing the gradient $\boldsymbol{G}(\boldsymbol{T})$ of the GW objective evaluated at the current estimate $\boldsymbol{T}$ given by equation (3.39).

ii) Updating $\boldsymbol{T}$ as the unique solution of an entropic OT problem with kernel $\boldsymbol{K} = \exp\left(-\boldsymbol{G}(\boldsymbol{T})/\varepsilon\right)$ using Sinkhorn's algorithm (Algorithm 1).

From the discussion on the theoretical complexity of the entropic linear OT solver, assuming $n = \overline{n}$, step (ii) can be considered as having $\widetilde{O}(\frac{n^2}{\varepsilon^2})$ *polylogarithmic complexity* (Dvurechensky

et al., 2018). So for well chosen precision estimation ε of a GW solution, the overall theoretical complexity of this algorithm in $O(n^3)$ is due to the gradient computation at step (i).
Like the linear OT case, the entropy term leads to diffuse mass displacements so the resulting regularized OT matrix is not sparse, and only approximates an unregularized OT matrix at precision ε. Moreover a trade-off guided by ε has to be taken into account, for instance seeking for better precision (smaller ε) might posit convergence issues as the learning rate $\tau = 1/\varepsilon$ gets higher.

Like entropic linear OT, the GW problem can be tackled using a Proximal Point algorithm with KL geometry. This approach comes down to endow the aforementioned Projected Gradient algorithm with adaptive learning rates, by taking as Gibbs' kernel $\boldsymbol{K} = \exp\left(-\boldsymbol{G}(\boldsymbol{T})/\varepsilon\right) \odot \boldsymbol{T}$ at each iteration (Xu et al., 2019b).

Other GW solvers. As the complexity of the existing solvers to approximate solutions to the GW problem remains at least cubic in the number of nodes, several recent approaches aim at reducing this computational cost to operate on large graphs.

Mémoli (2011) first proposed to estimate GW thanks to a lower bound resulting from decoupling the action of the OT matrix from node pairs to nodes leading to the problem:

$$\min_{\boldsymbol{T}^{(1)} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \sum_{kl} \left(\min_{\boldsymbol{T}^{(2)} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \sum_{ij} |C_{ik} - \overline{C}_{jl}|^p T^{(2)}_{ij} \right) T^{(1)}_{kl} \tag{3.43}$$

This problem is actually a Wasserstein of Wasserstein distances between 1D empirical distributions on the lines of $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$, such that the inner OT (over $\boldsymbol{T}^{(2)}$) problems can be solved with simple sort algorithms (see special cases mentioned in Section 3.1.2).

Other approaches seek for relevant upper-bounds to the original GW problem, leading to easier optimization problems. A first paradigm consists in imposing specific properties to the transport plans as being of *low rank*, hence admitting glued transport plans as decomposition (Scetbon et al., 2021a;b). Another one rather focuses on partitioning the structures, hardly or softly, to split their matching into the matchings of their respective sub-graphs. This divide-and-conquer strategy was first studied using the GW matchings to a reference structure, namely a GW barycenter (see next Section 3.2.5). As such, a soft partitioning of both input structures is performed simultaneously, based on which nodes of the input graphs are assigned to one or more nodes in the reference structure (Xu et al., 2019a). Then, Chowdhury et al. (2021) formalized and extended this divide-and-conquer strategy. As a result, authors proposed the *quantized Gromov-Wasserstein distance* which can rely on any partitioning scheme (Parés et al., 2017), while also considering node representative of the formed sub-graphs, selected *e.g* using maximal Page Rank (Brin & Page, 1998), to better reconstruct the global OT estimate from the sub-graph matchings.

A last family of methods to approximate the GW distance, further from the scope of this manuscript than the aforementioned approaches, investigate the use of iterative random sampling strategies that take into account both structures and coupling estimate (Vayer et al., 2019b; Kerdoncuff et al., 2021; Li et al., 2022; Fatras et al., 2021b).

3.2.5 Gromov-Wasserstein barycenter

Since the GW problem induces a distance, one can also define a notion of barycenter, in the same vein as the Wasserstein barycenters (Section 3.1.4). This GW barycenter estimation problem was first investigated in Peyré et al. (2016) for discrete probability measures, defined over various spaces endowed with symmetric inner costs such as *undirected graphs*.

Learning the barycenter structure. Let us consider a dataset of graphs $\{\mathcal{G}_i = (\boldsymbol{C}_i, \boldsymbol{h}_i)\}_{i\in[\![I]\!]}$ seen as discrete measurable networks (Chowdhury & Mémoli, 2019), where for a given graph, $\boldsymbol{C}_i$ is an arbitrary node pairwise connectivity matrix, and $\boldsymbol{h}_i$ a probability vector modeling the relative significance of the graph nodes. When the weighting of the barycenter is fixed and given by $\boldsymbol{\lambda} \in \Sigma_I$, the GW barycenter problem in its first formulation seeks for a target structure $\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n}\times\overline{n}}$, composed on $\overline{n}$ nodes whose relative importance is fixed to $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$, solving

$$\min_{\overline{\boldsymbol{C}}\in\mathbb{R}^{\overline{n}\times\overline{n}}} \sum_{i\in[\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) = \min_{\overline{\boldsymbol{C}}} \min_{\{\boldsymbol{T}_i\in\mathcal{U}(\boldsymbol{h}_i,\overline{\boldsymbol{h}})\}_i} \sum_{i\in[\![I]\!]} \lambda_i \mathcal{E}^{GW}(\boldsymbol{C}_i, \overline{\boldsymbol{C}}, \boldsymbol{T}_i) \tag{3.44}$$

This problem is non-convex in general, however the minimization sub-problem *w.r.t* $\overline{\boldsymbol{C}}$, with fixed OT plans, is convex while considering $p = 2$ in GW. Peyré et al. (2016) proposed to solve for problem (3.44) using a BCD procedure which alternates between two steps:

i) Solving I independent GW problems, with respective estimated OT matrices $\boldsymbol{T}_i$, from inputs $(\boldsymbol{C}_i, \boldsymbol{h}_i)$, respectively to the fixed and common target structure $\overline{\boldsymbol{C}}$.

ii) Updating $\overline{\boldsymbol{C}}$ for fixed $\boldsymbol{T}_i$ using a closed form formula for this minimization sub-problem.

Indeed, the gradient *w.r.t* to $\overline{\boldsymbol{C}}$ of the objective function in (3.44) reads as

$$\nabla_{\overline{\boldsymbol{C}}}(\cdot) = 2 \sum_{i\in[\![I]\!]} \lambda_i \{\overline{\boldsymbol{C}} \odot \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top - \boldsymbol{T}_i^\top \boldsymbol{C}_i \boldsymbol{T}_i\} \tag{3.45}$$

so the first optimality condition leads to the update

$$\overline{\boldsymbol{C}} \leftarrow \left(\sum_i \lambda_i \boldsymbol{T}_i^\top \boldsymbol{C}_i \boldsymbol{T}_i\right) \oslash \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \tag{3.46}$$

assuming that the entry-wise division by $\overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top$ is well-defined, *i.e* that $\overline{\boldsymbol{h}}$ does not have a null entry. Otherwise one can simply update the rows and columns of $\overline{\boldsymbol{C}}$ whose corresponding nodes have non-null probability mass. Note that any solver detailed in Section 3.2.4 can be used to solve the inherent OT problems. This GW barycenter problem has also been studied by leveraging the Riemannian geometry of the Gromov-Wasserstein space (Chowdhury & Needham, 2020). The estimation of GW barycenters has recently found many applications for instance for shape interpolation, image clustering (Peyré et al., 2016), graph clustering and partitioning (Xu, 2020), graphon estimation (Xu et al., 2021a), large-scale GW estimation (Xu et al., 2019a) and structured graph prediction (Brogat-Motte et al., 2022).

Fully learning the barycentric measure. This first formulation of the GW barycenter problem (3.44) enforces a *fixed* probability vector $\overline{\boldsymbol{h}}$ to the barycentric measure, despite being unknown in practice, which can be detrimental to certain applications, for instance those relying on partitioning of the input structures such as in Xu et al. (2019a). In that paper, authors use a barycenter of small order as intermediate matching, to estimate the GW matching between large input graphs without comparing them directly (divide and conquer strategy). In this case, the mass conservation constraint impose to get partitions of the input structures whose proportions suit entries of $\overline{\boldsymbol{h}}$.
As such, Xu et al. (2019a) proposed heuristics to estimate these proportions directly from degree distributions of the input measures. However the resulting estimated $\overline{\boldsymbol{h}}$ is most likely limited in many applications. For instance, it is easy to generate synthetic graphs from well-known Stochastic Block Models whose degrees would be uncorrelated to cluster proportions. These considerations advocate for also learning the barycenter probability vector $\overline{\boldsymbol{h}}$, solving for the following optimization problem:

$$\min_{\overline{\boldsymbol{C}}\in\mathbb{R}^{\overline{n}\times\overline{n}},\overline{\boldsymbol{h}}\in\Sigma_{\overline{n}}} \quad \sum_{i\in[\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \tag{3.47}$$

However, any first-order optimization procedure would require to compute a subgradient of the GW distance *w.r.t* this parameter. To the best of our knowledge no theoretical results existed in the literature for finding such subgradients. To this end, we provided in our paper Vincent-Cuaz et al. (2021), fully reported in Chapter 5, a simple way to circumvent to this limitation. Interestingly, subgradients with respect to the probability vectors involved in the GW matching can be computed from subgradients of the well-known Wasserstein distance:

Theorem 5 (subgradient w.r.t masses of GW problem) *Let $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ be two graphs. Let $\boldsymbol{T}^*$ be an optimal coupling of the GW problem between $(\boldsymbol{C}, \boldsymbol{h}), (\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. We define the following cost matrix $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* = \left(\sum_{kl}(C_{ik} - \overline{C}_{jl})^2 T^*_{kl}\right)_{ij}$. Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the following linear OT problem:*

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{3.48}$$

Then $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ (resp. $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$) is a subgradient of the function $\mathrm{GW}_2^2(\boldsymbol{C}, \cdot, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ (resp. $\mathrm{GW}_2^2(\boldsymbol{C}, \boldsymbol{h}, \overline{\boldsymbol{C}}\, \cdot\,)$).

Sketch of the proof of Theorem 5. The detailed proof can be found in Annex (8.1.1). A first step to prove Theorem 5 consists in relating a solution $\boldsymbol{T}^*$ of the GW problem to a solution of the Linear Program (LP) given in Equation (3.48) using (Murty, 1988, Theorem 1.12), reported in Theorem 12. Denoting $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ and $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ an optimal solution to the dual problem of (3.48), we have by strong duality

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle = \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F = \mathrm{GW}(\boldsymbol{C}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \tag{3.49}$$

Then the objective is to show that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \boldsymbol{q})$ (by symmetry the result will be true for $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$). We will do so by leveraging the weak-duality of the GW problem as described in the next lemma:

Lemma 2 *For any vectors $\boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\nu} \in \mathbb{R}^m$ we define:*

$$\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\nu}) := \min_{\boldsymbol{T}\geq 0} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} - \boldsymbol{\mu}\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^\top, \boldsymbol{T}\rangle$$

Let $\boldsymbol{T}^*$ be an optimal solution of the GW problem. Consider:

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{3.50}$$

where $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*$. Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the problem in (3.48). If $\mathcal{G}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ then $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{GW}_2(\boldsymbol{C}, \boldsymbol{C}, \boldsymbol{h}, \boldsymbol{q})$

After proving Lemma 2, we conclude the proof of Theorem 5 by showing that $\mathcal{G}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ thanks to equation (3.49), where $\mathcal{G}$ is defined in Lemma's equation (2) □

Algorithm 3 Projected Gradient Descent solver for extended GW barycenter problem (3.47).

1: Inputs: Graphs $\{(\boldsymbol{C}_i, \boldsymbol{h}_i)\}$, learning rates η_C and η_h.
2: **repeat**
3: Compute OT matrices with corresponding dual potentials $(\boldsymbol{T}_i, \boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ of independent GW matching of $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$.
4: Perform the following updates of $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{h}}$ using gradients respectively given in equations (3.45) and (3.52):

$$\overline{\boldsymbol{C}} \leftarrow \overline{\boldsymbol{C}} - \eta_C \nabla_{\overline{\boldsymbol{C}}} \quad \text{and} \quad \overline{\boldsymbol{h}} \leftarrow Proj_{\Sigma_{\overline{n}}}(\overline{\boldsymbol{h}} - \eta_h \nabla_{\overline{\boldsymbol{h}}}) \tag{3.53}$$

5: **until** convergence.

The proposition above shows that the subgradient of GW *w.r.t.* the probability vectors can be found by solving a linear OT problem which relates to a Wasserstein distance. The ground cost $\boldsymbol{M}(\boldsymbol{T}^*)$ of this Wasserstein problem is moreover the gradient *w.r.t.* the couplings of the optimal GW loss (up to a factor 2) when structure matrices are assumed symmetric (see equation (3.39)). Theorem 5 can be equivalently written using $\boldsymbol{M}(\boldsymbol{T}^*) = \frac{1}{2}\{\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + \mathcal{L}(\boldsymbol{C}^\top, \overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T}^*\}$, to highlight the gradient of the GW loss in the general case to include asymmetric structure matrices.
In practice, when the GW problem is solved with a Conditional Gradient algorithm (Section 3.2.4), the latter already requires to solve this linear OT problem at each iteration. Thus a subgradient *w.r.t.* the weights can be extracted for free from the last iteration of the CG algorithm.

Therefore using Theorem 5, the computation of a subgradient *w.r.t* $\overline{\boldsymbol{h}}$ of the objective function of the extended GW barycenter problem (3.47) can be achieved, for a fixed $\overline{\boldsymbol{C}}$, using the following primal-dual like relation:

$$\sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) = \sum_{i \in [\![I]\!]} \lambda_i \left(\langle \boldsymbol{\mu}_i, \boldsymbol{h}_i \rangle + \langle \boldsymbol{\nu}_i, \overline{\boldsymbol{h}} \rangle \right) \tag{3.51}$$

where $(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ are the dual potentials associated to the linear program given in equation (3.48), for each independent GW matching between $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. Note that any dual optimum $(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ is determined up to an additive constant, since for any real value $x \in \mathbb{R}$ the pair $(\boldsymbol{\mu}_i + x\mathbf{1}_{n_i}, \boldsymbol{\nu}_i - x\mathbf{1}_{\overline{n}})$ is also a feasible solution to the underlying linear problem (Cuturi & Avis, 2014). As such, we consider a normalized version of $\boldsymbol{\nu}_i$ which sums to zero, denoted $\widetilde{\boldsymbol{\nu}}_i$, so that the subgradient *w.r.t* $\overline{\boldsymbol{h}}$ reads as

$$\nabla_{\overline{\boldsymbol{h}}} \left(\sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{GW}_2^2(\boldsymbol{C}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{h}}) \right) = \sum_{i \in [\![I]\!]} \lambda_i \widetilde{\boldsymbol{\nu}}_i \tag{3.52}$$

We then propose to solve for (3.47) using a projected (sub)gradient descent algorithm, summarized in Algorithm 3. At each iteration, we first find an OT matrix $\boldsymbol{T}_i$ with corresponding dual potential $\widetilde{\boldsymbol{\nu}}_i$ for each GW matching from the input graph $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ to $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. Then we perform a projected gradient update of $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{h}}$, whose respective gradients are evaluated at fixed pairs $\{(\boldsymbol{T}_i, \widetilde{\boldsymbol{\nu}}_i)\}$ by using the Envelope Theorem (Bonnans & Shapiro, 2000). In practice, we suggest to use accelerated projected gradient descent using *e.g.* Adam optimizer (Kingma & Ba, 2015).

Experiments on graph multi-partitioning. We briefly compare now both approaches, namely the vanilla GW barycenter problem (3.44) and its extended version (3.47), on a simple

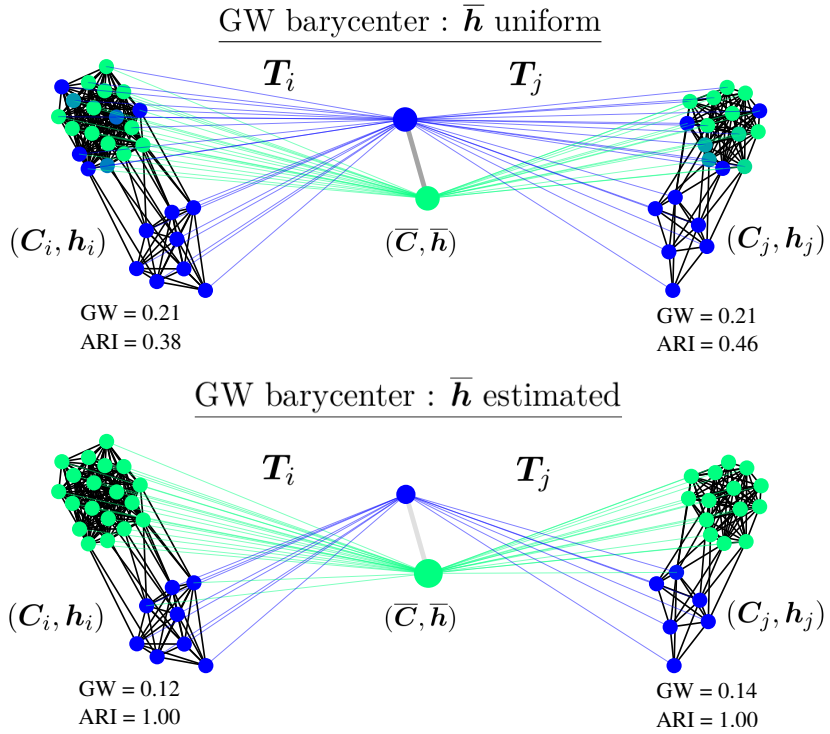

FIGURE 3.6: Illustration of the multi-partitioning tasks performed by learning barycenters of the simulated graphs, either fixing $\overline{\boldsymbol{h}}$ to uniform (top) or estimating it (bottom). Colors are given to nodes of the barycenter, then node colors of the input graphs are assigned by transporting colors of the barycenter nodes according to $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ respectively.

task of graph multi-partitioning. To this end, we design synthetic datasets of 10 homogeneous graphs with orders varying in $\{20, 22, ..., 38\}$, generated via Stochastic Block Model (SBM, Holland et al. (1983)). Each graph is composed of two clusters in imbalanced proportions $[30\%; 70\%]$, both with intra-cluster and inter-cluster connection probabilities of 0.8 and 0.2, respectively.

To perform the simultaneous partitioning of the generated graphs, we first estimate a GW barycenter $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$, either fixing $\overline{\boldsymbol{h}}$ to uniform (equation (3.44)) or learning it (equation (3.47)), with as many nodes as the true number of clusters (*i.e.* $\overline{n} = 2$). Then we recover the OT matrix $\boldsymbol{T}_i$ from each GW problem between an input graph $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ and the estimated barycenter $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$, already computed during the barycenter computation. Finally, the cluster assignment of a node j for a given graph $(\boldsymbol{C}_i, \boldsymbol{h}_i)$ is taken as the *argmax* over the i^{th} row $\boldsymbol{T}_i$ of the OT matrix. The quality of the multi-partitioning of the graphs is then measured by means of Adjusted Rand Index (ARI), between the predicted cluster assignments and the *known* ground truth, and taken on average over all graphs. For both barycenter problems, we run these experiments on 5 simulated datasets, then the respective averaged ARIs and estimated optimal loss values are reported in Table 3.2. Moreover, we provide an illustration of the GW matchings of two simulated samples to the learned barycenters in Figure 3.6.

First, we can conclude that estimating the barycenter masses $\overline{\boldsymbol{h}}$ is crucial to simultaneously partition these graphs perfectly. As expected this approach also leads to a better overall averaged GW reconstruction of input graphs, hence the learned representation (*i.e* the barycenter) relates to a better understanding of the structures. Interestingly, the loss while estimating weights exhibits a larger variance than while considering uniform weights. As input graphs have a few nodes sampled from SBM, they naturally have more diversity in their actual structures, which might explain this variance.

We postulate that our approach can also lead to improvement for the simultaneous

TABLE 3.2: Graph Multi-partitioning Performances of both GW barycenter problems, namely with fixed uniform weights (unif.) and estimated ones (est.).

GW bary.	ARI	loss
$\overline{\boldsymbol{h}}$ unif	0.35(0.03)	2.07(0.01)
$\overline{\boldsymbol{h}}$ est.	**1.00(0.00)**	**1.39(0.09)**

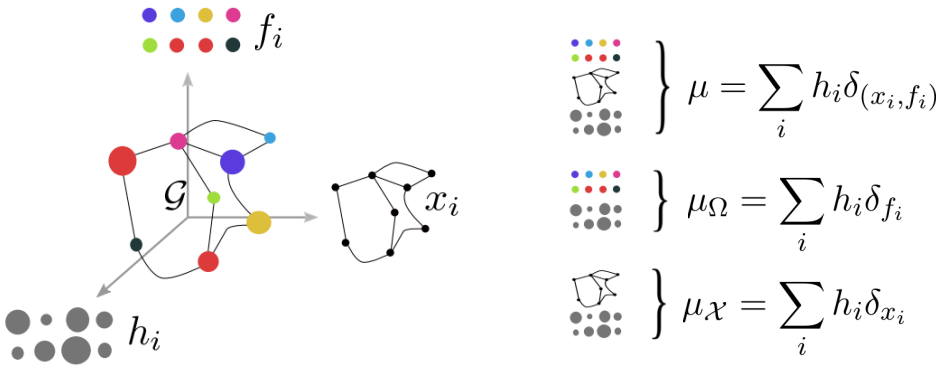

FIGURE 3.7: Illustration of an attributed graph $\mathcal{G}$ (left) with node features $\{\boldsymbol{f}_i\}_i$ and node locations $\{\boldsymbol{x}_i\}_i$. The graph is further endowed with a mass vector $\boldsymbol{h} = (h_i)_i$ measuring the relative significance of its nodes. As such $\mathcal{G}$ can be represented by a fully supported probability measure μ over the product space of feature (Ω, d_Ω) and structure $(\mathcal{X}, c)$ with respective marginals μ_Ω and $\mu_\mathcal{X}$.

partitioning of *heterogeneous* graphs, as Xu et al. (2019a) studied for barycenters endowed of fixed probability weights. In such scenario, the partitioning has to be performed recursively, by iteratively learning a barycenter of graphs before splitting them into subgraphs.

However considering the broader paradigm of Graph Representation Learning (Chapter 2), representing graphs as a single structure like their GW barycenter might be too limiting to perform well on down-stream tasks. As such, we proposed novel GW based Representation Learning methods that will be developed in Chapters 4, 5 and 6.

3.3 Optimal Transport across incomparable spaces endowed with feature information

A Large class of Graph Machine Learning problems are composed of *attributed* graphs, *i.e* nodes, depicted by their pairwise interactions designated as *structure information*, and additionally endowed with *feature information*. The aforementioned Gromov-Wasserstein (Section 3.2) or Wasserstein (Section 3.1) distances, that focus solely on structures and features, respectively, are not able to exploit jointly both information. At least not without dedicated pre-processing schemes.
To go beyond this limitation, Vayer et al. (2019a; 2020) proposed a new OT distance that unveils the geometric nature of the attributed graphs space, leveraging both structure and feature information and consequently called the Fused Gromov-Wasserstein (FGW) distance. In the following, we detail the OT problem inherent to FGW (Section 3.3.1) and its geometric properties (Section 3.3.2). Then we introduce solvers designed for this problem (Section 3.3.3) and the FGW barycenter estimation.

3.3.1 Problem statement

In the OT context, attributed graphs of finite order are a specific instance of structured objects defined over an ambient feature metric space, denoted here (Ω, d_Ω), also designated as *feature space*.

Attributed graphs as discrete probability measures. Let us consider an attributed graph $\mathcal{G}$ modeled as a tuple (V, E), where $V = \{v_i\}_{i\in[\![n]\!]}$ refers to its set of vertices (or nodes) and E to its edges. In the vein of Section 3.2.1, we detail next how to model $\mathcal{G}$ as a fully supported measure illustrated in Figure 3.7.
First, assume the existence of an application $s : V \to \mathcal{X}$ mapping a vertex $v_i \in V$ from the graph to its structure representation $\boldsymbol{x}_i = s(v_i)$ in a structure space $(\mathcal{X}, c)$ where $\mathcal{X}$ is assumed to be a Polish space, and $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a measurable cost function quantifying the connectivity between the nodes in the graph. Second, we associate each vertex $v_i \in V$ to a feature $\boldsymbol{f}_i = a(v_i)$ belonging to the feature space (Ω, d_Ω), via a labeling function $a : V \to \Omega$. Both mappings allow the graph to be entirely represented by a fully support probability measure over the product of the structure $(\mathcal{X}, c)$ and feature (Ω, d_Ω) spaces, as $\mu = \sum_{i\in[\![n]\!]} h_i \delta_{(\boldsymbol{x}_i, \boldsymbol{f}_i)}$. For conciseness, we summarize node pairwise structural relations in the matrix $\boldsymbol{C} = (c(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j\in[\![n]\!]} \in \mathbb{R}^{n\times n}$, and node features in $\boldsymbol{F} = (\boldsymbol{f}_i)_{i\in[\![n]\!]} \in \mathbb{R}^{n\times d}$, assuming $\Omega \subset \mathbb{R}^d$. Therefore the measure μ assigned to the graph $\mathcal{G}$ is encoded as the tuple $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$.

The Fused Gromov-Wasserstein problem. We now consider any pair of graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$, of any order n and $\overline{n}$, with measures μ and $\overline{\mu}$ respectively encoded as $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. The Fused Gromov-Wasserstein (FGW) problem relates to a soft graph matching problem associating pairs of structure and feature points, with similar connectivities within each structure pair and with similar features. The FGW problem defined up to a trade-off parameter $\alpha \in [0, 1]$, weighting linearly the importance of the structure and feature information in the graph matching, reads as follow

$$\begin{aligned} \mathrm{FGW}^p_{p,\alpha}(\mu, \overline{\mu}) &= \mathrm{FGW}^p_{p,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \\ &= \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \alpha \boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}})^p \otimes \boldsymbol{T} + (1-\alpha)\boldsymbol{D}^p(\boldsymbol{F}, \overline{\boldsymbol{F}}), \boldsymbol{T}\rangle \\ &= \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \sum_{ijkl} \{\alpha |C_{ij} - \overline{C}_{kl}|^p + (1-\alpha) d^p_\Omega(\boldsymbol{f}_i, \overline{\boldsymbol{f}}_k)\} T_{ik} T_{jl} \end{aligned} \tag{3.54}$$

for any $p \in \mathbb{N}^*$[6]. Where $\boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}})$ is the 4D tensor involved in the objective of the GW problem (3.34), and $\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) = (d_\Omega(\boldsymbol{f}_i, \overline{\boldsymbol{f}}_j))_{i,j\in[\![n]\!]\times[\![\overline{n}]\!]}$ is the pairwise distance between features of both matched graphs.
The objective function in problem (3.54) consists in a convex combination, guided by α, of the GW cost (3.34) between structures $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ and a linear OT cost (3.8) between node feature matrices $\boldsymbol{F}$ and $\overline{\boldsymbol{F}}$. To simply ensure the existance of an OT matrix, one can assume both $\boldsymbol{L}$ and $\boldsymbol{D}$ to be bounded. Then, the OT matrix $\boldsymbol{T}$ minimizing equation (3.54) performs the desired soft attributed graph matching, by simultaneously handling minimizations of the structure transport distortion through the GW loss and of the transport of node features $\boldsymbol{F}$ and $\overline{\boldsymbol{F}}$ seen as point clouds in the metric space (Ω, d_Ω). The latter metric d_Ω can be adapted to the feature nature of the matched graphs, for instance it can be taken as the euclidean norm to compare continuous features of $\mathbb{R}^d$ or as the Hamming distance if features are discrete.

[6] A more generic formulation of the FGW problem can be found Definition 3.5.7 of Vayer et al. (2020) where fused ground costs are considered with exponents $q \geq 1$, namely $\left(\alpha|C_{ij} - \overline{C}_{kl}|^p + (1-\alpha) d^p_\Omega(f_i, \overline{f}_k)\right)^q$.

3.3.2 Fused Gromov-Wasserstein properties

We now develop the properties of the matching cost resulting from the FGW problem (3.54). Combining the studies of Vayer et al. (2019a; 2020) and Chowdhury & Mémoli (2019), we detail invariants of the problem for any inner costs over structures for which the matching problem is well-defined, before formalizing its resulting metric properties.

Invariants of the FGW problem. At first, Vayer et al. (2019a; 2020) introduced the Fused Gromov-Wasserstein problem to operate on graphs whose structures are depicted by *metric* inner costs. As the FGW and GW problems are closely related by definition, the generalization of GW to *measurable networks* endowed with any structure representation investigated in Chowdhury & Mémoli (2019) can also be envisioned for the FGW problem (Section 3.2.3). Actually as such an adaptation is rather straigth-forward, we detail in the following the properties of the FGW problem between any *discrete measurable attributed networks*, whose inner structure costs might be *metric* or *non-metric* ones, hence also potentially asymetric.

First, the aforementioned strong isomorphism (Definition 1) relation stating invariance to nodes permutation of unattributed graphs now reads as follows:

Definition 3 (Strong isomorphism between attributed graphs) *Two measurable attributed graphs* $(\mathcal{X} \times \Omega, c, \mu)$ *and* $(\overline{\mathcal{X}} \times \Omega, \overline{c}, \overline{\mu})$ *with finite orders* n *and* $\overline{n}$*, respectively modeled as* $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ *and* $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$*, are said to be strongly isomorphic if and only if there exists a permutation matrix* $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ *such that*

$$\overline{\boldsymbol{C}} = \boldsymbol{P}\boldsymbol{C}\boldsymbol{P}^\top, \quad \overline{\boldsymbol{F}} = \boldsymbol{P}\boldsymbol{F} \quad \textit{and} \quad \overline{\boldsymbol{h}} = \boldsymbol{P}\boldsymbol{h}. \tag{3.55}$$

Second, the relaxation to *non-metric* inner structure costs allowing multiple nodes to have the same incoming and outgoing edge weights, and also *the same node features*, induces that such nodes can be merged or split without information loss. This idea is formalized with the notion of *weak isomorphism*:

Definition 4 (Weak isormorphism between attributed graphs) *Two measurable attributed graphs* $(\mathcal{X} \times \Omega, c_\mathcal{X}, \mu_\mathcal{X})$ *and* $(\mathcal{Y} \times \Omega, c_\mathcal{Y}, \mu_\mathcal{Y})$ *are weakly isomorphic if there exists* $(\mathcal{Z} \times \Omega, c_\mathcal{Z}, \mu_\mathcal{Z})$ *with* $\text{supp}(\mu_\mathcal{Z}) = \mathcal{Z}$ *and measurable maps* $\phi : \mathcal{Z} \times \Omega \to (\phi_1, \phi_2) \in \mathcal{X} \times \Omega$ *and* $\psi : \mathcal{Z} \times \Omega \to (\psi_1, \psi_2) \in \mathcal{Y} \times \Omega$ *such that for all* $(z, f_z), (z', f'_z) \in \mathcal{Z} \times \Omega$*,*

i) $c_\mathcal{Z}(z, z') = c_\mathcal{X}(\phi_1(z), \phi_1(z')) = c_\mathcal{Y}(\psi_1(z), \psi_1(z'))$.

ii) $d_\Omega(f_z, f'_z) = d_\Omega(\phi_1(f_z), \phi_1(f'_z)) = d_\Omega(\psi_1(f_z), \psi_1(z'))$.

ii) $\phi_\# \mu_\mathcal{Z} = \mu_\mathcal{X}$ *and* $\psi_\# \mu_\mathcal{Z} = \mu_\mathcal{Y}$.

So the weak isomorphism still relates to a "tripod structure" including, both weakly isomorphic attributed graphs $(\mathcal{X} \times \Omega, c_\mathcal{X}, \mu_\mathcal{X})$ and $(\mathcal{Y} \times \Omega, c_\mathcal{Y}, \mu_\mathcal{Y})$, and a *canonical graph representation* $(\mathcal{Z} \times \Omega, c_\mathcal{Z}, \mu_\mathcal{Z})$ resulting from the aggregation of weakly isomorphic nodes of both inputs. As for unattributed graphs, this notion of isomorphism is mostly relevant for *non-metric* inner structure costs, as for metric ones strong and weak isomorphisms are equivalent, which can be easily deduce from the proof of Sturm (2012), Lemma 1.10.

Metric properties of FGW. The Fused Gromov-Wasserstein problem acts as a generalization of the Wasserstein and Gromov-Wasserstein problems other many aspects. The fact that FGW exhibits metric or pseudo-metric properties is one. The next Theorem aims at unifying FGW metric properties mostly addressed in Vayer et al. (2019a; 2020) for *metric* inner

structure costs that we genelarize to non-metric ones following recent findings of Chowdhury & Mémoli (2019)[7]:

Theorem 6 (Metric properties of FGW) *Let $(\mathcal{X} \times \Omega, c_{\mathcal{X}}, \mu_{\mathcal{X}})$ and $(\mathcal{Y} \times \Omega, c_{\mathcal{Y}}, \mu_{\mathcal{Y}})$ two measurable attributed networks. For any $p \in \mathbb{N}^*$ and $\alpha \in [0, 1]$,*

i) $\mathrm{FGW}^p_{p,\alpha}$ *is symmetric and positive.*

ii) $\mathrm{FGW}^p_{p,\alpha}(\mu_{\mathcal{X}}, \mu_{\mathcal{Y}}) = 0$ *if and only if $(\mathcal{X} \times \Omega, c_{\mathcal{X}}, \mu_{\mathcal{X}})$ and $(\mathcal{Y} \times \Omega, c_{\mathcal{Y}}, \mu_{\mathcal{Y}})$ are weakly isomorphic.*

iii) if $c_{\mathcal{X}}$ and $c_{\mathcal{Y}}$ are metrics, then $\mathrm{FGW}_{p,\alpha}(\mu_{\mathcal{X}}, \mu_{\mathcal{Y}}) = 0$ *if and only if $(\mathcal{X} \times \Omega, c_{\mathcal{X}}, \mu_{\mathcal{X}})$ and $(\mathcal{Y} \times \Omega, c_{\mathcal{Y}}, \mu_{\mathcal{Y}})$ are strongly isomorphic.*

iv) $\mathrm{FGW}^p_{p,\alpha}$ *satisfies the triangle inequality if $p = 1$ and a relaxed triangle inequality for $p > 1$. So for any measurable attributed network $(\mathcal{Z} \times \Omega, c_{\mathcal{Z}}, \mu_{\mathcal{Z}})$ we have for any $p \in \mathbb{N}^*$:*

$$\mathrm{FGW}^p_{p,\alpha}(\mu_{\mathcal{X}}, \mu_{\mathcal{Y}}) \leq 2^{p-1}\{\mathrm{FGW}^p_{p,\alpha}(\mu_{\mathcal{X}}, \mu_{\mathcal{Z}}) + \mathrm{FGW}^p_{p,\alpha}(\mu_{\mathcal{Z}}, \mu_{\mathcal{Y}})\} \tag{3.56}$$

Proof of Theorem 6. The positiveness and the symmetry in (i) are rather straightforward. Then, we refer the reader to the proof of Theorem 3.2 in Vayer et al. (2019a) for the relaxed triangle inequality (ii). A detailed inspection of the proof confirms that this property is independent of any metric-like assumptions over inner structure costs, so Theorem 3.2 (Vayer et al., 2019a) can be extended to *non-metric* costs $c_{\mathcal{X}}, c_{\mathcal{Y}}$ and $c_{\mathcal{X}}$, as Chowdhury & Mémoli (2019) did for non-attributed networks. Finally, (iii) and (iv) can be easily deduced from coupling Theorem 3.2 of Vayer et al. (2019a) and Theorem 18 of Chowdhury & Mémoli (2019). □

Theorem 6 endows the space of all measurable attributed networks over (Ω, d_{Ω}) with a topology induced by Fused Gromov-Wasserstein distances. Similarly than for GW, this assertion holds if both structure and feature measurable costs are finite, hence reducing the space to measurable attributed networks with finite "sizes" (Vayer, 2020, Definition 3.5.2). As such, Theorem 4 on the metric (resp. pseudo-metric) nature of Gromov-Wasserstein, over quotient spaces induced by the notion of strong isomorphism (resp. weak isomorphism), can be extended to Fused Gromov-Wasserstein if $p = 1$, otherwise if $p > 1$ their respective *semi-metric* relaxations have to be taken into account (see Theorem 6, iv)).
Interestingly, the space of all measurable metric networks doted of the topology induced by the FGW distance also allows the definition of constant speed geodesic (Vayer et al., 2020, Theorem 3.6).

3.3.3 Solving Fused Gromov-Wasserstein problems

We detail now the optimization problem inherent to the FGW distance (3.21) between measurable attributed networks of finite support, before describing solvers used to estimate solutions of the problem, which are closely related to those for the GW distance (Section 3.2.4).
We respectively encode attributed graphs, as tuples $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times d} \times \Sigma_n$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \in \mathbb{R}^{\overline{n} \times \overline{n}} \times \mathbb{R}^{\overline{n} \times d} \times \Sigma_{\overline{n}}$. In general, the optimization problem inherent to the FGW distance (3.54) is a non-convex quadratic program which inherits its non-convexity and its

[7] Notice that for various OT problems such as GW and FGW, the triangle inequality can be proved from the relaxed triangle inequality as stated in assertion iv), over the loss function taken with exponent p. The proof is analogous to the one of the Minkowski inequality via Hölder's inequality.

Algorithm 4 Conditional Gradient algorithm for FGW estimation

1: Inputs: input graph $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$, target graph $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$, trade-off parameter α.
2: **repeat**
3: Compute the gradient $\nabla_{\boldsymbol{T}} \mathcal{E}_\alpha^{FGW}$ evaluated at current estimate $\boldsymbol{T}$ using equation (3.61).
4: Compute the descent-direction $\boldsymbol{X}$, solving for the linear OT problem (3.40) with cost matrix $\nabla_{\boldsymbol{T}} \mathcal{E}_\alpha^{FGW}$
5: Optimal Line-search step using Algorithm 5: denoting $\boldsymbol{Z}(\gamma) = \boldsymbol{T} + \gamma(\boldsymbol{X} - \boldsymbol{T}) = \boldsymbol{T} + \gamma \Delta\, \boldsymbol{T}$,

$$\gamma^\star = \arg\min_{\gamma \in (0,1)} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{Z}(\gamma)) = \arg\min_{\gamma \in (0,1)} a\gamma^2 + b\gamma + c \tag{3.59}$$

with

$$\begin{aligned} a &= -2\alpha \langle \boldsymbol{C} \Delta \boldsymbol{T} \overline{\boldsymbol{C}}^\top, \Delta \boldsymbol{T} \rangle_F \\ b &= \langle (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) + \alpha \boldsymbol{c}_{\boldsymbol{C}, \overline{\boldsymbol{C}}}, \Delta \boldsymbol{T} \rangle_F - 2\alpha \{ \langle \boldsymbol{C} \boldsymbol{X} \overline{\boldsymbol{C}}^\top, \boldsymbol{T} \rangle_F + \langle \boldsymbol{C} \boldsymbol{T} \overline{\boldsymbol{C}}^\top, \boldsymbol{X} \rangle_F \} \end{aligned} \tag{3.60}$$

6: $\boldsymbol{T} \leftarrow \boldsymbol{Z}(\gamma^*)$
7: **until** convergence

computational complexity from the GW loss. As such the specific case $p = 2$, also benefits from the advantageous factorization of the tensor-matrix multiplication given in equation (3.35), calculable within $O(n^2\overline{n} + \overline{n}^2 n)$ operations (Peyré et al., 2016, proposition 1). In this scenario, the FGW distance between $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$ can be expressed for any $\alpha \in [0, 1]$ as

$$\begin{aligned} \mathrm{FGW}_{2,\alpha}^2(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) &= \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \alpha \boldsymbol{c}_{\boldsymbol{C}, \overline{\boldsymbol{C}}} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}), \boldsymbol{T} \rangle_F - 2\alpha \langle \boldsymbol{C} \boldsymbol{T} \overline{\boldsymbol{C}}^\top, \boldsymbol{T} \rangle_F \\ &= \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \mathcal{E}_\alpha^{FGW}\left(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}\right) \end{aligned} \tag{3.57}$$

where $\mathcal{E}_\alpha^{FGW}$ refers to the FGW cost evaluated in any *admissible* coupling $\boldsymbol{T}$. Then the problem (3.57) admits as equivalent classical QP formulation:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \frac{1}{2} \mathrm{vec}(\boldsymbol{T})^\top \boldsymbol{Q}_\alpha \mathrm{vec}(\boldsymbol{T}) + \mathrm{vec}(\alpha \boldsymbol{c}_{\boldsymbol{C}, \overline{\boldsymbol{C}}} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}))^\top \mathrm{vec}(\boldsymbol{T}) \tag{3.58}$$

with $\boldsymbol{Q}_\alpha = -4\alpha \boldsymbol{C} \otimes_K \overline{\boldsymbol{C}}$ emphasizing that the FGW problem (3.57) is in general a non-convex QP, as its Hessian is the same for the GW problem up to a trade-off factor α, resulting from the inclusion of *feature* information.

Adapting GW solvers to FGW ones. Exactly like the GW problem, the FGW problem expressed in its classical QP form (3.58) can be treated as a Wasserstein problem with a non-convex quadratic regularization (Ferradans et al., 2014; Flamary et al., 2014). As such, most algorithms based on first-order derivatives of the GW cost $\mathcal{E}^{GW}$ will be adapted to the FGW cost $\mathcal{E}_\alpha^{FGW}$ by simply changing the gradient computation step to the adequate one:

$$\begin{aligned} \nabla_{\boldsymbol{T}} \mathcal{E}_\alpha^{FGW} &= \alpha \nabla_{\boldsymbol{T}} \mathcal{E}^{GW} + (1-\alpha) \nabla_{\boldsymbol{T}} \mathcal{E}^{W} \\ &= \alpha \{ \boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + \boldsymbol{L}(\boldsymbol{C}^\top, \overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T} \} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) \end{aligned} \tag{3.61}$$

Algorithm 5 Line-search for any second order polynom $a\gamma^2 + b\gamma + c$.

1: Inputs: coefficients a and b.
2: **If** $a > 0$: $\gamma^* \leftarrow \min\left(1, \max\left(0, \frac{-b}{2a}\right)\right)$
3: **Else if** $a + b < 0$: $\gamma^* \leftarrow 1$.
4: **Else**: $\gamma^* \leftarrow 0$.

For instance, the Projected Gradient descent (PGD) algorithm proposed in Peyré et al. (2016) to solve for the entropically regularized GW problem (3.41) with iterates given by equation (3.42), comes down for FGW to

i) Compute the gradient $\nabla_{\boldsymbol{T}} \mathcal{E}_\alpha^{FGW}(\boldsymbol{T})$ evaluated at current estimate $\boldsymbol{T}$.

ii) Update $\boldsymbol{T}$ using Sinkhorn Algorithm 1 with kernel $\boldsymbol{K} = \exp\left(-\nabla_{\boldsymbol{T}} \mathcal{E}_\alpha^{FGW}(\boldsymbol{T})/\varepsilon\right)$

where $\varepsilon > 0$ is the entropic regularization coefficient. FGW also admits as exact solver related to the PGD, a Proximal Point algorithm, using the KL geometry, which comes down to consider adaptive learning rates through the Gibbs' kernel $\boldsymbol{K} = \exp\left(-\nabla_{\boldsymbol{T}} \mathcal{E}_\alpha^{FGW}(\boldsymbol{T})/\varepsilon\right) \odot \boldsymbol{T}$ at each iteration (Xu et al., 2019b).

Vayer et al. (2019a; 2020) originally proposed to use an exact Conditional Gradient (CG) solver allowing estimated solution to be sparse. The solver is detailed in Algorithm 4, which resembles the one for GW. To complete previous explanations given in Section 3.2.4, we detail the line-search step. It comes down to solving for the constraint minimization problem (3.59), whose objective function can be expressed as a second degree polynomial function in γ with coefficients given in equation (3.60). This problem admits closed form solutions computed using the Algorithm 5. Let us mention that one can recover the detailed intermediate steps of the Conditional Gradient solver for GW, simply setting $\alpha = 1$ in Algorithm 4.

To conclude on the OT solvers for the FGW problem, the complexity of the existing solvers to approximate solutions results from the GW term whose computation remains at least cubic in the number of nodes. But the recent approaches detailed in Section 3.2.4 can also be adapted to the FGW problem to reduce its computational complexity, *e.g* Scetbon et al. (2021a); Xu et al. (2019a); Chowdhury et al. (2021).

3.3.4 Fused Gromov-Wasserstein barycenter

Since FGW has metric properties, Vayer et al. (2019a; 2020) proposed a notion of FGW barycenter as a Fréchet mean of attributed graphs. Authors demonstrated the relevance of this approach *e.g.* on graph denoising and the clustering of graphs and the nodes of the graphs. Then in the vein of the GW barycenter problems (Section 3.2.5), we detail next the FGW barycenter problem, first as introduced by fixing the barycentric distribution to arbitrary probability weights $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$(Vayer et al., 2019a), then while additionally learning $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$ thanks to a novel result adapted from Theorem 5 to compute sub-gradients *w.r.t.* this parameter (Vincent-Cuaz et al., 2021).

Learning the FGW barycenter support. We consider a dataset of attributed graphs $\{\mathcal{G}_i = (\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i \in [\![I]\!]}$ seen as discrete measures (Section 3.3.1). When the weighting of the barycenter is fixed and given by $\lambda \in \Sigma_I$, the FGW barycenter problem in its first formulation seeks for target structure matrix $\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n} \times \overline{n}}$ and feature matrix, composed on $\overline{n}$ nodes whose relative importance is fixed to $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$, solving for

$$\min_{\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n} \times \overline{n}}, \overline{\boldsymbol{F}} \in \mathbb{R}^{\overline{n} \times \overline{d}}} \quad \sum_{i \in [\![I]\!]} \lambda_i \, \mathrm{FGW}_{2,\alpha}^2(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \tag{3.62}$$

This problem is non-convex in general, however for fixed transport matrices $\{\boldsymbol{T}_i\}_i$, the block-minimization sub-problems *w.r.t* $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{F}}$ are convex while considering $\text{FGW}_{2,\alpha}$. Vayer et al. (2019a) then proposed to solve for problem (3.62) adapting the BCD procedure for GW of Peyré et al. (2016) to FGW as follow:

i) Solving I independent FGW problem for a target structure $\overline{\boldsymbol{C}}$ and node features $\overline{\boldsymbol{F}}$ with respective estimated OT matrices $\boldsymbol{T}_i$.

ii) Updating $\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}$ for fixed $\boldsymbol{T}_i$ using closed form formulas for both minimization sub-problems *w.r.t* $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{F}}$. As the gradient *w.r.t* $\overline{\boldsymbol{C}}$ of the FGW cost satisfies $\nabla_{\overline{\boldsymbol{C}}}\mathcal{E}^{FGW}_{\alpha} = \alpha\nabla_{\overline{\boldsymbol{C}}}\mathcal{E}^{GW}$ whose latter is given by equation (3.45), then $\overline{\boldsymbol{C}}$ admits the close form update already given in equation (3.46). Then the update *w.r.t* $\overline{\boldsymbol{F}}$ can be computed *e.g* with Equation 8 of Cuturi & Doucet (2014):

$$\overline{\boldsymbol{F}} \leftarrow \text{diag}(\boldsymbol{1}_{\overline{n}} \oslash \overline{\boldsymbol{h}}) \sum_{i\in[\![I]\!]} \lambda_i \boldsymbol{T}_i^\top \boldsymbol{F}_i \tag{3.63}$$

Fully learning the FGW barycenter measure. This first formulation of the FGW barycenter problem (3.62) enforces an *a priori unknown* probability vector $\overline{\boldsymbol{h}}$ to the barycentric measure, which can be detrimental to certain applications. For instance, those relying on the *simultaneous* partitioning of input structures and node features, such as Xu et al. (2019a) one, previously described for GW in Section 3.2.5. To this end we proposed to *learn* this target distribution $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$, adapting Theorem 5 to the FGW distance, which provides sub-gradients *w.r.t.* the probability weights for the GW problem. Similarly than for GW, a subgradient for FGW weights can be derived from subgradients of the well-known Wasserstein distance:

Theorem 7 (subgradient w.r.t masses of FGW) *Let* $(\boldsymbol{C}, , \boldsymbol{F}, \boldsymbol{h})$ *and* $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$ *be two attributed graphs. Let* $\boldsymbol{T}^*$ *be an optimal coupling of the FGW problem between* $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}), (\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. *We define the following cost matrix* $\boldsymbol{M}(\boldsymbol{T}^*) := \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) = \left(\alpha \sum_{kl}(C_{ik} - \overline{C}_{jl})^2 T^*_{kl} + (1-\alpha)\|\boldsymbol{F}_{i,:} - \overline{\boldsymbol{F}}_{j,:}\|_2^2\right)_{ij}$. *Let* $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ *be the dual variables of the following linear OT problem:*

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{3.64}$$

Then $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ *(resp.* $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$*) is a subgradient of the function* $\text{FGW}^2_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \cdot, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$ *(resp.* $\text{FGW}^2_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \cdot)$*).*

Proof of Theorem 7. The detailed proof can be found in Annex 8.1.2 and follows an analog scheme than the proof for GW given in 8.1.1. □

The ground cost $\boldsymbol{M}(\boldsymbol{T}^*)$ of this Wasserstein problem is moreover the gradient *w.r.t.* the couplings of the optimal FGW loss (up to a factor 2 for the term from the GW loss) when structure matrices are assumed symmetric. Theorem 7 can also be equivalently written using $\boldsymbol{M}(\boldsymbol{T}^*) = \frac{\alpha}{2}\{\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + \mathcal{L}(\boldsymbol{C}^\top, \overline{\boldsymbol{C}}^\top) \otimes \boldsymbol{T}^*\} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}})$, to highlight the gradient of the FGW loss in the general case to include asymmetric structure matrices.
Note that in practice the FGW problem is solved with a CG algorithm which already requires to solve this linear OT problem at each iteration. In this way, after convergence, the gradient *w.r.t.* the weights can be extracted for free from the last iteration of the CG algorithm.

Then, one can endow the FGW barycenter problem of a learnable barycentric probability vector $\overline{\boldsymbol{h}}$, solving for the following optimization problem:

$$\min_{\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\overline{\boldsymbol{h}}} \quad \sum_{i\in[\![I]\!]} \lambda_i \, \mathrm{FGW}^2_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \tag{3.65}$$

Therefore using Theorem 7, the computation of a sub-gradient *w.r.t* $\overline{\boldsymbol{h}}$ of the objective function in problem (3.65) can be achieved, for fixed $\overline{\boldsymbol{C}}$ and $\overline{\boldsymbol{F}}$, using the following primal-dual like relation:

$$\sum_{i\in[\![I]\!]} \lambda_i \, \mathrm{FGW}^2_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) = \sum_{i\in[\![I]\!]} \lambda_i \left(\langle \boldsymbol{\mu}_i, \boldsymbol{h}_i \rangle + \langle \boldsymbol{\nu}_i, \overline{\boldsymbol{h}} \rangle \right) \tag{3.66}$$

where $(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ are the dual *centered* potentials (Section 3.2.5) associated to the linear program given in equation (3.64), for each independent FGW matching between $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. Then the sub-gradient *w.r.t* $\overline{\boldsymbol{h}}$ reads as

$$\nabla_{\overline{\boldsymbol{h}}} \left(\sum_{i\in[\![I]\!]} \lambda_i \, \mathrm{FGW}^2_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \right) = \sum_{i\in[\![I]\!]} \lambda_i \boldsymbol{\nu}_i \tag{3.67}$$

So as for GW, we propose to solve for (3.65) using a projected (sub)gradient descent algorithm, adapting the previously detailed Algorithm 3. At each iteration, we first find an OT matrix $\boldsymbol{T}_i$ with corresponding dual potential $\widetilde{\boldsymbol{\nu}}_i$ for each FGW matching from the input graph $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$ to $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. Then we perform a projected gradient update of $\overline{\boldsymbol{C}}$, $\overline{\boldsymbol{F}}$ and $\overline{\boldsymbol{h}}$, whose respective gradients are evaluated at fixed pairs $\{(\boldsymbol{T}_i, \widetilde{\boldsymbol{\nu}}_i)\}$ by using the Envelope Theorem (Bonnans & Shapiro, 2000)

3.4 Conclusion

We introduced in this chapter, the various OT distances that can be used to operate on graphs potentially endowed with node attributes. First, the Wasserstein distances (Section 3.1) may be of use to compare graphs modeled as point clouds omitting the graph topology. Then the Gromov-Wasserstein (GW, Section 3.2) and Fused Gromov-Wasserstein (FGW, Section 3.2) distances provide an interpretable way to compare graphs while taking into account their respective structure information, and jointly with their feature information if graphs also have node features.

These latter OT problems dedicated to graphs have shown seminal promising results on Graph Representation Learning thanks to the estimation of barycentric measures, either using the GW (Section 3.2.5) or FGW (Section 3.3.4) distances (Peyré et al., 2016; Vayer et al., 2019a; Vincent-Cuaz et al., 2021). The next chapters, namely Chapters 4-5-6, aim at developing novel (F)GW based models where the computation of dedicated (F)GW target measures will be central to enhance desired properties of the learned representation, such as interpretability or discriminatory capacity.

Chapter 4

Optimal Transport distances for graphs meet Graph Neural Networks

Contents

This chapter presents the results from the paper Vincent-Cuaz et al. (2022c) which focuses on Graph Representation Learning in the context of *supervised* machine learning at the *instance-level*. From the observation of a dataset of several graphs assigned to classes (*classification task*) or values to predict (*regression task*), the goal is to learn graph representations which are discriminant with respect to these assignments.

The two most effective ways to address this problem nowadays are:

i) By designing kernels or discriminant similarity functions from distances or divergences between graphs as detailed in section 2.1.

ii) By learning in an end-to-end fashion a vector representation of a graph, coming from the aggregation of its node embeddings learned through a message-passing (see subsection 2.2) scheme which implicitly encodes the structural/topological information of the graph.

We propose in this work a novel point of view, which places distances to some learnable graph templates at the core of the graph representation. This distance embedding is constructed thanks to the Fused Gromov-Wasserstein (FGW) distance. We postulate that the vector of FGW distances to a set of template graphs can have a strong discriminative power, which is

then fed to a non-linear classifier for final predictions. Distance embedding can be seen as a new pooling layer, and can leverage on existing message passing techniques to learn sensible feature representations. Interestingly enough, in our work the optimal set of template graphs is also learnt in an end-to-end fashion by differentiating through this layer. After describing the corresponding learning procedure, we empirically validate our claim on several synthetic and real life graph classification datasets, where our method is competitive or surpasses kernel and GNN state-of-the-art approaches. We complete our experiments by an ablation study and a sensitivity analysis to parameters.

4.1 Introduction

Attributed graphs are characterized by *i)* the relationships between the nodes of the graph (structural or topological information) and *ii)* some specific features or attributes endowing the nodes themselves. Learning from those data is ubiquitous in many research areas (Battaglia et al., 2018), *e.g.* image analysis (Harchaoui & Bach, 2007; Bronstein et al., 2017), brain connectivity (Ktena et al., 2017), biological compounds (Jumper et al., 2021) or social networks (Yanardag & Vishwanathan, 2015), to name a few. Various methodologies detailed in the chapter 2 approach the inherent complexity of those data, such as signal processing (Shuman et al., 2013), Bayesian and kernel methods on graphs (Perrin et al., 2003; Ng et al., 2018; Kriege et al., 2020) or more recently Graph Neural Networks (GNN) (Wu et al., 2020) in the framework of the geometric deep learning (Bronstein et al., 2017; 2021b).

We are interested in this work in the classification of attributed graphs *at the instance level*. One existing approach consists in designing kernels that leverage topological properties of the observed graphs (Borgwardt & Kriegel, 2005; Feragen et al., 2013; Gärtner et al., 2003; Shervashidze et al., 2009) (see Section 2.1). For instance, the popular Weisfeiler-Lehman (WL) kernel (Shervashidze et al., 2011) iteratively aggregates for each node the features of its k-hop neighborhood (Section 2.1.2). Alternative approaches aim at learning vectorial representations of the graphs that can encode the graph structure (see Section 2.2). In this domain, GNN lead to state-of-the-art performances with end-to-end learnable embeddings (Wu et al., 2020). At a given layer, these architectures typically learn the node embeddings via local permutation-invariant transformations aggregating its neighbour features (Gori et al., 2005b; Maron et al., 2019b; Kipf & Welling, 2016; Hamilton et al., 2017; Xu et al., 2019c). In order to obtain a representation of the whole graph suitable for classification, GNNs finally operate a pooling (Knyazev et al., 2019; Mesquita et al., 2020) of the node embeddings, either global (*e.g* summation over nodes (Xu et al., 2019c)), or hierarchical (*e.g* by iteratively clustering nodes (Zhang et al., 2018; Ying et al., 2018; Lee et al., 2019)).

Another line of works targets the construction of meaningful distances that integrate simultaneously the structural and feature information, and that are based on optimal transport (OT) (Villani, 2009; Peyré & Cuturi, 2019), as detailed in Chapter 3. Originally designed to compare probability distributions based on a geometric notion of optimality, OT allows defining very general loss functions between various objects, modeled as probability distributions. In a nutshell, it proceeds by constructing a *coupling* between the distributions that minimizes a specific *cost*. Some approaches dealing with graphs rely on non-parametric models that first embed the graphs into a vectorial space and then match them via OT (Nikolentzos et al., 2017; Togninalli et al., 2019; Kolouri et al., 2021; Maretic et al., 2019). Recently Chen et al. (2020a) proposed the OT-GNN model, that embeds a graph as a vector of the Wasserstein distances between the nodes' embeddings (after GNN pre-processing) and learnt point clouds, acting as templates.

Building further from OT variants, the Gromov-Wasserstein (GW) distance (see Section 3.2) directly handles graphs through the symmetric matrix $\boldsymbol{C}$ that encodes the distance/similarity between each pair of nodes (*e.g.* adjacency, shortest path), and the *weight*

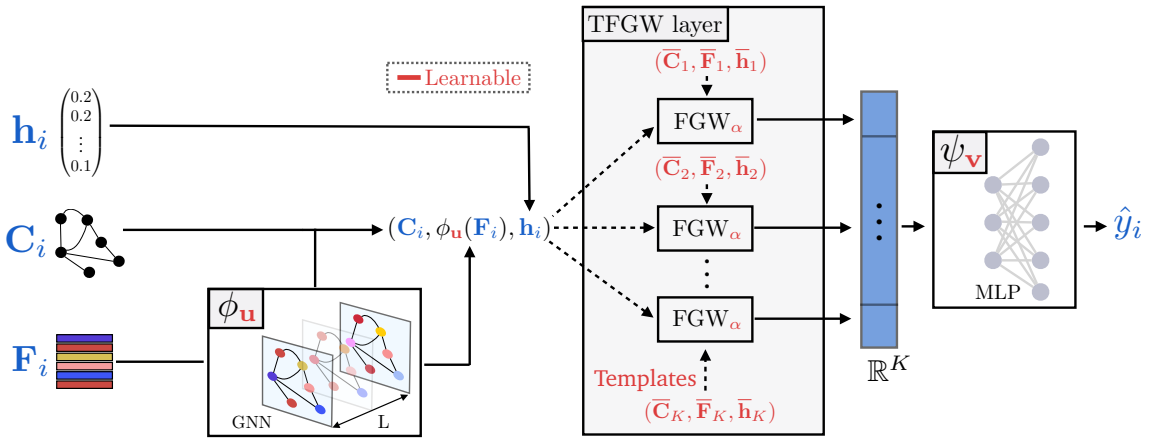

FIGURE 4.1: Illustration of the proposed model. **(left)** The input graph is represented as a triplet $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$ where the matrix $\boldsymbol{C}_i$ encodes the structure, $\boldsymbol{F}_i$ the features, $\boldsymbol{h}_i$ the nodes' weights. A GNN $\phi_{\boldsymbol{u}}$ is applied to the raw features in order to extract a meaningful node representations. **(center)** The TFGW layer is applied to the filtered graph and provides a vector representation as FGW distances to templates. **(right)** a final MLP $\psi_{\boldsymbol{v}}$ is applied to this vector in order to predict the final output of the model. All objects red in are parameters that are learned from the data.

vector $\boldsymbol{h}$, a probability mass function encoding the nodes' relative importance. GW has proven to be useful for tasks such as graph matching and partitioning (Xu et al., 2019a; Chowdhury & Needham, 2021) or unsupervised graph dictionary learning (Xu, 2020; Vincent-Cuaz et al., 2021; 2022a), which will be further addressed in the next chapters of this manuscript. GW has been also extended to directed graphs (Chowdhury & Mémoli, 2019) and to attributed graphs via the Fused Gromov-Wasserstein (FGW, Section 3.3) distance (Vayer et al., 2019a; 2020), that realizes a trade-off between an OT distance with a cost on node features and the GW distance between the similarity matrices. Despite its recent successes on complex unsupervised tasks such as graph clustering (Xu, 2020), FGW has never been explored as part of an end-to-end model for graph classification. In this work, we fill this gap by introducing a novel "layer" that embeds an attributed graph into a vector, whose coordinates are FGW distances to few (learned) graph templates. While FGW can be performed directly on raw data (*i.e.* the input structured graph without any pre-processing), we also consider the case where features representations are learnt from a GNN, similarly to OT-GNN (Chen et al., 2020a), and thus also realizing a particular type of aggregation.

Contributions. We introduce a new GNN layer, named *TFGW* for *Template-based FGW* and illustrated in the middle of Figure 4.1 (Section 4.2.1). From an input graph, it computes a vector of FGW distances to learnable graph templates. This layer can be seen as an alternative to global pooling layers and can be integrated into any neural network architecture. We detail the optimization strategy that enables learning simultaneously GNN pre-processing layers and graph templates relevant for a downstream task in an end-to-end fashion (Section 4.2.2). Then we detail its properties and the associated invariances (Section 4.2.4). We empirically demonstrate the relevance of our model in terms of performances compared to several state-of-the-art architectures. Remarkably, we show that a simple GNN model leveraging on our new layer can surpass state-of-the-art performances by a relatively large margin (Sections 4.3.1 & 4.3.2). Finally, we also provide some illustrative interpretations of our method and a sensitivity analysis of our model parameters (Sections 4.3.3 to 4.3.5).

4.2 Template based Graph Neural Network with Optimal Transport Distances

4.2.1 Model definition

Building upon the FGW distance and its properties, both detailed in Section 3.3, we propose a simple layer for a GNN that takes a graph $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ as input and computes its FGW distances to a list of K *template graphs* $\overline{\mathcal{G}} := \{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}_{k \in [\![K]\!]}$ as follows :

$$\mathrm{TFGW}_{\overline{\mathcal{G}},\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}) := \left[\mathrm{FGW}^2_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\right]_{k=1}^{K} \tag{4.1}$$

We postulate that this graph representation can be discriminant between the observed graphs. This claim relies on the theory of Balcan et al. (2008) detailed in Section 2.1.1, allowing one to learn provably strongly discriminant classifiers based on the distances from the observed graphs and templates that are sampled from the dataset. Notable instances of such learning approach can be found e.g. in Rakotomamonjy et al. (2018) using OT distances, and Johansson & Dubhashi (2015) combining the optimal assignment kernel and geometric embeddings to operate on graphs.

However such an approach often requires a large amount of templates which might be prohibitive if the distance is costly to compute. Instead, we propose to *learn* the graph templates $\overline{\mathcal{G}}$ in a supervised manner. In the same way, we also learn the trade-off parameter α from the data. As such, the TFGW layer can automatically adapt to data whose discriminating information can be discovered either in the features or in the structure of the graphs, or in a combination of the two. Moreover, the template structures can leverage on any type of input representation $\boldsymbol{C}$ since they are learnt directly from the data. Indeed, in the numerical experiments we implemented the model using either adjacency matrices (ADJ) that provide more interpretable templates (component $C_{ij} \in [0, 1]$ can be seen as a probability of link between nodes) or shortest path matrices (SP) that are more complex to interpret but encode global relations between the nodes.

The TFGW layer can be used directly as a first layer to build a graph representation feeding a fully connected network (MLP) for *e.g.* graphs classication. In order to enhance the discriminating power of the model, we propose to put a GNN (denoted by $\phi_{\boldsymbol{u}}$ and parametrized by $\boldsymbol{u}$) on top of the TFGW layer. We assume in the remainder that this GNN model $\phi_{\boldsymbol{u}}$ is injective in order to preserve isomorphism relations between graphs (see Xu et al. (2019c) and Section 2.2 for more details). With a slight abuse of notation, we write $\phi_{\boldsymbol{u}}(\mathbf{F})$ to denote the feature matrix of an observed graph after being processed by the GNN.

4.2.2 Learning problem and solver.

Learning with TFGW-GNN. We focus on a classification task where we observe a dataset $\mathcal{D}$ of I graphs $\{\mathcal{G}_i = (\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i \in [\![I]\!]}$ with variable number of nodes $\{n_i\}_{i \in [\![I]\!]}$ and where each graph is assigned to a label $y_i \in \mathcal{Y}$, with $\mathcal{Y}$ a finite set.[1] The full model is illustrated in Figure 4.1. We first process the features of the nodes of the input graphs via the GNN $\phi_{\boldsymbol{u}}$, then use the TFGW layer to represent the graphs as vectors in $\mathbb{R}^K$. Finally we use the final MLP model $\psi_{\boldsymbol{v}} : \mathbb{R}^K \to \mathcal{Y}$ parameterized by $\boldsymbol{v}$, to predict the label for any input graph. The whole model is learned in a end-to-end fashion by minimizing the cross-entropy

[1] We focused on graph classification because it is an archetypal graph-level task that is most common in the literature. However, our framework can be easily adapted to, for example, multi-label graph classification or graph regression by changing the loss function $\mathcal{L}$ and the final MLP model $\psi_{\boldsymbol{v}}$.

loss on the whole dataset leading to the following optimization problem :

$$\min_{\boldsymbol{u},\boldsymbol{v},\{(\overline{\boldsymbol{C}}_k,\overline{\boldsymbol{F}}_k,\overline{\boldsymbol{h}}_k)\},\alpha} \quad \frac{1}{I}\sum_{i=1}^{I}\mathcal{L}\left(y_i,\psi_{\boldsymbol{v}}\left(\mathrm{TFGW}_{\overline{\mathcal{G}},\alpha}\left(\boldsymbol{C}_i,\phi_{\boldsymbol{u}}(\boldsymbol{F}_i),\boldsymbol{h}_i\right)\right)\right). \tag{4.2}$$

Notable parameters of (4.2) are the template graphs in the embeddings $\{(\overline{\boldsymbol{C}}_k,\overline{\boldsymbol{F}}_k,\overline{\boldsymbol{h}}_k)\}$ and more precisely their pairwise node relationship $\overline{\boldsymbol{C}}_k$, node features $\overline{\boldsymbol{F}}_k$, and finally the distribution on the nodes $\overline{\boldsymbol{h}}_k$ on the simplex using Theorem 7. The last parameter reweighs individual nodes in each template and performs nodes selection when some weights are exactly 0 as illustrated in the computation of FGW barycenters in Section 3.3.4. Finally, the global parameter α is also learnt from the whole dataset. Although it is possible to learn a different α per template, we observed that this extra level of flexibility is prone to overfitting, so we will not consider it in the core of the experimental Section 4.3 and refer to the Annex 8.2.4 the reader interested in the results supporting this claim.

Optimization and differentiation of TFGW. We propose to solve the optimization problem in (4.2) using stochastic gradient descent. The FGW distances are computed by adapting the conditional gradient solver described in Section 3.2.4 and implemented in the POT toolbox (Flamary et al., 2021), where graphs are assumed undirected as in most graph classification problems.

The solver was designed to allow backward propagation of the gradients *w.r.t.* all the parameters of the distance and was adapted to also compute the gradient *w.r.t.* the parameter α. The gradients are obtained using the Envelop Theorem (Afriat, 1971) allowing to keep $\boldsymbol{T}^\star$ constant. We used Pytorch (Paszke et al., 2017) to implement the model. The template structure $\overline{\boldsymbol{C}}_k$, node weights $\overline{\boldsymbol{h}}_k$ and α are updated with a projected gradient respectively on the set of symmetric matrices $\mathbb{S}_{\overline{n}_k}(\mathbb{R}_+)$ ($\mathbb{S}_{\overline{n}_k}([0,1])$ when $\boldsymbol{C}_i$ are adjacency matrices), the simplex $\Sigma_{\overline{n}_k}$ and $[0,1]$. The projection onto the probability simplex of the node weights leads to sparse solutions (Condat, 2016), therefore the size of each $(\overline{\boldsymbol{C}}_k,\overline{\boldsymbol{F}}_k,\overline{\boldsymbol{h}}_k)$ can decrease along iterations hence reducing the *effective* number of their parameters to optimize. This way the numerical solver can leverage on the fact that many computations are unnecessary as soon as the weights are set to zero.

Moreover, the optimal coupling $\boldsymbol{T}^\star$ resulting from TFGW between $(\boldsymbol{C}_i,\phi_{\boldsymbol{u}}(\boldsymbol{F}_i),\boldsymbol{h}_i)$ and the template $(\overline{\boldsymbol{C}}_k,\overline{\boldsymbol{F}}_k,\overline{\boldsymbol{h}}_k)$, will encode correspondances between the nodes of the graph and the nodes of the template that will be propagated during the backward operation. The size of the inputs, the size of the templates and their respective weight $\overline{\boldsymbol{h}}_k$ will play a crucial role regarding this operation. Also note that, since the templates are estimated here to optimize a supervised task, they will promote discriminant distance embedding instead of graph reconstruction quality as proposed in other FGW unsupervised learning methods (Xu, 2020).

Note that the FGW solver from POT uses an OT solver implemented in C++ on CPU which means that it comes with some overhead (memory transfer between GPU and CPU) when training the model on GPU. Still the multiple FGW distances computation has been implemented in parallel on CPU with a computational time that remains reasonable in practice (see Section 4.3). While a GPU solver can be found when using entropy regularized FGW, it introduces a new parameter related to the regularization strength which is more cumbersome to set, and that we did not consider in the experiments. We refer the reader to Section 3.2.4 for a more detailed discussion about these computational considerations.

4.2.3 TFGW as a generic OT based model.

One can observe that through the interpolation properties of the FGW distance regarding the trade-off parameter α (see Theorem 3.1 in Vayer et al. (2019a) and Section 3.3.2), the

model and its learning procedure described above can be adapted to the Wasserstein (resp. Gromov-Wasserstein) distance template-based embedding by fixing $\alpha = 0$ (resp. $\alpha = 1$).

Wasserstein template-based pooling. As GNN comes with node embedding schemes which takes *implicitly* into account the graph structure through message-passing (Section 2.2.1), a template based Wasserstein pooling (denoted TW) might be relevant to discriminate point clouds outputed by GNNs $\phi_{\boldsymbol{u}}$, while operating on any graph. Notably, Chen et al. (2020a) proposed the OT-GNN model which is exactly an instance of our TFGW model while fixing $\alpha = 0$. Defining by $\overline{\mathcal{F}}$ the set of point cloud templates $\{(\overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}_{k \in [\![K]\!]}$, the learning problem stated for TFGW in equation (4.2) becomes for TW the following one:

$$\min_{\boldsymbol{u}, \boldsymbol{v}, \{(\overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}} \quad \frac{1}{I} \sum_{i=1}^{I} \mathcal{L}\left(y_i, \psi_{\boldsymbol{v}}\left(\mathrm{TW}_{\overline{\mathcal{F}}}(\phi_{\boldsymbol{u}}(\boldsymbol{F}_i), \boldsymbol{h}_i)\right)\right). \tag{4.3}$$

This approach mostly relies on the ability of the GNN pre-processing $\phi_{\boldsymbol{u}}$ to produce discriminant node features aware of the graph structure. On the contrary, as the graph structure is explicitly provided to the TFGW based templates, it may help the GNN pre-processing to focus on producing discriminant node embeddings. Both approaches will be compared extensively in the experimental section.

Gromov-Wasserstein template-based pooling. Adopting the Gromov-Wasserstein distance within our distance embedding instead of Fused Gromov-Wasserstein could also be of interest. Most likely, it would be relevant for graphs without node features and would lead to an end-to-end model where the pre-processing step achieved by $\phi_{\boldsymbol{u}}$ is omitted.

However, as illustrated in many works on graph kernels and GNNs, the common practice is to augment such graphs with features encoding node properties, such as degree information (*e.g* normalized degree distribution, one-hot encoding of node degree) or node centrality measures (Saxena & Iyengar, 2020). Once graphs are augmented, most GNN approaches can be integrated to embed node features and therefore advocates for the use of our TFGW layer as the resulting graph representation is an attributed graph.

One unexplored way to reduce the number of templates' learnable parameters by omitting their node features, while learning on attributed graphs, would be to pre-process graphs using Structure Learning (Li et al., 2018; Jiang et al., 2019; Zhang et al., 2019b) approaches. This paradigm is for now mostly studied for specific variant of GNN based on graph hierarchical pooling (see Section 2.2). Structure Learning consists in learning node pairwise similarities resulting from both the node features and the initial graph structure. More formally, from an attributed input graph $(\boldsymbol{C}_i, \boldsymbol{F}_i)$, it consists in an application $\phi_{\boldsymbol{u}}$ applied in $(\boldsymbol{C}_i, \boldsymbol{F}_i)$ outputing an embedded structure $\widetilde{\boldsymbol{C}}_i = \phi_{\boldsymbol{u}}(\boldsymbol{C}_i, \boldsymbol{F}_i)$. Hence, once the graphs $\{(\boldsymbol{C}_i, \boldsymbol{F}_i)\}_i$ are pre-processed by $\phi_{\boldsymbol{u}}$, one could aim at classifying them based on the Gromov-Wasserstein distances from their embedding $\widetilde{\boldsymbol{C}}_i$ to learnable graph templates.

4.2.4 Model properties

Invariances of the TFGW layer. We now discuss invariance properties of the proposed layer resulting from the properties of FGW (see Section 3.3). Let us start with the following lemma whose proof can be found in Annex 8.2.3,

Lemma 3 (TFGW invariant 1) *The* TFGW *embeddings are invariant to strong isomorphism.*

This lemma directly stems from the fact that FGW is invariant to strong isomorphism of one of its inputs, without any assumption on the involved structure matrices. This proposition

implies that two graphs with any aforementioned representation which only differ by a permutation of the nodes will share the same TFGW embedding.

Since as the more general concept of weak isomorphism between graphs intervene rarely in such scenarios, Lemma 3 stating the invariance to strong isomorphism of the TFGW may be enough for most real-world graph classification datasets. The invariance to weak isomorphism might be needed when graphs have nodes which are exact duplicates from each other. These considerations can be found in the proofs of (Chowdhury & Mémoli, 2019, Theorem 16) in the restrained context of unattributed networks using the Gromov-Wasserstein distance. Moreover they were emphasized in (Kerdoncuff et al., 2022, Theorem 1, Definition 1) which introduced a notion of "canonical representations" which amounts to merging the nodes which are exact duplicates and aggregate their respective masses. For the sake of completeness, we discussed these considerations for attributed networks in Sections 3.2-3.3, and we extend here Lemma 3 to any graph or network, in its most generic formulation as defined in Chowdhury & Mémoli (2019), further endowed with node attributes:

Lemma 4 (TFGW invariant 2) *The* TFGW *embeddings are invariant to weak isomorphism.*

Lemma 4 shows that two attributed graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with any aforementioned representation which are *weakly isomorphic* will share the same TFGW embedding. Moreover, the *canonical representation* $\mathcal{G}_c$ of $\mathcal{G}_1$ and $\mathcal{G}_2$, which does not admit any duplicate node and is unique up to any permutation of its nodes, will also share the same TFGW embedding. As this relation of weak isomorphism is transitive and consists in splitting/aggregating masses of duplicates nodes, we can first remark that there exists an infinite number of graphs weakly isomorphic to $\mathcal{G}_c$ which share the same TFGW embedding. Second, as strongly isomorphic graphs are consequently weakly isomorphic Lemma 3 can be seen as a corollary of Lemma 4.

The invariance property stated in Lemma 4 holds for any mapping ϕ_u, in particular for an injective one, such as a Multi-Layer Perceptron (MLP) (Hornik et al., 1989) or any GNN with a sum aggregation scheme as described in Xu et al. (2019c) and as considered in our end-to-end model presented in Subsection 4.2.2.

Perspectives. One primary concern of newly designed GNN is their *expressivity* (Balcilar et al., 2021; Sato, 2020), or in other words, their ability to distinguish any *non-isomorphic* graphs (see detailed discussion in Section 2.2). Accessing the expressivity of our TFGW model consists in studying whether the converse of Lemma 4 holds true or not, *i.e* if two graphs have the same TFGW embedding, are they isomorphic ? We let for future works this theoretical question involving advanced concept of geometry and aim at addressing this question *empirically* in Section 4.3.1.

Another recent concern emerging from the GNN literature relates to the generalization abilities of these models (Knyazev et al., 2019; Garg et al., 2020). Also this point is addressed *empirically* in Subsection 4.3.2.

Finally, it would be of interest to study whether our TFGW model satisfies a new kind of universal approximation of functions (Hornik et al., 1989; Lu & Lu, 2020) defined directly on graphs. Such property would check whether our class of models can approximate arbitrarily well any real function defined on the product space of attributed graphs (see Section 3.3), *e.g* the unknown function which assigns a graph from a dataset to its true label (Cybenko, 1989). Although the notion of universality does not indicate how easy it is in practice to learn the correct function, it can at least guarantee the absence of a fundamental bottleneck of our model. We let this study, which may be well-guided by the seminal work of Brüel Gabrielsson (2020), for future works.

However, there exists encouraging results about universal approximation of functions defined *on point clouds* of OT-GNN Chen et al. (2020a), being a specific instance of our TFGW model. Indeed, authors showed that the standard architecture, *i.e* using sum pooling (see Section 2.2), lacks a fundamental property of *universal approximation of functions* defined on point clouds, and that their template-based Wasserstein pooling overcomes this limitation. To state these results more formally, let us recall a characterization of universality. We consider in the following, a kernel $k : \mathcal{X}^2 \to \mathbb{R}$ on a set $\mathcal{X}$ in its generic form, without any assumption on its positivity or definiteness (see Subsection 2.1). We denote by $\mathcal{X}_d^n$ the set of point clouds $\boldsymbol{X} = \{x_i\}_{i\in[\![n]\!]}$ of size n of $\mathbb{R}^d$.

Definition 5 (Universal kernels) *A kernel k defined on $\mathcal{X}_d^n$ is said to be* universal *if for any compact subset $\mathcal{X} \subset \mathcal{X}_d^n$, the set of functions of the form $\{\sum_{j\in[\![m]\!]} \alpha_j \sigma(k(\cdot, \theta_j) + \beta_j) | \forall m \in \mathbb{N}^*, (\alpha_j, \beta_j) \in \mathbb{R}^2, \theta_j \in \mathcal{X}_d^n\}$ [2] is dense in the set of continuous functions from $\mathcal{X}$ to $\mathbb{R}$, w.r.t the supremum norm defined on $\mathcal{X}$.*

Theorem 8 (Universality properties Chen et al. (2020a)) *On $\mathcal{X}_d^n$,*

i) The aggregation kernel defined for $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{X}_d^n$ by $agg(\boldsymbol{X}, \boldsymbol{Y}) = \langle \sum_i x_i, \sum_j y_j \rangle$ is not universal.

ii) The Wasserstein distance defines an universal *kernel.*

Sketch of the proof of Theorem 8: i) Consider a function $f \in \mathcal{C}(\mathcal{X})$ such that there exists $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{X}$, such that $f(\boldsymbol{X}) \neq f(\boldsymbol{Y})$ and $\sum_i x_i = \sum_j y_j$. Then any function of the form $\sum_i \alpha_i \sigma(agg(\boldsymbol{W}_i, \cdot) + \theta_i)$ would take equal values on $\boldsymbol{X}$ and $\boldsymbol{Y}$ and hence would not approximate f arbitrarily well. ii) The proof is inspired from the proof of universality of neural networks from Cybenko (1989). It consists in using that σ is discriminatory w.r.t to a kernel (Cybenko, 1989, Definition 1) is a sufficient condition for the kernel universality (Cybenko, 1989, Theorem 1). Then it remains to prove that σ is discriminatory w.r.t the Wasserstein distance (Chen et al., 2020a, Lemma 2) □

Theorem 8 states that a priori the template-based Wasserstein model is more suitable for graphs classification than other GNN models using sum or mean pooling. Nevertheless, OT-GNN can only distinguish graphs that have distinct multi-sets of node embeddings, e.g. all Weisfeiler-Lehman distinguishable graphs while using GNN as node pre-processing. To empirically study these notions of universality, generalization and expressiveness, we will extensively compare the various template-based distance (FGW or Wasserstein) pooling with sum pooling in the following numerical experiments using similar node embedding schemes $\phi_{\boldsymbol{u}}$.

4.3 Experimental results

This section aims at illustrating the performances of our approach for graph classification in synthetic and real-world datasets. First, we showcase the relevance of our TFGW layer on existing synthetic datasets known to require expressiveness beyond the 1-WL test (Section 4.3.1). Then we benchmark our model with state-of-the-art approaches on well-known real-world datasets (Section 4.3.2). We finally discuss our results through a sensitivity analysis of our models (Sections 4.3.3 to 4.3.5).

[2] Here σ is the sigmoid function defined as $\sigma(x) = \left(1 + e^{-x}\right)^{-1}$

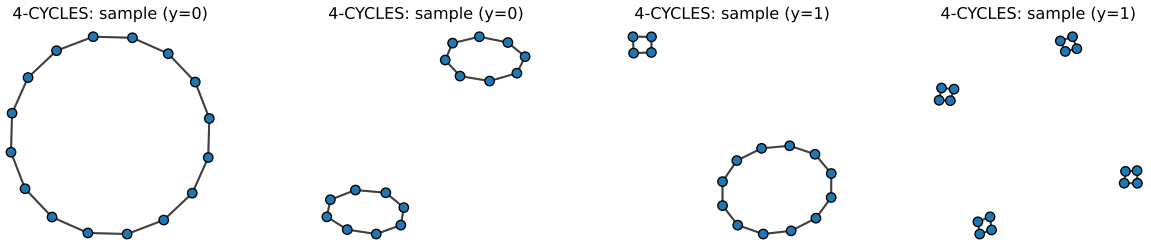

FIGURE 4.2: Few samples with different labels $y \in \{0, 1\}$ from the dataset 4-CYCLES.

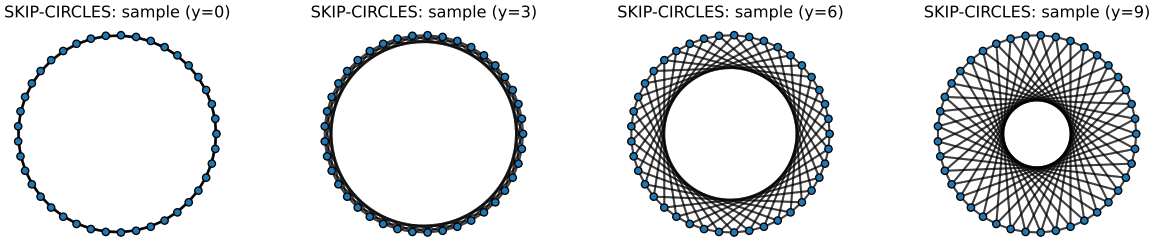

FIGURE 4.3: Unique sample from different labels $y \in \{0, 3, 6, 9\}$ corresponding respectively to $\{2, 5, 11, 16\}$ hops from the dataset SKIP-CIRCLES.

4.3.1 Analysis of the expressiveness of template-based OT models

Synthetic datasets beyond WL test. Identification of graphs beyond the WL test is one important challenge faced by the GNN community. In order to test the ability of TFGW to handle such fundamentally difficult problems we consider two synthetic datasets: 4-CYCLES (Loukas, 2020; Papp et al., 2021) contains graphs with (possibly) disconnected cycles where the label y_i is the presence of a cycle of length 4; SKIP-CIRCLES (Chen et al., 2019) contains circular graphs with skip links and the labels (10 classes) are the lengths of the skip links among $\{2, 3, 4, 5, 6, 9, 11, 12, 13, 16\}$.

Benchmarked methods. We compare the performances of the TFGW layer for embedding such graphs with various kinds of GNN[3]:

i) GIN with sum pooling (Xu et al., 2019c) as baseline model proved to be at least as expressive as the 1-WL test.

ii) GIN independently coupled with 3 augmentation techniques, which consist in adding features to nodes or edges to make nodes with similar neighborhoods distinguishable, provably enhancing GIN's expressiveness: GIN-port adding port numbers on edges (Sato et al., 2019), GIN-id adding unique IDs for nodes (Sato et al., 2021), GIN-rf adding random features on nodes (Loukas, 2020).

iii) DropGIN (Papp et al., 2021) which for a given graph $\mathcal{G}_i$ drops out nodes randomly several times forming multiple perturbed versions of $\mathcal{G}_i$ independently processed by GIN and averaged to produce a finale representation of $\mathcal{G}_i$.

iv) OT-GNN (Chen et al., 2020a) for which the same GIN architecture is considered except for the pooling step, and the same number of templates than for TFGW is considered (see below).

[3]More detailed description of most of these models and analysis of their expressiveness can be found in Section 2.2.1

TABLE 4.1: Accuracy on synthetic datasets beyond 1-WL tests averaged over 10 different runs. First and second best methods highlighted in bold. We additionally report for each method the numerical complexity to predict the label of a graph with n nodes whose maximum node degree is denoted d_{max}.

model	complexity	4-CYCLES	SKIP-CIRCLES
TFGW (ours)	$0(n^2\overline{n}_{max})$	**0.99(0.03)**	**1.00(0.00)**
TFGW-fix (ours)	$0(n^2\overline{n}_{max})$	0.63(0.11)	**1.00(0.00)**
OT-GNN	$0(n^2\overline{n}_{max})$	0.50(0.00)	0.10(0.00)
GIN	$0(nd_{max})$	0.50(0.00)	0.10(0.00)
GIN-port	$0(nd_{max})$	0.84(0.07)	0.14(0.08)
GIN-id	$0(nd_{max})$	0.58(0.07)	0.10(0.09)
GIN-rf	$0(nd_{max})$	0.77(0.05)	0.16(0.05)
DropGIN	$0(rnd_{max})$	**1.00(0.01)**	0.82(0.28)
PPGN	$0(n^4)$	**1.00(0.01)**	**0.90(0.11)**

v) PPGN (Maron et al., 2019a) that interleaves applications of standard MLP applied to the feature dimension and matrix multiplication to be provably as expressive as a 3-WL test.

We build upon the benchmark of Papp et al. (2021) by considering for all methods except PPGN and TFGW, 4 GIN layers for 4-CYCLES, and 9 GIN layers for SKIP-CIRCLES as the skip links can form cycles of up to 17 hops. For PPGN we kept the best performing model among its 3 standard architectures fixing the learning rate to 0.001 and its decay to 0.5. Since the graphs do not have features we use directly the TFGW on the raw graph representation with $\alpha = 1$ hence computing only the GW distance. The GNN methods above artificially add a feature equal to 1 on all nodes as they have the same degree. For these experiments we use adjacency matrices for $\boldsymbol{C}_i$ and we investigate two flavours of TFGW:

1) In TFGW-fix we fix the templates by sampling *one template per class* from the training dataset (this can be seen as a simpler FGW feature extraction).

2) For TFGW we learn the templates from the training data (as many as the number of classes) as proposed in the previous sections.

Results. Performances quantified by accuracy are averaged over 10 runs and reported in Table 4.1. TFGW based methods perform very well on both datasets with impressive results on SKIP-CIRCLE when GNN have limited performances. This is due to the fact that different samples from one class of SKIP-CIRCLE are generated by permuting nodes of the same graph and FGW distances are invariant to these permutations, so setting as many templates as classes for this task makes the problem quite easy to handle for TFGW. 4-CYCLES has a more complex structure with intra-class heterogeneity and requires more than two templates to perform as good as DropGIN. To illustrate this we have computed the accuracy on this dataset as a function of the number of templates K in Figure 4.4. We can see that a perfect classification is reached up to $K = 4$ for TFGW, while TFGW-fix still struggles to generalize at $K = 20$. This illustrates that learning the templates is essential to keep K (and numerical complexity) small while ensuring good performances.

Stress test on the number of templates. To further emphasize the discriminative power of the TFGW embeddings, we report here additional experiments conducted on the SKIP-CIRCLES simulated datasets. For adjacency (ADJ) and shortest-path (SP) matrices

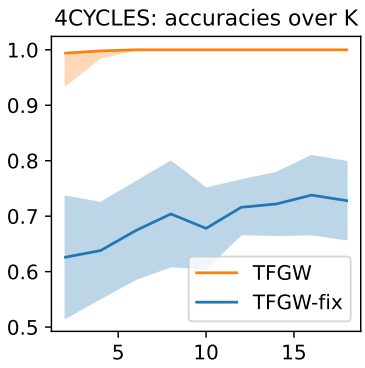

FIGURE 4.4: Test accuracy distributions by number of templates either fixed or learned.

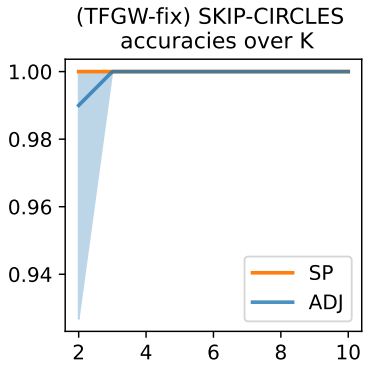

FIGURE 4.5: Test accuracies on SKIP-CIRCLES of TFGW-fix for $K \in \{2, ..., 10\}$.

as $\boldsymbol{C}_i$, instead of using one template per class ($K = 10$), we stress the TFGW-fix models by learning on $K \in \{2, ..., 10\}$ fixed templates sampled from the dataset. We report in Figure 4.5, the test accuracies averaged over 10 simulations with the same other settings than for the previously reported experiments. The averaged accuracies are illustrated in bold, while the intervals between the minimum and the maximum accuracy across runs is illustrated with a lower intensity. We can see that both methods perfectly distinguish the classes using at most 3 templates. Moreover only 2 suffice to achieve such performance using SP matrices, which is not the case for ADJ matrices. As these toy experiments seem to highlight the importance of a well-chosen input representation, both ADJ and SP will be considered in the next numerical experiments (Subsections 4.3.2 - 4.3.5).

4.3.2 Graph classification benchmark

We now evaluate and compare the performances of our TFGW GNN with a number of state-of-the-art graph classifiers, from kernel methods to GNN. The numerical experiments are conducted on real life graph datasets to provide a fair benchmark of all methods on several heterogeneous graph structures.

Datasets. We use 8 well-known graph classification datasets (Kersting et al., 2016): 5 bioinformatics datasets among which 3 have discrete node features (MUTAG, PTC, NCI1 (Kriege & Mutzel, 2012; Shervashidze et al., 2011)) and 2 have continuous node features (ENZYMES, PROTEINS (Borgwardt et al., 2005)) and 3 social network datasets (COLLAB, IMDB-B, IDBM-M (Yanardag & Vishwanathan, 2015)). In order to analyse them with all methods, we augment unattributed graphs from social networks with node degree features. Detailed description and statistics on these datasets are reported in Table 8.1 of the Annex.

Baselines. We benchmark our approaches to the following state-of-the-art baselines for graphs classification, split into 3 categories:

i) *kernel based approaches*, including FGW (Vayer et al., 2019a) operating on adjacency and shortest-path matrices, the WL subtree kernel (Shervashidze et al., 2011, WL) and the Wasserstein WL kernel (Togninalli et al., 2019, WWL).

ii) *OT based representation learning* models, including WEGL (Kolouri et al., 2021) and OT-GNN (Chen et al., 2020a).

TABLE 4.2: Test set classification accuracies from 10-fold CV. The first (resp. second) best performing method is highlighted in bold (resp. underlined).

category	model	MUTAG	PTC	ENZYMES	PROTEIN	NCI1	IMDB-B	IMDB-M	COLLAB
Ours	TFGW ADJ (L=2)	**96.4(3.3)**	**72.4(5.7)**	73.8(4.6)	**82.9(2.7)**	**88.1(2.5)**	**78.3(3.7)**	**56.8(3.1)**	**84.3(2.6)**
	TFGW SP (L=2)	94.8(3.5)	70.8(6.3)	**75.1(5.0)**	82.0(3.0)	86.1(2.7)	74.1(5.4)	54.9(3.9)	80.9(3.1)
OT emb.	OT-GNN (L=2)	91.6(4.6)	68.0(7.5)	66.9(3.8)	76.6(4.0)	82.9(2.1)	67.5(3.5)	52.1(3.0)	80.7(2.9)
	OT-GNN (L=4)	92.1(3.7)	65.4(9.6)	67.3(4.3)	78.0(5.1)	83.6(2.5)	69.1(4.4)	51.9(2.8)	81.1(2.5)
	WEGL	91.0(3.4)	66.0(2.4)	60.0(2.8)	73.7(1.9)	75.5(1.4)	66.4(2.1)	50.3(1.0)	79.6(0.5)
GNN	PATCHYSAN	91.6(4.6)	58.9(3.7)	55.9(4.5)	75.1(3.3)	76.9(2.3)	62.9(3.9)	45.9(2.5)	73.1(2.7)
	GIN	90.1(4.4)	63.1(3.9)	62.2(3.6)	76.2(2.8)	82.2(0.8)	64.3(3.1)	50.9(1.7)	79.3(1.7)
	DropGIN	89.8(6.2)	62.3(6.8)	65.8(2.7)	76.9(4.3)	81.9(2.5)	66.3(4.5)	51.6(3.2)	80.1(2.8)
	PPGN	90.4(5.6)	65.6(6.0)	66.9(4.3)	77.1(4.0)	82.7(1.8)	67.2(4.1)	51.3(2.8)	81.0(2.1)
	DIFFPOOL	86.1(2.0)	45.0(5.2)	61.0(3.1)	71.7(1.4)	80.9(0.7)	61.1(2.0)	45.8(1.4)	80.8(1.6)
Kernels	FGW - ADJ	82.6(7.2)	55.3(8.0)	72.2(4.0)	72.4(4.7)	74.4(2.1)	70.8(3.6)	48.9(3.9)	80.6(1.5)
	FGW - SP	84.4(7.3)	55.5(7.0)	70.5(6.2)	74.3(3.3)	72.8(1.5)	65.0(4.7)	47.8(3.8)	77.8(2.4)
	WL	87.4(5.4)	56.0(3.9)	69.5(3.2)	74.4(2.6)	85.6(1.2)	67.5(4.0)	48.5(4.2)	78.5(1.7)
	WWL	86.3(7.9)	52.6(6.8)	71.4(5.1)	73.1(1.4)	85.7(0.8)	71.6(3.8)	52.6(3.0)	81.4(2.1)
	Gain with TFGW	**4.3**	**4.4**	**2.9**	**4.9**	**2.4**	**5.3**	**4.2**	**2.9**

iii) *GNN models*, with global or more sophisticated pooling operations, including PATCHY-SAN (Niepert et al., 2016), DIFFPOOL (Ying et al., 2018), PPGN (Maron et al., 2019a), GIN (Xu et al., 2019c) and its augmented version through structure perturbations DropGIN (Papp et al., 2021).

For methods (i) that do not require a stopping criterion dependent on a validation set, we report results using for parameter validation a 10-fold nested cross-validation (Vayer et al., 2019a; Kriege et al., 2020) repeated 10 times. For methods (ii) - (iii), we adopt the hyper-parameters suggested in the respective papers, but with a slightly different model selection scheme, as detailed in the next paragraph.

Benchmark settings. Recent GNN literature (Maron et al., 2019b; Xu et al., 2019c; Maron et al., 2019a; Papp et al., 2021) successfully addressed many limitations in terms of model expressiveness compared to the WL tests. Within that scope, they suggested to benchmark their models using a 10-fold cross-validation (CV) where the best average accuracy on the validation folds was reported. We suggest here to quantify the generalization capacities of GNN based models by performing a 10-fold cross validation, but measuring performances on a holdout test set never seen during training. For each split, we track the accuracy on the validation fold every 5 epochs, then the model whose parameters maximize that accuracy is retained. Finally, the model used to predict on the holdout test set is the one with maximal validation accuracy averaged across all folds. This setting is more realistic than a simple 10-fold CV and allows a better understanding of the generalization performances (Bengio & Grandvalet, 2003). This point explains why some existing approaches have here different performances than those reported in their original paper.

For all the TFGW based approaches we empirically study the impact of the input structure representation by considering adjacency (ADJ) and shortest-path (SP) matrices $\boldsymbol{C}_i$. For all template based models, we set the size of the templates to the median size of the observed graphs.

We validate the number of templates K in $\{\beta|\mathcal{Y}|\}_\beta$, with $\beta \in \{2, 4, 6, 8\}$ and $|\mathcal{Y}|$ the number of classes. Only for ENZYMES with 6 classes of 100 graphs each, we validate $\beta \in \{1, 2, 3, 4\}$. All parameters of our TFGW layers highlighted in red in Figure 4.1 are learned while $\phi_{\boldsymbol{u}}$ is a GIN architecture (Xu et al., 2019c) composed of $L = 2$ layers aggregated using the Jumping Knowledge scheme (Xu et al., 2018) known to prevent overfitting in global pooling frameworks. For OT-GNN we validate the number of GIN layers in $L \in \{2, 4\}$. Finally, we validate the number of hidden units within the GNN layers and the application of dropout on the final MLP for predictions, similarly to GIN and DropGIN.

TABLE 4.3: Number of parameters and averaged prediction time per graph in milliseconds when running on CPU and GPU.

model	PTC			PROTEIN		
	parameters	CPU runtimes	GPU runtimes	parameters	CPU runtimes	GPU runtimes
(ours) TFGW	25.1k	12.1	20.7	47.7k	45.9	21.8
OT-GNN	30.8k	7.6	8.8	95.2k	27.1	11.1
GIN	29.9k	0.19	0.06	45.9k	2.5	0.09
DropGIN	44.1k	14.3	2.1	45.9k	79.8	4.7

Results. The results of the comparisons in terms of accuracy are reported in Table 4.2. Our TFGW approach consistently *outperforms with significant margins* the state-of-the-art approaches from all categories. Even if most of the benchmarked models can perfectly fit the train sets by learning implicitly the graphs structure (Xu et al., 2019c; Papp et al., 2021; Maron et al., 2019a), enforcing such knowledge explicitly as our TFGW layer does (through FGW distances) leads to considerably stronger generalization performances. On 7 out of 8 datasets, TFGW leads to better performances while operating on adjacency matrices (TFGW ADJ) than on shortest-path ones (TFGW SP). Interestingly, this ranking with respect to those input representations does not necessarily match the one of the FGW kernel which extracts knowledge from the graph structures $\boldsymbol{C}_i$ through FGW, as our TFGW layer. These different dependencies to the provided inputs may be due to the GNN pre-processing of node features which suggests the study of its ablation. Finally to complete this analysis, we report in Table 4.3 the number of parameters of best selected models across various methods, for the datasets PTC and PROTEIN. Our TFGW leads to considerably better classification performances while having comparable number of parameters than these competitors. We also reported for these models, their averaged prediction time per graph. These measures were taken both on CPUs (Intel Core i9-9900K CPU, 3.60 GHz) and a GPU Quatro RTX 4000, in order to fairly compare the numerical complexity of these methods, as OT solver used in TFGW and OT-GNN are currently limited to CPUs (see the discussion in Subsection 4.2.2). Although the theoretical complexity of our approach is *at most* cubic in the number of nodes, we still get in practice a fairly good speed for classifying graphs, in comparison to the other competitive methods. More precisely, TFGW has approximately the same or lower averaged prediction time per graph on CPU as recent DropGIN architecture. However, GIN and DropGIN get a 3-30 times speedup on GPU when TFGW is slower on GPU for PTC and 2x faster for PROTEIN (acceleration of matrix product for large graphs but overhead time for transfering between CPU and GPU for small graphs). This shows that TFGW is not yet accelerated on GPU but remains reasonable in practice, so one can still benefit from it in the GNN and MLP models.

4.3.3 TFGW-GIN: Ablation study and embedding visualization

In this Section we inspect the role of some of the model parameters (α in FGW, weights estimation in the templates, depth of the GNN $\phi_{\boldsymbol{u}}$) in terms of the classification performance. To this end, we first conduct on all datasets an ablation study on the graph template weights $\overline{\boldsymbol{h}}_k$ and the number of GIN layers in $\phi_{\boldsymbol{u}}$.

Ablation Study. Following the same procedure as in Section 4.3.2, we benchmark the following settings for our TFGW models: for adjacency (ADJ) and shortest path (SP) representations $\boldsymbol{C}_i$, we learn the distance layers either directly, on the raw data (*i.e.* $L = 0, \phi_{\mathbf{u}} = \mathrm{id}$), or after embedding the data with ($L = 1$) or ($L = 2$) GIN layers. For $L = 0$ we either fix the graph template weights $\overline{\boldsymbol{h}}_k$ uniformly or learn them.

TABLE 4.4: Classification results from 10-fold cross-validation of our TFGW models in various scenarios: for $L \in \{0, 1, 2\}$ GIN layers, we either fix templates weights $\overline{\boldsymbol{h}}_k$ to uniform distributions or learn them. The first and second best performing method are respectively highlighted in bold and underlined.

model	inputs	$\overline{\boldsymbol{h}}_k$	MUTAG	PTC	ENZYMES	PROTEIN	NCI1	IMDB-B	IMDB-M	COLLAB
TFGW (L=0)	ADJ	uniform	92.1(4.5)	63.6(5.0)	67.4(7.3)	78.0(2.0)	80.3(1.5)	69.9(2.5)	49.7(4.1)	78.7(3.1)
	ADJ	learnt	94.2(3.0)	64.9(4.1)	72.1(5.5)	78.8(2.2)	82.1(2.5)	71.3(4.3)	52.3(2.5)	80.9(2.7)
	SP	uniform	94.8(3.7)	66.5(6.7)	72.7(6.9)	77.5(2.4)	79.6(3.7)	68.1(4.4)	48.3(3.6)	78.4(3.4)
	SP	learnt	95.9(4.1)	67.9(5.8)	**75.1(5.6)**	79.5(2.9)	83.9(2.0)	72.6(3.1)	53.1(2.5)	79.8(2.5)
TFGW(L=1)	ADJ	learnt	94.8(3.1)	68.7(5.8)	72.7(5.1)	81.5(2.8)	85.4(2.8)	76.3(4.3)	55.9(2.4)	82.6(1.8)
	SP	learnt	95.4(3.5)	70.9(5.5)	74.9(4.8)	82.1(3.4)	85.7(3.1)	73.8(4.8)	54.2(3.3)	81.1(2.5)
TFGW (L=2)	ADJ	learnt	**96.4(3.3)**	**72.4(5.7)**	73.8(4.6)	**82.9(2.7)**	**88.1(2.5)**	**78.3(3.7)**	**56.8(3.1)**	**84.3(2.6)**
	SP	learnt	94.8(3.5)	70.8(6.3)	**75.1(5.0)**	82.0(3.0)	86.1(2.7)	74.1(5.4)	54.9(3.9)	80.9(3.1)

The results (test accuracy) are reported in Table 4.4. Learning the weights systematically improves the generalization capabilities of our models of at least 1% or 2% for both ADJ and SP graph representations. For a given number of graph templates, the weights learning allows to better fit the specificities of the classes (e.g. varying proportion of nodes in different parts of the graphs). Moreover, as weights can become sparse in the simplex during training they also allow the model to have templates whose number of nodes adapts to the classification objective, while bringing computational benefits as discussed in Section 4.2.2. Those observations explain why we learnt them by default in the benchmark of the previous Section.

Next, we see in Table 4.4 that using GNN layers as a pre-processing for our TFGW layer enhances generalization powers of our models, whose best performances are obtained for $L = 2$. Interestingly, for $L = 0$, TFGW with SP matrices outperforms TFGW with ADJ matrices, meaning that the shortest path distance brings more discriminant information on raw data. But when $L \geq 1$ (*i.e.* when a GNN pre-processes the node features), TFGW with ADJ matrices improves the accuracy. An explanation could be that the GNN $\phi_{\boldsymbol{u}}$ can somehow replicate (and outperform) a SP metric between nodes. This emphasizes that the strength of our approach clearly exhibited in Table 4.2 lies in the inductive bias of our FGW distance embedding. This last point is further reinforced by additional experiments reported in Section 4.3.5 where GIN layers (used by default for our TFGW model) are replaced by GAT layers from Veličković et al. (2018) (see also Section 2.2.1, equation (2.9)).

Visualizing the TFGW embedding. In order to interpret the TFGW embeddings, we first illustrate in Figure 4.6 the PCA projection of our distance embeddings learned on PTC with $L = 0$ and $L = 2$ and the number of templates K varying in $\{4, 8\}$.

For this experiment, we have chosen PCA because it allows to have a more interpretable low dimensional projection that preserves the geometry compared to local neighbourhood based embeddings such as TSNE (Van der Maaten & Hinton, 2008) or UMAP (McInnes et al., 2018). As depicted in the figure, the learned templates are extreme points in the embedding space of the PCA. This result is particularly interesting because existing unsupervised FGW representation learning methods tend toward estimating templates that belong to the data manifold, or to form a "convex enveloppe" of the data to ensure good reconstruction (Xu, 2020) (see Chapters 5 and 6 for this matter). On the contrary, the templates learned through our approach seem to be located on a plane in the PCA space while the samples evolve orthogonally to this plane (when the FGW distance increases). In a classification context, this means that the learned templates will not actually represent realistic graphs from the data but might encode "exaggerated" or "extreme" features in order to maximize the margin between classes in the embedding. An instance of such learned templates are illustrated in Figure 4.7. By comparing these templates (on the left) with samples from the dataset (on

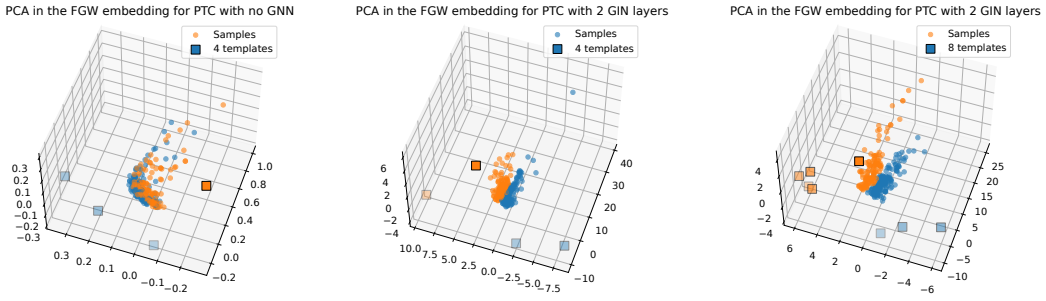

FIGURE 4.6: PCA projections of the template based embeddings for different models and number of templates.

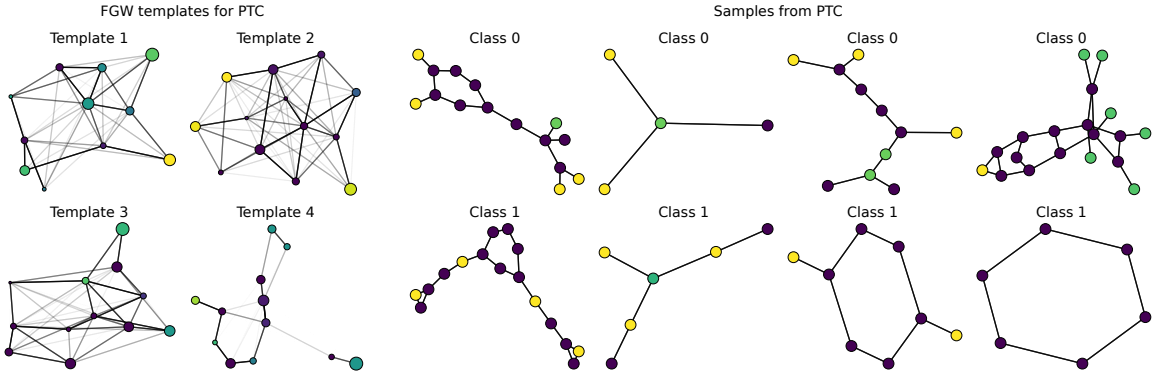

FIGURE 4.7: Illustration of the templates learned on PTC with $K = 4$ and $L = 0$ (on the left side), and some samples from this dataset (on the right sight). The graph structures are represented using the entries of $\overline{\boldsymbol{C}}_k$ (resp. $\boldsymbol{C}_i$) as repulsive strength and the corresponding edges are colored in shades of grey (black being the maximum). The node colors are computed based on their features $\overline{\boldsymbol{F}}_k$ (resp. $\boldsymbol{F}_i$). The nodes size are made proportional to the weights $\overline{\boldsymbol{h}}_k$ (resp. $\boldsymbol{h}_i$).

the right), we can clearly see that the learned templates do not represent realistic graphs from the data.

Finally, in Figure 4.7, the samples are coloured *w.r.t.* their class and the templates are coloured by their predicted class. Interestingly, the classes are already well separated with 4 templates but the separation is clearly non-linear whereas using GNN pre-processing and a larger number of templates leads to a linear separation of the two classes.

4.3.4 TFGW-GIN: Sensitivity analysis

In this section, we take a closer look at the estimated trade-off parameters α and provide a sensitivity analysis *w.r.t.* the number of templates and the number of GIN layers for our TFGW models.

Importance of the structure/feature aspect of FGW. To the best of our knowledge we are the first to actually learn the trade-off parameter α of FGW in a supervised way. For this matter, we verify that our models did not converge to degenerated solutions where either the structure ($\alpha = 0$) or the features ($\alpha = 1$) are omitted.

To this end we report in Figure 4.8 the distributions of the estimated α for some models learnt on datasets PTC and IMDB-B, where features are respectively existing in the dataset or created using node degrees. We can see that for both kinds of input graph representations, α parameters are strictly between 0 and 1. One can notice the variances of those distributions

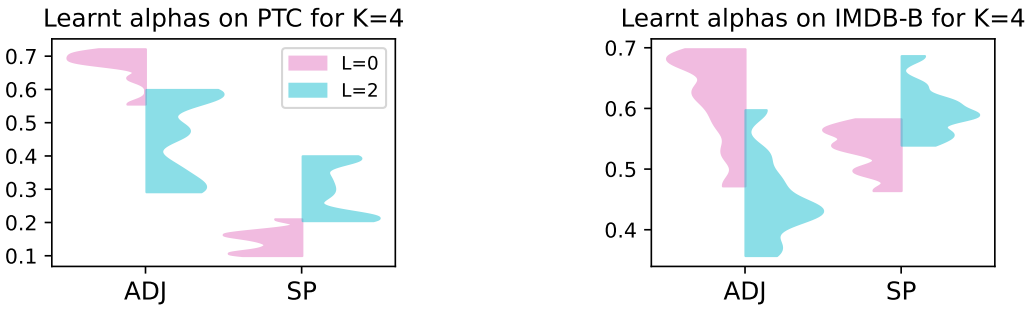

FIGURE 4.8: Distributions of estimated α on the PTC and IMDB-B datasets, using TFGW with $L \in \{0, 2\}$ GIN layers while learning on adjacency (ADJ) or shortest-path (SP) input graph representations.

illustrating the non-uniqueness of this trade-off parameter coming from the non-convexity of our optimization problem (a given value of α can potentially be compensated by the scaling of the GNN output). Unfortunately, the analysis of the real relative importance between structures and features can not be achieved only by looking at those values as the node embeddings and templates are different across models and data splits. In order to be exhaustive, we reported in Annex 8.2.4 the α distribution for each dataset of the best selected TFGW models with $L = 2$ whose performances are given in Table 4.2.

Sensitivity to the number of GIN layers and templates. To illustrate the sensitivity of our TFGW layer to the number of templates K and the number of GIN layers L in $\phi_{\mathbf{u}}$, we learned our models on the PTC dataset with L varying in $\{0, 1, 2, 3, 4\}$. We follow the same procedure than in the benchmark of Section 4.3.2 regarding the validation of K and the learning process, while fixing the number of hidden units in the GNN layers to 16. The test accuracy distributions for all settings are reported in Figure 4.9. Two phases are clearly distinguishable. The first one for $L \leq 2$, where for each L we see that the performance across the number of templates steadily increases, and the second for $L > 2$ where this performance progressively decreases as a function of L. Moreover, in the first phase performances are considerably dependent on K to compensate for a simple node representation, while this dependency is mitigated in the second which exhibits a slight overfitting. Note that these deeper models still lead to competitive results in comparison with benchmarked approaches in Table 4.2, with best averaged accuracies of 70.6 ($L = 3$) and 67.4 ($L = 4$). On one hand, these observations led us to set our number of layers to $L = 2$ for all benchmarked datasets which lead to strong generalization power. On the other hand, deeper models might be a way to benefit from our FGW embeddings with very few templates which can be interesting from a computational perspective on larger graph datasets.

To complete the sensitivity analysis given above, we also studied the dynamics of GIN models with sum aggregation.

Sensitivity of GIN to the number of layers. For a number of GIN layers varying in $L \in \{1, 2, ..., 6\}$, using analog settings and validation. The test accuracies of the validated models for each L are reported in Figure 4.10. First, the model with $L = 4$ (default for the method (Xu et al., 2019c)) leads to best performances on average. Whereas $L = 5$ leads to worst performances but $L = 6$ leads to the second best model in this benchmark. Then, no clear pattern of overfitting is observed when L increases when using sum pooling contrary to our TFGW based pooling. Such behavior may come from the Jumping Knowledge scheme (with concatenation) (Xu et al., 2018), as argued by the authors.

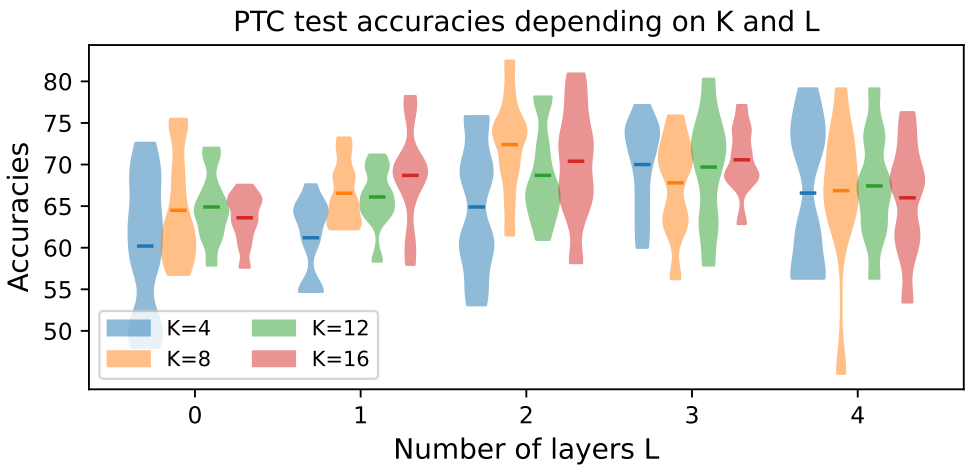

FIGURE 4.9: Test accuracy distributions by number of templates K and number of GIN layers L, using TFGW as pooling layer with $K \in \{4, 8, 12, 16\}$ templates.

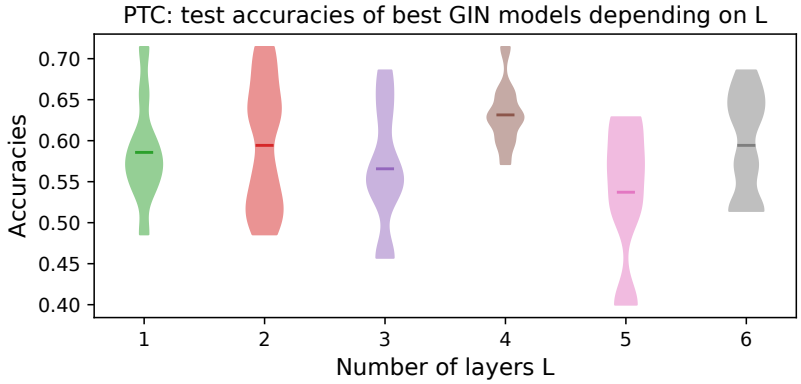

FIGURE 4.10: Test accuracies on PTC of GIN using sum pooling with a varying number of layers $L \in \{1, ..., 6\}$.

4.3.5 TFGW-GAT: sensitivity analysis

As we focused until here on the behavior of our models while using GIN layers, one remaining question is how does TFGW behave with other GNN architectures than GIN? Our intuition regarding this matter is that using our TFGW layer to obtain the graph representations consistently leads to better performances than simple sum pooling over graph nodes, regardless of the GNN architectures. Moreover, performances of both approaches should be correlated. To support these affirmations, we investigate here the use of Graph Attention networks (Veličković et al., 2018, GAT) for graph classification, either using a simple sum pooling or using the TFGW layer.

GAT with sum pooling. First, as the GAT architecture was investigated by its authors on node classification tasks and not graph-level ones, we study the behavior of GAT layers within a comparable framework than the one proposed by GIN, on 3 bioinformatic datasets. We use the same Jumping Knowledge scheme than GIN, i.e we concatenate features produced at each GAT layer, then we sum them across all nodes to get the graph representation fed to the final classifier $\psi_{\boldsymbol{v}}$. To fit to the benchmark reported in Table 2 of the main paper, we validate the features dimension in $\{16, 32\}$ (using a single attention head in GAT layers) and the dropout ratios applied to $\psi_{\boldsymbol{v}}$ in $\{0, 0.2, 0.5\}$, while considering $L \in \{1, 2, 3, 4\}$ GAT layers.

TABLE 4.5: Test set classification accuracies from 10-fold CV. The first (resp. second) best performing method is highlighted in bold (resp. underlined).

	MUTAG	PTC	PROTEIN
GAT (L=1)	89.4(1.0)	53.1(3.4)	**77.8(1.7)**
GAT (L=2)	91.1(2.5)	52.0(4.0)	76.3(3.1)
GAT (L=3)	88.9(1.7)	53.4(3.8)	75.9(2.3)
GAT (L=4)	**91.2(2.8)**	50.9(5.8)	77.6(2.7)
GIN (L=4)	90.1(4.4)	**63.1(3.9)**	76.2(2.8)

TABLE 4.6: Test set classification accuracies from 10-fold CV of the TFGW models using GAT or GIN as $\phi_{\boldsymbol{u}}$. The first (resp. second) best in bold (resp. underlined).

category	model	MUTAG	PTC	PROTEIN
Ours	TFGW ADJ (L=2)	95.4(3.5)	68.7(5.8)	**83.4(2.8)**
	TFGW SP (L=2)	96.2(3.0)	67.9(5.8)	82.6(2.9)
$\phi_{\boldsymbol{u}}$ = GAT	TFGW ADJ (L=1)	94.8(3.1)	66.9(5.4)	82.1(3.3)
	TFGW SP (L=1)	**96.4(3.3)**	68.3(6.0)	82.3(3.1)
Ours	TFGW ADJ (L=2)	**96.4(3.3)**	**72.4(5.7)**	82.9(2.7)
	TFGW SP (L=2)	94.8(3.5)	70.8(6.3)	82.0((3.0))
$\phi_{\boldsymbol{u}}$ = GIN	TFGW ADJ (L=1)	94.8(3.1)	68.7(5.8)	81.5(2.8)
	TFGW SP (L=1)	95.4(3.5)	70.9(5.5)	82.1(3.4)
sum	GAT (L=4)	91.2(2.8)	50.9(5.8)	77.6(2.7)
pooling	GIN (L=4)	90.1(4.4)	63.1(3.9)	76.2(2.8)

The results in terms of accuracy are reported in Table 4.5. GAT provides competitive performances on MUTAG and PROTEIN datasets compared to GIN, while GIN largely outperforms GAT on the PTC dataset. Also, it seems harder to find a consensus across datasets on L for GAT-based architectures compared to GIN's ones. Finally, we observed that using multi-head attention was prone to overfitting and led to lower performances on these graph classification tasks, so such suggestions from GAT's authors on node classification tasks seem to not hold for these graph-level tasks.

GAT with TFGW pooling. Finally, we investigate the use of our TFGW layer with GAT layers, as $\phi_{\boldsymbol{u}}$ to produce node embeddings. We follow a similar validation than for TFGW coupled with GIN, setting $L \in \{1, 2\}$. The results are reported in Table 4.6. We can see that TFGW with GAT leads to competitive performances compared to TFGW with GIN, at least on MUTAG and PROTEIN. GAT clearly struggles on the PTC dataset, however with our TFGW layer instead of a sum pooling, we observe a boost of performances from 53.4% to 68.7%. Even if TFGW coupled with GIN on PTC is still considerably better than TFGW with GAT. Therefore, the choice of GNN architectures to produce node embeddings to feed to the TFGW layer matters, but the gain from using TFGW seems to be independent of the GNN architecture.

4.4 Discussion and conclusion

We have introduced a new GNN layer whose goal is to represent a graph by its distances to template graphs, according to the optimal transport metric FGW. The proposed layer can be used directly on raw graph data, *i.e* without node embedding step, as the first layer of a GNN. Or it can also benefit from more involved node embedding using classical GNN layers, such as

the studied ones, GIN (Xu et al., 2019c) and GAT (Veličković et al., 2018), or potentially any other ones. In a graph classification context, we combined this TFGW layer with a simple MLP model. We demonstrated on several benchmark datasets that this approach compared favorably with state-of-the-art GNN and kernel based classifiers. A sensitivity analysis and an ablation study were presented to justify the choice of several parameters explaining the good generalization performances.

We believe that the new way to represent complex structured data provided by TFGW will open the door to novel and hopefully more interpretable GNN architectures. From a practical perspective, future works will be dedicated to combine TFGW with fast GPU solvers for network flow (Shekhovtsov & Hlaváč, 2013). This would greatly accelerate our approach and more generally OT based deep learning methods. We also believe that the FGW distance and its existing extensions can be used with other learning strategies including the semi-relaxed FGW divergence (Vincent-Cuaz et al., 2022a;b) for sub-graph detection, as detailed in Chapter 6.

Chapter 5

Fused Gromov-Wasserstein Linear Dictionary of graphs

Contents

This chapter presents the results from the paper Vincent-Cuaz et al. (2021) which focuses on unsupervised Graph Representation Learning (GRL) through the lens of Dictionary Learning.
Dictionary Learning basically consists into modeling the observed data as a linear combination of a few basic elements. Yet, this analysis is not amenable in the context of graph learning, as graphs usually belong to different metric spaces. We fill this gap by proposing a new Graph Dictionary Learning (GDL) approach, which either uses the Gromov-Wasserstein (Section 3.2) or the Fused Gromov-Wasserstein (Section 3.3) distance for the data fitting term, depending on the presence of node attributes.

Contributions. In this chapter, we use OT distances between graphs encoded through their nodes' pairwise relations and features, to design a linear and online DL for graphs. This novel factorization model relies on the (F)GW as data fitting term to model graphs as convex combination of graph atoms *i.e.* dictionary elements. Our proposal is depicted in Figure 5.1 in the case of unattributed graphs.
We propose an online stochastic algorithm to learn the dictionary which scales to large real-world data (Section 5.2), and uses extensively new derivations of sub-gradients of the

(F)GW distance (Theorem 7, Section 3.3). An unmixing procedure projects the graph into an embedding space defined *w.r.t.* the dictionary (Section 5.2.2).
Interestingly enough, we prove that the (F)GW distance in this embedding is upper-bounded by a Mahalanobis distance over the space of *unmixing* weights, providing a reliable and fast approximation of (F)GW (Section 5.2.3). Moreover, this approximation defines a kernel that can be efficiently used for clustering and classification of graphs datasets (Section 5.3.2).
Finally, we empirically demonstrate the relevance of our approach for online subspace estimation, subspace tracking by designing streams of graphs (Section 5.3.5) and graphs completion (Section 5.2.6) over two datasets.

5.1 Introduction

The question of how to build machine learning algorithms able to go beyond vectorial data and to learn from structured data such as graphs has been of great interest in the last decades. Notable applications can be found in molecule compounds (Kriege et al., 2018), brain connectivity (Ktena et al., 2017), social networks (Yanardag & Vishwanathan, 2015; Brouard et al., 2011), time series (Cuturi & Blondel, 2018), trees (Day, 1985) or images (Harchaoui & Bach, 2007; Bronstein et al., 2017).

Designing good representations for these data is challenging, as their nature is by essence non-vectorial, and requires a dedicated modeling. Graph Representation Learning problems have traditionally been handled using implicit representations such as graph kernels (Section 2.1), which may also leverage OT based distances (Section 3.2 and 3.3). However, one limit of kernel methods is that the representation of the graph is fixed *a priori* and cannot be adapted to specific datasets. On the other hand, Geometric deep learning approaches (Bronstein et al., 2017)(see also Section 2.2) attempt to learn a representation of structured data by means of deep learning (Scarselli et al., 2008; Perozzi et al., 2014; Niepert et al., 2016). Graph Neural Networks (Wu et al., 2020) have shown impressive performance for end-to-end supervised learning problems. However both kernel methods and many deep learning based representations for graphs suffer from the fundamental *pre-image* problem, that prevents recovering actual graph objects from the embeddings.

Here we attack the *unsupervised graph representation learning* problem, where additionally the entirety of the data might not be known beforehand, and is rather produced continuously by different sensors, and available through streams. In this setting, tackling the non-stationarity of the underlying generating process is challenging (Ditzler et al., 2015). Good examples can be found, for instance, in the context of dynamic functional connectivity (Heitmann & Breakspear, 2018) or network science (Masuda & Lambiotte, 2020). As opposed to recent approaches focusing on dynamically varying graphs in online or continuous learning (Yang et al., 2018; Vlaski et al., 2018; Wang et al., 2020), we rather suppose in this work that *distinct* graphs are made progressively available (Zambon et al., 2017; Grattarola et al., 2019). This setting is particularly challenging as the structure, the attributes or the number of nodes of each graph observed at a time step can differ from the previous ones. We propose to tackle this problem by learning a linear representation of graphs with online dictionary learning.

Dictionary Learning (DL). Dictionary Learning (Mairal et al., 2009; Schmitz et al., 2018) is a field of unsupervised learning that aims at estimating a linear representation of the data, *i.e.* to learn a linear subspace defined by the span of a family of vectors, called *atoms*, which constitute a *dictionary*. These atoms are inferred from the input data by minimizing a reconstruction error. These representations have been notably used in statistical frameworks such as data clustering (Ng et al., 2002), recommender systems (Bobadilla et al., 2013) or dimensionality reduction (Candès et al., 2011).
While DL methods mainly focus on vectorial data, it is of prime interest to investigate flexible

and interpretable factorization models applicable to *structured data*. We aim at designing a DL method overcoming such a limit relying on a vectorial representation in the dictionary, and which also allows a direct reconstruction of interpretable graphs.

We also consider the dynamic or time varying version of the problem, where the data generating process may exhibit non-stationarity over time, yielding a problem of subspace change or tracking (see *e.g.* Narayanamurthy & Vaswani (2018)), where one wants to monitor changes in the subspace best describing the data.

A recent contribution potentially overcoming the pre-image problem and also considering the dynamic or time varying version of the problem is Grattarola et al. (2019). In that paper, a variational autoencoder is indeed trained to embed the observed graphs into a constant curvature Riemannian manifold. The aim of that paper is to represent the graph data into a space where the statistical tests for change detection are easier. We look instead for a latent representation of the graphs that remains interpretable as much as possible. In this work, we propose to address both challenges of the pre-image problem and continuous learning thanks to a novel DL relying on Optimal Transport (Chapter 3) as a fidelity term to compare these structured data.

Optimal Transport for structured data. Optimal Transport (OT) theory (Chapter 3) provides a set of methods for comparing probability distributions, using, *e.g.* the well-known Wasserstein distance (Section 3.1). It has been notably used by the machine learning community in the context of distributional unsupervised learning (Arjovsky et al., 2017; Schmitz et al., 2018; Peyré & Cuturi, 2019). Broadly speaking the interest of OT lies in its ability to provide correspondences, or relations, between sets of points. Consequently, it has recently garnered attention for learning tasks where the points are described by graphs/structured data (see *e.g.* Niepert et al. (2016); Maretic et al. (2019); Togninalli et al. (2019); Xu et al. (2019a); Chen et al. (2020a)). One of the key ingredient in this case is to rely on the so called Fused Gromov-Wasserstein (FGW) distance (see Section 3.3) which is an OT problem adapted to the scenario in which the supports of the probability distributions lie in different metric spaces.

Graph Dictionary Learning meets Optimal Transport. Note that OT divergences as losses for linear and non-linear DL over vectorial data have already been proposed in Bonneel et al. (2016); Rolet et al. (2016); Schmitz et al. (2018) and further details on these approaches can be found in Section 3.1.4. However the case of structured data remains quite unaddressed.

In a recent work, Xu (2020) proposed a non-linear factorization of graphs using a regularized version of (F)GW barycenters (Peyré et al., 2016) and called it Gromov-Wasserstein Factorization (GWF). Given a dataset of I graphs $\{(\boldsymbol{C}_i, \boldsymbol{h}_i)\}_{i\in[\![I]\!]}$, authors propose to learn a dictionary $\{\overline{\boldsymbol{C}}_k \in \mathbb{R}^{\overline{N}_k \times \overline{N}_k}\}_{k\in[\![K]\!]}$ by solving the following DL problem

$$\min_{\{\boldsymbol{w}_i\}\subset\Sigma_K, \{\overline{\boldsymbol{C}}_k\}} \sum_{i\in[\![I]\!]} \mathrm{GW}_2^2\left(\widetilde{\boldsymbol{B}}(\boldsymbol{w}_i; \{\overline{\boldsymbol{C}}_k\}), \boldsymbol{C}_i\right) \tag{5.1}$$

where $\widetilde{\boldsymbol{B}}(\boldsymbol{w}_i; \{\overline{\boldsymbol{C}}_k\}) \in \arg\min_{\boldsymbol{B}} \sum_k w_{ik}\, \mathrm{GW}_2^2(\boldsymbol{B}, \overline{\boldsymbol{C}}_k)$ is a non-linear GW barycenter (Section 3.2.5) of the learned graph atoms $\{\overline{\boldsymbol{C}}_k\}_{k\in[\![K]\!]}$ and $\boldsymbol{w}_i \in \Sigma_K$ is the embedding of $\boldsymbol{C}_i$ in the dictionary. As a consequence, the projection of $\boldsymbol{C}_i$ onto the dictionary requires solving a complex bi-level optimization problem that is computationally expensive and might suffer from a lack of interpretability. In these terms, we propose a novel linear Dictionary Learning that we develop in the next Section 5.2, and whose relevance to overcome these two limitations will be empirically verified in Section 5.3.

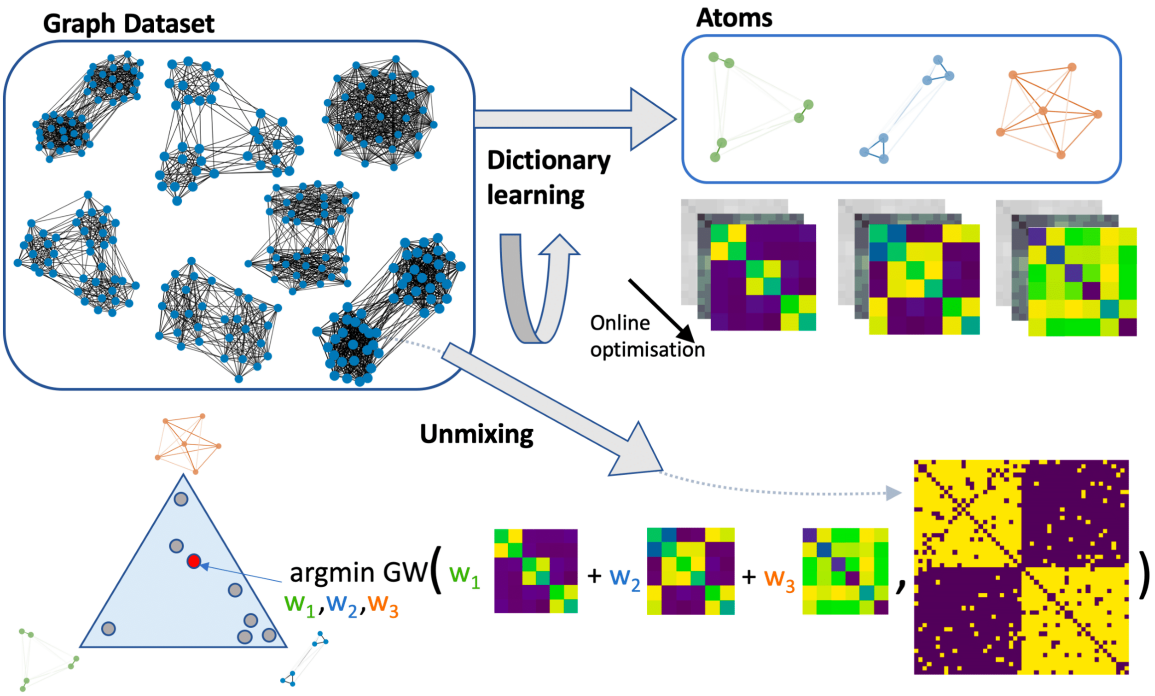

FIGURE 5.1: From a dataset of graphs without node features with different number of nodes, our method builds a dictionary of graph atoms with an online procedure. In this scenario, it uses the Gromov-Wasserstein distance as data fitting term between a convex combination of the atoms and a pairwise relations representation for graphs from the dataset.

5.2 Fused Gromov-Wasserstein Linear Dictionary Learning on graphs

In this section, we first detail how we model graphs as convex combinations of graph atoms (Section 5.2.1). An unmixing or projection step, studied in Section 5.2.2, is performed to get linear approximations as faithful as possible. After proving interesting properties of our model (Section 5.2.3), we propose an online algorithm to estimate optimal graph atoms from dataset of graphs (Section 5.2.4). Then we propose an extended DL (Section 5.2.5) to fully exploit the (F)GW formalism by estimating simultaneously the graph structures and distributions. Finally, we study in Section 5.2.6 the generative abilities of our DL through graphs completion.

5.2.1 Linear modeling of graphs

We propose to model a (labeled) graph as weighted sums of pairwise relation matrices and node feature matrices. More precisely, given a graph $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and a *dictionary* $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}_{k\in[K]}$ composed of graph with the same order $\overline{N}$, we want to find linear representations of the graph topology $\boldsymbol{C}$ and its node features $\boldsymbol{F}$, as faithful as possible. In a first place, we assume that all graph atoms share the same node distribution, *i.e* $\forall k \in [\![K]\!]$, $\overline{\boldsymbol{h}}_k = \overline{\boldsymbol{h}} \in \Sigma_{\overline{N}}$. Linear representations are parameterized by $\boldsymbol{w} \in \Sigma_K$ and are then expressed as

$$\widetilde{\boldsymbol{C}}(\boldsymbol{w}) = \sum_{k\in[\![K]\!]} w_k \overline{\boldsymbol{C}}_k \qquad \text{and} \qquad \widetilde{\boldsymbol{F}}(\boldsymbol{w}) = \sum_{k\in[\![K]\!]} w_k \overline{\boldsymbol{F}}_k \tag{5.2}$$

while their nodes relative significance is fixed to the shared probability vector $\overline{\boldsymbol{h}}$. Notice that this assumption is consistent with the common choice of setting node distributions to uniform ones.

The dictionary is made of pairwise relation matrices and feature matrices of labeled graphs with order $\overline{N}$. Thus, each tuple $(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k) \in \mathbb{R}^{\overline{N}\times\overline{N}} \times \mathbb{R}^{\overline{N}\times d}$ is called an *atom*, and $\boldsymbol{w} = (w_k)_{k\in[K]} \in \Sigma_K$ is designated as *embedding* and denotes the coordinate of the graph $\mathcal{G}$ in the dictionary as illustrated for unattributed graphs in Figure 5.1. We rely on the FGW distance to assess the quality of our linear approximations and propose to minimize it to estimate its optimal embedding.

In addition to being interpretable by linearity, the convex combination used to model the graph structure can convey certain properties of the atom structures. For instance, when the pairwise matrices $\boldsymbol{C}$ are adjacency matrices and the dictionary atom structures are weighted adjacency matrices with components in $[0, 1]$, the model $\widetilde{\boldsymbol{C}}(\boldsymbol{w})$ provides a matrix whose components can be interpreted as probabilities of connection between the nodes.
Moreover if atom structures are distance matrices, such as shortest-path matrices, the model will also be a distance matrix (as diagonal entries will be null and its off-diagonal entries will satisfy the triangle inequality). Inner product properties (symmetry, linearity, positive-definiteness) of the atoms can also be inherited by the linear representations of the inputs, however Euclidean distance properties can not (Section 3.3). Finally if atom structures are positive semi-definite (PSD) matrices, the model $\widetilde{\boldsymbol{C}}(\boldsymbol{w})$ will also be PSD matrices. These last properties can be of interest when considering the optimization problem inherent to the FGW distance (Section 3.3).

5.2.2 Fused Gromov-Wasserstein linear unmixing

We first study the unmixing problem that consists in projecting a graph on the linear representation discussed above, *i.e.* estimate the optimal embedding $\boldsymbol{w}$ of a graph $\mathcal{G}$. The unmixing problem can be expressed as the minimization of the FGW distance between any graph $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and its linear representations in the dictionary following equation (5.2):

$$\min_{\boldsymbol{w}\in\Sigma_K} \mathrm{FGW}^2_{2,\alpha}\left(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \overline{\boldsymbol{h}}\right) \Leftrightarrow \min_{\boldsymbol{w}\in\Sigma_K, \boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \mathcal{E}^{FGW}_{\alpha}\left(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T}\right) \tag{5.3}$$

where the FGW cost $\mathcal{E}^{FGW}_{\alpha}$ is given in equation (3.57) of Section 3.3. Once more, since $\overline{\boldsymbol{h}}$ is the same for all atoms and models, it does not depend on $\boldsymbol{w}$. This is why we called it $\overline{\boldsymbol{h}}$, instead of $\widetilde{\boldsymbol{h}}$.

Without any assumption over structures involved in the FGW matching, the sub-problem w.r.t $\boldsymbol{T}$ is non-convex, as detailed in Section 3.3. The sub-problem w.r.t $\boldsymbol{w}$ is a *convex* quadratic program as detailed in the following lemma proven in Annex 8.3.1:

Lemma 5 (FGW unmixing sub-problem w.r.t w) *For any input graph $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$, any dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}})\}_{k\in[\![K]\!]}$ and any admissible coupling $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$, the unmixing sub-problem*

$$\min_{\boldsymbol{w}\in\Sigma_K} \mathcal{E}^{FGW}_{\alpha}\left(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T}\right) \tag{5.4}$$

is a convex *problem which is equivalent to the following canonical quadratic program*

$$\min_{\boldsymbol{w}\in\Sigma_K} \boldsymbol{w}^\top \boldsymbol{Q}_\alpha \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{c} \tag{5.5}$$

where

$$\boldsymbol{Q}_\alpha = 2\left(\alpha \boldsymbol{M}^{GW} + (1-\alpha)\boldsymbol{M}^{W}\right) \tag{5.6}$$

is PSD as a convex combination of the PSD matrices, $\boldsymbol{M}^{GW} = \left(\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \overline{\boldsymbol{C}}_p, \overline{\boldsymbol{C}}_q \boldsymbol{D}_{\overline{\boldsymbol{h}}} \rangle_F\right)_{p,q \in [\![K]\!]^2}$ *and* $\boldsymbol{M}^{W} = (\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{F}}_p, \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{F}}_q \rangle_F)_{p,q \in [\![K]\!]^2}$ *while denoting* $\boldsymbol{D}_{\overline{\boldsymbol{h}}} = diag(\overline{\boldsymbol{h}})$. *Finally,* $\boldsymbol{c} = (c_k)_{k \in [\![K]\!]}$ *satisfies* $c_k = -2\alpha \langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}}_k \rangle_F - 2(1-\alpha) \langle \boldsymbol{T}^\top \boldsymbol{F}, \overline{\boldsymbol{F}}_k \rangle_F$.

Algorithm 6 BCD for unmixing problem 5.3

1: Inputs: Input graph $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$, Dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}})\}_k$, trade-off parameter α.
2: Initialize $\boldsymbol{w} = \frac{1}{K} \mathbf{1}_K$
3: **repeat**
4: Compute OT matrix $\boldsymbol{T}$ of $\text{FGW}^2_{2,\alpha}\left(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \overline{\boldsymbol{h}}\right)$, with CG Algorithm 4 of Section 3.3.
5: Compute the optimal $\boldsymbol{w}$ solving (5.3) for a fixed $\boldsymbol{T}$ with CG algorithm.
6: **until** convergence

Interestingly, the Hessian of this quadratic form depends on linear interactions between the atom structures and features given by the matrices $\boldsymbol{M}^{GW}$ and $\boldsymbol{M}^{W}$. However, even if this sub-problem is convex, the overall problem is non-convex so we propose to tackle it using a Block Coordinate Descent (BCD) algorithm (Tseng, 2001).

The BCD given in Algorithm 6 works by alternatively updating the OT matrix of the FGW distance and the embeddings $\boldsymbol{w}$. When $\boldsymbol{w}$ is fixed the problem is a classical FGW that we solve using the Conditional Gradient (CG) algorithm (Jaggi, 2013) based on Vayer et al. (2019a), further detailed in Section 3.3.3. Note that the use of an exact solver for FGW, instead of a regularized proxy allowed us to keep a sparse OT matrix as well as to preserve "high frequency" components of the graph, as opposed to regularized versions of GW (Peyré et al., 2016; Solomon et al., 2016; Xu, 2020) that promotes dense OT matrices and leads to smoothed/averaged pairwise matrices (Sections 3.2.4 and 3.3.3). For a fixed OT matrix $\boldsymbol{T}$, we propose to estimate solution of the sub-problem w.r.t $\boldsymbol{w}$ using the CG solver reported in Algorithm 7. The CG consists in 3 steps:

i) Computing the gradient w.r.t $\boldsymbol{w}$ of the FGW cost $\mathcal{E}^{FGW}$ satisfying for all $k \in [\![K]\!]$

$$\begin{aligned}\left(\nabla_{\boldsymbol{w}} \mathcal{E}^{FGW}_{\alpha}\right)_k = 2\alpha \left\{ \langle \overline{\boldsymbol{C}}_k \odot \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle_F - \langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}}_k \rangle_F \right\} \\ + 2(1-\alpha) \langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \widetilde{\boldsymbol{F}}(\boldsymbol{w}) - \boldsymbol{T}^\top \boldsymbol{F}, \overline{\boldsymbol{F}}_q \rangle_F\end{aligned} \tag{5.7}$$

An efficient implementation consists in using operations on 3D tensors to leverage the linearity of the gradient w.r.t $\boldsymbol{w}$. At the BCD algorithm initialization, one can store both PSD matrices $\boldsymbol{M}^{GW}$ and $\boldsymbol{M}^{W}$ (see Lemma 5) to reuse them as such to compute both terms in the gradient which depends on $\boldsymbol{w}$. Then terms depending on $\boldsymbol{T}$ can also be stored at the first CG iteration to avoid redundant computation and used as such to fasten the line-search described below. Therefore the overall computation of the gradient comes with $O\left(K(N^2\overline{N} + N\overline{N}^2 + \overline{N}^3)\right)$ operations made at initialization. Then at next iterations, every gradient can be computed using $O(K)$ operations. Such scheme is efficient as we expect $\overline{N}$ and K to be small compared to N in a factorization context.

ii) Solve the direction-finding subproblem in equation (5.9) which consists in minimizing the linear approximation of the problem given by the first-order Taylor approximation of $\mathcal{E}^{FGW}_{\alpha}$ around $\boldsymbol{w}$. This problem admits *closed-form solutions* x^*, given for instance by finding the entries $\mathcal{K} \subset [\![K]\!]$ s.t $\forall k \in \mathcal{K}, g_k = \min_j g_j$ and forming x^* from entries

Algorithm 7 CG for the sub-problem w.r.t $\boldsymbol{w}$

Inputs: Input graph $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$, Dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}})\}_k$, trade-off parameter α.
repeat
  Compute $\boldsymbol{g}$, the gradient *w.r.t* $\boldsymbol{w}$ of $\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T})$ following (5.7).
  Direction-finding subproblem:

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x} \in \Sigma_K} \boldsymbol{x}^T \boldsymbol{g} \tag{5.9}$$

  Line-search: denoting $\boldsymbol{z}(\gamma) = \boldsymbol{w} + \gamma(\boldsymbol{x}^\star - \boldsymbol{w}) = \boldsymbol{w} + \gamma \Delta \boldsymbol{w}$,

$$\gamma^\star = \arg\min_{\gamma \in (0,1)} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \widetilde{\boldsymbol{F}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) = \arg\min_{\gamma \in (0,1)} a\gamma^2 + b\gamma + c \tag{5.10}$$

  where

$$\begin{aligned} a &= \alpha \langle \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w})^2, \overline{\boldsymbol{h}\boldsymbol{h}}^\top \rangle + (1-\alpha) \langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \widetilde{\boldsymbol{F}}(\Delta \boldsymbol{w}), \widetilde{\boldsymbol{F}}(\Delta \boldsymbol{w}) \rangle \\ b &= 2\alpha \{ \langle \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w}) \odot \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \overline{\boldsymbol{h}\boldsymbol{h}}^\top \rangle - \langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w}) \rangle \} \\ &\quad + 2(1-\alpha) \langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \widetilde{\boldsymbol{F}}(\boldsymbol{w}) - \boldsymbol{T}^\top \boldsymbol{F}, \widetilde{\boldsymbol{F}}(\Delta \boldsymbol{w}) \rangle \end{aligned} \tag{5.11}$$

  $\boldsymbol{w} \leftarrow \boldsymbol{z}(\gamma^\star)$
until convergence

$x_k^* = \frac{1}{|\mathcal{K}|}, \forall k \in \mathcal{K}$. Note that the solution is unique if and only if the gradient $\boldsymbol{g}$ admits a unique coordinate achieving $\min_j g_j$.

iii) Determine the exact step-size to update $\boldsymbol{w}$ by solving the problem in equation (5.10) admitting closed-form solutions given in Algorithm 5, Section 3.3.3. Coefficients a and b can be computed in $O(K^2)$ operations using pre-computed terms detailed in (i).

Note that for non-convex problems the CG algorithm is known to converge to a local stationary point (Lacoste-Julien, 2016, Theorem 1) at a rate $O(1/\sqrt{t})$, where t relates to the number of algorithm iterations. Whereas for convex problems the CG algorithm is known to convergence to a global optimum (Jaggi, 2013, Theorem 1) at a rate $O(1/t)$.

We also propose to promote sparsity in the weights $\boldsymbol{w}$ similarly to sparse coding (Chen et al., 2001). To this end, we propose to use a *negative* quadratic regularization promoting sparsity on the simplex as discussed in Li et al. (2016), expanding the unmixing (5.3) problem to the following one

$$\min_{\boldsymbol{w} \in \Sigma_K} \quad \mathrm{FGW}_{2,\alpha}^2 \left(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \overline{\boldsymbol{h}} \right) - \lambda \|\boldsymbol{w}\|_2^2 \tag{5.8}$$

where $\lambda \in \mathbb{R}^+$. The BCD and the CG, respectively given in Algorithms 6 and 7, are straightforward to adapt to this regularized unmixing problem. However the sub-problem w.r.t $\boldsymbol{w}$ becomes non-convex for large values of λ, which reduces our theoretical guarantees. In practice, we observed for both unmixing problems (5.3) and (5.8) a typical convergence of the CGs in a few tens of iterations. The BCD itself converges in less than 10 iterations.

5.2.3 Fast upper bound for GW

Interestingly, when two graphs belong to the linear subspace defined by our dictionary, there exists a proxy of the FGW distance using a dedicated Mahalanobis distance as described in the next:

Theorem 9 (Fused Gromov-Wasserstein upper-bound in the embedding) *For two embedded graphs with node features, with embeddings $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ over the set of pairwise relation matrices $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)\}_{k\in[K]} \subset \mathbb{R}^{\overline{N}\times\overline{N}} \times \mathbb{R}^{\overline{N}\times d}$, and a shared masses vector $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$, the following inequality holds $\forall \alpha \in [0,1]$,*

$$\mathrm{FGW}_{2,\alpha}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}_1), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_1), \widetilde{\boldsymbol{C}}(\boldsymbol{w}_2), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_2)\right) \leq \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_{\boldsymbol{Q}_\alpha/2} \tag{5.12}$$

where $\boldsymbol{Q}_\alpha$ is the PSD matrix in equation (5.6), hence engenders a Mahalanobis distance between embeddings.

Proof of Theorem 9. Let us consider two embedded graphs with node features, with embeddings $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ over a dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)\}_{s\in[\![K]\!]} \subset \mathbb{R}^{\overline{N}\times\overline{N}} \times \mathbb{R}^{\overline{N}\times d}$, and a masses vector $\overline{\boldsymbol{h}}$ shared by linear models and dictionary atoms. We introduce for any admissible coupling $\boldsymbol{T} \in \mathcal{U}(\overline{\boldsymbol{h}}, \overline{\boldsymbol{h}})$, the application providing the FGW cost given $\boldsymbol{T}$ between both embedded graphs $f_{\boldsymbol{T}} : \boldsymbol{w} \in \Sigma_k \to \mathcal{E}^{FGW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}_1), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_1), \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T}\right)$. This function is further characterized in Lemma 5. As $f_{\boldsymbol{T}}$ is a second-order polynomial function, it satisfies for any $\boldsymbol{w} \in \Sigma_K$,

$$f_{\boldsymbol{T}}(\boldsymbol{w}) = f_{\boldsymbol{T}}(\boldsymbol{w}_1) + \nabla_{\boldsymbol{w}} f_{\boldsymbol{T}}(\boldsymbol{w}_1)^\top (\boldsymbol{w} - \boldsymbol{w}_1) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_1)^\top \boldsymbol{Q}_\alpha (\boldsymbol{w} - \boldsymbol{w}_1) \tag{5.13}$$

where $\boldsymbol{Q}_\alpha$ is given by Lemma 5 and for any $k \in \Sigma_k$,

$$\begin{aligned}\nabla_{\boldsymbol{w}} f_{\boldsymbol{T}}(\boldsymbol{w}_1)_k = 2\alpha\{\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \widetilde{\boldsymbol{C}}(\boldsymbol{w}_1) \boldsymbol{D}_{\overline{\boldsymbol{h}}}, \overline{\boldsymbol{C}}_k\rangle - \langle \boldsymbol{T}^\top \widetilde{\boldsymbol{C}}(\boldsymbol{w}_1) \boldsymbol{T}, \overline{\boldsymbol{C}}_k\rangle\} \\ + 2(1-\alpha)\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \widetilde{\boldsymbol{F}}(\boldsymbol{w}_1) - \boldsymbol{T}^\top \widetilde{\boldsymbol{F}}(\boldsymbol{w}_1), \overline{\boldsymbol{F}}_q\rangle\end{aligned} \tag{5.14}$$

following intermediate equations (8.77) and (8.81) of the lemma's proof. Then denoting $\boldsymbol{T}^*$ the optimal coupling resulting from the FGW problem, we have $\boldsymbol{T}^*$ which minimizes the application $\boldsymbol{T} \in \mathcal{U}(\overline{\boldsymbol{h}}, \overline{\boldsymbol{h}}) \to f_{\boldsymbol{T}}(\boldsymbol{w}_2)$. Therefore as $\boldsymbol{D}_{\overline{\boldsymbol{h}}}$ is an admissible coupling, we have $f_{\boldsymbol{T}^*}(\boldsymbol{w}_2) \leq f_{\boldsymbol{D}_{\overline{\boldsymbol{h}}}}(\boldsymbol{w}_2)$, which can be expanded using equation (5.13) as

$$f_{\boldsymbol{T}^*}(\boldsymbol{w}_2) \leq f_{\boldsymbol{D}_{\overline{\boldsymbol{h}}}}(\boldsymbol{w}_1) + \nabla_{\boldsymbol{w}} f_{\boldsymbol{D}_{\overline{\boldsymbol{h}}}}(\boldsymbol{w}_1)^\top (\boldsymbol{w}_2 - \boldsymbol{w}_1) + \frac{1}{2}(\boldsymbol{w}_2 - \boldsymbol{w}_1)^\top \boldsymbol{Q}_\alpha (\boldsymbol{w}_2 - \boldsymbol{w}_1) \tag{5.15}$$

Using equation (5.14) it is easy to see that $\nabla_{\boldsymbol{w}} f_{\boldsymbol{D}_{\overline{\boldsymbol{h}}}}(\boldsymbol{w}_1) = \boldsymbol{0}$. Moreover $f_{\boldsymbol{D}_{\overline{\boldsymbol{h}}}}(\boldsymbol{w}_1) = 0$ as it corresponds to the FGW distance between $\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}_1), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_1), \overline{\boldsymbol{h}}\right)$ and itself. So using these relations and the definition of $f_{\boldsymbol{T}^*}(\boldsymbol{w}_2)$, we have the desired upper-bound of equation (5.12). □

As detailed in equation (5.15) of the proof, the upper-bound given in Theorem 9 is obtained by considering the FGW cost between the linear models calculated using the admissible coupling $\boldsymbol{D}_{\overline{\boldsymbol{h}}}$. The latter coupling assumes that both graph representations are aligned and therefore is a priori suboptimal. As such, this bound is not tight in general. However, when the embeddings are close, the optimal coupling matrix should be close to $\boldsymbol{D}_{\overline{\boldsymbol{h}}}$ so that Theorem 9 provides a reasonable proxy to the FGW distance into our embedding space. In practice, this upper bound can be used to compute efficiently pairwise kernel matrices or to do retrieval of closest samples (see numerical experiments, Section 5.3).

5.2.4 Dictionary learning and online algorithm

Assume now that the dictionary $\overline{\mathcal{D}} = \{\overline{\mathcal{G}}_k : (\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}})\}_{k\in[K]}$ is not known and has to be estimated from the data. We define a dataset $\mathcal{D}$ of I attributed graphs $\mathcal{D} =$

$\{\mathcal{G}_i : (\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i\in[I]}$. Recall that each graph $\mathcal{G}_i$ of order N_i is summarized by its pairwise relation matrix $\boldsymbol{C}_i \in \mathbb{R}^{N_i\times N_i}$, its node feature matrix $\boldsymbol{F}_i \in \mathbb{R}^{N_i\times d}$ and weights $\boldsymbol{h}_i \in \Sigma_{N_i}$ over nodes, as described in Section 3.3.

The DL problem, that aims at estimating the optimal dictionary for a given dataset can be expressed as:

$$\min_{\{\boldsymbol{w}^{(i)}\}_i, \{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)\}_k} \sum_{i=1}^{I} \mathrm{FGW}_{2,\alpha}\left(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i), \overline{\boldsymbol{h}}\right) - \lambda\|\boldsymbol{w}_i\|_2^2 \tag{5.16}$$

where $\boldsymbol{w}_i \in \Sigma_K, \overline{\boldsymbol{C}}_k \in \mathbb{R}^{\overline{N}\times\overline{N}}$ and $\overline{\boldsymbol{F}}_k \in \mathbb{R}^{\overline{N}\times d}$. Let us denote the objective function in problem (5.16) by $\mathcal{F}_{\mathcal{D},\lambda}(\{\boldsymbol{w}_i\}, \overline{\mathcal{D}})$. The optimization problem above is a classical sparsity promoting dictionary learning on a linear subspace but with the important novelty that the reconstruction error is computed by means of the FGW distance. This allows us to learn a graphs subspace of fixed order $\overline{N}$ using a dataset of graphs with various orders. The sum over the errors in equation (5.16) can be seen as an expectation and we propose to devise an online strategy to optimize the problem similarly to the online DL proposed in Mairal et al. (2009).

The main idea is to update the dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k\}_k$ with a stochastic estimation of the gradients on a minibatch of B graphs $\mathcal{B} = \{\mathcal{G}_i\}_i$. At each stochastic update the unmixing problems providing optimal $\{(\boldsymbol{w}_i, \boldsymbol{T}_i)\}_i$ are solved independently for each graph $\mathcal{G}_i$ of the minibatch using a fixed dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)\}_k$, according to the procedure described in Section 5.2.2. Then one can *estimate* gradients *w.r.t* $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)\}_k$ of the objective function $\mathcal{F}_{\mathcal{D},\lambda}(\{\boldsymbol{w}_i\}, \overline{\mathcal{D}})$ of our DL problem (5.16), by gradients computed over the minibatch, given for each atom structure by

$$\widetilde{\nabla}_{\overline{\boldsymbol{C}}_k}\left(\mathcal{F}_{\mathcal{D},\lambda}(\{\boldsymbol{w}_i\}, \overline{\mathcal{D}})\right) = 2\alpha \sum_{i\in[\![B]\!]} w_{ik}\{\widetilde{\boldsymbol{C}}(\boldsymbol{w}_i) \odot \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top - \boldsymbol{T}_i^\top \boldsymbol{C}_i \boldsymbol{T}_i\} \tag{5.17}$$

and for each atom feature matrix by

$$\widetilde{\nabla}_{\overline{\boldsymbol{F}}_k}\left(\mathcal{F}_{\mathcal{D},\lambda}(\{\boldsymbol{w}_i\}, \overline{\mathcal{D}})\right) = 2(1-\alpha) \sum_{i\in[\![B]\!]} w_{ik}\{\boldsymbol{D}_{\overline{\boldsymbol{h}}}\widetilde{\boldsymbol{F}}(\boldsymbol{w}_i) - \boldsymbol{T}_i^\top \boldsymbol{F}_i\} \tag{5.18}$$

knowing that terms depending on $\boldsymbol{T}_i$ are already computed by unmixings solver. Note that if the input graph structures $\{\boldsymbol{C}_i\}$ are symmetric (undirected graphs) and atom structures are initialized as symmetric matrices, the updated atom structures following equation (5.17) will remain symmetric. When operating on adjacency or shortest path matrices (see numerical experiments in Section 5.3), we propose to add a projected gradient step to constraint atom structures to be non-negative. The stochastic update of the proposed algorithm is detailed in Algorithm 8. This process can be used on a finite dataset with possibly several epochs on the whole dataset or online in the presence of streaming graphs. We provide an example of such subspace tracking in Section 5.3.5. We will refer to our approach as GDL in the rest of the manuscript.

Numerical complexity The numerical complexity of GDL depends on the complexity of each update. The main computational bottleneck is the unmixing procedure that relies on multiple resolution of FGW problems. The complexity of solving a FGW with the CG algorithm between two graphs of order N and M and computing its gradient is dominated by $\mathcal{O}\left(N^2M + M^2N\right)$ operations (Peyré et al., 2016; Vayer et al., 2019a) (see details in Section 3.3.3). Thus given dictionary atoms of order $\overline{N}$, the worst case complexity can be only **quadratic** in the highest graph order $N_{\max} = \max_{i\in[\![I]\!]} N_i$ in the dataset, in a factorization context where $\overline{N}$ and K are assumed small compared to $N_{\max}$.

Algorithm 8 GDL: stochastic update of atoms $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)\}_{k\in[K]}$

1: Sample a minibatch of B graphs $\mathcal{B} := \{(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}$.
2: Compute optimal $\{(\boldsymbol{w}_i, \boldsymbol{T}_i)\}_{i\in[\![B]\!]}$ by solving B independent unmixing problems with Algorithm 6.
3: Projected gradient step with estimated gradients $\widetilde{\nabla}_{\overline{\boldsymbol{C}}_k}$ and $\widetilde{\nabla}_{\overline{\boldsymbol{F}}_k}$ given (5.17) and (5.18), $\forall k \in [K]$:

$$\overline{\boldsymbol{C}}_k \leftarrow Proj_{\mathbb{R}_+}(\overline{\boldsymbol{C}}_k - \eta_C \widetilde{\nabla}_{\overline{\boldsymbol{C}}_k}) \qquad \text{and} \qquad \overline{\boldsymbol{F}}_k \leftarrow \overline{\boldsymbol{F}}_k - \eta_F \widetilde{\nabla}_{\overline{\boldsymbol{F}}_k} \tag{5.19}$$

For instance, estimating embedding on dataset IMDB-M (Section 5.3.2) over 12 atoms takes on average 44 ms per graph (on processor i9-9900K CPU 3.60GHz)[1]. Note that in addition to scale well to large datasets thanks to the stochastic optimization, our method also leads to important speedups when using the representations as input feature for other ML tasks. For instance, we can use the upper bound of Theorem 9 to compute efficiently kernels between graphs instead of computing all pairwise FGW distances. Moreover, we show in Section 5.3.2 that our GDL representation technique compares favorably to the non-linear GW based DL GWF (see equation (5.1)), both in terms of numerical complexity and performance.

5.2.5 Learning the graph structure and distribution

Recent researches have studied the use of potentially more general distributions $\boldsymbol{h}$ on the nodes of graphs than the naive uniform ones commonly used. Xu et al. (2019a) empirically explored the use of distributions induced by degrees, such as parameterized power laws, $h_i = \frac{p_i}{\sum_i p_i}$, where $p_i = (deg(x_i) + a)^b$ with $a \in \mathbb{R}_+$ and $b \in [0, 1]$. They demonstrated the interest of this approach but also highlighted how hard it is to calibrate, which advocates for learning these distributions.
With this motivation, we extend our GDL model defined in equation (5.16) and propose to learn atoms of the form $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}_{k\in[K]}$. In this setting we have two independent dictionaries modeling the relative importance of the nodes via $\overline{\boldsymbol{h}}_k \in \Sigma_{\overline{N}}$, and their pairwise relations and node features through $(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k)$. This dictionary learning problem reads:

$$\min_{\{(\boldsymbol{w}_i,\boldsymbol{v}_i)\}_i, \{(\overline{\boldsymbol{C}}_k,\overline{\boldsymbol{F}}_k)\}_k} \sum_{i=1}^{I} \mathrm{FGW}^2_{2,\alpha}\left(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i)\right) - \lambda\|\boldsymbol{w}_i\|_2^2 - \rho\|\boldsymbol{v}_i\|_2^2 \tag{5.20}$$

where $\boldsymbol{w}_i \in \Sigma_K$ and $\boldsymbol{v}_i \in \Sigma_K$ are the structure and distribution embeddings, and the linear models are defined for any $i \in [\![I]\!]$ as

$$\widetilde{\boldsymbol{C}}(\boldsymbol{w}_i) = \sum_{k\in[\![K]\!]} w_{ik}\overline{\boldsymbol{C}}_k, \qquad \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i) = \sum_{k\in[\![K]\!]} w_{ik}\overline{\boldsymbol{F}}_k, \qquad \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i) = \sum_{k\in[\![K]\!]} v_{ik}\overline{\boldsymbol{h}}_k, \tag{5.21}$$

with $\overline{\boldsymbol{C}}_k \in \mathbb{R}^{\overline{N}\times\overline{N}}$, $\overline{\boldsymbol{F}}_k \in \mathbb{R}^{\overline{N}\times d}$ and $\overline{\boldsymbol{h}}_k \in \Sigma_K$. We denote the objective function in problem (5.20) by $\mathcal{F}_{\mathcal{D},\lambda,\rho}(\{\boldsymbol{w}_i, \boldsymbol{v}_i\}, \overline{\mathcal{D}})$.

Here we fully exploit the FGW formalism by estimating simultaneously the graph distribution $\widetilde{\boldsymbol{h}}$, its geometric structure $\widetilde{\boldsymbol{C}}$ and its node features $\widetilde{\boldsymbol{F}}$. The optimization problem (5.20) can be solved by an adaptation of the stochastic Algorithm 8, detailed in Algorithm 10. It consists in the two following steps:

[1] To update with new implementation

Algorithm 9 BCD for GDL extended unmixing problem providing optimal $(\boldsymbol{w}_i, \boldsymbol{v}_i, \boldsymbol{T}_i)$ for input graph $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$.

1: Inputs: Input graph $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$ and dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}_k$.
2: Initialize embeddings such as $\boldsymbol{w}_i = \boldsymbol{v}_i = \frac{1}{K}\mathbf{1}_K$
3: **repeat**
4: Compute OT matrix $\boldsymbol{T}_i$ of $\text{FGW}_{2,\alpha}\left(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i)\right)$, with CG algorithm (Section 3.3). From the finale iteration of CG, get dual potentials $(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ of the corresponding linear OT problem (see (5.23)).
5: Compute the optimal $\boldsymbol{v}$ by minimizing (5.23) *w.r.t* $\boldsymbol{v}$ using closed-form solutions (5.24) if $\rho = 0$ or using CG algorithm (5.25) if $\rho > 0$.
6: Compute the optimal $\boldsymbol{w}_i$ solving (5.8) given $\boldsymbol{T}$ and $\boldsymbol{v}$, setting $\overline{\boldsymbol{h}} = \sum_k v_{ik}\overline{\boldsymbol{h}}_k$, with CG algorithm 7.
7: **until** convergence

Algorithm 10 extended GDL: stochastic update of atoms $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}}_k)\}_{k\in[\![K]\!]}$

1: Sample a minibatch of B graphs $\mathcal{B} := \{(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i\in\mathcal{I}\subset[\![I]\!]}$.
2: Compute optimal embeddings $\{(\boldsymbol{w}_i, \boldsymbol{v}_i)\}_{i\in[\![B]\!]}$ coming jointly with the set of OT variables $(\boldsymbol{T}_i, \mu_i, \nu_i)$ by solving B independent unmixing problems with BCD algorithm 9.
3: Projected gradient step with estimated gradients $\widetilde{\nabla}_{\overline{\boldsymbol{C}}_k}$ (5.26), $\widetilde{\nabla}_{\overline{\boldsymbol{F}}_k}$ (5.27) and $\widetilde{\nabla}_{\overline{\boldsymbol{h}}_k}$ (5.28), $\forall k \in [\![K]\!]$:

$$\overline{\boldsymbol{C}}_k \leftarrow Proj_{\mathbb{R}_+}(\overline{\boldsymbol{C}}_k - \eta_C\widetilde{\nabla}_{\overline{\boldsymbol{C}}_k}), \quad \overline{\boldsymbol{F}}_k \leftarrow \overline{\boldsymbol{F}}_k - \eta_F\widetilde{\nabla}_{\overline{\boldsymbol{F}}_k} \quad \text{and} \quad \overline{\boldsymbol{h}}_k \leftarrow Proj_{\Sigma_N}(\overline{\boldsymbol{h}}_k - \eta_h\widetilde{\nabla}_{\overline{\boldsymbol{h}}_k}) \tag{5.22}$$

i) **Solve the extended unmixing problem.** We estimate the structure/node weights unmixings $(\boldsymbol{w}_k, \boldsymbol{v}_k)$ over a minibatch of graphs with an extension of the BCD Algorithm 6. This extended BCD is summarized in Algorithm 9. First, for fixed unmixings $(\boldsymbol{w}_i, \boldsymbol{v}_i)$, we compute the OT matrix $\boldsymbol{T}_k$ of $\text{FGW}_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i))$ using a CG algorithm (Section 3.3). At the last iteration of this CG, a Wasserstein problem is solved providing $\boldsymbol{T}_i$ and its corresponding optimal dual variables $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_i$ satisfying:

$$\text{FGW}_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i)) = \langle \boldsymbol{\mu}_i, \boldsymbol{h}_i\rangle + \langle \boldsymbol{\nu}_i, \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i)\rangle \tag{5.23}$$

So we can optimize $\mathcal{E}_\alpha^{FGW}$ w.r.t $\boldsymbol{v}_i$ at fixed $\boldsymbol{T}_i$ and $\boldsymbol{w}_i$ solving for

$$\min_{\boldsymbol{v}\in\Sigma_K} \langle \boldsymbol{\mu}_i, \boldsymbol{h}_i\rangle + \langle \boldsymbol{\nu}_i, \widetilde{\boldsymbol{h}}(\boldsymbol{v})\rangle \Leftrightarrow \min_{\boldsymbol{v}\in\Sigma_K} \boldsymbol{v}^\top \boldsymbol{c} \tag{5.24}$$

where $\boldsymbol{c} = (c_k = \langle \boldsymbol{\nu}_i, \overline{\boldsymbol{h}}_k\rangle)_{k\in[\![K]\!]}$. This problem is a classic Linear Program under simplex constraints which admits closed-form solutions, as discussed in Step 2 of Algorithm 7 (Section 5.2.2).
If we further promote sparsity of $\boldsymbol{v}_i$ ($\rho > 0$) using the negative quadratic regularization of equation (5.20), the unmixing sub-problem w.r.t $\boldsymbol{v}_i$ at fixed $\boldsymbol{T}_i$ and $\boldsymbol{w}_i$ becomes

$$\min_{\boldsymbol{v}\in\Sigma_K} -\rho\boldsymbol{v}^\top\boldsymbol{v} + \boldsymbol{v}^\top\boldsymbol{c} \tag{5.25}$$

which is a concave Quadratic Program that can be solved using a simple Conditional Gradient algorithm.

Finally, once $\boldsymbol{v}_i$ is updated for a fixed $\boldsymbol{T}_i$ and $\boldsymbol{w}_i$, we can update $\boldsymbol{w}_i$ using the CG Algorithm 7 by posing $\overline{\boldsymbol{h}} = \sum_k v_{ik}\overline{\boldsymbol{h}}_k$ in equations (5.7) and (5.10).

Algorithm 11 GDL: graphs completion

1: Inputs: Observed graph $\boldsymbol{C}_{obs}$, Dictionary $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}})\}$, size of the imputed graph and trade-off parameter α.
2: Initialize randomly the entries $\boldsymbol{C}_{imp}$ by iid sampling from $\mathcal{N}(0.5, 0.01)$ (In a symmetric manner if $\boldsymbol{C}_{obs}$ is symmetric).[2]
3: **repeat**
4: Compute optimal $(\boldsymbol{T}, \boldsymbol{w})$ of $(\boldsymbol{C}_{est}, \boldsymbol{F}_{est}, \boldsymbol{h})$ into the dictionary (5.8).
5: Perform a projected gradient step on $\boldsymbol{C}_{est}$ and $\boldsymbol{F}_{est}$ using gradients from equations (5.31) and (5.32).
6: **until** convergence.

ii) **Stochastic gradient updates.** Then once unmixing problems providing $(\boldsymbol{w}_i, \boldsymbol{v}_i, \boldsymbol{T}_i, \boldsymbol{\mu}_i, \boldsymbol{\nu}_i)$ are solved independently, we perform simultaneously a projected gradient step update of $\{\overline{\boldsymbol{C}}_k\}_k$ and $\{\overline{\boldsymbol{h}}_k\}_k$, and a gradient step update of $\{\overline{\boldsymbol{C}}_k\}_k$. For which stochastic gradients of the objective function $\mathcal{F}_{\mathcal{D},\lambda,\rho}(\{\boldsymbol{w}_i, \boldsymbol{v}_i\}, \overline{\mathcal{D}})$ of problem (5.20) read for each atom structure,

$$\widetilde{\nabla}_{\overline{\boldsymbol{C}}_k}\left(\mathcal{F}_{\mathcal{D},\lambda,\rho}(\{\boldsymbol{w}_i, \boldsymbol{v}_i\}, \overline{\mathcal{D}})\right) = \frac{2}{B}\sum_{i\in[\![B]\!]} w_{ik}\{\widetilde{\boldsymbol{C}}(\boldsymbol{w}_i) \odot \widetilde{\boldsymbol{h}}(\boldsymbol{v}_i)\widetilde{\boldsymbol{h}}(\boldsymbol{v}_i)^\top - \boldsymbol{T}_i^\top \boldsymbol{C}_i \boldsymbol{T}_i\}, \quad (5.26)$$

each atom feature matrices,

$$\widetilde{\nabla}_{\overline{\boldsymbol{F}}_k}\left(\mathcal{F}_{\mathcal{D},\lambda,\rho}(\{\boldsymbol{w}_i, \boldsymbol{v}_i\}, \overline{\mathcal{D}})\right) = \frac{2}{B}\sum_{i\in[\![B]\!]} w_{ik}\{\mathrm{diag}(\widetilde{\boldsymbol{h}}(\boldsymbol{v}_i))\widetilde{\boldsymbol{F}}(\boldsymbol{w}_i) - \boldsymbol{T}_i^\top \boldsymbol{F}_i\} \quad (5.27)$$

and each atom distributions

$$\widetilde{\nabla}_{\overline{\boldsymbol{h}}_k}\left(\mathcal{F}_{\mathcal{D},\lambda,\rho}(\{\boldsymbol{w}_i, \boldsymbol{v}_i\}, \overline{\mathcal{D}})\right) = \frac{1}{B}\sum_{i\in[\![B]\!]} v_{ik}\boldsymbol{\nu}_i \quad (5.28)$$

5.2.6 GDL for graphs completion

The GDL dictionaries estimated using equation (5.16)[3] on the dataset $\mathcal{D} = \{(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}$ are expected to capture a simplified semantic within $\mathcal{D}$, that might be considered as the hidden generative process from which observed graphs were sampled. This way the dictionary can be used to infer/complete a new graph that is only partially observed and assumed to be generated by the same generative process observed in $\mathcal{D}$.
In this setting, we aim at recovering the full structure $\boldsymbol{C} \in \mathbb{R}^{N\times N}$ and node features $\boldsymbol{F} \in \mathbb{R}^{N\times d}$, while only a subset of relations and features among $N_{obs} < N$ nodes are observed, denoted as $\boldsymbol{C}_{obs} \in \mathbb{R}^{N_{obs}\times N_{obs}}$ and $\boldsymbol{F}_{obs} \in \mathbb{R}^{N_{obs}\times d}$. This amounts to solving:

$$\min_{\boldsymbol{C}_{imp},\boldsymbol{F}_{imp}} \min_{\boldsymbol{w}} \mathrm{FGW}_{2,\alpha}\left(\boldsymbol{C}_{est}, \boldsymbol{F}_{est}, \boldsymbol{h}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \overline{\boldsymbol{h}}\right) - \lambda\|\boldsymbol{w}\|_2^2 \quad (5.29)$$

with

$$\boldsymbol{C}_{est} = \left[\begin{array}{c|c} \boldsymbol{C}_{obs} & \vdots \\ \hline \cdots & \boldsymbol{C}_{imp} \end{array}\right] \quad \text{and} \quad \boldsymbol{F}_{est} = \left[\begin{array}{c} \boldsymbol{F}_{obs} \\ \hline \boldsymbol{F}_{imp} \end{array}\right] \quad (5.30)$$

where only $N^2 - N_{obs}^2$ coefficients collected into $\boldsymbol{C}_{imp}$ and $(N - N_{obs})d$ coefficients collected into $\boldsymbol{F}_{imp}$ are optimized, and thus imputed.

[3]For simplicity, we consider here that the atoms share the same distribution $\overline{\boldsymbol{h}}$. The graph completion procedure described in this Section can be extended to GDL with learned distributions simply using the corresponding unmixing procedure.

We solve the optimization problem (5.29) by a classical projected gradient descent. At each iteration, we find an optimal pair of coupling and embedding $(\boldsymbol{T}, \boldsymbol{w})$ by solving the unmixing problem (5.8) for a fixed $(\boldsymbol{C}_{imp}, \boldsymbol{F}_{imp})$. Then $(\boldsymbol{T}, \boldsymbol{w})$ is used to calculate the gradient w.r.t $(\boldsymbol{C}_{imp}, \boldsymbol{F}_{imp})$ of the FGW loss between $(\boldsymbol{C}_{est}, \boldsymbol{F}_{est}, \boldsymbol{h})$ and $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}\boldsymbol{w}, \overline{\boldsymbol{h}})$. The latter is obtained thanks to the imputed components of the gradient of the FGW cost function evaluated at the fixed optimal coupling $\boldsymbol{T}$ and embedding $\boldsymbol{w}$ by using the Envelope Theorem (Bonnans & Shapiro, 2000). The updates read as follow, for the imputed structure

$$\boldsymbol{C}_{imp} \leftarrow \left[2\alpha\left(\boldsymbol{C}_{est} \odot \boldsymbol{h}\boldsymbol{h}^\top - \boldsymbol{T}\widetilde{\boldsymbol{C}}(\boldsymbol{w})\boldsymbol{T}^\top\right)\right]_{\text{imputed } i,j} \tag{5.31}$$

and for the imputed feature matrix

$$\boldsymbol{F}_{imp} \leftarrow \left[2(1-\alpha)\left(\boldsymbol{D}_{\boldsymbol{h}}\boldsymbol{F}_{est} - \boldsymbol{T}\widetilde{\boldsymbol{F}}(\boldsymbol{w})\right)\right]_{\text{imputed } i} \tag{5.32}$$

A projection step for $\boldsymbol{C}_{imp}$ can be applied to enforce known properties of $\boldsymbol{C}_{est}$, such as positivity and symmetry based on the observed subgraph $\boldsymbol{C}_{obs}$ to impute.
In practice the estimated $\boldsymbol{C}_{imp}$ will have continuous values, so *e.g* is $\boldsymbol{C}_{obs}$ is an adjacency matrix, one has to apply apply a thresholding with value 0.5 on $\boldsymbol{C}_{imp}$ to recover a completed binary adjacency matrix.

5.3 Experimental results

This section aims at illustrating the behavior of the approaches introduced so far for both clustering and classification of graph datasets (Sections 5.3.2 and 5.3.4), online subspace tracking (Section 5.3.5) and graphs completion (Section 5.2.6).

Implementation details. The base OT solvers that are used in the algorithms rely on the POT toolbox (Flamary et al., 2021). For our experiments, we considered the Adam algorithm (Kingma & Ba, 2015) as an adaptive strategy for the update of the atoms with a fixed dataset, but used SGD with constant step size for the online experiments in Section 5.3.5.

5.3.1 GDL on simulated datasets

The GDL approach discussed in this section refers to equation (5.16) where the GW distance is used as data fitting term (*i.e.* $\alpha = 1$). First we illustrate it on datasets simulated according to the well understood Stochastic Block Model (SBM, Holland et al., 1983; Wang & Wong, 1987) and show that we can recover embeddings and dictionary atoms corresponding to the generative structure.

Datasets description. We consider two datasets of graphs, generated according to SBM, with various orders, randomly sampled in $\{10, 15, ..., 60\}$:

1. The first scenario (D_1) adopts three different generative structures (also referred to as *classes*): dense (no clusters), two clusters and three clusters (see Figure 5.2). Nodes are assigned to clusters into equal proportions. For each generative structure 100 graphs are sampled.

2. The second scenario (D_2) considers the generative structure with two clusters, but with varying proportions of nodes for each block (see top of Figure 5.4), 150 graphs are simulated accordingly.

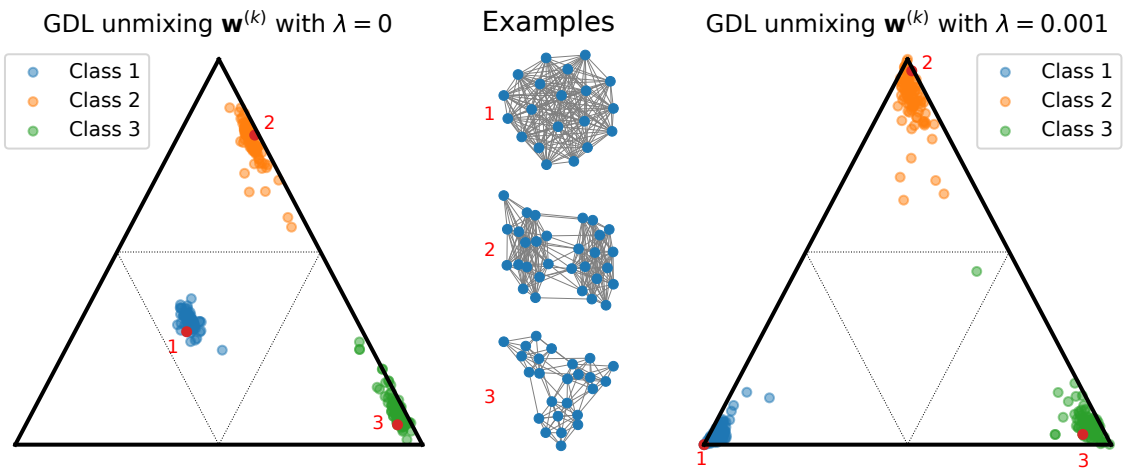

FIGURE 5.2: Visualizations of the embeddings of the graphs from D_1 with our GDL on 3 atoms. The positions on the simplex for the different classes are reported with no regularization (left) and sparsity promoting regularization (right). Three simulated graphs from D_1 are shown in the middle and their positions on the simplex reported in red.

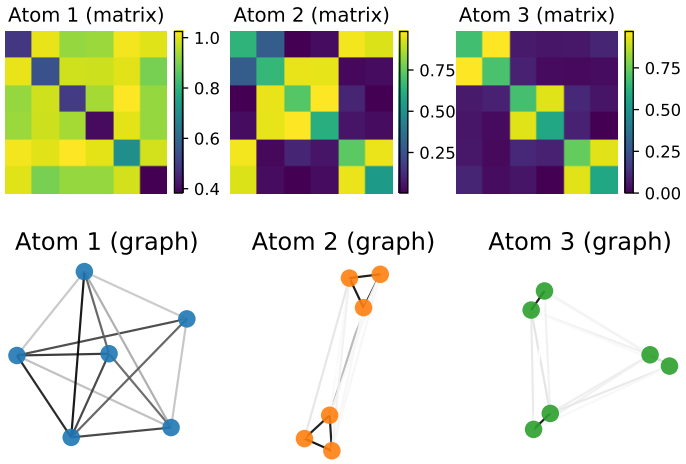

FIGURE 5.3: Visualizations of the GDL atoms learned from the graphs from D_1. Nodes of each graph atom are colored according to the labels color of the corresponding atom given in Figure 5.2. Edges are assigned with color intensities proportionally to the corresponding entries of the atom structure matrices $\overline{\boldsymbol{C}}_k$.

In both scenarios we fix $p = 0.1$ as the probability of inter-cluster connectivity and $1 - p$ as the probability of intra-cluster connectivity. We consider adjacency matrices for representing the structures of the graphs in the datasets and uniform weights on the nodes.

Results and interpretations. First we learn on dataset D_1 a dictionary of 3 atoms of order 6. The unmixing coefficients for the samples in D_1 are reported in Figure 5.2. On the left, we see that the coefficients are not sparse on the simplex but the samples are clearly well clustered and graphs sharing the same class (i.e. color) are well separated. When adding sparsity promoting regularization (right part of the figure) the different classes are clustered on the corners of the simplex, thus suggesting that regularization leads to a more discriminant representation. The estimated atoms for the regularized GDL are reported on top of Figure 5.3 as both matrices $\overline{\boldsymbol{C}}_k$ and their corresponding graphs. As it can be seen, the different SBM structures in D_1 are recovered.

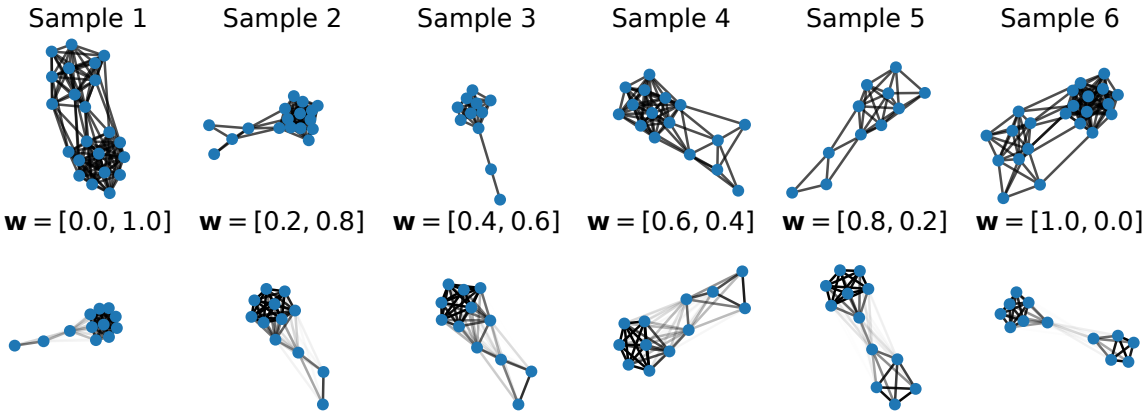

FIGURE 5.4: On the top, random samples of simulated graphs from D_2 (two blocks). On the bottom, reconstructed graphs as linear combination of two estimated atoms (varying proportions for each atom).

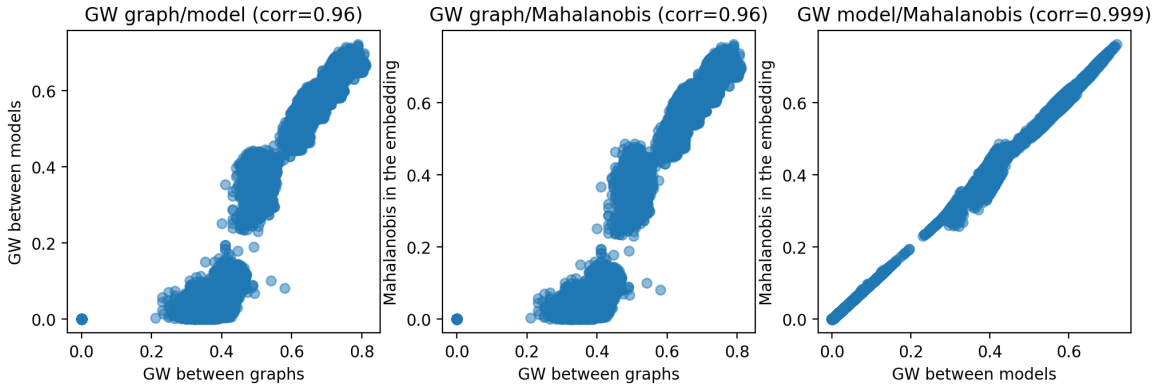

FIGURE 5.5: Plot of the pairwise distances in D_1 and their Pearson correlation coefficients. GW distance between graphs versus its counterpart between the models *i.e.* embedded graphs (left). GW distance between graphs versus Mahalanobis distance between the embeddings (middle). GW distance between the embedded graphs versus Mahalanobis between the corresponding embeddings (right).

Next we estimate on D_2 a dictionary with 2 atoms of order 12. The interpolation between the two estimated atoms for some samples is reported in Figure 5.4. As it can be seen, D_2 can be modeled as a one dimensional manifold where the proportion of nodes in each block changes continuously. We stress that the grey links on the bottom of Figure 5.4 correspond to the entries of the reconstructed adjacency matrices. Those entries are in $[0, 1]$, thus encoding a probability of connection (see Section 5.2.1). The darker the link, the higher the probability of interaction between the corresponding nodes. The possibility of generating random graphs using these probabilities opens the door to future researches.

GW approximation in the embedding. We evaluate in Figure 5.5 the quality of the Mahalanobis upper bound in Theorem 9 as a proxy for the GW distance on D_1. On the left, one can see that the linear model allows us to recover the true GW distances between graphs most of the time. Exceptions occur for samples in the same class (i.e. "near" to each other in terms of GW distance). The right part of the figure shows that the correlation between the Mahalanobis upper bound (cf. Theorem 9) and the GW distance between the embedded graphs is nearly perfect (0.999). This proves that our proposed upper bound provides a nice approximation of the GW distance between the input graphs, with a correlation of 0.96 (middle of the figure), at a much lower computational cost.

5.3.2 GDL for clustering of real-world datasets

We now show how our *unsupervised* GDL procedure can be used to find meaningful representations for well-known graph classification datasets. The knowledge of the classes will be employed as a ground truth to validate our estimated embeddings in *clustering* tasks.

Datasets and benchmark methods. We considered well-known benchmark datasets divided into three categories, whose detailed descriptions are provided in Table 8.1 of the Annex 8.2.4:

- IMDB-B and IMDB-M (Yanardag & Vishwanathan, 2015) gather graphs without node attributes derived from social networks.
- Graphs with discrete attributes representing chemical compounds from MUTAG (Debnath et al., 1991) and cuneiform signs from PTC-MR (Krichene et al., 2015).
- Graphs with real vectors as attributes, namely BZR, COX2 (Sutherland et al., 2003) and PROTEINS, ENZYMES (Borgwardt & Kriegel, 2005).

We benchmark our GDL models for clustering tasks with the following state-of-the-art OT models:

i) GWF (Xu, 2020) detailed in Section 5.1, using the proximal point algorithm to estimate FGW (see Section 3.3) and exploring two configurations, i.e. with either fixed atom order (GWF-f) or random atom order (GWF-r, default for the method).

ii) GW k-means (GW-k) which is a k-means algorithm using (F)GW distances and (F)GW barycenters (Peyré et al., 2016) (see Section 3.3.4).

iii) Spectral Clustering (SC) (Shi & Malik, 2000; Stella & Shi, 2003) applied to the pairwise GW distance matrices or the pairwise FGW distance matrices for graphs with attributes.

For both Dictionary Learning methods, namely GDL and GWF (i), we evaluate the embeddings $\{\boldsymbol{w}_i\}$ and the corresponding embedded graphs. Unlike GDL and GWF, GW-k and SC do not require any embedding learning step. Indeed, GW-k directly computes (a GW) k-means over the input graphs and SC is applied to the GW distance matrix obtained from the input graphs. We complete these clustering evaluations with an ablation study of the effect of the negative quadratic regularization proposed with our models. As introduced in equation (5.16), this regularization is parameterized by λ, so in this specific context we will distinguish GDL ($\lambda = 0$) from GDL$_\lambda$ ($\lambda > 0$).

Experimental settings. For the datasets with attributes involving FGW, we tested 15 values of the trade-off parameter α via a logspace search in $(0, 0.5)$ and symmetrically $(0.5, 1)$ and select the one minimizing our objectives. Moreover adjacency matrices are used as input representations for graphs and complementary results using shortest-path matrices are reported in Annex 8.3.3.
For our GDL methods as well as for GWF, a first step consists into learning the atoms. A variable number of $K = \beta k$ atoms is tested, where k denotes the number of classes and $\beta \in \{2, 4, 6, 8\}$, with a uniform number of atoms per class. When the order N of each atom is fixed, for GDL and GWF-f, it is set to the median order in the dataset. The atoms are initialized by randomly sampling graphs from the dataset with corresponding order. GDL atoms are learned for unattributed (resp. attributed) graphs over 50 epochs (resp. 100 epochs) with 16 as batch size and 0.01 as initial learning rate. Whereas GWF atoms are learned according to Xu (2020). We tested 4 regularization coefficients λ in for both methods,

TABLE 5.1: K-means on GDL and GWF embeddings (see (a)), and on input graphs (see (ii)). Clustering performances are measured by means of Rand Index. Best results (resp. second bests) are highlighted in bold (resp.italic). GDL performance gains *w.r.t.* all competitors are reported on the last row.

Models	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
GDL (ours)	51.3(0.30)	55.1(0.28)	71.9(0.48)	51.5(0.36)	62.6(1.68)	58.4(0.52)	71.8(0.18)	60.2(0.30)
GDL_λ (ours)	**51.7(0.59)**	55.4(0.20)	**72.3(0.17)**	**51.9(0.54)**	**66.4(1.96)**	**59.5(0.68)**	**72.9(0.28)**	**60.5(0.71)**
GWF-r	51.2 (0.02)	**55.5(0.03)**	68.8(1.47)	51.4(0.52)	52.4(2.48)	56.8(0.41)	72.1(0.19)	60.0(0.09)
GWF-f	50.5(0.34)	54.0(0.37)	59.0(1.91)	50.9(0.79)	51.7(2.96)	52.9(0.53)	71.6(0.31)	58.9(0.39)
GW-k	50.3(0.02)	53.7(0.07)	57.6(1.50)	50.4(0.35)	56.7(0.50)	52.5(0.12)	66.3(1.42)	50.1(0.01)
GDL gains	**+0.5**	-0.1	**+3.5**	**+0.5**	**+9.7**	**+2.7**	**+0.8**	**+0.5**

which relate to the sparsity promoting regularization for GDL, whereas λ relates to an entropic regularization for GWF.

The embeddings $\{\boldsymbol{w}_i\}$ and the corresponding embedded graphs are then computed and used as input for different clustering algorithms:

a) A k-means algorithm on the embeddings $\{\boldsymbol{w}_i\}$ respectively produced by both DL methods. However, whereas a standard Euclidean distance is used to implement k-means over the GWFs embeddings, we use the Mahalanobis distances given in Theorem 9 for the k-means clustering of the GDLs embeddings.

b) A Spectral Clustering algorithm on the embeddings $\{\boldsymbol{w}_i\}$ with the same geometry used than in a).

c) A Spectral Clustering algorithm on the embedded graphs corresponding to embeddings $\{\boldsymbol{w}_i\}$, *i.e* $\{(\widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i), \overline{\boldsymbol{h}})\}$ respectively given by equations (5.2) and (5.1) for GDL and GWF. For each model, the pairwise FGW distance matrix between embedded graphs is computed, then the kernel e^{-FGW} is used to perform spectral clustering.

The procedures *a)* and *b)* are the most appealing as they do not require to compute an expansive pairwise FGW distance matrix as in *c)*. However in a factorization context, the computation time of the FGW kernel for procedure *c)* can be greatly reduced compared to the one to perform spectral clustering directly on input graphs.

The cluster assignments are assessed by means of Rand Index (RI, Rand, 1971), computed between the true class assignment (known) and the one estimated by the different methods. For each parameter configuration (number of atoms, number of nodes and regularization parameter) we run each experiment five times, independently, with different random initializations. The mean RI was computed over the random initializations and the dictionary configuration leading to the highest RI was finally retained. We refer the interested reader to Annex 8.3.4 where corresponding results measured by Adjusted Rand Index (Steinley, 2004, ARI) are reported.

Results and interpretation. Clustering results achieved either by K-means algorithm (a) and Spectral Clustering algorithms (b)-(c) can be respectively seen in Tables 5.1 and 5.2. The mean RI and its standard deviation are reported for each dataset and method.

Our model outperforms or at least is comparable to the state-of-the-art OT based approaches for most of the datasets, independently of the chosen down-stream clustering algorithm. Results show that the negative quadratic regularization proposed with our models brings additional gains in performance.
Across all datasets, the evaluation of GDL thanks to Spectral clustering (SC) lead to an absolute gain of 1% to 7% in comparison to K-means. Interestingly, applying SC to the graphs

TABLE 5.2: Spectral Clustering on DL embeddings $\boldsymbol{w}$ (see (b)) and embedded graphs $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}))$, then on input graphs $(\boldsymbol{C}, \boldsymbol{F})$ (see (ii)). Clustering performances are measured by means of Rand Index (%). GDL performance gains *w.r.t.* all competitors are reported on the last row.

Models	Inputs	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
GDL (ours)	$\boldsymbol{w}$	51.9(0.19)	56.9(0.15)	**75.1(0.12)**	**52.7(0.10)**	66.5(0.01)	*66.2(0.15)*	72.1(0.15)	61.3(0.17)
GDL$_\lambda$ (ours)	$\boldsymbol{w}$	**52.2(0.24)**	57.1(0.09)	**75.1(0.05)**	*52.6(0.24)*	66.8(0.4)	*66.2(0.15)*	*73.4(0.37)*	61.7(0.25)
GDL (ours)	$\widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w})$	*52.0(0.08)*	*57.3(0.21)*	73.9(0.30)	52.6(0.18)	*67.1(0.05)*	*66.2(0.15)*	72.8(0.31)	61.9(0.20)
GDL$_\lambda$ (ours)	$\widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w})$	**52.2(0.06)**	**57.6(0.17)**	*74.5(0.01)*	52.4(0.35)	**67.5(0.11)**	**66.3(0.01)**	**73.6(0.50)**	**62.4(0.23)**
GWF-r	$\boldsymbol{w}$	51.6(0.07)	57.1(0.04)	71.6(0.18)	51.9(0.19)	55.1(0.18)	60.3(0.17)	72.7(0.12)	60.6(0.28)
GWF-f	$\boldsymbol{w}$	51.1(0.26)	55.9(0.28)	69.8(0.33)	51.4(0.07)	53.9(0.36)	58.3(0.28)	72.4(0.25)	59.7(0.21)
GWF-r	$\widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w})$	51.0(0.19)	55.6(0.1)	70.5(0.59)	50.9(0.20)	52.9(0.15)	59.9(0.26)	71.5(0.87)	60.3(0.34)
GWF-f	$\widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w})$	50.2(0.05)	54.3(0.23)	66.9(1.01)	50.9(0.31)	52.8(0.32)	58.5(0.46)	70.7(0.75)	60.2(0.40)
SC	$\boldsymbol{C}, \boldsymbol{F}$	50.4(0.03)	55.7(0.09)	*74.5(0.01)*	51.7(0.04)	53.1(4.95)	65.3(0.01)	71.2(1.00)	51.9(0.01)
GDL gains		**+0.6**	**+0.5**	**+3.5**	**+0.8**	**+12.4**	**+1.0**	**+0.9**	**+1.8**

embedded by GDL$_\lambda$ (GDL-g) lead to better performances than when the algorithm is applied to input graphs. Therefore GDL learned in an unsupervised way provides more discriminant graph representations. Moreover, SC evaluated on GDL-w also leads to better results than on input graphs and competes with GDL-g. So GDL-w provides the best trade-off between computation speed and clustering performances as the kernel required to apply SC on GDL-g is necessarily more expansive to compute.

To better understand these phenomenons, we report in Figure 5.6 the pairwise distributions between all considered distances, for the best selected models learned on MUTAG with GDL and GDL$_\lambda$. Both dictionaries are composed of 4 atoms and learned with $\alpha = 0.9997$, whereas λ takes value in $\{0, 0.01\}$.
On the right, one can see for both GDL and GDL$_\lambda$ that the linear models do not allow to recover well the FGW distances between graphs (correlations of 0.48 and 0.56), unlike our results on synthetic data sets. This was to be expected while factoring numerous graphs with heterogeneous structures and features within simply 4 atoms. However, as SC on graph models leads to comparable performances than on raw data for this dataset, the learned representations remain discriminant using both GDL and GDL$_\lambda$. We emphasize that in this scenario, GDL$_\lambda$ projects very different input graphs a priori on the graph model while clustering samples well, which highlights the denoising properties of the method.
In the middle, we can see that the Mahalanobis distances between embeddings is slightly better correlated to FGW distances between graphs than the ones between graph models, but the distributions are still quite different for the same reasons than above. Overall, both kind of correlations shown on the left and middle plots emphasize that great reconstruction and clustering performances are not that closely linked.
Finally, on the left we can observe that the correlation between the Mahalanobis upper bound and the FGW distance between embedded graphs is almost perfect (0.97) for GDL$_\lambda$ and consequently better than GDL one. So by promoting sparsity among unmixings, we enforce graph models to be *almost* aligned while preserving various and discriminant graph representations. Moreover, this plot for GDL$_\lambda$ shows that the Mahalanobis distance can sometimes be lower than the FGW distances between embedded graphs estimated using the CG solver in Algorithm 4, and then provide a better estimation of these FGW distances. These aforementioned improvements in terms of correlations and estimations might explain why the best clustering performances are achieved thanks to the SC on embeddings using the Mahalanobis distance.

To conclude on our analysis, we report in Table 5.3 averaged runtimes for the same relative precision of 10^{-4} to compute one graph embedding on learned dictionaries from datasets of social networks IMDB-B and IMDB-M, using a 3.60GHz CPU (processor i9-9900K). As

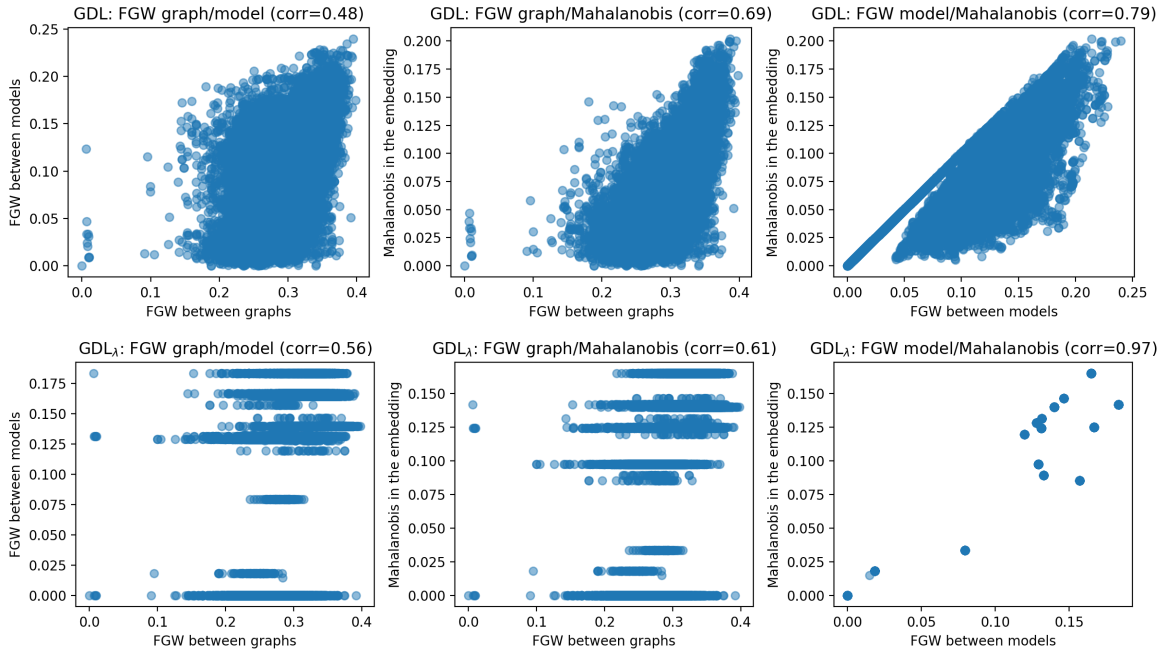

FIGURE 5.6: Plot of the pairwise distances and their Pearson correlation coefficients, for best GDL models learned on MUTAG, without (resp. with) sparsity promoting regularization on the first row (resp. second row). FGW distance between graphs versus its counterpart between the embedded graphs (left). FGW distance between graphs versus Mahalanobis distance between the embeddings (middle). FGW distance between the embedded graphs versus Mahalanobis between the corresponding embeddings (right).

TABLE 5.3: Averaged runtimes over the whole dataset to compute embeddings $\{\boldsymbol{w}_i\}$ respective to each method.

dataset	# atoms	GDL*	GWF	GWF/GDL
IMDB-B	12	52 ms	123 ms	2.4
	16	69 ms	186 ms	2.7
IMDB-M	12	44 ms	101 ms	2.3
	18	71 ms	168 ms	2.4

reported in Table 8.1 of the annex, IMDB-B contains 1000 graphs of orders 12 to 136 with a median order of 17. Whereas IMDB-M contains 1500 relatively smaller graphs of median order 10, with minimum and maximum orders of 7 and 89 respectively. Overall we observe a relative gain in speed of 2.3 to 2.7 for GDL. The best relative speed increase is achieved on biggest graphs on average for largest tested number of atoms. This further supports the use of GDL to factor datasets containing numerous and large graphs which may require larger number of atoms to be well represented.

5.3.3 Illustration of GDL dictionaries on real-world datasets

In the following, we first illustrate the graph atoms learned by GDL using the FGW distance on graphs with discrete node features from MUTAG. Then we show the relevance of our GDL extension, where a dictionary on the graph atom distributions is added, for learning representations of social networks from IMDB-B.

GDL model on attributed graphs. We represent in Figure 5.7 the atoms and corresponding closest graph samples based on their respective unmixings $\boldsymbol{w}_i$ for the GDL model leading

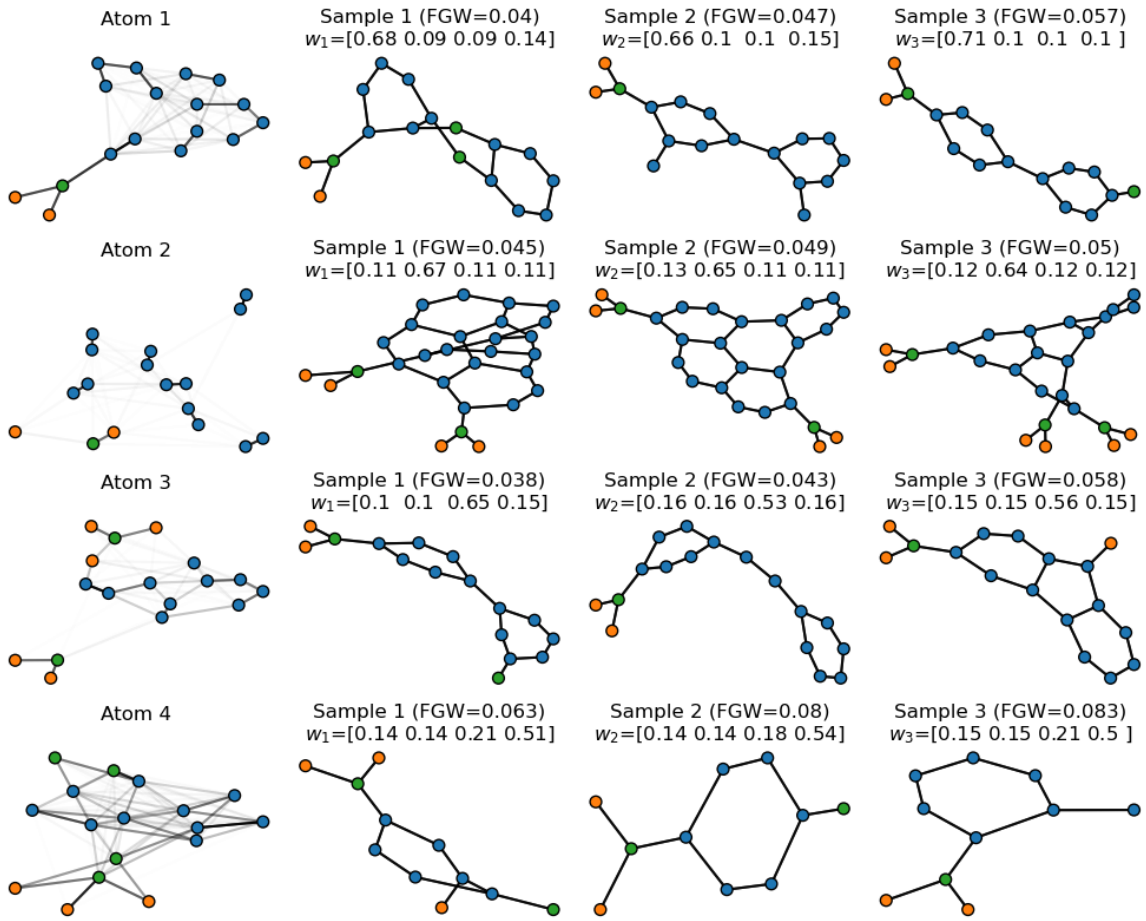

FIGURE 5.7: Atoms and their closest samples according to embeddings from GDL model learned on MUTAG with 4 atoms and $\alpha = 0.9997$. The corresponding FGW reconstruction errors are added next to samples.

to best clustering performances on MUTAG. To recover labels for learned node features, we perform a K-means algorithm on the features of graph models and retain the number of clusters whose clustering maximizes the Silhouette score. Interestingly, 3 clusters are identified, for which true node classes are recovered with an ARI of 97%, which coincides with the true class imbalance where 3 classes among 7 represent 98% of the node class distribution. That is why in Figure 5.7, node colors of the samples are assigned using true node classes, while those of the atoms are assigned using the overlap of the true and predicted node classes. Note also that the same procedure as in Figure 5.3 is used for edge colors.

This way we can see that all atoms capture rather well the ramifications composed of orange and green nodes, and the denser structures composed of blue nodes. Also the atoms recover substructures of blue nodes whose connectivity tend to be proportional to the inverse of the number of cycles and the inter-cycles connectivity in the samples. These last patterns are interesting as they involve notions of proportions which advocate learning atom distributions to enhance their interpretability. As this analysis is tedious in the presence of node features, we demonstrate how this distribution learning can be relevant in the following simplified scenario.

Extended GDL model on IMDB-B dataset. We illustrate in Figure 5.8 the interest of the extension of GDL with estimated weights for IMDB-M dataset. We can see in the middle-left part of the figure that, without estimating the weights, GDL can experience difficulties producing a model that preserves the global structure of the graph because of the

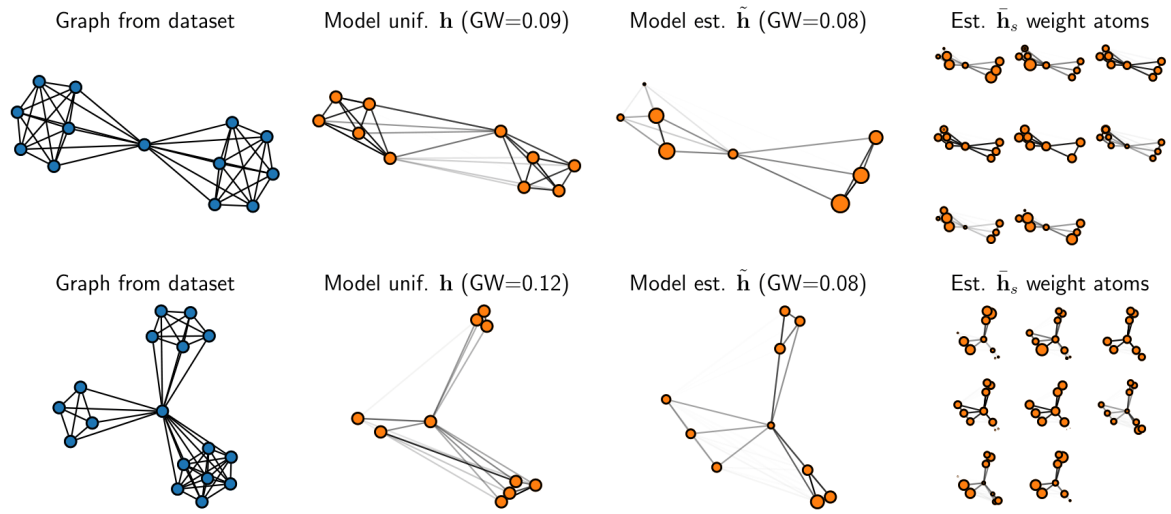

FIGURE 5.8: Modeling of two real life graphs from IMDB-M with our GDL approaches with 8 atoms of order 10. (left) Original graphs from the dataset, (middle left) linear model for GDL with uniform weights as in equation (5.16), (middle right) linear model for GDL with estimated weights as in equation (5.20) and (right) different $\overline{\boldsymbol{h}}_s$ on the estimated structure.

uniform weights on the nodes. In opposition, simultaneously estimating the weights brings a more representative modeling (in the GW sense), as illustrated in the middle-right columns. The weights estimation can re-balance and even discard non relevant nodes, in the vein of attention mechanisms.

5.3.4 GDL for classification of real-world datasets

We show in this section that the embeddings $\{\boldsymbol{w}_i\}$ and corresponding models $\{\widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i)\}$ provided by GDL can be beneficial to down-stream *supervised* classification tasks.
To this end, we train Support Vector Machines (SVM) on the embeddings and models using exponential kernels e^{-d} where d is respectively a Mahalanobis distance and a FGW distance. For each GDL configuration, one dictionary is randomly chosen among the 5 existing ones used for the clustering benchmark (Section 5.3.2). The embeddings and the Mahalanobis distance associated to the dictionary are used to form a first kernel. Then we compute a FGW_α kernel on corresponding embedded graphs, using the same α than for the DL. In the following, we denote GDL-w the SVM derived from then embeddings $\{\boldsymbol{w}_i\}$. Whereas GDL-g denotes the SVM derived from the embedded graphs $\{\widetilde{\boldsymbol{C}}(\boldsymbol{w}_i), \widetilde{\boldsymbol{F}}(\boldsymbol{w}_i)\}$.

Benchmark methods and settings. To consistently benchmark the proposed supervised evaluation of GDL embeddings, we consider the following state-of-the-art models:

i) First we benchmark our models to several graph kernel methods (see Section 2.1): (FGWK) The kernels e^{-FGW} derived from FGW (Section 3.3) distances computed on input graphs using either adjacency (ADJ) or shortest-path (SP) matrices as graph structures; (SPK) denotes the shortest path kernel (Borgwardt & Kriegel, 2005); (RWK) the random walk kernel (Gärtner et al., 2003); (WLK) the Weisfeler Lehman kernel (Vishwanathan et al., 2010); (GK) the graphlet count kernel (Shervashidze et al., 2009); (HOPPERK) the HOPPER kernel (Feragen et al., 2013); (PROPAK) the propagation kernel (Neumann et al., 2016). We build upon the GraKel library (Siglidis et al., 2020) to construct the kernels and perform the same hyperparameter validations as in Vayer et al. (2019a).
For all kernel methods under various configurations, as real graph datasets commonly

TABLE 5.4: Classification results measures by means of accuracy. Best results are highlighted in bolt independently of the model categories, and the best performances from not end-to-end supervised methods are reported in italic.

Models	IMDB-B	IMDB-M	MUTAG	PTC	BZR	COX2	ENZYMES	PROTEIN
GDL-w ADJ (ours)	70.1(3.1)	49.0(3.7)	81.1(7.8)	55.3(8.0)	87.3(3.6)	76.6(3.2)	70.7(3.4)	72.1(3.1)
GDL-w SP (ours)	65.4(3.7)	48.0(3.8)	84.6(6.7)	55.1(6.0)	84.0(5.5)	75.9(3.8)	70.0(5.0)	73.0(3.7)
GDL-g ADJ (ours)	*72.1(4.1)*	*50.6(4.4)*	85.8(6.9)	58.5(7.7)	*87.8(4.3)*	78.1(5.1)	71.4(4.2)	74.6(5.0)
GDL-g SP (ours)	68.24(4.4)	48.47(4.2)	*87.1(6.3)*	57.1(6.6)	84.6(5.9)	76.9(4.9)	71.5(6.0)	*74.9(4.4)*
FGWK ADJ	70.8(3.5)	48.9(3.9)	82.6(7.2)	56.2(8.9)	85.6(5.2)	77.0(4.2)	72.2(4.0)	72.4(4.7)
FGWK SP	65.0(3.7)	47.8(3.8)	84.4(7.3)	55.4(7.0)	84.2(6.4)	76.5(4.7)	70.5(6.2)	74.3(3.3)
GWF-r ADJ	65.1(2.9)	47.5(3.2)	-	-	83.6(5.0)	75.3(4.2)	*72.5(5.4)*	73.6(2.5)
GWF-f ADJ	64.7(2.3)	47.2(3.0)	-	-	83.7(5.1)	75.0(4.0)	72.1(5.0)	73.1(2.1)
GK (K=3)	57.1(3.5)	41.9(4.5)	82.9(7.9)	57.1(7.2)	NA	NA	NA	NA
SPK	56.2(2.9)	39.1(4.9)	83.3(8.0)	60.1(6.4)	NA	NA	NA	NA
RWK	NA	NA	79.5(7.9)	55.7(6.9)	NA	NA	NA	NA
WLK	NA	NA	86.4(8.0)	*63.1(6.6)*	NA	NA	NA	NA
HOPPERK	NA	NA	NA	NA	84.5(5.2)	*79.7(3.5)*	46.2(3.8)	72.1(3.1)
PROPAK	NA	NA	NA	NA	80.0(5.1)	77.8(3.8)	71.8(5.8)	61.7(4.5)
GIN	64.3(3.1)	50.9(1.7)	90.1(4.4)	63.1(3.9)	-	-	62.2(3.6)	76.2(2.8)
TFGW ADJ	**78.3(3.7)**	**56.8(3.1)**	**96.4(3.3)**	**72.4(5.7)**	-	-	**73.8(4.6)**	**82.9(2.7)**

used in machine learning literature show a high variance considering structure, we perform a nested cross validation (using 9 folds for training, 1 for testing, and report the average accuracy of this experiment repeated 10 times) by keeping same folds across methods. All splits are balanced *w.r.t* labels. In the following results, parameters of SVM are cross validated within $C \in \{10^{-7}, 10^{-6}, ..., 10^{7}\}$ and $\gamma \in \{2^{-10}, 2^{-9}, ..., 2^{10}\}$, using SVM implementation from Scikit-Learn (Buitinck et al., 2013).

ii) Then we include in the benchmark the end-to-end model derived from GWF (Xu, 2020). The model consists in minimizing a weighted sum of the DL reconstruction error (see equation (5.1)) and a classification loss, *e.g* cross-entropy. For a fixed dictionary state, the unmixings are computed then used to feed a Multi-Layer Perceptron (MLP) with non-linearities achieving final label predictions. Then the MLP and the graph atoms are optimized using automatic differentiation. Similarly than for the clustering benchmark we refer again to GWF-r and GWF-f when their dictionary atoms have random size (default for the method) or when we fix it to match GDL ones. We followed authors' choices for the validation of MLP architectures and trade-off hyperparameter to weight reconstruction and classification losses.

iii) Finally we include SOTA GNN models by reporting few results from Chapter 4, such as GIN with sum pooling (Xu et al., 2019c) and TFGW (Vincent-Cuaz et al., 2022c) which uses GIN as backbone and FGW distances from learnable graph templates as pooling layer while learning from ADJ or SP input graph representations.

Results on supervised classification. The benchmark results measured by means of accuracy are reported in Table 5.4. First, we observe as anticipated that the model TFGW (see Chapter 4) always outperforms other methods by significant margins, except on the dataset ENZYMES where the gains are less significant.
Interestingly (F)GW kernels over the embedded graphs (GDL-g) built thanks to our GDL approach consistently improve performances (from 1% to 3%) of (F)GW kernels over input graphs (FGWK), independently of the chosen input representations (ADJ or SP). Hence, it supports that our dictionaries (learned in an unsupervised way) are able to properly denoise and/or capture discriminant patterns of these graphs. GDL-g outperforms other models except GNN on all datasets except 2, PTC and COX2 where kernel methods WLK and HOPPERK lead respectively to the best performances.

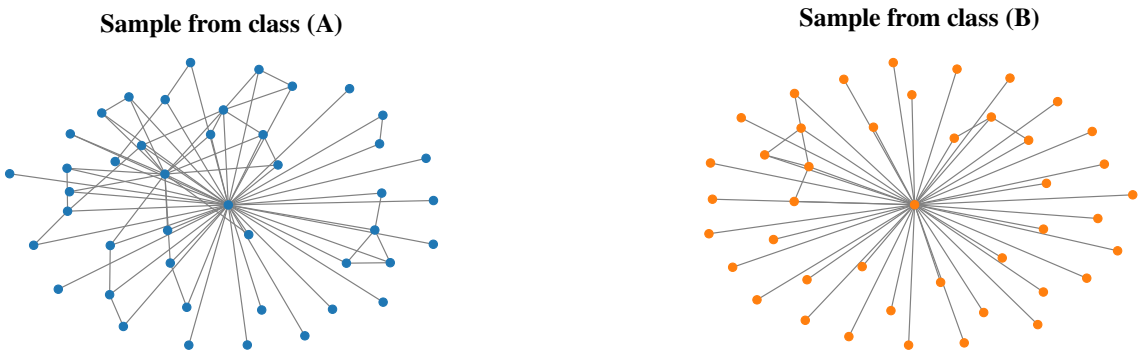

FIGURE 5.9: GDL online experiments on dataset TWITCH-EGOS: illustration of a typical sample from class A (left plot) and class B (right plot).

Moreover the kernels based on Mahalanobis distances over embeddings $\boldsymbol{w}$ (GDL-w) lead to performances comparable to FGWK. These results peculiarly support the use of our embedding schema as pre-processing for classification as GDL-w comes at a considerably lower expense than FGWK. Indeed to classify a new graph sample $\mathcal{G}$ of order N, GDL-w requires to first embed $\mathcal{G}$ within $O(N^2\overline{N})$ operations, then to compute Mahalanobis distances of its embedding to all I embeddings of the train dataset at a cost $O(IK)$ where K is the number of atoms in the dictionary. Whereas FGWK requires to compute all FGW distances from $\mathcal{G}$ to the I graphs composing its train dataset coming within $O(\sum_{i\in I} N^2 N_i + N_i^2 N)$ operations.

5.3.5 Online graph subspace estimation and change detection

In the following, we experiment on GDL for online graph subspace estimation on simulated and real life datasets. We show that our approach can be used for subspace tracking of graphs as well as for change point detection of subspaces.

Datasets and experiments. We consider here two new large graph classification datasets: TWITCH-EGOS (Rozemberczki et al., 2020) containing social graphs without attributes belonging to 2 classes and TRIANGLES (Knyazev et al., 2019) that is a simulated dataset of labeled graphs with 10 classes.

Here we investigate how our approach fits to online data, i.e in the presence of a stream of graphs. The experiments are designed with different time segments where each segment streams graphs belonging to the same classes (or group of classes). The aim is to see if the method learns the current stream and detects or adapts to abrupt changes in the stream, simulated according to the following processes:

- For TWITCH-EGOS, we first streamed all graphs of a class (A), then graphs of the other class (B), both counting more than 60.000 graphs. All these graphs consist in a unique high-frequency (a hub structure) with sparse connections between non-central nodes (sparser for class B). A typical sample from each class is illustrated in Figure 5.9.
- For TRIANGLES, the stream follows the three groups A,B and C, with 10,000 graphs each, where the labels associated with each group are: $A = \{4, 5, 6, 7\}$, $B = \{8, 9, 10\}$ and $C = \{1, 2, 3\}$.

Results and discussion The online (F)GW losses and a running mean of these losses are reported for each dataset on the left part of Figure 5.10. On the right part of the figure, we report the average losses computed on several datasets containing data from each stream at some time instant along the iterations.

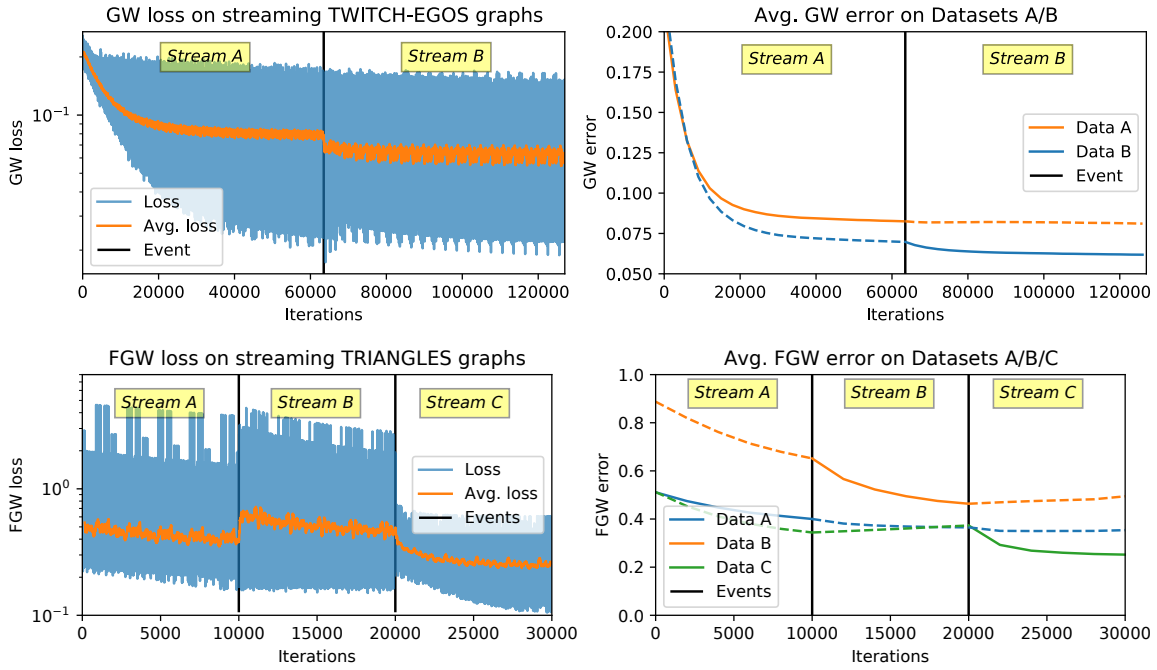

FIGURE 5.10: Online GDL on dataset TWITCH-EGOS with 2 atoms of 14 nodes each (top) and on TRIANGLES with 4 atoms of 17 nodes each (bottom).

First, the online learning for both datasets can be seen in the running means with a clear decrease of loss on each time segment. Also, note that at each event (change of stream) a jump in terms of loss is visible suggesting that the method can be used for change point detection.

Finally it is interesting to see on the TRIANGLES dataset that while the loss on Data B is clearly decreased during Stream B it increases again during Stream C, thus showing that our algorithm performs subspace tracking, adapting to the new data and forgetting old subspaces no longer necessary.

5.3.6 Applications to graph completion

Finally, we present graph completion results on the real world datasets IMDB-B and MUTAG, using the approach proposed in Section 5.2.6.

Experimental setting. Since the datasets do not explicitly contain graphs with missing nodes, we proceed as follow: first we split the dataset into a training dataset ($\mathcal{D}_{train}$) used to learn the dictionary and a test dataset ($\mathcal{D}_{test}$) reserved for the completion tasks. For each graph of $(\boldsymbol{C}, \boldsymbol{F}) \in \mathcal{D}_{test}$, we created incomplete graphs $(\boldsymbol{C}_{obs}, \boldsymbol{F}_{obs})$ by independently removing 10% and 20% of their nodes, uniformly at random. The partially observed graphs are then reconstructed using the procedure described in Section 5.2.6 and the performance of each method averaged over all imputed graphs is computed.
Note that the continuous versions of the imputed graphs $(\boldsymbol{C}_{imp}, \boldsymbol{F}_{imp})$ are not necessarily aligned with their ground truth $(\boldsymbol{C}, \boldsymbol{F})$. So a final step to access completion performances consists in aligning the imputed graphs with their respective ground truth thanks to the OT matrix $\boldsymbol{T}$ obtained by solving the (F)GW problem between both graphs. Then we apply a threshold to the aligned imputed structures $\boldsymbol{T}^\top \boldsymbol{C}_{imp} \boldsymbol{T}$ to recover an adjacency matrix estimating $\boldsymbol{C}$, and we estimate $\boldsymbol{F}$ with the aligned imputed features $\boldsymbol{T}^\top \boldsymbol{F}_{imp}$.

The hyperparameters of the dictionaries (learned exclusively on $\mathcal{D}_{train}$) are validated in the same way than in the clustering benchmark (Section 5.3.2), except that extreme values

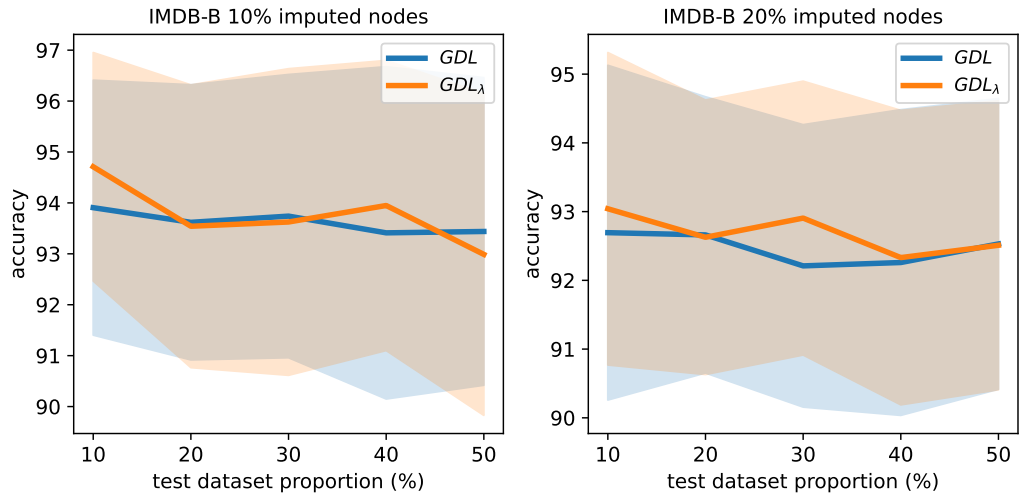

FIGURE 5.11: Completion experiments with GDL on IMDB-B dataset where 10% (right plot) and 20% (left plot) of the nodes are to be imputed, while dictionaries are learned over varying train/test dataset proportions.

0 and 1 for α are omitted. Then the unmixings inherent to our completion algorithm are computed using the same settings than the DL. By default, we initialize entries of $\boldsymbol{C}_{imp}$ by iid sampling from $\mathcal{N}(0.5, 0.01)$. For IMDB-B composed of social networks, we propose to leverage information from node degrees within $\boldsymbol{C}_{obs}$ to initialize connections between the observed nodes and the imputed ones. Specifically, new connections to a node p are sampled from $\mathcal{N}(\frac{d_p}{\max_q d_q}, 0.01)$ where for all q, d_q denotes the degree of node q observed within $\boldsymbol{C}_{obs}$. Finally for MUTAG, imputed features are initialized uniformly at random in the range of observed features.

Results on IMDB-B dataset. The completion results of unattributed graph from the IMDB-B datasets measured by accuracy are reported in Figure 5.11. Both GDL and GDL$_\lambda$ lead to comparable performances and are able to complete graphs with 10% (resp 20%) of nodes to impute with an averaged accuracy of 93%-95% (resp. 92%-93%). We observe small and gradual decreases in accuracy as the proportion of graphs used to learn the dictionaries also decreases from 90% to 50%, but these losses in performance remain reasonable. Finally, the sparsity promoting regularization seems beneficial to infer more nodes as GDL$_\lambda$ becomes slightly but consistently better than GDL across variable test dataset proportions. Since more complex structures have to be imputed when more nodes are removed, these gains support the fact that our regularized GDL leads to more refined atoms allowing such reconstruction.

Results on MUTAG dataset. We postulate for attributed graphs that the respective completions of their node connectivities and features must be performed jointly to be efficient. To this end, we propose to evaluate our completion performance on the MUTAG dataset by considering a non-preferential loss, *i.e.* one that equally considers completions of graph structures and features. This non-preferential loss reads as the mean of the accuracy taken over structures and the Pearson's linear coefficient of determination r^2 taken over features.

The performances quantified by our non-preferential losses, with the corresponding accuracies and r^2 coefficients are reported in Figure 5.12. First as for the IMDB-B dataset, we can see in these scenarios that both methods lead to good performances for this novel task of graph completion, while being rather robust to the proportion of graphs used to learn the dictionaries. Interestingly, GDL now slightly outperforms GDL$_\lambda$ considering all tested settings. This performance gap seems to grow as the proportion of nodes to be imputed

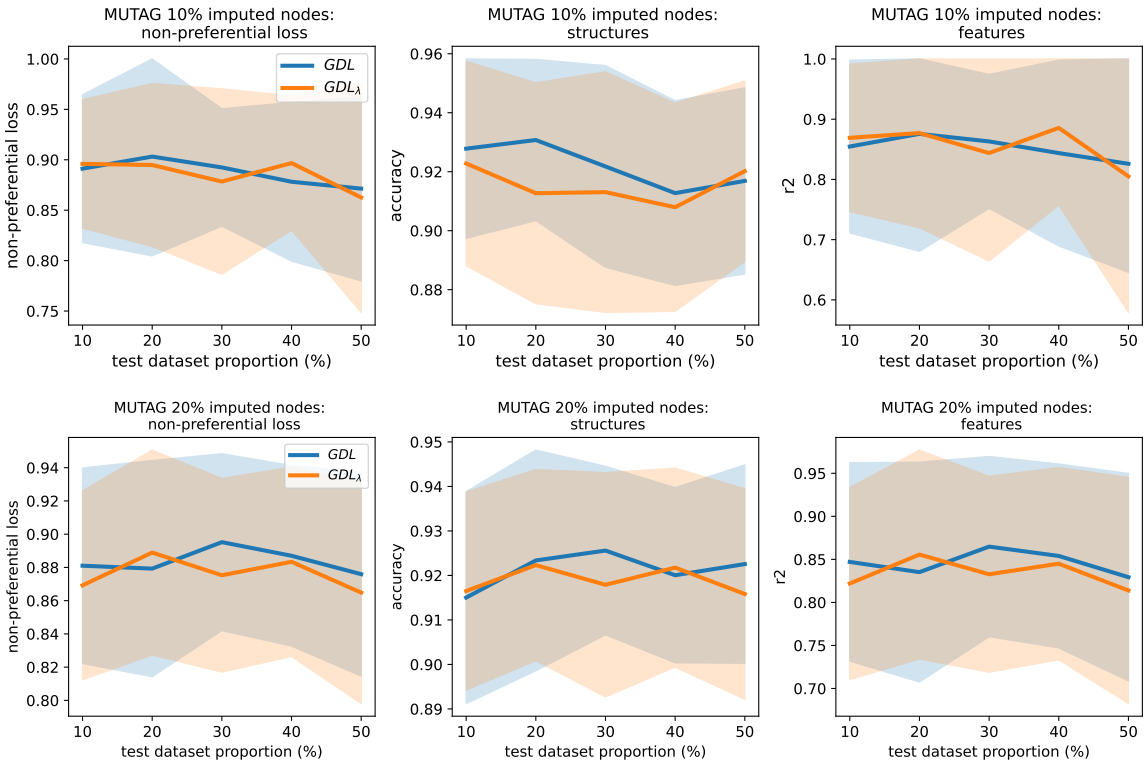

FIGURE 5.12: Completion experiments with GDL on MUTAG dataset where 10% (right plot) and 20% (left plot) of the nodes are to be imputed, while dictionaries are learned over varying train/test dataset proportions.

TABLE 5.5: Best α for each test dataset proportions for experiments on MUTAG dataset.

	10 % imputed nodes					20 % imputed nodes				
test dataset proportions (%)	10	20	30	40	50	10	20	30	40	50
GDL (α)	0.5	0.999999	0.75	0.9	0.9	0.5	0.25	0.9	0.9	0.9
GDL$_\lambda$ (α)	0.1	0.995	0.9	0.25	0.995	0.75	0.9	0.75	0.75	0.75

increases, which tends to contradict our previous analysis on the IMDB-B dataset. Since the regularized GDL can provide more refined atoms, the bottleneck for imputing more complex graphs such as attributed ones might lie in our choice of preserving the same regularization coefficient λ for the iterative unmixings in our completion scheme than in our DL. Such considerations are let for future works.

To complete our analysis, we report in Table 5.5 the trade-off parameters α leading to best performances depending on the test dataset and imputed nodes proportions. We observe that quite variable values of α are selected depending on completion settings, which advocates for the validation of this hyperparameter also in these completion tasks. Moreover, these values are not the same as the ones leading to most discriminant dictionaries (see Annex 8.3.4).

5.4 Discussion and conclusion

We introduced a new *linear* Dictionary Learning approach for graphs with different orders relying on the Fused Gromov-Wasserstein (FGW) divergence, where graphs are modeled as convex combination of graph atoms.

Such modeling requires a projection step onto the dictionary for which we provided an efficient Block Coordinate Descent algorithm. Then We design an online stochastic algorithm to efficiently learn our dictionary and propose a computationally light proxy to the (F)GW distance in the described graphs subspace.

Our experiments on discriminant tasks demonstrate the relevance of our DL, *e.g* to lead to state-of-the-art performances for the clustering of graphs dataset and to produce discriminant representations for down-stream classification tasks using Support Vector Machines. Moreover, our numerical results show the relevance of our unsupervised graph representation learning approach for online subspace tracking and graphs completion.

We envision several extensions to this work. Notably in the context of graph denoising or graph inpainting to observe whether our novel DL can be bring improvements. Then in a practical way for all tasks, it can be interesting to study new solvers for our GDL, *e.g* using entropically regularized solvers to allow faster parallelization on GPUs and potentially smooth the learning process (see Sections 3.2.4 and 3.3.3).

Finally, several theoretical future works can be envisioned for GDL. For instance, a light-proxy to (F)GW in the embedding space when learning graph distributions is yet to be found, and its search is particularly motivated by the encouraging results provided by our proxy to (F)GW when enforcing these distributions to be equal. Our proofs of concept for online subspace tracking and graphs completion motivate further investigations on these matters.

Chapter 6

Relaxing the Optimal Transport paradigm for unsupervised Graph Representation Learning

Contents

This Chapter presents the results of the papers Vincent-Cuaz et al. (2022a;b) where we proposed a relaxation of the Optimal Transport paradigm (Chapter 3) to better address challenges of unsupervised Graph Representation Learning.

Comparing structured objects such as graphs, potentially endowed with node attributes, is a fundamental operation involved in many learning tasks. To this end, the Gromov-Wasserstein (GW, Section 3.2) and Fused Gromov-Wasserstein (FGW, Section 3.3) distances, based on Optimal Transport (OT), have proven to be successful in handling the specific nature of the associated objects (see *e.g.* Chapters 4 and 5). More specifically, through the nodes connectivity relations, GW and FGW respectively operate on graphs and attributed graphs, seen as probability measures over specific spaces. At the core of OT is the idea of *conservation of mass*, which imposes a coupling between all the nodes from the two considered graphs. We argue in this chapter that this property can be detrimental for tasks such as graph dictionary (Chapter 5) or partition learning (Section 3.2.5).

Contributions. We introduce novel Optimal Transport based divergences between graphs or attributed graphs, respectively derived from the GW and FGW distances. We call them respectively the *semi-relaxed Gromov-Wasserstein* (srGW) and *semi-relaxed Fused Gromov-Wasserstein* (srFGW) divergences. Like FGW for GW, srFGW acts as a generalization of srGW, via a parameter (α) linearly weighting the importance between graph structure and feature information. Thus, in this chapter we focus on the more generic srFGW for attributed graphs. srGW (operating on unattributed graphs) will only be used to provide some insights in simplified scenarios.

After discussing srFGW properties and motivating its use in ML applications (Section 6.2.1), we propose efficient solvers for the corresponding optimization problem or regularized versions (Section 6.2.2). Our solvers better fit to modern parallel programming than exact solvers for FGW do. We empirically demonstrate the relevance of our divergence for graph partitioning (Sections 6.4.1 and 6.4.2), Dictionary Learning (DL), clustering (Section 6.4.3) or classification (Section 6.4.4) of graphs, and graph completion tasks (Section 6.3.2). With sr(F)GW, we recover or surpass SOTA performances on these tasks, at a significantly lower computational cost compared to methods based on pure (F)GW.

6.1 Introduction

As illustrated so far in this thesis, learning from graph data is an omnipresent challenge in a number of ML tasks. A first one relates to *Graph Representation Learning* (Chapter 2), that either relies on meaningful notions of similarity between graphs (Section 2.1) or graph embeddings produced by Graph Neural Networks (Section 2.2). Finally, it is often of interest either to establish meaningful structural correspondences between the nodes of different graphs, also known as *graph matching* (Zhou & De la Torre, 2012; Maron & Lipman, 2018; Bernard et al., 2018; Yan et al., 2016) or to find a representative partition of the nodes of a graph, which we refer to as *graph partitioning* (Chen et al., 2014; Nazi et al., 2019; Kawamoto et al., 2018; Bianchi et al., 2020a).

6.1.1 Optimal Transport for structured data

Based on the theory of Optimal Transport (OT) (Chapter 3), novel approaches to graph modeling have recently emerged from a series of works, partially covered in Chapters 4 and 5. Informally, the goal of OT is to match two probability distributions under the constraint of mass conservation and in order to minimize a given matching cost. OT originally tackles the problem of comparing probability distributions whose supports lie on the *same* metric space, by means of the so-called Wasserstein distance (Section 3.1).

Extensions to graph data analysis were introduced by either embedding the graphs in a space endowed with Wasserstein geometry (Nikolentzos et al., 2017; Togninalli et al., 2019; Maretic et al., 2019) or relying on the (Fused) Gromov-Wasserstein distances (Sections 3.2 and 3.3). The latter are variants of the classical OT in which one aims at comparing probability distributions whose supports lie on *different* spaces, such as graphs, potentially endowed with node features (Section 3.2). Consequently, the (F)GW distance computes both a soft assignment matrix between nodes of the two compared graphs and a notion of similarity between them (see the left part of Figure 6.1). These properties have proven to be useful for a wide range of tasks such as graph matching and partitioning (see *e.g.* experiments in Section 3.2.5), estimation of nonparametric graph models (*graphons*, Diaconis & Janson, 2007; Xu et al., 2021a) or for graph Dictionary Learning (Chapter 5).

6.1.2 On the limitations of Optimal Transport distances for graphs

Despite those recent successes, applications of (F)GW to graph modeling still have several limitations. First, computing the (F)GW distance remains challenging, as it boils down to solving a difficult non-convex Quadratic Program (equations (3.37) and (3.58)), which, in practice, limits the size of the graphs that can be processed (Sections 3.2.4 and 3.3.3). A second limit naturally emerges when a probability mass function is introduced over the set of the graph nodes. The mass associated with a node refers to its relative importance and, without prior knowledge, each node is either assumed to share the same probability mass or to have one proportional to its degree (Xu et al., 2019a).

OT for graph partitioning. The challenge of partitioning the nodes of a graph (*a.k.a* graph partitioning), by means of a GW matching, perfectly highlights the need to choose the distributions meticulously (Section 3.2.5). To further detail how this is achieved, let us recall that in the GW context, a graph $\mathcal{G}$ of order n is modeled as a pair $(\boldsymbol{C}, \boldsymbol{h})$ (Section 3.2.1). Where $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ is a matrix encoding the relationships between nodes in the topology of the graph (*i.e.* adjacency), and $\boldsymbol{h} \in \Sigma_n$ is a probability vector, referred here as distribution, modeling the relative importance of the nodes within the graph.
In the GW sense, the node partitioning of such a graph relies on its matching to an ideal graph atom $\overline{\mathcal{G}} = (\overline{\boldsymbol{D}}, \overline{\boldsymbol{h}})$ of m nodes, with one node corresponding to a cluster ($m << n$) and whose structure is encoded by a matrix $\overline{\boldsymbol{D}} \in \mathbb{R}^{m \times m}$, representing the cluster's connections, and its distribution $\overline{\boldsymbol{h}}$ estimates the proportion of the nodes in each cluster (Xu et al., 2019a). The OT plan between a graph $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{h})$ to the ideal graph $\overline{\mathcal{G}}$ can be used to recover the node clusters in $\mathcal{G}$. The graph partitioning is expected to be good, at least if the (true) number of clusters corresponds to the number of nodes in $\overline{\mathcal{G}}$, and the proportions of the clusters correspond to the node weights $\overline{\boldsymbol{h}}$. Xu et al. (2019a) suggested to empirically refine both graph distributions, by choosing $\boldsymbol{h}$ based on power-law transformations of the degree distribution of $\mathcal{G}$ and to deduce $\overline{\boldsymbol{h}}$ from $\boldsymbol{h}$ by linear interpolation. Chowdhury & Needham (2021) proposed to use heat kernels $(e^{-\lambda \boldsymbol{L}}, \lambda \in \mathbb{R}_+)$ from the Laplacian $\boldsymbol{L}$ of $\mathcal{G}$ (equation (2.5)), instead of its adjacency binary representation, and proved that the resulting GW partitioning is closely related to the well-known Spectral Clustering (Fiedler, 1973).

(F)GW for unsupervised Graph Representation Learning. (F)GW has also been used as a data fitting term for unsupervised Graph Representation Learning by means of Dictionary Learning (DL) (Chapter 5). In a nutshell, DL applied to graphs datasets consists in factorizing them as composition of graph primitives (or atoms) encoded as $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{h}}_k)\}_{k \in [\![K]\!]}$. The first approach proposed by Xu (2020) consists in a non-linear DL based on entropic GW barycenters (see equation (5.1)). On the other hand, we presented in Chapter 5 a *linear* graph DL method (GDL), by modeling graphs as a linear combination of graph atoms thus reducing the computational cost. We also observed limitation of GDL to obtain representative modeling for certain graphs when considering uniform weights on the nodes of graph atoms (Section 5.3.3). To this end, we extended GDL to simultaneously handle structure and distribution learning via respective and independent linear dictionaries (Section 5.2.5). In all cases, the *embedding problem* that consists in the projection of any graph on the learned graph subspace requires solving a computationally intensive optimization problem.

In this chapter, we argue that the *mass conservation* paradigm inherent to the (F)GW distance is detrimental to many applications, as the aforementioned ones, and *de facto* should be relaxed, with the additional benefit of lowering the computational complexity. Notice that various relaxations of this notion of conservation in OT exist, among which the Unbalanced (F)GW problems (Séjourné et al., 2021; Chapel et al., 2019) most resembles

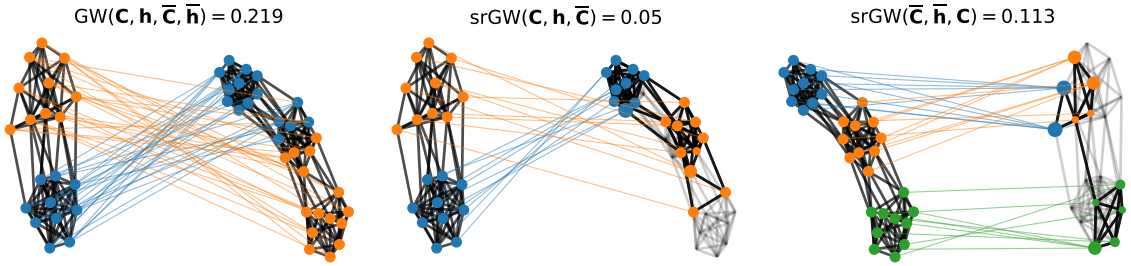

FIGURE 6.1: Comparison of the GW matching (left) and asymmetric srGW matchings (middle and right) between graphs $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ with uniform distributions. Nodes of the source graph are colored based on their clusters. The OT from the source to the target nodes is represented by arcs colored depending on the corresponding source node color. The nodes in the target graph are colored by averaging the (rgb) color of the source nodes, weighted by the entries of the OT plan.

our novel *semi-relaxed* (F)GW problems, which can be considered as an unexplored extreme scenario of the latter.

As a comparison between GW and our semi-relaxed version srGW, consider Figure 6.1: on the left image, the GW matching is given between two graphs, with respectively two and three clusters, associated with uniform weights on the nodes. By relaxing the weight constraints over the second (middle image) or first graph (right image) we obtain different matchings, that can better preserve the structure of the source graph by reweighing the target nodes and thus selecting a meaningful subgraph.

6.2 The semi-relaxed Fused Gromov-Wasserstein divergence

In what follows, we formally introduce our novel OT-based divergences between graphs and investigate their properties (Section 6.2.1). Then, we propose solvers to tackle the optimization problem inherent to the computation of these divergences and of some regularized versions (Section 6.2.2).

6.2.1 Definition and properties

Problem formulation. Given two observed attributed graphs $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and $\overline{\mathcal{G}} = (\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$ of n and $\overline{n}$ nodes, we propose to find a correspondence between them while relaxing the weights $\overline{\boldsymbol{h}}$ on the second graph. To this end we introduce the *semi-relaxed Fused Gromov-Wasserstein divergence* as :

$$\text{srFGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}) = \min_{\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}} \text{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \tag{6.1}$$

for any $\alpha \in [0, 1]$, which might be referred as srFGW in the following for conciseness. Note that a more generic setting relates to $\text{srFGW}_{p,\alpha}$, derived from $\text{FGW}_{p,\alpha}$ (Section 3.3.1), for any $p \in \mathbb{N}^\star$. Here we adopt $p = 2$ for computational reasons as discussed in Section 3.3.3.

Problem (6.1) means that we search for a reweighing of the nodes of $\overline{\mathcal{G}}$ leading to an attributed graph with structure $\overline{\boldsymbol{C}}$ and node features $\overline{\boldsymbol{F}}$, with minimal FGW distance from $\mathcal{G}$. We emphasize that setting $\alpha = 1$ in Equation (6.1) comes down to minimizing the GW distance between $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ *w.r.t* $\overline{\boldsymbol{h}}$. Hence this latter problem will be referred as the *semi-relaxed Gromov-Wasserstein divergence*, denoted srGW.

While the optimization problem (6.1) above might seem complex to solve, the next Proposition states that it is actually equivalent to a FGW problem where the mass constraints on the second marginal of $\boldsymbol{T}$ are relaxed.

Proposition 2 (srFGW equivalent problem) *Problem (6.1) is equivalent to the following optimization problem:*

$$\text{srFGW}^2_{2,\alpha}(\boldsymbol{C},\boldsymbol{F},\boldsymbol{h},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}}) = \min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \sum_{ijkl} \left\{\alpha|C_{ij}-\overline{C}_{kl}|^2 + (1-\alpha)\|\boldsymbol{F}_i-\overline{\boldsymbol{F}}_k\|_2^2\right\} T_{ik}T_{jl} \tag{6.2}$$

where $\mathcal{U}_{\overline{n}}(\boldsymbol{h})$ denotes the set of admissible coupling with first marginal $\boldsymbol{h}$ and relaxed second marginal:

$$\mathcal{U}_{\overline{n}}(\boldsymbol{h}) = \left\{\boldsymbol{T}\in\mathbb{R}_+^{n\times\overline{n}} | \boldsymbol{T}\mathbf{1}_{\overline{n}} = \boldsymbol{h}\right\} \tag{6.3}$$

Proof of Proposition 2. See annex 8.4.1.

From an optimal coupling $\boldsymbol{T}^\star$ of problem (6.2), the optimal weights $\overline{\boldsymbol{h}}^\star$ expressed in problem (6.1) can be recovered by computing $\boldsymbol{T}^\star$'s second marginal, *i.e* $\overline{\boldsymbol{h}}^\star = \boldsymbol{T}^{\star\top}\mathbf{1}_n$. This reformulation with relaxed marginal has been investigated in the context of the Wasserstein distance (Rabin et al., 2014; Flamary et al., 2016b) and for relaxations of the GW problem in Schmitzer & Schnörr (2013) but was never investigated for the GW distance itself. To the best of our knowledge, the most similar related work is the Unbalanced (F)GW (Séjourné et al., 2022; Thual et al., 2022; Séjourné et al., 2021; Liu et al., 2020; Chapel et al., 2019). Briefly, the unbalanced OT problems consist in relaxing the mass conservation constraints inherent to balanced problems (like GW or FGW), hence allowing mass creation or destruction in the matching. This is achieved by seeking for a coupling $\boldsymbol{U}\in\mathbb{R}_+^{n\times\overline{n}}$, that minimizes a transportation cost penalized by two divergence terms (*e.g.* Kullback-Leibler divergences), quantifying the deviations of $\boldsymbol{U}$'s marginals from prior marginals, and respectively weighted by coefficients ρ_s and ρ_t. Thus one could recover sr(F)GW with different weighting over the marginal relaxations ($\rho_s=\infty$ on the first marginal and $\rho_t=0$ on the second) but this specific case was not discussed nor studied in those (or previous) works.

Properties. A first interesting property of srFGW is that since $\overline{\boldsymbol{h}}$ is optimized in the simplex $\Sigma_{\overline{n}}$, its optimal value $\overline{\boldsymbol{h}}^\star$ can be sparse. As a consequence, parts of the graph $\overline{\mathcal{G}}$ can be omitted in the comparison, similarly to a partial matching. This behavior is illustrated for srGW in the Figure 6.1, where two unattributed graphs with uniform distributions and structures $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ forming respectively 2 and 3 clusters are matched. The GW matching (left) between both graphs forces nodes of different clusters from $\boldsymbol{C}$ to be transported on one of the three clusters of $\overline{\boldsymbol{C}}$, leading to a high GW cost where clusters are not preserved. Whereas srGW can find a reasonable approximation of the structure of the left graph either though transporting on only two clusters (middle) or finding a structure with 3 clusters in a subgraph of the target graph with two clusters (right).

A second natural observation resulting from the dependence of $\text{srFGW}_{2,\alpha}$ to only one input distribution is its asymmetry, *i.e.* $\text{srFGW}_{2,\alpha}(\boldsymbol{C},\boldsymbol{F},\boldsymbol{h},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}}) \neq \text{srFGW}_{2,\alpha}(\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\overline{\boldsymbol{h}},\boldsymbol{C},\boldsymbol{F})$. Interestingly, $\text{srFGW}_{2,\alpha}$ shares similar properties than FGW as described in the next lemma:

Lemma 6 (srFGW properties) *For any attributed graphs $\mathcal{G}_1 = (\boldsymbol{C}_1,\boldsymbol{F}_1,\boldsymbol{h}_1)$ and $\mathcal{G}_2 = (\boldsymbol{C}_2,\boldsymbol{F}_2,\boldsymbol{h}_2)$, the following assertions hold true for any $\alpha\in[0,1]$*

i) $\text{srFGW}_{2,\alpha}(\boldsymbol{C}_1,\boldsymbol{F}_1,\boldsymbol{h}_1,\boldsymbol{C}_2,\boldsymbol{F}_2) = 0$ *iff there exists a reweighed sub-graph of $\mathcal{G}_2$ which is weakly isomorphic to $\mathcal{G}_1$ (Definition 4).*

ii) If $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are distance matrices then $\text{srFGW}_{2,\alpha}(\boldsymbol{C}_1,\boldsymbol{F}_1,\boldsymbol{h}_1,\boldsymbol{C}_2,\boldsymbol{F}_2) = 0$ *iff there exists a reweighed sub-graph of $\mathcal{G}_2$ which is strongly isomorphic to $\mathcal{G}_1$ (Definition 3)*

Then considering any third attributed graph $\mathcal{G}_3 = (\boldsymbol{C}_3, \boldsymbol{F}_3, \boldsymbol{h}_3)$*, and* $\alpha \in]0,1[$*,*

iii) For any optimal reweighing $\boldsymbol{h}^\star_{(1|2)}$ *from* $\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2)$*, we have*

$$\begin{aligned}\mathrm{srFGW}^2_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_3, \boldsymbol{F}_3) \leq 2(&\,\mathrm{srFGW}^2_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2) \\ &+ \mathrm{srFGW}^2_{2,\alpha}(\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}^\star_{(1|2)}, \boldsymbol{C}_3, \boldsymbol{F}_3))\end{aligned} \tag{6.4}$$

iv) For any optimal reweighing $\boldsymbol{h}^\star_{(1,2)}$ *from* $\mathrm{srGW}(\boldsymbol{C}_1, \boldsymbol{h}_1, \boldsymbol{C}_2)$*, we have*

$$\mathrm{srGW}^2_2(\boldsymbol{C}_1, \boldsymbol{h}_1, \boldsymbol{C}_3) \leq 2\left(\mathrm{srGW}^2_2(\boldsymbol{C}_1, \boldsymbol{h}_1, \boldsymbol{C}_2) + \mathrm{srGW}^2_2(\boldsymbol{C}_2, \boldsymbol{h}^\star_{(1|2)}, \boldsymbol{C}_3)\right) \tag{6.5}$$

Proof of Lemma 6. See Annex 8.4.2.

In other words the assertions i) and ii) say that $\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ vanishes iff there exists a reweighing $\overline{\boldsymbol{h}}^\star \in \Sigma_{\overline{n}}$ of the nodes of the second graph which cancels the $\mathrm{FGW}_{2,\alpha}$ distance. When it is the case, the induced graphs $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}^\star)$ are weakly or strongly isomorphic depending on the set of structure matrices considered (see Definitions 3 and 4 in Chapter 3). Then assertions iii) and iv) *resemble* (relaxed) triangular inequalities, up to the fact that they depend on a specific reweighing $\boldsymbol{h}^*_{(1|2)}$ of the intermediate graph $\mathcal{G}_2$ as $(\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}^*_{(1|2)})$ given from the srFGW projection of $\mathcal{G}_1$ onto $\mathcal{G}_2$. Overall these assertions suggest that sr(F)GW could define a quasi-metric onto the space of measurable networks quotiented by a specific subgraph isomorphism notion. However, a such assertion is for the moment a mere conjecture that requires further investigation left for future works. In the following, we rather focus on the benefits springing from our divergences in applications.

6.2.2 Optimization and algorithms

In this section we discuss the computational aspects of the srFGW divergence and propose an algorithm to solve the related optimization problem (6.2). We also discuss variations resulting from entropic or/and sparse regularization(s) of the initial problem.

Solving for srFGW. First, notice that the srFGW problem (6.2) is equivalent to the following non-convex Quadratic Program:

$$\min_{\boldsymbol{T} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h})} \alpha \,\mathrm{vec}(\boldsymbol{T})^\top \left(\overline{\boldsymbol{C}}^2 \otimes_K \mathbf{1}_n \mathbf{1}_n^\top - 2\overline{\boldsymbol{C}} \otimes_K \boldsymbol{C}\right) \mathrm{vec}(\boldsymbol{T}) + (1-\alpha)\,\mathrm{vec}(\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}))^\top \mathrm{vec}(\boldsymbol{T}) \tag{6.6}$$

where $\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) = \left(\|\boldsymbol{F}_i - \overline{\boldsymbol{F}}_j\|_2^2\right)_{i,j \in [\![n]\!] \times [\![\overline{n}]\!]}$ is the matrix of pairwise Euclidean distances between node features. Hence the srFGW problem is a non-convex Quadratic Program similar to the one of FGW with the important difference that the linear constraints are independent.

Consequently, we propose to solve problem (6.2) with a Conditional Gradient (CG) algorithm (Jaggi, 2013) that can benefit from those independent constraints and is known to converge to local stationary point on non-convex problems (Lacoste-Julien, 2016).
This algorithm (Algorithm 12), consists in solving at each iteration (t) a linearization $\boldsymbol{X} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h}) \to \langle \boldsymbol{X}, \boldsymbol{G}^{(t)} \rangle$ of (6.2), given by equation (6.7), where $\boldsymbol{G}^{(t)}$ is the gradient of the objective in (6.2). The solution of the linearized problem provides a *descent direction* $\boldsymbol{X}^{(t)} - \boldsymbol{T}^{(t)}$. Then a line-search is performed, and the optimal step size γ^* can be found in closed form, adapting the line-search step used in the CG solver for FGW (See Algorithms 4 and 5).
While both srFGW and FGW CG require at each iteration the computation of a gradient with complexity $O(n^2\overline{n} + \overline{n}^2 n)$ (Section 3.2.4), the main source of efficiency of our algorithm comes from the computation of the descent directions (6.7). In the FGW case, one needs to solve

Algorithm 12 CG solver for srFGW

1: Inputs: Input graph $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$, target graph $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$, trade-off parameter α.
2: **repeat**
3: $\quad \boldsymbol{G}^{(t)} \leftarrow$ Compute gradient w.r.t $\boldsymbol{T}$ of (6.1) given by equation (3.61).
4: $\quad$ Find the descent direction solving:

$$\boldsymbol{X}^{(t)} \leftarrow \min_{\substack{\boldsymbol{X}\mathbf{1}_m = \boldsymbol{h} \\ \boldsymbol{X} \geq 0}} \langle \boldsymbol{X}, \boldsymbol{G}^{(t)} \rangle_F \tag{6.7}$$

5: $\quad \boldsymbol{T}^{(t+1)} \leftarrow \boldsymbol{T}^{(t)} + \gamma^\star(\boldsymbol{X}^{(t)} - \boldsymbol{T}^{(t)})$ with $\gamma^\star \in [0, 1]$ from exact-line search detailed in Annex 8.4.3.
6: **until** convergence.

an exact linear OT problem, while in our case, one just needs to solve several *independent linear problems* under a simplex constraint:

$$\forall i \in [\![n]\!], \quad \boldsymbol{X}^{(t)}_{i,:} \leftarrow \min_{\boldsymbol{x} \in h_i \Sigma_{\overline{n}}} \langle \boldsymbol{x}, \boldsymbol{G}^{(t)}_{i,:} \rangle_2 \tag{6.8}$$

This simply amounts to finding minimum over each row $\boldsymbol{G}^{(t)}_{i,:}$ of $\boldsymbol{G}^{(t)}$, as discussed *e.g* in Section 5.2.2, within $O(n\overline{n})$ operations, that can be parallelized with GPUs. Performance gains are illustrated in the experimental Sections 6.4.1, 6.4.3 and 6.4.4.

Entropic regularization. Recent OT applications have shown the interest of adding an entropic regularization to the exact problem (6.2), *e.g.* Cuturi (2013); Peyré et al. (2016). It makes the optimization problem smoother and more robust while densifying the optimal transport plans from the unregularized problem. Similarly to Peyré et al. (2016); Xu et al. (2019b); Xie et al. (2020b), we can use a *Mirror-Descent* (MD) scheme *w.r.t.* the Kullback-Leibler divergence (KL) to solve entropically regularized srFGW.
The problem boils down to find, at iteration (t), the coupling $\boldsymbol{T}^{(t+1)} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h})$ that minimizes the first-order approximation of the FGW cost function $\mathcal{E}^{FGW}_\alpha$ (see equation 3.57) evaluated at $\boldsymbol{T}^{(t)}$, with an added KL proximity term:

$$\boldsymbol{T}^{(t+1)} \leftarrow \operatorname*{arg\,min}_{\boldsymbol{T} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h})} \mathcal{E}^{FGW}_\alpha(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}^{(t)}) + \langle \boldsymbol{T} - \boldsymbol{T}^{(t)}, \boldsymbol{G}^{(t)} \rangle_F + \varepsilon \, \mathrm{KL}(\boldsymbol{T} | \boldsymbol{T}^{(t)}) \tag{6.9}$$

Where $\varepsilon > 0$, $\boldsymbol{G}^{(t)}$ denotes the gradient of the FGW loss at iteration (t) and $\mathrm{KL}(\boldsymbol{T}|\boldsymbol{T}^{(t)}) = \sum_{ij} T_{i,j} \log(\frac{T_{i,j}}{T^{(t)}_{i,j}}) - T_{i,j} + T^{(t)}_{i,j}$ is the KL divergence between the variable $\boldsymbol{T}$ and the current estimate $\boldsymbol{T}^{(t)}$. Problem (6.9) can be reduced to the next equivalent problem:

$$\boldsymbol{T}^{(t+1)} \leftarrow \operatorname*{arg\,min}_{\boldsymbol{T} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h})} \langle \boldsymbol{T}, \boldsymbol{G}^{(t)} \rangle_F + \varepsilon \, \mathrm{KL}(\boldsymbol{T} | \boldsymbol{T}^{(t)}) \tag{6.10}$$

Let us denote the Entropy of any $\boldsymbol{T} \in \mathbb{R}^{n \times m}_+$ by $\mathrm{H}(\boldsymbol{T}) = -\sum_{ij} T_{ij}(\log T_{ij} - 1)$, then the following relation can be proven

$$\langle \boldsymbol{G}^{(t)}, \boldsymbol{T} \rangle_F - \varepsilon \, \mathrm{H}(\boldsymbol{T}) = \varepsilon \, \mathrm{KL}\left(\boldsymbol{T} | \exp(-\frac{\boldsymbol{G}^{(t)}}{\varepsilon})\right) \Leftrightarrow \varepsilon \, \mathrm{KL}(\boldsymbol{T} | \boldsymbol{G}^{(t)}) = \langle -\varepsilon \log \boldsymbol{G}^{(t)}, \boldsymbol{T} \rangle_F - \varepsilon \, \mathrm{H}(\boldsymbol{T}) \tag{6.11}$$

Algorithm 13 MD solver for entropic srFGW.

1: Inputs: Input and target graphs, trade-off parameter α, entropic parameter ε.
2: **repeat**
3: $\boldsymbol{G}^{(t)} \leftarrow$ Compute gradient *w.r.t* $\boldsymbol{T}$ of (6.1) given by equation (3.61) and applied in $\boldsymbol{T}^{(t)}$.
4: Compute the matrix $\boldsymbol{K}^{(t)}(\varepsilon)$ following equation (6.14).
5: Get $\boldsymbol{T}^{(t+1)}$ with the scaling of $\boldsymbol{K}^{(t)}(\epsilon)$ following equation (6.15).
6: **until** convergence based on the loss relative variation between steps (t) and $(t+1)$.

and leads to this equivalent formulation of equation (6.10):

$$\boldsymbol{T}^{(t+1)} \leftarrow \underset{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})}{\arg\min} \langle \boldsymbol{G}^{(t)} - \varepsilon \log \boldsymbol{T}^{(t)}, \boldsymbol{T}\rangle_F - \varepsilon\, \mathrm{H}(\boldsymbol{T}). \tag{6.12}$$

Denoting $\boldsymbol{M}^{(t)}(\varepsilon) = \boldsymbol{G}^{(t)} - \varepsilon \log \boldsymbol{T}^{(t)}$, overall we end up with the following iteration

$$\boldsymbol{T}^{(t+1)} \leftarrow \underset{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})}{\arg\min} \langle \boldsymbol{M}^{(t)}(\varepsilon), \boldsymbol{T}\rangle_F - \varepsilon\, \mathrm{H}(\boldsymbol{T}). \tag{6.13}$$

which is equivalent to

$$\boldsymbol{T}^{(t+1)} \leftarrow \underset{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})}{\arg\min}\, \varepsilon\, \mathrm{KL}(\boldsymbol{K}^{(t)}(\varepsilon)|\boldsymbol{T}) \quad \text{where} \quad \boldsymbol{K}^{(t)}(\varepsilon) := \exp\{-\boldsymbol{M}^{(t)}(\varepsilon)/\varepsilon\}. \tag{6.14}$$

in force of equation (6.11). Following the seminal work of Benamou et al. (2015), an optimal solution $\boldsymbol{T}^{(t+1)}$ to problem (6.14) is given by a simple scaling of the matrix $\boldsymbol{K}^{(t)}(\varepsilon)$ reading as

$$\boldsymbol{T}^{(t+1)} = \mathrm{diag}\left(\frac{\boldsymbol{h}}{\boldsymbol{K}^{(t)}(\varepsilon)\mathbf{1}_{\overline{n}}}\right)\boldsymbol{K}^{(t)}(\varepsilon). \tag{6.15}$$

The resulting mirror-descent algorithm is summarized in Alg.13. This approach is adapted to srGW by simply changing the gradient accordingly.

Even if the objective function of the srFGW problem is not convex, one can prove the following convergence guaranty of our MD algorithm:

Proposition 3 (Mirror-Descent convergence) *For any attributed graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$, the Mirror-Descent algorithm 13 converges to a stationary point non-asymptotically.*

Proof of Proposition 3. The proof reported in Annex 8.4.4 provides the non-asymptotic stationary convergence of the MD algorithms for a slightly more generic objective which encompasses various sr(F)GW regularized objectives:

$$\min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \langle a\boldsymbol{L}(\boldsymbol{C},\overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + b\boldsymbol{D}, \boldsymbol{T}\rangle_F, \tag{6.16}$$

where $(a,b) \in \mathbb{R}_+^2$ denote any positive scalar constants, $\boldsymbol{L}(\boldsymbol{C},\overline{\boldsymbol{C}})$ is a 4D tensor (see *e.g.* problem (3.34)) and $\boldsymbol{D}$ any matrix in $\mathbb{R}^{n\times\overline{n}}$. This proof of convergence relies on sufficient conditions provided by Scetbon et al. (2021a), Proposition 2, among which the smoothness of the generic objective function in equation (6.16), relatively to the entropy H on the set $\mathcal{U}_{\overline{n}}(\boldsymbol{h})$. The proof of the relative smoothness of our objective function is handled by using a sufficient condition given by Zhang et al. (2020), which boils down to proving the next Lemma:

Lemma 7 *For any* $(a, b, \varepsilon) \in \mathbb{R}_+^3$ *and* $\boldsymbol{T} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h})$*, we denote the reduced objective function from equation* (6.16) *by*

$$F_{a,b,\varepsilon}(\boldsymbol{T}) = a\langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle_F - 2a\langle \boldsymbol{T}\overline{\boldsymbol{C}}\boldsymbol{T}^\top, \boldsymbol{C}\rangle_F + b\langle \boldsymbol{D}, \boldsymbol{T}\rangle_F - \varepsilon H(\boldsymbol{T}) \tag{6.17}$$

Then for all $\boldsymbol{T}_1$ *and* $\boldsymbol{T}_2$ *in* $\mathcal{U}_{\overline{n}}(\boldsymbol{h})$*, we have*

$$\|\nabla F_{a,b,\varepsilon}(\boldsymbol{T}_1) - \nabla F_{a,b,\varepsilon}(\boldsymbol{T}_2)\|_F \leq L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon} \|\nabla H(\boldsymbol{T}_1) - \nabla H(\boldsymbol{T}_2)\|_F \tag{6.18}$$

where $L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon} = a\{n\|\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2\|_F + 4\|\boldsymbol{C}\|_F\|\overline{\boldsymbol{C}}\|_F\} + \varepsilon$.

Overall, Algorithm 13 ensures the convergence to a local optimum $\boldsymbol{T}^*$ of the regularized srFGW problem, that lies in the interior of the set $\mathcal{U}_{\overline{n}}(\boldsymbol{h})$, as $\boldsymbol{T}^*$ can not have null entries. As such, the entropic regularization of srFGW problem induces a bias as $\text{srFGW}_\varepsilon(\mathcal{G}, \mathcal{G}) \neq 0$, in the vein of the linear OT problem Genevay et al. (2018). We *empirically* investigate the effect of these biased solutions (Section 6.4), and let the study of unbiased versions for future works.

Sparsity promoting regularization. As illustrated in Figure 6.1, srGW naturally leads to sparse solutions in $\overline{\boldsymbol{h}}$. To compress even more the localization over a few nodes of $\overline{\boldsymbol{C}}$, we can promote the sparsity of $\overline{\boldsymbol{h}}$ through a penalization $\Omega(\boldsymbol{T}) = \sum_j \sqrt{\sum_i T_{ij}} = \sum_j \sqrt{\overline{h}_j}$ which defines a concave function in the positive orthant $\mathbb{R}_+^{n\times\overline{n}}$. This results in the following optimization problem:

$$\min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}) + \lambda_g \Omega(\boldsymbol{T}) \tag{6.19}$$

with $\lambda_g \in \mathbb{R}_+$ an hyperparameter. Note that an analog procedure can be applied when an entropic regularization is already enforced. In both scenarios, we adapt the Majorisation-Minimisation (MM) of Courty et al. (2014) that was introduced to solve classical OT with a similar regularization. MM consists in iteratively minimizing an upper bound of the objective function which is tight at the current iterate (Hunter & Lange, 2004). With this procedure, the objective function is guaranteed to decrease at every iteration. To this end we only need to major the penalty term $\Omega(\boldsymbol{T})$ to obtain a tractable function. Consequently, we consider the concavity of the composite functions $\overline{h}_j \to \sqrt{\overline{h}_j}$ of Ω, using the tangent inequalities:

$$\sqrt{\overline{h}_j} \leq \sqrt{\overline{h}_j^{(t)}} + \frac{1}{2\sqrt{\overline{h}_j^{(t)}}}(\overline{h}_j - \overline{h}_j^{(t)}) \tag{6.20}$$

for all $j \in [\![\overline{n}]\!]$. This essentially linearizes the regularization terms at $\overline{\boldsymbol{h}}^{(t)}$ whose contributions can then be absorbed into the inner product inherent to $\mathcal{E}_\alpha^{FGW}$ (see equation (3.54)), leading to the intermediate optimization problem:

$$\min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \langle \alpha \boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) + \boldsymbol{R}^{(t)}, \boldsymbol{T}\rangle \tag{6.21}$$

where $\boldsymbol{R}^{(t)} = \frac{\lambda_g}{2}\left(\overline{h}_j^{(t)-1/2}\right)_{i,j\in[\![n]\!]\times[\![\overline{n}]\!]}$. This way our MM algorithms summarized in Algorithm 14 consist at each (t) in the two steps:

i) Solving the srFGW problem or its entropically regularized version srFGW_ε, with the additional linear OT term from the local linearization given in Equation (6.21) at

Algorithm 14 MM solver for srFGW$_g$ and srFGW$_{e+g}$

1: Inputs: Input and target graphs, trade-off parameter α, sparsity promoting parameter λ and entropic parameter ε if using srGW$_{e+g}$.
2: Set $\boldsymbol{R}^{(0)} = \mathbf{0}$.
3: **repeat**
4: Get optimal transport $\boldsymbol{T}^{(t)}$ with second marginal $\overline{\boldsymbol{h}}^{(t)}$ from CG solver (Alg. 12) (srFGW$_g$) or MD solver (Alg. 13) (srFGW$_{e+g}$) with additional linear OT term $\boldsymbol{R}^{(t-1)}$.
5: Compute $\boldsymbol{R}^{(t)} = \frac{\lambda_g}{2}(\overline{\boldsymbol{h}}_j^{(t)})_{ij}^{-1/2}$ the new local linearization of $\Omega(\boldsymbol{T}^{(t)})$.
6: **until** convergence.

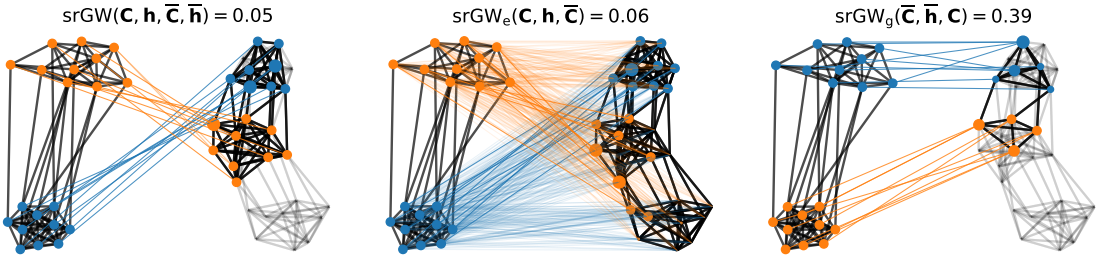

FIGURE 6.2: Comparison of the srGW matchings resulting respectively from the unregularized problem (6.2) (left) and its regularized versions, namely entropic regularization srGW$_\varepsilon$ (6.10) (middle) and sparsity promoting regularization srGW$_g$ (6.19) (right). Nodes of the source graph are colored based on their clusters. The OT from the source to the target nodes is represented by arcs colored depending on the corresponding source node color. The nodes in the target graph are colored by averaging the (rgb) color of the source nodes, weighted by the entries of the OT plan.

iteration $(t-1)$, adapting Alg. 12 or Alg. 13 accordingly. This step provides the new estimate $\boldsymbol{T}^{(t)}$.

ii) Updating the penalty term $\boldsymbol{R}^{(t)} = \frac{\lambda_g}{2}\left(\overline{h}_j^{(t)-1/2}\right)_{i,j}$ from $\boldsymbol{T}^{(t)}$ with second marginal $\overline{\boldsymbol{h}}^{(t)}$.

Comparison of srGW solvers. We illustrate in Figure 6.2, the optimal couplings associated to the computation of the srGW divergence, and its regularized versions, from a graph $(\boldsymbol{C}, \boldsymbol{h})$ with two communities, to a graph with structure $\overline{\boldsymbol{C}}$ composed of three communities. We show on the *left plot* the unregularized srGW (6.2) matching between both graphs estimated using the CG Algorithm 12. The natural sparsity induced by the constraint set $\mathcal{U}_{\overline{n}}(\boldsymbol{h})$ results in an optimal solution that highlights a subgraph of $\overline{\boldsymbol{C}}$ containing two clusters as $(\boldsymbol{C}, \boldsymbol{h})$.
Then, we illustrate in the *middle plot* the effect of entropic regularization (6.10) using our Mirror-Descent Algorithm 13, that prevents optimal couplings from srGW$_\varepsilon$ to be sparse. Consequently, all nodes of the target structure $\overline{\boldsymbol{C}}$ are assigned a strictly positive mass. However, looking at the nodes with larger masses, two clusters are still clearly identifiable. So the high-frequencies in an embedding $\overline{\boldsymbol{h}}$ can still preserve discriminant structure properties of the input graph, while imposing the consideration of the complete target structure.
Finally, the *right plot* shows the effect of sparsity promoting regularization into the srGW problem (6.19). The optimal coupling estimated using our Majorization-Minimization Algorithm 14 finds a smaller sub-structure in $\overline{\boldsymbol{C}}$ than the one found by the unregularized srGW problem, that still represents well the input graph. As such, srGW$_g$ lead to denoised or compressed embeddings.

Algorithm 15 Stochastic update of the srFGW dictionary atom $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$

1: Sample a minibatch of B graphs $\mathcal{B} := \{(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i \in [\![B]\!]}$.
2: Get OT matrices $\{\boldsymbol{T}_i^\star\}_{i \in [\![B]\!]}$ from srFGW$(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ with any solver discussed in Section 6.2.2.
3: Compute sub-gradients $\widetilde{\nabla}_{\overline{\boldsymbol{C}}}\left(\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})\right)$ and $\widetilde{\nabla}_{\overline{\boldsymbol{F}}}\left(\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})\right)$ of srFGW, respectively given in Equations (6.24) and (6.25) for fixed OT matrices, to finally perform the next projected gradient steps:

$$\overline{\boldsymbol{C}} \leftarrow \mathrm{Proj}_{\mathcal{S}(\mathbb{R}_+^{\overline{n}\times\overline{n}})}(\overline{\boldsymbol{C}} - \eta_C \widetilde{\nabla}_{\overline{\boldsymbol{C}}}\left(\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})\right)) \quad \text{and} \quad \overline{\boldsymbol{F}} \leftarrow \overline{\boldsymbol{F}} - \eta_F \widetilde{\nabla}_{\overline{\boldsymbol{F}}}\left(\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})\right) \quad (6.22)$$

6.3 The semi-relaxed (Fused) Gromov-Wasserstein barycenter as a natural Dictionary Learning problem

A dataset of I attributed graphs $\mathcal{D} = \{(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i \in [\![I]\!]}$, with heterogeneous structures, features and a variable number of nodes denoted by $\{n_i\}_{i \in [\![I]\!]}$ is now considered. In the following, we introduce a novel graph Dictionary Learning (DL) whose peculiarity is to learn *a unique* dictionary element. Then we discuss how this dictionary can be used to perform graph completion, *i.e.* jointly reconstruct the full structure and node features of a graph from an observed attributed subgraph.

6.3.1 A novel Graph Dictionary Learning

Semi-relaxed Fused Gromov-Wasserstein embedding. We first discuss how an observed graph can be represented in a dictionary with a unique element $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ (or *atom*), with structure $\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n}\times\overline{n}}$ and node features $\overline{\boldsymbol{F}} \in \mathbb{R}^{\overline{n}\times d}$, both assumed to be known or designed through expert knowledge.
First, for a given input graph $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$, one computes $\mathrm{srFGW}^2_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ using the algorithmic solutions and regularization strategies detailed in Section 6.2.2. From the estimated optimal coupling $\boldsymbol{T}_i^\star$, the optimal weights for the target graph $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ are recovered by $\overline{\boldsymbol{h}_i^\star} = \boldsymbol{T}_i^{\star\top}\mathbf{1}_{n_i}$. The graph $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}_i^\star})$ can be seen as a projection of $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$ on $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ in the FGW sense and the probability vector $\overline{\boldsymbol{h}_i^\star}$ is an *embedding* of $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$. Representing a graph as a vector of weights $\overline{\boldsymbol{h}_i^\star}$ on a graph $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ is a new and elegant way to define a graph subspace that is orthogonal to other DL methods, that either rely on GW barycenters (Xu, 2020) or linear representations as those introduced in Chapter 5. One particularly interesting aspect of this modeling is that when $\overline{\boldsymbol{h}_i^\star}$ is sparse, only the subgraph of $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ corresponding to the nodes with non-zero weights in $\overline{\boldsymbol{h}_i^\star}$ is used.

srFGW Dictionary Learning and online algorithm. Given a dataset of attributed graphs $\mathcal{D} = \{\mathcal{G}_i := (\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i \in [I]}$, we propose to learn the graph dictionary $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}) \in \mathbb{R}^{\overline{n}\times\overline{n}}$ from the observed data, by optimizing:

$$\min_{\overline{\boldsymbol{C}} \in \mathbb{R}^{\overline{n}\times\overline{n}}, \overline{\boldsymbol{F}} \in \mathbb{R}^{\overline{n}\times d}} \frac{1}{I}\sum_{i=1}^{I} \mathrm{srFGW}^2_{2,\alpha}(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}). \quad (6.23)$$

for any $\alpha \in [0, 1]$. Let us denote $\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ the objective function in equation (6.23). This problem is denoted as **srFGW Dictionary Learning**. It can be seen as a srFGW barycenter problem (Peyré et al., 2016) where we look for a graph structure $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ for which there exist node weights $(\overline{\boldsymbol{h}_i^\star})_{i \in [\![I]\!]}$ leading to a minimal FGW error. Interestingly this DL model requires

only to solve the srFGW problem to compute the embedding $\overline{\boldsymbol{h}}_i^\star$ since it can be recovered from the solution $\boldsymbol{T}_i^\star$ of the problem.

We solve the non-convex optimization problem above with an online algorithm similar to the one first proposed in Mairal et al. (2009) for vectorial data and adapted by Vincent-Cuaz et al. (2021) for graph data (Section 5.2.4). The core of the stochastic algorithm is depicted in Algorithm 15.

The main idea is to update the dictionary $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ with a stochastic estimation of the gradients on a minibatch of B graphs $\mathcal{B} = \{\mathcal{G}_i\}_i$. At each stochastic update, we solve the embedding problems $\{\overline{\boldsymbol{h}}_i^* := \boldsymbol{T}_i^{*\top}\mathbf{1}_{n_i}\}_{i\in[\![B]\!]}$ (Section 6.2.2), *independently for each graph* in the mini-batch, using a fixed dictionary $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$.

Then, we estimate sub-gradients *w.r.t.* the srFGW DL objective function $\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$. These estimates computed over the mini-batch are given, for the atom structure by

$$\widetilde{\nabla}_{\overline{\boldsymbol{C}}}\Big(\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})\Big) = 2\alpha \sum_{i\in[\![B]\!]} \{\overline{\boldsymbol{C}} \odot \overline{\boldsymbol{h}}_i^* \overline{\boldsymbol{h}}_i^{*\top} - \boldsymbol{T}_i^{*\top}\boldsymbol{C}_i\boldsymbol{T}_i^*\} \tag{6.24}$$

and for the atom feature matrix by

$$\widetilde{\nabla}_{\overline{\boldsymbol{F}}}\Big(\mathcal{F}_{\mathcal{D}}(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})\Big) = 2(1-\alpha) \sum_{i\in[\![B]\!]} \Big\{\mathrm{diag}(\overline{\boldsymbol{h}}_i^*)\overline{\boldsymbol{F}} - \boldsymbol{T}_i^{*\top}\boldsymbol{F}_i\Big\} \tag{6.25}$$

Remark that according to equations (6.24) and (6.25), an input $\mathcal{G}_i$ only contributes to the update of the entries of $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ that have non-null values in the embedding $\overline{\boldsymbol{h}}_i^*$. Finally given these sub-gradient estimates, we perform a projection on the set $\mathcal{S}(\boldsymbol{R}_+^{\overline{n}\times\overline{n}})$ of symmetric non-negative matrices to update $\overline{\boldsymbol{C}}$, while operating on adjacency or shortest-path input matrices $\{\boldsymbol{C}_i\}_{i\in[\![I]\!]}$. In practice we use Adam optimizer (Kingma & Ba, 2015) in all our experiments.

Numerical complexity. The complexity of this stochastic algorithm is mostly bounded by computing the gradients, which can be done in $O(n_i^2\overline{n} + n_i\overline{n}^2)$ (Section 6.2.2). Hence, in a factorization context *i.e.* $\max_i(n_i) >> \overline{n}$, the overall learning procedure has a quadratic complexity *w.r.t.* the maximum graphs size. Since the embedding $\overline{\boldsymbol{h}}_k^\star$ is a by-product of computing the different srFGW, we do not need an iterative solver to estimate it. Consequently, it leads to a speed up on CPU of 2 to 3 orders of magnitude compared to our main competitors (see Section 6.4) whose DL methods, instead, require such iterative scheme.

Relation with FGW barycenters. A specific scenario relates to considering a unique input graph ($I = 1$) in the DL problem (6.23). In this setting, our srFGW Dictionary Learning problem is equivalent to fully estimating the support of a FGW barycenter, as studied in Section 3.3.4 (see equation (3.62)).

We originally proposed to solve the latter FGW problem using a Projected Gradient Descent algorithm whose iterate consists in: i) solving a FGW problem given the current FGW barycenter estimation $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. ii) Performing projected gradient steps to update $\overline{\boldsymbol{C}}$, $\overline{\boldsymbol{F}}$, and finally $\overline{\boldsymbol{h}}$ whose sub-gradient is given by Theorem 7. Therefore, Algorithm 15 to estimate the srFGW DL provides a novel solver to this specific problem with $I = 1$, that benefits from our relaxation of the mass conservation paradigm.

6.3.2 DL-based model for graphs completion

As with GDL (Section 5.2.6), the dictionary atom $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$ estimated on the dataset $\mathcal{D}$ can be used to infer/complete a new graph from the dataset that is only partially observed. In this setting, we aim at recovering the full structure $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ and feature matrix $\boldsymbol{F} \in \mathbb{R}^{n\times d}$,

Algorithm 16 srFGW: graphs completion

1: Inputs: Observed graph $\boldsymbol{C}_{obs}$, Dictionary $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$, size of the imputed graph and trade-off parameter α.
2: Initialize randomly the entries $\boldsymbol{C}_{imp}$ by iid sampling from $\mathcal{N}(0.5, 0.01)$ (In a symmetric manner if $\boldsymbol{C}_{obs}$ is symmetric).
3: **repeat**
4: Compute optimal transport $\boldsymbol{T}^*$, with second marginal $\overline{\boldsymbol{h}}^*$, from $\text{srFGW}_{2,\alpha}(\boldsymbol{C}_{est}, \boldsymbol{F}_{est}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}})$.
5: Perform projected gradient steps on $\boldsymbol{C}_{est}$ and $\boldsymbol{F}_{est}$ using gradients from Equations (6.28) and (6.29).
6: **until** convergence.

while only a subset of relations and features among $n_{obs} < n$ nodes is observed, denoted as $\boldsymbol{C}_{obs} \in \mathbb{R}^{n_{obs} \times n_{obs}}$ and $\boldsymbol{F}_{obs} \in \mathbb{R}^{n_{obs} \times d}$. This boils down to solve:

$$\min_{\boldsymbol{C}_{imp}, \boldsymbol{F}_{imp}} \quad \text{srFGW}^2_{2,\alpha}\left(\widetilde{\boldsymbol{C}}, \widetilde{\boldsymbol{F}}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}\right) \tag{6.26}$$

with

$$\boldsymbol{C}_{est} = \left[\begin{array}{c|c} \boldsymbol{C}_{obs} & \vdots \\ \hline \cdots & \boldsymbol{C}_{imp} \end{array}\right] \quad \text{and} \quad \boldsymbol{F}_{est} = \left[\begin{array}{c} \boldsymbol{F}_{obs} \\ \hline \boldsymbol{F}_{imp} \end{array}\right] \tag{6.27}$$

where only $n^2 - n^2_{obs}$ coefficients collected into $\boldsymbol{C}_{imp}$ and $(n - n_{obs})d$ coefficients collected into $\boldsymbol{F}_{imp}$ are optimized (and thus imputed).

We solve the optimization problem above by a classical Projected Gradient Descent algorithm. At each iteration we find an optimal coupling $\boldsymbol{T}^\star$ of srFGW that is used to calculate the gradient of srFGW *w.r.t* $\overline{\boldsymbol{C}}_{imp}$ and $\overline{\boldsymbol{F}}_{imp}$. The latter is obtained as the gradient of the srFGW cost function evaluated at the fixed optimal coupling $\boldsymbol{T}^\star$ by using the Envelope Theorem (Bonnans & Shapiro, 2000). The updates read as follows, for the imputed structure

$$\boldsymbol{C}_{imp} \leftarrow \left[2\alpha\left(\boldsymbol{C}_{est} \odot \boldsymbol{h}\boldsymbol{h}^\top - \boldsymbol{T}^* \overline{\boldsymbol{C}} \boldsymbol{T}^{*\top}\right)\right]_{\text{imputed } i,j} \tag{6.28}$$

and for the imputed feature matrix

$$\boldsymbol{F}_{imp} \leftarrow \left[2(1-\alpha)\left(\text{diag}(\boldsymbol{h})\boldsymbol{F}_{est} - \boldsymbol{T}^* \overline{\boldsymbol{F}}\right)\right]_{\text{imputed } i} \tag{6.29}$$

The projection step is here to enforce known properties of $\boldsymbol{C}$, such as positivity and symmetry. In practice the estimated $\boldsymbol{C}_{imp}$ will have continuous values, so one has to apply a thresholding (with value 0.5) on $\boldsymbol{C}_{imp}$ to recover a binary adjacency matrix.

6.4 Numerical experiments

This section aims at illustrating the behavior of the sr(F)GW divergence (Section 6.2) and the resulting Dictionary Learning (Section 6.3) on synthetic and real-world problems. We first investigate the use of srGW on the task of unattributed *graph partitioning*, that focuses on the processing of a single graph (Sections 6.4.1 and 6.4.2). Then we test our srGW and srFGW dictionaries for both clustering and classification of graph datasets (Sections 6.4.3 & 6.4.4), and graphs completion (Section 6.3.2).

6.4.1 Graph partitioning: Benchmarks

As discussed in Section 6.1.2, it is possible to achieve graph partitioning via OT by estimating a GW matching between the graph to partition $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{h})$ (n nodes) and a smaller graph $\overline{\mathcal{G}} = (\boldsymbol{I}_Q, \overline{\boldsymbol{h}})$, with $Q \ll n$ nodes. The atom $\boldsymbol{I}_Q$ is set as the identity matrix to enforce the emergence of densely connected groups (i.e. *communities*). The distribution $\overline{\boldsymbol{h}}$ estimates the proportion of nodes in each cluster. We recall that $\overline{\boldsymbol{h}}$ must be given to compute the GW distance, whereas it is estimated with srGW.

All partitioning performances are measured by Adjusted Mutual Information (AMI, Vinh et al., 2010). These measures will be completed thanks to the Adjusted Rand Index (ARI) on real-world datasets. AMI and ARI compare the nodes partition provided by the OT matrix in srGW with the "true" (it would be wiser to say the "annotated") one available with the datasets analyzed. The comparison between ARI and AMI has been thoroughly investigated in Romano et al. (2016) and led to the following conclusion: ARI should be used when the reference clustering has large equal sized clusters; AMI should be used when the reference clustering is unbalanced and there exist small clusters.

On srGW algorithm initialization. Chowdhury & Needham (2021) discussed the sensitivity of GW solvers to the initialization depending on few choices of input graph representations. This way it is arguable to compare GW and srGW by using the product $\boldsymbol{h}\overline{\boldsymbol{h}}^\top$, usually chosen as default initialization, but methods to select better starting points have not been investigated yet in the GW literature.

However for these partitioning tasks we should/can not initialize the transport plan of our srGW solver using the product of $\boldsymbol{h} \in \Sigma_N$ with a uniform target distribution $\overline{\boldsymbol{h}}^{(0)} = \frac{1}{Q}\mathbf{1}_Q$, leading to $\boldsymbol{T}^{(0)} = \frac{1}{Q}\boldsymbol{h}\mathbf{1}_Q^\top$. Indeed, for any symmetric input representation $\boldsymbol{C}$ of the graph, the partial derivative of our objective *w.r.t* the $(p, q) \in [\![N]\!] \times [\![Q]\!]$ entries of $\boldsymbol{T}$, satisfies

$$\begin{aligned}\frac{\partial \mathcal{E}^{GW}}{\partial T_{pq}}(\boldsymbol{C}, \boldsymbol{I}_Q, \boldsymbol{T}^{(0)}) &= 2\sum_{ij}(C_{ip} - \delta_{jq})^2 T_{ij}^{(0)} = \frac{2}{Q}\sum_{ij}(C_{ip}^2 + \delta_{jq} - 2C_{ip}\delta_{jq})h_i \\ &= \frac{2}{Q}\{Q\sum_i C_{ip}^2 h_i + 1 - 2\sum_i C_{ip}h_i\}\end{aligned} \tag{6.30}$$

This expression is independent of $q \in [\![Q]\!]$, so taking the minimum value over each row in the direction finding step of our CG Algorithm 12, will lead to $\boldsymbol{X} = \boldsymbol{T}^{(0)}$. Then the line-search step involving, for any $\gamma \in [0, 1]$, $\boldsymbol{Z}^{(0)}(\gamma) = \boldsymbol{T}^{(0)} + \gamma(\boldsymbol{X} - \boldsymbol{T}^{(0)})$ will be independent of γ as $\boldsymbol{Z}^{(0)}(\gamma) = \boldsymbol{T}^{(0)}$. This implies that the algorithm would terminate with optimal solution $\boldsymbol{T}^\star = \boldsymbol{T}^{(0)}$ being a non-informative coupling.

Nevertheless, we observed for srGW that using an initialization based on the hard assignments of a K-means performed on the rows of the input representations (up to the left scaling $\mathrm{diag}(\boldsymbol{h})$), leads to better srGW estimation and graph partitioning performances. Hence we applied this scheme for our method in the next graph partitioning benchmarks. We will illustrate later in this section, how srGW refines these hard assignments by soft ones through OT, on real-word graph datasets.

Experiments on simulated data. In Figure 6.3, we illustrate the behavior of GW and srGW partitioning on a toy graph, simulated according to SBM (Holland et al., 1983) with 3 clusters of different proportions. We see that miss-classification occurs either when the proportions $\overline{\boldsymbol{h}}$ do not fit the true ones or when $q \neq 3$. On the other hand, clustering from srGW can simultaneously recover any cluster proportions (since it estimates them) and can select the actual number of clusters, using the sparse values in $\overline{\boldsymbol{h}}$.

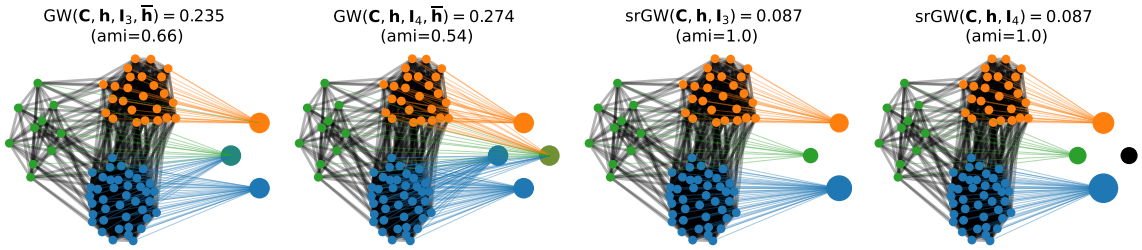

FIGURE 6.3: GW vs srGW partitioning of $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{h} = \mathbf{1}_n/n)$ with 3 clusters of varying proportions to $\overline{\mathcal{G}} = (\boldsymbol{I}_q, \overline{\boldsymbol{h}})$ where $\overline{\boldsymbol{h}}$ is fixed to uniform for GW (left) and estimated for srGW (right) for $Q = 3$ and $Q = 4$. Nodes of $\mathcal{G}$ are colored based on their cluster assignments while those of $\overline{\mathcal{G}}$ are interpolated based on linear interpolation of node colors of $\mathcal{G}$ linked to them through their OT (colored line between nodes) if these links exist, otherwise default nodes color is black. Node sizes of both graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ are proportional to their respective masses $\boldsymbol{h}$ and $\overline{\boldsymbol{h}}$ (except for black nodes of zero mass).

TABLE 6.1: Partitioning benchmark: Datasets statistics.

Datasets	# nodes	# communities	connectivity rate (%)
Wikipedia	1998	15	0.09
EU-email	1005	42	3.25
Amazon	1501	12	0.41
Village	1991	12	0.42

Real-world datasets. In order to benchmark the srGW partitioning on real (directed and undirected) graphs, we consider 4 datasets (whose statistics are provided in Table 6.1), using the same pre-processing routine as in Chowdhury & Needham (2021):

1. A Wikipedia hyperlink network (Yang & Leskovec, 2015, Wikipedia), preprocessed by choosing 15 webpage categories and extracting their induced directed subgraphs.

2. A directed graph of email interactions between departments of a European research institute (Yin et al., 2017, EU-email).

3. An Amazon product network (Yang & Leskovec, 2015, Amazon), where the subgraph induced by top 12 product categories was selected.

4. A network of interactions between indian villages (Banerjee et al., 2013, Village)

For the directed graphs (Wikipedia & EU-email), we also construct symmetric versions following the procedure described in Chowdhury & Needham (2021) that consists in adding the reciprocal edges.

Benchmark settings. Our main competitors are the two GW based partitioning methods proposed by Xu et al. (2019b) and Chowdhury & Needham (2021). The former (GWL) relies on adjacency matrices, the latter (SpecGWL) adopts heat kernels on the graph normalized laplacians (SpecGWL). The GW solver of Flamary et al. (2021) was used for these methods. For fairness, we also consider these two representations for srGW partitioning (namely srGW and srSpecGW). All these OT based methods depend on hyperparameters which can be tuned in an unsupervised way (*i.e.* without knowing the ground truth partition) based on modularity maximization (Chowdhury & Needham, 2021). Following their numerical experiments, we considered as input distribution $\boldsymbol{h}$ for the observed graph the parameterized power-law transformations of the form $h_i = \frac{p_i}{\sum_i p_i}$ where $p_i = (deg(i)+a)^b$, with $deg(i)$ the i-th node degree and real parameters $a \in \mathbb{R}$ and $b \in [0, 1]$. If the graph has no isolated nodes we chose $a = 0$ and $a = 1$ otherwise. b is validated within 10 values $\{0, 0.0001, 0.005, ..., 0.1, 0.5, 1\}$

TABLE 6.2: Partitioning performances on real datasets measured by AMI. We see in bold (resp. italic) the first (resp. second) best method. NA: non applicable.

	Wikipedia		EU-email		Amazon	Village
	asym	sym	asym	sym	sym	sym
srGW (ours)	*56.92*	56.92	49.94	50.11	48.28	81.84
srSpecGW	50.74	**63.07**	49.08	50.60	76.26	*87.53*
srGW_e	**57.13**	*57.55*	**54.75**	*55.05*	50.00	83.18
srSpecGW_e	53.76	61.38	*54.27*	50.89	*85.10*	84.31
GWL	38.67	35.77	47.23	46.39	38.56	68.97
SpecGWL	40.73	48.98	45.89	49.02	65.16	77.85
FastGreedy	NA	55.30	NA	45.89	77.21	**93.66**
Louvain	NA	54.72	NA	**56.12**	76.30	**93.66**
InfoMap	46.43	46.43	54.18	49.10	**94.33**	**93.66**
srGW gain	**+10.7**	**+7.77**	**+0.57**	-1.07	-9.23	-6.13

which progressively transform the input distribution from the uniform to the normalized degree ones. An ablation study of this parameter is reported in Table 6.6. The heat parameter for SpecGWL and srSpecGW is tuned for each dataset within the range $[1, 100]$ by recursively splitting this range into 5 values, find the parameter leading to maximum modularity, and repeat the same process on the induced new interval with this best parameter as middle point. This process is stopped based on the relative variation of maximum modularity between two successive iterates of precision 10^{-3}. A similar scheme can be used to fine-tune the entropic parameter.

Finally, three competing methods specialized in graph partitioning are also considered: FastGreedy (Clauset et al., 2004), Louvain (with validation of its resolution parameter, Blondel et al., 2008) and Infomap (Rosvall & Bergstrom, 2008).

Results on real-world datasets. The graph partitioning performances, measured by means of AMI and ARI, are respectively reported in Table 6.2 and 6.3. Our method srGW always outperforms the GW based approaches for a given type of input structure representations (adjacency vs heat kernels), moreover the entropic regularization seems to improve the performance on these applications. One can also observe that the choice of performance metric (AMI or ARI) leads to slight variations of rankings, which seem to occur more often for methods explicitly based on modularity maximization. We want to stress that our *general purpose* divergence srGW outperforms methods that have been specifically designed for nodes clustering tasks on 3 (based on AMI) or 4 (based on ARI) out of 6 datasets.

Runtimes comparison on real datasets. To complete our analysis we report in Table 6.4 the runtimes on CPUs for the partitioning benchmark on real datasets. For the sake of comparison, we re-ran the srGW partitioning experiments using CPUs instead of a GPU (Tesla K80). In this setting, we can observe that GW partitioning of raw adjacency matrices (GWL) is the most competitive in terms of computation time (while being the last in terms of clustering performances), on par with InfoMap. Overall, our srGW partitioning methods seem slower than GW based ones in terms of speed, even if we remain in the same order of complexity. This might be due to several practical aspects. The linear OT problems involved in each CG iteration of GW based methods are solved thanks to a C solver (POT's implementation (Flamary et al., 2021)), whereas our srGW CG solver is fully implemented in Python. This can compensate the computational benefits of srGW solver regarding GW solver. So the computation time became mostly affected by the number of computed gradients. We observed in these experiments that for the same precision level, srGW needed

TABLE 6.3: Partitioning performances on real datasets measured by ARI, corresponding to best configurations reported in Table 6.2. We see in bold (resp. italic) the first (resp. second) best method. NA: non applicable.

	Wikipedia		EU-email		Amazon	Village
	asym	sym	asym	sym	sym	sym
srGW (ours)	33.56	33.56	30.99	29.91	30.08	67.03
srSpecGW	32.85	**58.91**	32.76	31.28	51.94	*80.36*
srGW_e	*36.03*	36.27	*35.91*	**36.87**	31.83	69.58
srSpecGW_e	**37.71**	*57.24*	**38.22**	30.74	*75.81*	74.71
GWL	5.70	13.85	19.35	26.81	24.96	48.35
SpecGWL	25.23	30.94	26.32	30.17	46.66	67.22
FastGreedy	NA	55.61	NA	17.23	45.80	**93.97**
Louvain	NA	52.16	NA	*32.79*	42.64	**93.97**
InfoMap	35.48	35.48	16.70	16.70	**89.74**	**93.97**
srGW gains	**+2.23**	**+3.3**	**+21.52**	**+4.08**	- 13.93	-13.61

TABLE 6.4: Runtimes (seconds) on real datasets measured on CPU for all partitioning methods and corresponding best configurations.

	Wikipedia		EU-email		Amazon	Village
	asym	sym	asym	sym	sym	sym
srGW (ours)	4.62	4.31	4.18	4.22	4.31	4.68
srSpecGW	2.91	2.71	2.49	2.83	3.06	3.11
srGW_e	2.69	2.48	2.31	2.81	2.87	2.53
srSpecGW_e	2.35	1.96	2.15	2.03	2.16	2.58
GWL	**0.17**	**0.17**	**0.13**	**0.12**	**0.13**	**0.16**
SpecGWL	**0.77**	1.25	1.55	0.97	1.01	0.88
FastGreedy	NA	0.56	NA	2.31	0.37	1.26
Louvain	NA	0.52	NA	0.31	*0.20*	*0.49*
InfoMap	**0.17**	*0.18*	**0.12**	**0.14**	**0.13**	**0.16**

considerably more iterations to converge than GW. This might be due to the difference in conditioning of the respective Quadratic Programs that respectively admit for Hessian $\left(\overline{\boldsymbol{C}}^2 \otimes_K \mathbf{1}_n \mathbf{1}_n^\top - 2\overline{\boldsymbol{C}} \otimes_K \boldsymbol{C}\right)$ and $-2\overline{\boldsymbol{C}} \otimes_K \boldsymbol{C}$. So using a non-informative structure $\overline{\boldsymbol{C}} = \boldsymbol{I}_Q$ might negatively impact the convergence speed of srGW, and advocate for further studies regarding srGW based graph partitioning using more informative $\overline{\boldsymbol{C}}$.

srGW runtimes: CPU vs GPU. Finally, we stress that we observed runtimes 10 to 20 times slower for srGW using CPUs instead of a GPU. To illustrate this matter, we generated 10 graphs according to Stochastic Block Model with 10 clusters, a varying number of nodes in $\{100, 200, ..., 2900, 3000\}$ and the same connectivity matrix. We report in Figure 6.4, the averaged runtimes of one CG iteration depending on the size of the input graphs. For small graphs with a few hundreds of nodes, performances on CPUs or a single GPU are comparable. However, operating on GPU becomes very beneficial once graphs of a few thousand of nodes are processed.

6.4.2 Graph partitioning: Initialization and parameterization

In this section, we provide complementary insights on the specific choices that led to performances reported in the previous Section 6.4.1, namely the initialization of srGW algorithms

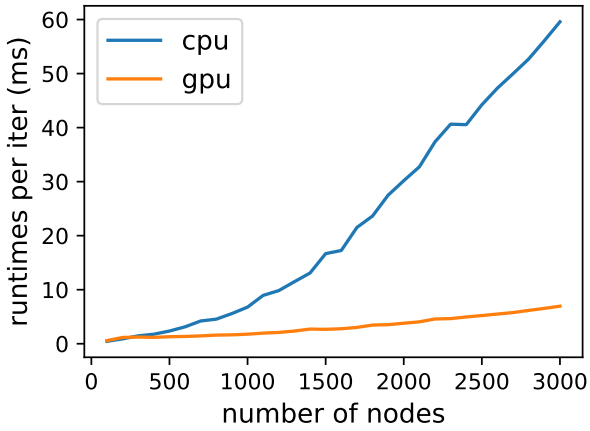

FIGURE 6.4: Runtimes of srGW's CG algorithm over increasing graph sizes.

TABLE 6.5: Partitioning performances on real datasets measured by AMI. Comparison between srGW and K-means whose hard assignments are used to initialize srGW.

	Wikipedia		EU-email		Amazon	Village
	asym	sym	asym	sym	sym	sym
srGW (ours)	*56.92*	56.92	49.94	50.11	48.28	81.84
srGW_e	**57.13**	*57.55*	**54.75**	*55.05*	50.00	83.18
Kmeans (adj)	29.40	29.40	36.59	34.35	34.36	60.83
srGW gains	**+27.73**	**+28.15**	**+18.16**	**+20.7**	**+25.64**	**+22.35**

using a K-means algorithm, and the (validated) parametrization of the input graph distributions according to power laws.

Initializations. As mentioned in Section 6.4.1, for the srGW based graph partitioning that uses a non-informative structure, we proposed to initialize its CG and MD solvers using a K-means algorithm. The latter is applied over the rows of the *rescaled* input graph structures $\text{diag}(\boldsymbol{h})\boldsymbol{C}$ if it is symmetric, or its symmetrization $\text{diag}(\boldsymbol{h})(\boldsymbol{C}+\boldsymbol{C}^\top)/2$ otherwise. As this initialization already leads to a partitioning of the graph, we compare the K-means performances with the ones of the srGW algorithms that aim at refining these solutions. For the K-means algorithm an analog validation than for srGW methods over $\boldsymbol{h}$ is performed to select the best graph partitioning. The performances measured by means of AMI, while using the adjacency matrix for $\boldsymbol{C}$, are reported in Table 6.5. These support the use of our srGW methods for graph partitioning that improve the K-means solution from 18 to 28 points of AMI.

Parameterized input distributions: Ablation study. To conclude on our graph partitioning benchmark, we perform an ablation study of the parameter $b \in [0, 1]$ introduced on the input graph distributions, first suggested by Xu et al. (2019a). The results in terms of AMI are reported on all tested datasets in Table 6.6. In most scenarios and in every OT based methods, a parameter $b \in]0, 1[$ leads to best AMI performances (except for the dataset Village), while the common assumption of uniform distribution remains competitive. The use of raw normalized degree distributions consequently reduces partitioning performances of all methods. Hence these results first indicate that further research on the input distributions could be beneficial. They also suggest that the commonly used uniform input distributions provide a good compromise in terms of performances while being parameter-free.

TABLE 6.6: Partitioning performances on real datasets measured by AMI: Ablation study of the parameter involved in the power-law transformations parameterized by $b \in [0, 1]$ of normalized degree distributions for srGW and GW based methods. We denote different types of transformation by 'unif' ($b = 0$), 'deg' ($b = 1$) and 'inter' ($0 < b < 1$). We see in bold (resp. italic) the first (resp. second) best model. We also highlight distributions leading to the first (bold) and the second (italic) highest scores across all methods.

	Wikipedia						EU-email						Amazon			Village		
	asym			sym			asym			sym			sym			sym		
	unif	deg	**inter**	*unif*	deg	**inter**	*unif*	deg	**inter**	*unif*	deg	**inter**	*unif*	deg	**inter**	**unif**	deg	*inter*
srGW (ours)	52.7	47.4	*56.9*	52.7	47.4	56.9	49.7	43.6	49.9	49.5	39.9	50.1	40.8	42.7	48.3	74.9	62.0	81.8
srSpecGW	48.9	44.8	50.7	58.2	55.4	**63.0**	47.8	44.3	49.1	46.8	43.6	50.6	75.7	69.5	76.3	**87.5**	78.1	*86.1*
srGW$_e$	54.9	48.5	**57.1**	54.3	47.8	57.6	53.9	48.6	**54.8**	*53.5*	42.1	**55.1**	48.2	42.9	50.0	83.2	69.1	82.7
srSpecGW$_e$	51.7	45.2	53.8	59.0	54.9	*61.4*	52.1	47.9	*54.3*	47.8	43.2	50.9	*83.8*	76.9	**85.1**	84.3	77.6	83.9
GWL	33.8	8.8	38.7	33.15	14.2	35.7	47.2	35.1	43.6	37.9	46.3	45.8	32.0	27.5	38.5	68.9	43.3	66.9
SpecGWL	36.0	28.2	40.7	29.3	33.2	48.9	43.2	40.7	45.9	48.8	47.1	49.0	64.5	64.8	65.1	77.3	64.9	77.8

6.4.3 Clustering of graphs datasets

We now show how the embeddings $\{\overline{\boldsymbol{h}}_i^\star\}_{i \in [\![I]\!]}$ provided by the sr(F)GW Dictionary Learning can be particularly useful for the task of clustering a dataset of graphs $\mathcal{D} = \{(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)\}_{i \in [\![I]\!]}$.

Datasets and methods. We consider here three types of datasets, which were already studied in Section 5.3.2 of Chapter 5:

i) Social networks without node attributes from IMDB-B and IMDB-M (Yanardag & Vishwanathan, 2015).

ii) Graphs with discrete features representing chemical compounds from MUTAG (Debnath et al., 1991) and cuneiform signs from PTC-MR (Krichene et al., 2015)

iii) Graphs with continuous features, namely BZR, COX2 (Sutherland et al., 2003) and PROTEINS, ENZYMES (Borgwardt & Kriegel, 2005).

Our main competitors are the following OT-based SOTA models:

i) GDL (Vincent-Cuaz et al., 2021) and its regularized version GDL$_\lambda$, both introduced in Chapter 5.

ii) GWF (Xu, 2020), with both fixed (GWF-f) and random (GWF-r, default setting for the method) atom size.

iii) GW kmeans (GW-k), a K-means algorithm equipped with (F)GW distances and barycenters (Peyré et al., 2016; Vayer et al., 2019a).

Experimental settings. For all experiments we follow the benchmark proposed in Vincent-Cuaz et al. (2021). The clustering performances are measured by means of Rand Index (RI, Rand, 1971). This metric, that comes down to an accuracy up to class rearrangement, is chosen to provide a comparison with the performances measured via accuracy for the *supervised* evaluation of our embeddings given in Section 6.4.4. We refer the interested reader to the Annex 8.4.5 for the performances comparison in terms of ARI and AMI.

The standard Euclidean distance is used to implement K-means over srGW and GWFs embeddings, but we use for GDL the dedicated Mahalanobis distance as described in Theorem 8 introduced in Chapter 5. GW-k does not use any embedding since it directly computes (a GW) K-means over the input graphs.

For DL based approaches, a first step consists into learning the atoms. srGW dictionary sizes are tested in $M \in \{10, 20, 30, 40, 50\}$, the atom is initialized by randomly sampling its entries from $\mathcal{N}(0.5, 0.01)$ and made symmetric. The extension of srGW to attributed

TABLE 6.7: Clustering performances on real datasets measured by Rand Index. In bold (resp. italic) we highlight the first (resp. second) best method.

	NO ATTRIBUTE		DISCRETE ATTRIBUTES		REAL ATTRIBUTES			
MODELS	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
srGW (ours)	51.6(0.10)	54.9(0.29)	71.3(0.39)	51.5(0.12)	62.6(0.96)	60.1(0.27)	71.7(0.12)	59.7(0.10)
srGW_g	**52.4(0.46)**	**56.7(0.34)**	72.3(0.51)	51.8(0.39)	66.8(0.38)	**62.2(0.27)**	**72.5(0.10)**	*60.7(0.29)*
srGW_e	51.8(0.56)	55.4(0.14)	*74.4(0.84)*	*52.4(0.42)*	*67.6(1.17)*	59.8(0.39)	70.9(0.33)	60.0(0.21)
srGW_{e+g}	*52.2(0.83)*	*55.9(0.68)*	**74.7(0.73)**	**52.5(0.47)**	**67.8(0.94)**	*60.5(0.36)*	71.3(0.52)	**60.8(0.43)**
GDL (Chap.5)	51.3(0.27)	55.1(0.35)	71.9(0.48)	51.5(0.31)	62.8(1.60)	58.4(0.52)	69.8(0.33)	60.2(0.28)
GDL_λ	51.7(0.56)	55.4(0.22)	72.3(0.17)	51.8(0.47)	66.3(1.71)	59.6(0.74)	71.0(0.36)	60.5(0.65)
GWF-r	51.0(0.30)	55.1(0.46)	69.1(1.02)	51.5(0.59)	52.5(2.41)	56.9(0.46)	*72.1(0.21)*	60.0(0.11)
GWF-f	50.4(0.29)	54.2(0.27)	59.1(1.87)	50.8(0.81)	51.8(2.84)	52.8(0.53)	71.6(0.31)	58.9(0.41)
GW-k	50.3(0.03)	53.7(0.07)	57.6(1.45)	50.4(0.33)	56.8(0.53)	52.5(0.13)	66.4(1.37)	50.1(0.01)
srGW gains	**+0.7**	**+1.3**	**+2.4**	**+0.7**	**+1.5**	**+2.6**	**+0.4**	**+ 0.3**

TABLE 6.8: Embedding computation times (in ms) averaged over whole datasets at a convergence precision of 10^{-5} on learned dictionaries. $(-)$ (resp. $(+)$) denotes the fastest (resp. slowest) runtimes regarding DL configurations. We report here runtimes using $FGW_{0.5}$ for datasets with nodes attributes. Measures taken on Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz.

	NO ATTRIBUTE				DISCRETE ATTRIBUTES				REAL ATTRIBUTES							
	IMDB-B		IMDB-M		MUTAG		PTC-MR		BZR		COX2		ENZYMES		PROTEIN	
	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)	(-)	(+)
srGW (ours)	1.51	2.62	0.83	1.59	0.86	1.83	0.40	1.01	0.43	0.79	0.51	0.90	0.62	0.95	0.46	0.60
srGW_g	1.95	6.11	1.06	5.53	3.68	5.98	1.65	3.38	0.89	2.88	0.97	4.60	1.35	4.73	1.57	2.96
GWF-f	219	651	103	373	236	495	191	477	181	916	129	641	93	627	78	322
GDL	108	236	43.8	152	102	514	100	509	73.2	532	48.7	347	38	301	29	151

graphs, namely srFGW, is referred as srGW for conciseness. One efficient way to initialize atoms features for minimizing our resulting reconstruction errors is to use a (Euclidean) K-means algorithm seeking for $\overline{n}$ clusters on the nodes features observed in the dataset. For GDL and GWF, the same validation of hyper-parameters (number of atoms, regularization parameter) than detailed in Section 5.3.4 is performed. Note that for srGW_g, srGW_e and srGW_{e+g}, the sparsity promoting and/or entropic regularization coefficients are validated within $\{0.001, 0.01, 0.1, 1.0\}$.

For each parameter configuration (number of atoms, number of nodes and regularization parameters) we run each experiment five times, independently. The mean RI over the five runs is computed and the dictionary configuration leading to the highest RI for each method is reported.

Results and discussion. Clustering performances and running times are reported in Tables 6.7 and 6.8, respectively. All variants of srGW DL are at least comparable with the SOTA GW based methods. Remarkably, the sparsity promoting variants always outperform other methods. Notably Table 6.8 shows embedding computation times of the order of the millisecond for srGW, two order of magnitude faster than the competitors. Moreover, srGW computation times were measured on CPUs but could be greatly enhanced through parallelization on GPUs, contrary to the second best performing method GDL. Overall, these results clearly support the use of srGW for the clustering of graphs.

Complementary experiments could be considered to evaluate our embeddings, *e.g.* using a Spectral Clustering algorithm on the pairwise FGW distances between embedded graphs $\{(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}_i)\}_{i \in [\![I]\!]}$, as performed for GDL in Section 5.3.2. As these latter experiments lead to rather analog rankings between OT methods, than when GDL embedded graphs are validated in a supervised manner using SVM (Section 5.3.4), we only evaluate embedded graphs in this supervised setting in Section 6.4.4.

Visualizations of srGW embeddings. We provide in Figure 6.5 some examples of graphs embeddings from the dataset IMDB-B learned on a srGW dictionary $\overline{\boldsymbol{C}}$ of 10 nodes. We assigned different colors to each nodes of the graph atom (forth column) in order to visualize the correspondences recovered by the OT plan $\boldsymbol{T}^\star$ resulting from the projection of the respective sample $(\boldsymbol{C}, \boldsymbol{h})$ onto $\overline{\boldsymbol{C}}$ in the srGW sense (third column). As the atom has continuous values we set the edges intensity of grey proportional to the entries of $\overline{\boldsymbol{C}}$. By coloring nodes of the observed graphs based on the OT to its respective embedding $(\overline{\boldsymbol{C}})$(second column) we clearly observe that key subgraphs information, such as clusters and hubs are captured within the embedding.

Impact of the graph atom size. If we increase the size of the dictionary, our embeddings are refined and can bring complementary structural information is brought at a higher resolution *e.g.* finding substructures or variable connectivity between nodes in the same cluster. We illustrate these resolution patterns in Table 6.6 for embeddings learned on the IMDB-B dataset. We represent the embedded graph size distributions depending on the sizes of the graph atom learned thanks to srGW and its sparse variant srGW_g. For any graph atom size, the mean of each embedded graph size distribution represented with a white dot is below the atom size, hence embeddings are sparse and subparts of the atom are indeed selected. Moreover, promoting sparsity of the embeddings (srGW_g) lead to more concentrated embedded graph size distributions with lower averaged sizes than its unregularized counterpart (srGW), as expected. Finally, these distributions seem to reach a stable configuration when the graph atom size is large enough. This argues in favor of the existence of a (rather small) threshold on the atom size where the heterogeneity of all the graphs contained in the dataset is well summarized in the dictionaries.

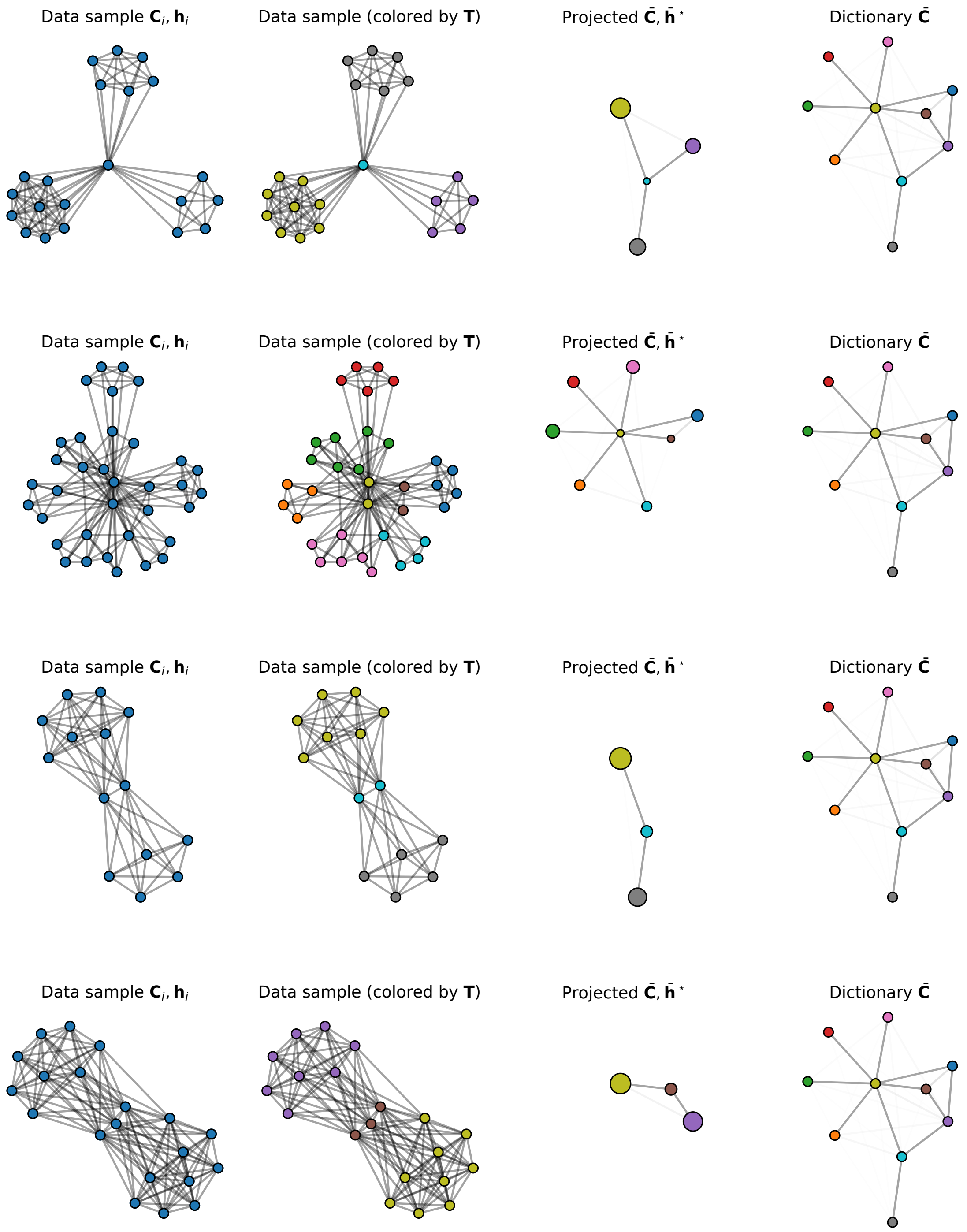

FIGURE 6.5: Illustration of the embedding of different graphs from the IMDB dataset on the estimated dictionary $\overline{\boldsymbol{C}}$. Each row corresponds to one observed graph and we show its graph (left), its graph with nodes colored corresponding to the OT plan (center left), the projected graph on the dictionary with optimal weight $\overline{\boldsymbol{h}}^\star$ and the full dictionary with uniform mass (right).

6.4.4 Classification of graphs datasets

Benchmark settings. We now aim at analyzing if our *unsupervised* DL methods produce discriminative graph representations. As the resulting (attributed) graph subspace is endowed with the (F)GW geometry, we perform classification as a downstream task, using SVM with (F)GW kernels computed on embedded graphs. For all experiments we mimic the benchmark proposed in GDL classification benchmark (Section 5.3.4), using adjacency matrices as input

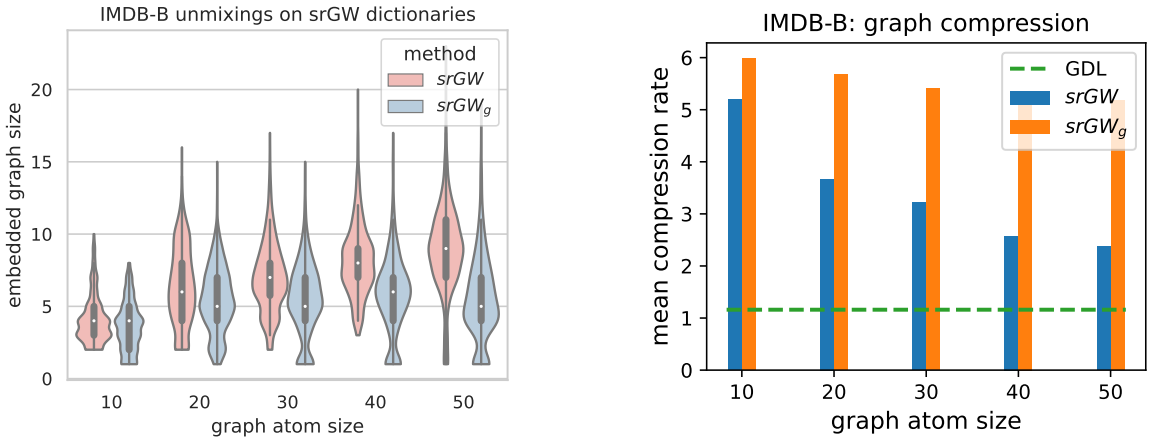

FIGURE 6.6: On the left: Evolution of the embedded graph sizes over the graph atom size validated in $\{10, 20, 30, 40, 50\}$ for dictionaries learned with srGW and srGW$_g$ (with $\lambda = 0.01$). On the right: Averaged ratios of input and corresponding embedded graph sizes using benchmarked DL methods.

TABLE 6.9: Classification performances on real datasets: We highlight the **1**$^{\text{st}}$ (resp. 2$^{\text{nd}}$) best method in bold (resp. italic). Unfilled values (-) when methods are specific to certain type of graph features.

		No attribute		Discrete attributes		Real attributes			
Categories	Models	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
OT DL	srGW (ours)	*72.1(4.1)*	49.2(3.6)	*89.1(5.9)*	*64.5(7.8)*	*88.0(4.2)*	77.7(2.7)	*72.3(5.7)*	*72.9(5.1)*
	srGW$_g$ (ours)	**73.2(4.3)**	**51.3(3.4)**	**90.3(5.4)**	**64.5(6.9)**	**88.5(3.9)**	**79.8(1.8)**	**73.6(4.3)**	**74.1(4.8)**
	GDL (Chap. 5)	70.1(3.3)	49.1(4.6)	87.4(5.0)	56.4(6.5)	85.9(4.3)	77.4(3.1)	70.7(3.9)	71.6(3.9)
	GDL$_\lambda$	71.5(4.1)	*50.1(4.8)*	88.1(7.8)	59.5(8.4)	86.5(5.4)	*78.1(4.4)*	71.5(4.2)	72.9(5.8)
OT kernel	FGWK	70.8(3.5)	48.9(3.9)	82.6(7.2)	56.2(8.9)	85.6(5.2)	77.0(4.2)	72.2(4.0)	72.4(4.7)
Kernels	SPK	56.2(2.9)	39.1(4.9)	83.3(8.0)	60.6(6.4)	-	-	-	-
	WLK	-	-	86.4(8.0)	63.1(6.6)	-	-	-	-
	HOPPERK	-	-	-	-	84.5(5.2)	79.7(3.5)	46.2(3.8)	72.1(3.1)
	PROPAK	-	-	-	-	80.0(5.1)	77.8(3.8)	71.8(5.8)	61.7(4.5)

graph structures.

Hence, FGWK refers to (F)GW kernels between raw input graphs, and other methods are SOTA graph kernels unrelated with OT.We perform 10-fold nested cross-validations repeated over 10 train/test splits, using same folds across methods, and same validated values for SVM's hyperparameters.

Results. Classification performances measured by means of accuracy are reported in Table 6.9. All variants of srGW DL lead to more discriminant graph representations than those of GDL, while consistently improving performances provided by FGWK opereting on raw graphs represented by their adjacency matrix. Notably srGW$_g$ consistently outperforms all benchmarked methods.

Moreover, our srGW DL naturally leads to embedded graphs of variable resolutions, contrary to GDL, with considerably small number of nodes relatively to their input representations. This behavior illustrated on IMDB-B in the figure 6.6 helps to drastically reduce the runtimes required to compute the GW pairwise matrices used in SVM, especially while promoting sparsity of our embeddings, as reported in Table 6.10. Therefore, the coupled discriminant denoising abilities and computational efficiency of our methods show that it could even be considered as a pre-processing step for GW based analysis.

TABLE 6.10: GW Kernel computation times (in ms) on different graph embeddings and input graphs, averaged over all corresponding pairs of graphs (499500 symmetric pairs in IMDB-B).

Models	Runtimes (ms)	
	min	max
srGW (ours)	4.7	11.1
srGW_g (ours)	1.7	3.7
GDL (Chap.5)	19.1	19.9
FGWK	24.7	

6.4.5 Graphs completion

Finally, we present graph completion results on the real world datasets IMDB-B and MUTAG, using the approach proposed in Section 6.3.2. We follow an analog procedure than in GDL experiments on graph completion detailed in Section 5.3.6. Then we report the benchmark of both GDL and srGW DL based completion methods.

Settings. The simulation of completion tasks on datasets IMDB-B and MUTAG are the same for GDL and srGW DL. The hyperparameters of the dictionaries (learned exclusively on $\mathcal{D}_{train}$) are validated in the same way as in the clustering benchmarks (Sections 5.3.2 and 6.4.3), except that extreme values 0 and 1 for α are omitted. Then the unmixings inherent to our completion algorithm are computed using the same settings than the DL. For both DL methods, we initialize the entries of $\boldsymbol{C}_{imp}$ by iid sampling from $\mathcal{N}(0.5, 0.01)$. For IMDB-B composed of social networks, we propose to leverage information from node degrees within $\boldsymbol{C}_{obs}$ to initialize connections between the observed nodes and the imputed ones. Specifically, new connections to a node p are sampled from $\mathcal{N}(\frac{d_p}{\max_q d_q}, 0.01)$ where for all q, d_q denotes the degree of node q observed within $\boldsymbol{C}_{obs}$. Finally for MUTAG, imputed features are initialized uniformly at random in the range of observed features.

Since the datasets do not explicitly contain graphs with missing nodes, we proceed as follow: first we split the dataset into a training dataset ($\mathcal{D}_{train}$) used to learn the dictionary and a test dataset ($\mathcal{D}_{test}$) reserved for the completion tasks. We then considered two different scenarios:

- Scenario 1: We vary the test dataset proportions from $\{10\%, 20\%, ..., 50\%\}$. Then for each graph of $\boldsymbol{C} \in \mathcal{D}_{test}$, we created incomplete graphs $\boldsymbol{C}_{obs}$ by independently removing 10% and 20% of their nodes, uniformly at random.
- Scenario 2: We fix the proportion of the test dataset to 10% and make percentage of imputed nodes vary in $\{10, 15, 20, 25, 30\}$.

For both scenarios, the partially observed graphs are then reconstructed using the procedure described in Section 5.3.6 and the average performance of each method is computed for 5 different dataset splits.

Results. The graph completion results for Scenario 1 are reported in Figures 6.7 and 6.8. Our srGW dictionary learning outperforms GDL consistently, when enough data is available to learn the atoms. When the proportion of train/test data varies, we can see that the linear GDL model that maintains the marginal constraints tends to be less sensitive to the scarcity of data. This can come form the fact that srGW is more flexible thanks to the optimization of $\overline{\boldsymbol{h}}$ but can slightly overfit when few data is available. Sparsity promoting

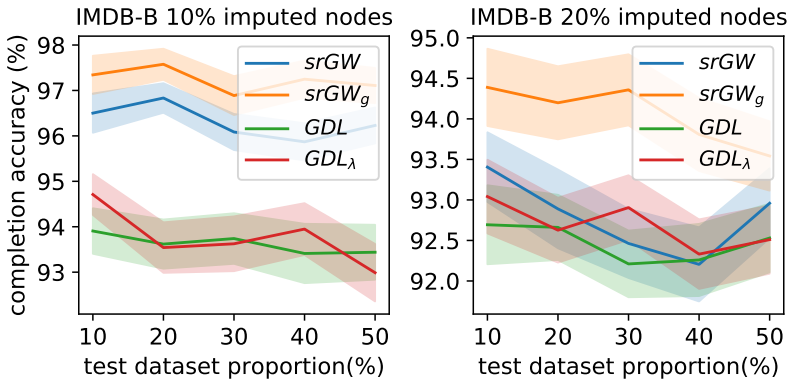

FIGURE 6.7: Completion performances for IMDB-B dataset, measured by means of accuracy for structures, respectively averaged over all imputed graphs.

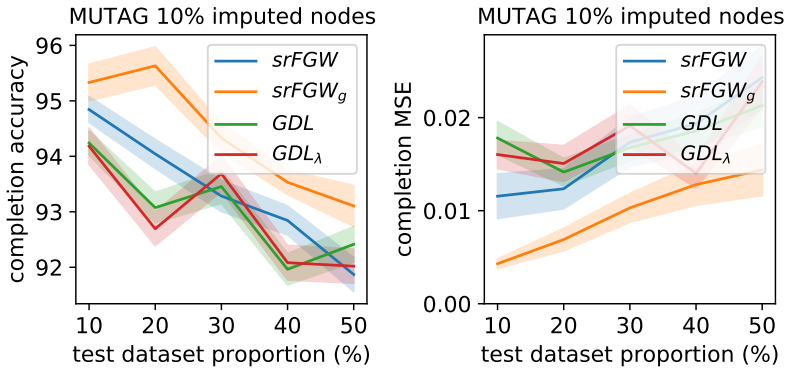

FIGURE 6.8: Completion performances for MUTAG dataset, measured by means of accuracy for structures and Mean Squared Error for node features, respectively averaged over all imputed graphs.

regularization can clearly compensate this overfitting and systematically leads to the best completion performances (high accuracy, low Mean Squared Error).

The graph completion results for Scenario 2 are reported in Figures 6.9 and 6.10. Our sr(F)GW dictionary learning and its regularized variants outperform GDL and GDL$_\lambda$ consistently when the percentage of imputed nodes is not too high ($< 20\%$), whereas this trend is reversed for high percentage of imputed nodes. Indeed, as sr(F)GW dictionaries capture subgraph patterns of variable resolutions from the input graphs, the scarcity of prior information in an observed graph leads to a too high number of valid possibilities to complete it. Whereas GDL dictionaries based on (F)GW lead to more steady performances as they keep their focus on global structures. Interestingly, the sparsity promoting regularization can clearly compensate this kind of overfitting over subgraphs for higher levels of imputed nodes and systematically leads to better completion performances (high accuracy, low Means Square Error). Moreover, the entropic regularization of srGW (srGW$_e$ and srGW$_{e+g}$) can be favorably used to compensate this overfitting pattern for high percentages of imputed nodes ($> 20\%$) and also pairs well with the sparse regularization (srGW$_{e+g}$).

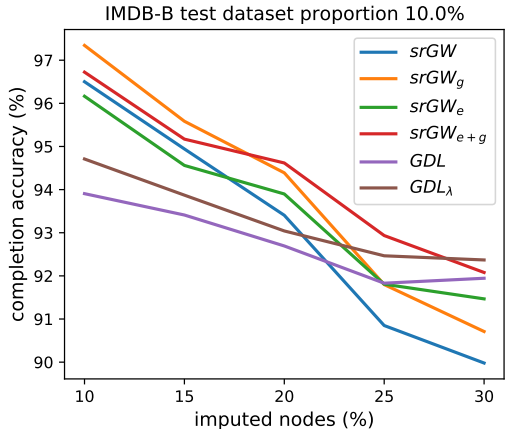

FIGURE 6.9: Completion performances for IMDB-B dataset, measured by means of accuracy for structures averaged over all imputed graphs.

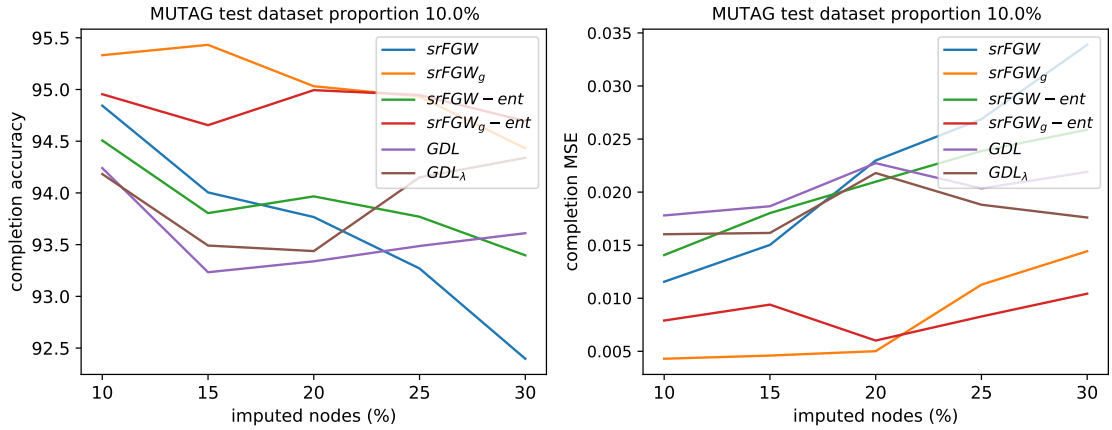

FIGURE 6.10: Completion performances for MUTAG dataset, measured by means of accuracy for structures and Mean Squared Error for node features, respectively averaged over all imputed graphs.

6.5 Conclusion

We introduced in this chapter new transport based divergences between structured data by relaxing the mass constraint on the second distribution of the Gromov-Wasserstein and Fused Gromov-Wasserstein problems (Section 6.2.1). After designing efficient solvers to estimate these divergences (Section 6.2.2), called the semi-relaxed (Fused) Gromov-Wasserstein sr(F)GW, we suggest to learn a unique graph to describe a dataset of graphs in the sr(F)GW sense. This novel modeling can be seen as a Dictionary Learning approach where graphs are embedded as a subgraph of a single atom.
Numerical experiments highlight the interest of our methods for graph partitioning (Sections 6.4.1 and 6.4.2), and representation learning of graphs datasets whose evaluation is conducted through clustering (Section 6.4.3), classification (Section 6.4.4) and completion (Section 6.4.5).

We believe that these new divergences will unlock the potential of (F)GW for graphs with unbalanced proportions of nodes. The associated fast numerical solvers allow to handle large size graph datasets, which was not possible with current (F)GW solvers. One interesting future research direction includes an analysis of sr(F)GW to perform parameters estimation of well-known random graph models such as Stochastic Block Models. Also, as relaxing the second marginal constraint in the original optimization problem gives more degrees of freedom to the underlying problem, one can expect dedicated regularization schemes, over

e.g. the level of sparsity of $\overline{\boldsymbol{h}}$, to address a variety of application needs, including pooling in Graph Neural Networks (See Chapters 2 and 4). Finally, our method can be seen as a special relaxation of the subgraph isomorphism problem. It remains to theoretically understand in which sense this relaxation holds.

Chapter 7

Conclusion

Contents

This last Chapter is a short conclusion to the manuscript. It summarizes the contributions presented in this thesis, while discussing research directions that seem relevant according to me.

7.1 Brief overview of the contributions

This thesis focuses on Graph Representation Learning (GRL) by means of Optimal Transport (OT) across incomparable spaces, both introduced in Chapter 2 and Chapter 3 respectively.

7.1.1 Optimal Transport for Graph Representation Learning

OT provides a set of distances for comparing graphs, modeled as discrete probability measures, where the nodes supporting them are whether assumed to belong to a common ambient space, or legitimately left as such in their respective graph topology. Both paradigms relate to the Wasserstein (Villani, 2009) and the (Fused) Gromov-Wasserstein ((F)GW) distances (Mémoli, 2011; Sturm, 2012; Vayer et al., 2020), respectively.
The latter carry a strong discriminant power for graph-level tasks (Vayer et al., 2019a), that we successfully leveraged in our new Graph Neural Networks (GNN, Section 2.2), called TFGW, presented in Chapter 4. Specifically, we showed that representing a graph via a vector of its FGW distances to learnable templates, acting as a *global pooling* in GNN (Section 2.2.2), enhances GNN expressivity and leads to state-of-the-art performances on graph classification tasks (Section 4.3).

Another convenient aspect of these distances between a source distribution and a well-chosen or estimated target one lie in the soft correspondences between nodes that they provide. Overall these properties can be used for a myriad of graph ML tasks. For instance, an OT matrix can be relevant for node-level tasks, such as the partitioning of a single graph (Xu et al., 2019a; Chowdhury & Needham, 2021) or of multiple ones simultaneously (Section 3.2.5).
The distances, themselves, can be used as fidelity terms for unsupervised GRL, while their intrinsic OT matrices, linking both input and learned graph representations, alleviate the pre-image problem that most GRL approaches suffer from. We illustrated this in Chapter 5 by introducing a novel Graph Dictionary Learning (GDL), that models a graph as a convex

combination of graph atoms. Learning such a dictionary from a given dataset provides a linear graph subspace, where the graphs, once embedded, can for example be better clustered or classified (Section 5.3). These downstream discriminative tasks are performed either by using (F)GW over embedded graphs with reduced orders (in a factorization context), or by using a dedicated Mahalanobis distance directly on embeddings that acts as a reasonable proxy for (F)GW (Section 5.2.3).

7.1.2 Learning the graph distribution

Nevertheless, the mass conservation at the core of OT, imposing a coupling between all the nodes from the two compared graphs, has specific implications in GRL. In many scenarios, learning only the structure and feature representations of a graph limits the modeling potential. The latter is fully exploited when the relative importance of nodes is also learned, as now possible using sub-gradients provided in Theorems 5 and 7 (Sections 3.2.5 & 3.3.4). Managing this extra degree of freedom inherent in OT has, *e.g.* improved the respective performance of TFGW and GW-based graph multi-partitioning, with minimal additional computational cost. We also extended GDL by adding a second and independent linear dictionary that models the relative importance of the nodes (Section 5.2.5).
This second dictionary mitigates the fixed resolution enforced by the dictionary over the structure, as node weights of the learned templates can become sparse. This way the (simultaneous) node weights estimation of the embedded graph can re-balance and even discard irrelevant nodes, in the vein of attention mechanisms, leading to a more representative modeling of the input. However, this extension of GDL has some drawbacks. First, it complicates both the unmixing mechanism and the estimation of dictionaries. Second, the Mahalanobis distance introduced for the vanilla GDL model can no longer be used to approximate (F)GW in the embedding space, so its potential generalization in this framework remains an open question.

Finally, we proposed in Chapter 6 to fully embrace this distribution learning paradigm by directly addressing the mass conservation constraint inherent in (F)GW. To this end, we propose to find correspondences between two graphs, while searching for a reweighed subgraph of the target graph at a minimum (F)GW distance from the input graph. This leads to the definition of a novel OT-based discrepancy, called the *semi-relaxed* (Fused) Gromov-Wasserstein divergence (sr(F)GW, Section 6.2). The latter can be estimated more efficiently than (F)GW (Section 6.2.2), while *e.g.* consequently enhancing GW-based approaches to partition a single input graph (Section 6.4.1). Moreover, considering a form of srFGW barycenter problem induces a novel DL, orthogonal to other GW-based DL (Chapter 5), where graphs are embedded as reweighed subgraphs of a single graph atom (Section 6.3). The learned representations compete favorably with other DL-based competitors, e.g. for graph clustering or classification, while being considerably faster to compute (section 6.4).

7.2 Perspectives for Optimal Transport on graphs

In the following, we discuss potential research directions motivated by an analysis of the pros and cons of the models proposed in this manuscript, or of related approaches. We identify two building blocks, where OT on graphs can lead to new models (or combinations thereof) that are potentially relevant for addressing graph ML tasks, namely: i) Discriminant modeling; ii) Dictionary Learning and generative modeling.

Notice that the computational complexity of OT on graphs is another fundamental limitation, that prevents its use for large scale graph ML tasks. We refer the reader to Section 3.2.4 for a brief overview of approaches that could be useful to relax the computational burden of (F)GW estimation in GRL.

7.2.1 Discriminant modeling

The conclusive results obtained with our TFGW *global pooling* motivate further research on its relevance to answer current questions of the GNN literature, *e.g.* regarding heterophily (Zhu et al., 2020; Zheng et al., 2022; Luan et al., 2022) and (over)smoothing (Topping et al., 2021; Chen et al., 2020b).
Both challenges, which are naturally dependent, relate to the neighborhood-based message-passing mechanism used in most GNN, to produce discriminant node features that also encode the graph topology. At present, these issues are mainly studied on node-level tasks, rather than on potentially more complex graph-level tasks, since graphs in a dataset can exhibit different levels of intra- and inter-class heterogeneity.

Heterophily in Graph Neural Networks. Graph heterophily points out that neighboring nodes in a graph have different labels, or distant features. This property is often observed in real-world scenarios, for example in molecular networks, as protein structures are more often composed of different types of amino acids linked together (Zhu et al., 2020). Such an information can be negatively affected, first by non-discriminant message passing scheme, such as neighborhood mean aggregation (*e.g.* GIN (Xu et al., 2019c)), second by an overly simplistic global pooling of the node embeddings. Addressing the first source of limitation has been shown to be relevant for graph-level task (*e.g.* (Ye et al., 2022)), using mean pooling, and might be further improved using OT. However the second source is still unexplored, and global pooling schemes where the graph structure is explicitly provided, like TFGW, should mitigate the information loss and be further studied theoretically.

Moreover, as discussed for TFGW in Section 4.2.2, the OT matrix encodes correspondences between respective nodes of the input and a template that will be propagated during the backward operation. So, enforcing or controlling the heterophily of template graphs, for instance through regularization losses, can be a way to preserve such a signal in the input graph.

Finally, exploring novel template-based global pooling using other graph (dis)similarities from the OT literature could provide a better trade-off between computation speed, accuracy and expressivity. For instance, it could be interesting to study this trade-off using srFGW. More specifically, the second marginal of an OT matrix ($\overline{\boldsymbol{h}}_i = \boldsymbol{T}^\top \mathbf{1}_n$), from srFGW of an input $\mathcal{G}_i$ onto a template graph $\overline{\mathcal{G}}_k$, can act as a relational global pooling. Learning from such representations requires "differentiation" through the matrix $\boldsymbol{T}$ of the FGW loss, which can be performed using *e.g.* our Mirror-Descent solver 13, whose iterations are auto-differentiable. The latter could compete with TFGW while using less templates, as graph embeddings will be nourished by knowledge from every sub-graphs contained in a template, instead of its global structure.
One might also envision to perform global pooling via Graph Diffuse Wasserstein distances (Barbe et al., 2021), from an input to graph templates encoded as $\left\{\exp\left(-\overline{t}_k \overline{\boldsymbol{L}}_k\right) \overline{\boldsymbol{F}}_k\right\}_{k\in[\![K]\!]}$, where $\overline{t}_k$, $\overline{\boldsymbol{L}}_k$ and $\overline{\boldsymbol{F}}_k$, are respectively the heat parameter, the Laplacian and feature matrices of a template. The use of Wasserstein discrepancies instead of FGW-based ones is interesting to reduce the computational complexity of the pooling operation. However, the explicit introduction of diffusion in the learned graph representations could also be subject to excessive smoothing, detrimental to the encoding of a heterophilic behavior.

Over-smoothing in Graph Neural Networks. On the other hand, *over-smoothing* (*i.e.* that GNN performances do not necessarily increase with the number of layers) also affect GNN with TFGW pooling (see *e.g.* Figures 4.9 and 4.10).
Theoretical studies on this phenomenon are for now dedicated to node-level tasks, and seminal results on linear GNN indicate that a trade-off in their depth has to be found (Keriven, 2022).

Studying this phenomenon on graph-level tasks with linear GNN and TFGW pooling (finally fed to a Logistic model), could be an interesting first step to better explain performances reached in Chapter 4 and build upon it.
Analyses that consider non-linear GNN as GIN are for now only studied asymptotically *w.r.t* the number of layers, tracking for instance the Dirichlet energy of node embeddings across layers (Cai & Wang, 2020). The latter resonates with common energies used to define special types of evolution equations as *Gradient Flows* to describe dynamical particle systems (Kichenassamy et al., 1995; Santambrogio, 2017). De facto, several GNN have recently been studied as discrete dynamical systems following specific energies (Di Giovanni et al., 2022), and considering FGW as such might be of real interest.

Novel frameworks for graph pooling. One drawback of TFGW is that the learned templates are not interpretable and correspond to extreme points of an envelop in the distance embedding (Section 4.3.3). Learning templates as barycenters of inputs belonging to a given class could enhance model interpretability. A distribution over classes can then be obtained using a softmax over FGW distances of an input to the barycentric templates (Snell et al., 2017).
However, iterative FGW barycenter estimation can have a prohibitive computational cost. Relevant barycenters could rather be estimated from srFGW embeddings of a graph, onto a (small) graph template that prescribes the barycenters' topology. Even if FGW remains the natural distance to compare the corresponding embedded graphs, establishing new relations between FGW and *e.g.* a linear OT problem using the graph topology as ground-cost could bring new theoretical and practical benefits. Notice that OT on a given graph metric space has already been studied in various settings that might provide some leads (*e.g.* (Tong et al., 2021; 2022; Le et al., 2022)).
Another perspective consists in using embedded graphs provided by the srFGW projection of an input onto a template, directly as a *learnable hierarchical graph pooling* (Section 2.2.2). Such an approach could bring a novel notion of "convolution" in the GNN literature.

7.2.2 Dictionary Learning and Generative modeling

On one hand, we addressed in Chapters 5 and 6 diverse unsupervised graph ML tasks, such as graph clustering and completion, by means of Dictionary Learning (DL). These methods come with their own limitations, such as their robustness, which would benefit from further study. On the other hand, lots of novel Graph autoencoders (GAE) have emerged from the GNN literature to learn network embeddings, mostly for node-level tasks, or generate new graphs (Section 6, Wu et al. (2020)). GAE are deep neural architectures which map nodes of a graph into a latent feature space and decode graph information from these latent representations. The position of the two types of approaches is still unknown as to their respective performances on these unsupervised tasks, plus the mixing of both paradigms is almost unexplored, and should be investigated in a near future.

Going further on (sr)FGW-based DL. Classical OT to compare probability distributions is known to be sensitive to outliers and missing data. Unbalanced OT, which compares arbitrary positive measures whose total mass can vary, is a well studied way to limit these sensitivity issues (Séjourné et al., 2022). This comparison translates to (F)GW which can suffer from structural or feature noise naturally observed in real-world graphs. So we can expect representations learned with our FGW based DL approaches to be sensitive to those. The same holds to some extent for our srFGW DL, although it is an extreme case of unbalanced FGW (UFGW, Thual et al. (2022)), the total amount of mass remains the same between both distributions. So exploring novel settings in UFGW for graph DL could be a direct

path to better and more robust representations, with promising relations to the literature on inexact graph matching (Bunke & Allermann, 1983; Gori et al., 2005a).

Robustness could also be addressed thanks to a better understanding of the variations of the OT matrix from (sr)FGW between two graphs, w.r.t common graph edits, like edge removal or node feature transformations. The latter could lead to novel OT discrepancies, where structural or feature perturbations can be directly included in the matching loss. At a higher level, linear unmixing inherent in GDL, whose variations were partly studied in Section 5.2.2, could also be made robust by including *e.g.* additive perturbations with a bounded norm.

Finally, node feature embeddings which are ubiquitous in the GNN literature, deserve more attention from FGW based DL methods, in my opinion. As most neighborhood aggregation schemes are not invertible, such a feature processing is not relevant for DL, where one wants to be able to reconstruct the graph from its embedding. Interestingly, a novel class of invertible propagation schemes has been proposed for Graph Normalizing Flows (GNF) (Liu et al., 2019). The latter could be used to define non-trivial multi-resolution DL, for instance by learning a srFGW atom fitting well the graphs produced at each forward-pass in a GNF layer. A consensus from unmixings computed at each resolution could outperform vanilla srFGW DL on graphs clustering and completion.

FGW-based generative modeling. The seminal work of Xu et al. (2021a) made a first link between GW barycenter estimation and generative graph models called (L^p) graphons (Diaconis & Janson, 2007; Borgs et al., 2019). The latter are graph limits formalized as (symmetric) measurable functions $W : \Omega^2 \to \mathcal{Y} = [0, 1]$, with Ω a measurable space, satisfying *e.g.* $\Omega = [0, 1]$ without loss of generality. Graphs of any order n can be sampled from a graphon, first sampling n points $\{x_i\}_{i\in[\![n]\!]}$ uniformly in $\Omega = [0, 1]$ or using a predefined grid. Then sampling the graph edges such that $A_{ij} \sim \text{Bernouilli}(W(x_i, x_j))$. Such a modeling is known to produce dense graphs, and can be adapted to handle sparse graphs setting $\mathcal{Y} = \mathbb{R}$ and modifying the edge sampling (Borgs et al., 2016; Fabian et al., 2022).
Graphons can be approximated by step-functions, which can be estimated via a GW barycenter of observed graphs (Xu et al., 2021a), whose structure and node weights dictate respectively the graph edge sampling and the partitioning of Ω. Naturally, the barycenter order limits the resolution of the graphon estimation. A first solution to this problem is to model the GW barycenter implicitly via a neural network $f_\theta : \Omega^2 \to [0, 1]$ that outputs edge connections from a pair of points sampled in Ω (Xia et al., 2022; Sitzmann et al., 2020).

Any GW-based DL can then be interpreted as a manifold of graphons from which graphs can be sampled. This theory can also be extended to attributed graphs, and has been used to derive a Graphon Autoencoder in Xu et al. (2021b). This paradigm allows to envision a wide-range of novel generative models for unsupervised learning. And also might be used to design adversarial training (Goodfellow et al., 2020; Miyato et al., 2018; Zhang et al., 2019a; Fatras et al., 2021a) of GNN so they can lead to more robust representations.

On the theoretical side, relations with GW barycenter and graphons as limits of random graphs, motivates the investigation of (sr)GW-based parameter estimation of well-known random graph models, being actually graphons, such as Stochastic Block Models (Holland et al. (1983)), or even Random Geometric Graph Models (Araya Valdivia & Yohann, 2019). Moreover, more generic class of graph limits called Graphexes have been recently proposed (Borgs et al., 2021), so understanding their relations to GW, or more generally to OT, might open a novel door to graph generative modeling.

Chapter 8

Annexes

8.1 Proofs of Chapter 3

8.1.1 Proof of Theorem 5: subgradient w.r.t distributions of GW

In this section we will prove the following result:

Theorem 4 (subgradient w.r.t masses of GW problem) *Let $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ be two graphs. Let $\boldsymbol{T}^*$ be an optimal coupling of the GW problem between $(\boldsymbol{C}, \boldsymbol{h}), (\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$. We define the following cost matrix $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* = \left(\sum_{kl}(C_{1,ik} - C_{2,jl})^2 T^*_{kl}\right)_{ij}$. Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the following linear OT problem:*

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T} \rangle_F \tag{8.1}$$

Then $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ (resp $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$) is a subgradient of the function $\mathrm{GW}_2^2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \cdot\,, \overline{\boldsymbol{h}})$ (resp $\mathrm{GW}_2^2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \cdot\,)$).

Before proving the Theorem 5, let us state the following theorem which relates a solution of a Quadratic Program (QP) with a solution of a Linear Program (LP):

Theorem 12 (Murty (1988), Section 1) *Consider the following (QP):*

$$\begin{array}{ll} \min_{\boldsymbol{x}} f(\boldsymbol{x}) & = \mathbf{c}\boldsymbol{x} + \boldsymbol{x}^T \mathbf{Q} \boldsymbol{x} \\ s.t. & \mathbf{A}\boldsymbol{x} = \mathbf{b},\ \boldsymbol{x} \geq 0 \end{array} \tag{8.2}$$

Then if $\boldsymbol{x}_*$ is an optimal solution of (8.2) it is an optimal solution of the following (LP):

$$\begin{array}{ll} \min_{\boldsymbol{x}} f(\boldsymbol{x}) & = (\mathbf{c} + \boldsymbol{x}_*^T \mathbf{Q})\boldsymbol{x} \\ s.t. & \mathbf{A}\boldsymbol{x} = \mathbf{b},\ \boldsymbol{x} \geq 0 \end{array} \tag{8.3}$$

Proof of the Theorem 4.

Step 1. In the following $\boldsymbol{T} \geq 0$ should be understood as $\forall i, j\ T_{ij} \geq 0$. Let $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ be two graphs of order n and m with $\boldsymbol{C}^1 \in S_n(\mathbb{R}), \boldsymbol{C}^2 \in S_m(\mathbb{R})$ and $(\boldsymbol{h}, \overline{\boldsymbol{h}}) \in \Sigma_n \times \Sigma_m$. Let $\boldsymbol{T}^*$ be an optimal solution of the GW problem *i.e.* $\mathrm{GW}_2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \overline{\boldsymbol{h}}) = \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*, \boldsymbol{T}^* \rangle_F$. We define $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*$. We consider the problem:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T} \rangle_F = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*, \boldsymbol{T} \rangle_F \tag{8.4}$$

We will first show that the optimal coupling for the Gromov-Wasserstein problem is also an optimal coupling for the problem (8.4), *i.e* $\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T} \rangle_F = \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}^* \rangle_F$.

Applying Theorem 12 to our case gives exactly that:

$$\boldsymbol{T}^* \in \underset{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})}{\arg\min} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{8.5}$$

since $\boldsymbol{T}^*$ is an optimal solution of the GW problem and so $\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})}\langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F$.

Now let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be an optimal solution to the dual problem of (8.4). Then by strong duality it implies that:

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle = \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F \tag{8.6}$$

Since $\langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F = \mathrm{GW}_2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \overline{\boldsymbol{h}})$ we have:

$$\mathrm{GW}_2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \overline{\boldsymbol{h}}) = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \tag{8.7}$$

Step 2. To prove the Theorem 5 the objective is to show that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \boldsymbol{q})$ (by symmetry the result will be true for $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$). In other words we want to prove that:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle - \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \leq F(\boldsymbol{q}) - F(\overline{\boldsymbol{h}}) \tag{8.8}$$

We will show in the following an equivalent formulation of this condition. The dual variable $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{GW}_2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \boldsymbol{q})$ if and only if:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle \leq F(\boldsymbol{q}) \tag{8.9}$$

Indeed $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient if and only if:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle - \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \leq F(\boldsymbol{q}) - F(\overline{\boldsymbol{h}}) \tag{8.10}$$

However using (8.7) and the definition of F we have:

$$F(\overline{\boldsymbol{h}}) = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \tag{8.11}$$

So overall:

$$\begin{aligned} &\langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle - \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \leq F(\boldsymbol{q}) - (\langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle) \\ &\iff \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle \leq F(\boldsymbol{q}) \end{aligned} \tag{8.12}$$

Step 3. In order to prove Theorem 5 we have to prove that the equivalent condition given by equation (8.12) is satisfied. We will do so by proving the following lemma leveraging the weak-duality of the GW problem:

Lemma 1 *For any vectors $\boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\nu} \in \mathbb{R}^m$ we define:*

$$\mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\nu}) := \min_{\boldsymbol{T}\geq 0} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} - \boldsymbol{\mu}\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^\top, \boldsymbol{T}\rangle$$

Let $\boldsymbol{T}^*$ be an optimal solution of the GW problem. Consider:

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{8.13}$$

where $\boldsymbol{M}(\boldsymbol{T}^*) := \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*$. *Let* $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ *be the dual variables of the problem in* (8.13). *If* $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ *then* $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ *is a subgradient of* $F : \boldsymbol{q} \to \mathrm{GW}_2(\boldsymbol{C}, \boldsymbol{C}, \boldsymbol{h}, \boldsymbol{q})$

Proof of lemma 1. Let $\boldsymbol{q} \in \Sigma_m$ be any weights vector be fixed. Recall that $F : \boldsymbol{q} \to \mathrm{GW}_2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \boldsymbol{q})$ so that:

$$F(\boldsymbol{q}) = \mathrm{GW}_2(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{h}, \boldsymbol{q}) = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \boldsymbol{q})} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}, \boldsymbol{T} \rangle \tag{8.14}$$

The Lagrangian associated to (8.14) reads:

$$\min_{\boldsymbol{T} \geq 0} \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \text{ where } \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) := \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}, \boldsymbol{T} \rangle + \langle \boldsymbol{h} - \boldsymbol{T}\mathbf{1}_m, \boldsymbol{\mu} \rangle + \langle \boldsymbol{q} - \boldsymbol{T}^\top \mathbf{1}_n, \boldsymbol{\nu} \rangle \tag{8.15}$$

Moreover by weak Lagrangian duality:

$$\min_{\boldsymbol{T} \geq 0} \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \geq \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \min_{\boldsymbol{T} \geq 0} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \tag{8.16}$$

However:

$$\begin{aligned} \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \min_{\boldsymbol{T} \geq 0} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) &= \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \langle \boldsymbol{\mu}, \boldsymbol{h} \rangle + \langle \boldsymbol{\nu}, \boldsymbol{q} \rangle + \min_{\boldsymbol{T} \geq 0} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} - \boldsymbol{\mu}\mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^\top, \boldsymbol{T} \rangle \\ &= \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \langle \boldsymbol{\mu}, \boldsymbol{h} \rangle + \langle \boldsymbol{\nu}, \boldsymbol{q} \rangle + \mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\nu}) \end{aligned}$$

So by considering the dual variable $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ defined previously we have:

$$\max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \min_{\boldsymbol{T} \geq 0} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \geq \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h} \rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q} \rangle + \mathcal{F}(\boldsymbol{\mu}^*(\mathcal{F}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \tag{8.17}$$

Now combining (8.16) and (8.17) we have:

$$\min_{\boldsymbol{T} \geq 0} \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \geq \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h} \rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q} \rangle + \mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \tag{8.18}$$

Since $F(\boldsymbol{q}) = \min_{\boldsymbol{T} \geq 0} \max_{\boldsymbol{\mu}, \boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu})$ we have proven that:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q} \rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h} \rangle + \mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \leq F(\boldsymbol{q}) \tag{8.19}$$

However equation (8.12) states that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of F if and only if:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q} \rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h} \rangle \leq F(\boldsymbol{q}) \tag{8.20}$$

So combining (8.19) with (8.12) proves:

$$\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \geq 0 \implies \boldsymbol{\nu}^*(\boldsymbol{T}^*) \text{ is a subgradient of } F \tag{8.21}$$

However we have $F(\overline{\boldsymbol{h}}) = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h} \rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}} \rangle$ by (8.11). So $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \leq 0$ using (8.19) with $\boldsymbol{q} = \overline{\boldsymbol{h}}$. So we can only hope to have $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$. □

Step 4. The previous lemma states that it is sufficient to look at the quantity $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*))$ in order to prove that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of F. Interestingly the condition $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ is satisfied as proven in the following:

We want to study:

$$\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = \min_{\boldsymbol{T} \geq 0} \langle \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top, \boldsymbol{T} \rangle$$

We define $H(\boldsymbol{T}) := \langle Ł(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top, \boldsymbol{T}\rangle$. Since $\boldsymbol{T}^*$ is optimal coupling for $\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})}\langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F$ by (8.5) then for all i, j we have $T^*_{ij}(\boldsymbol{M}(\boldsymbol{T}^*)_{ij} - \alpha^*_i(\boldsymbol{T}^*) - \beta^*_j(\boldsymbol{T}^*)) = 0$ by the property of the optimal couplings for the Wasserstein problems. Equivalently:

$$\forall (i,j) \in [n] \times [m],\ T^*_{ij}([\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*]_{ij} - \alpha^*_i(\boldsymbol{T}^*) - \beta^*_j(\boldsymbol{T}^*)) = 0 \tag{8.22}$$

Then:

$$\begin{aligned} H(\boldsymbol{T}^*) &= \mathrm{Tr}\left(\boldsymbol{T}^{*\top}(\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top)\right) \\ &= \sum_{ij} T^*_{ij}(\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top)_{ij} \\ &= \sum_{ij} T^*_{ij}([\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*]_{ij} - \alpha^*_i(\boldsymbol{T}^*) - \beta^*_j(\boldsymbol{T}^*)) = 0 \end{aligned} \tag{8.23}$$

Which proves $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$. □

8.1.2 Proof of Theorem 7: subgradient w.r.t distributions of FGW

In this section we will prove the following theorem 7 which extends the theorem 5 to the FGW distance:

Theorem 6 (subgradient w.r.t masses of FGW) *Let $(\boldsymbol{C}, , \boldsymbol{F}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$ be two attributed graphs. Let $\boldsymbol{T}^*$ be an optimal coupling of the FGW problem between $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}), (\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. We define the following cost matrix $\boldsymbol{M}_\alpha(\boldsymbol{T}^*) := \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) = \left(\alpha\sum_{kl}(C_{1,ik} - C_{2,jl})^2 T^*_{kl} + (1-\alpha)\|\boldsymbol{F}_i - \overline{\boldsymbol{F}}_j\|_2^2\right)_{ij}$. Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the following linear OT problem:*

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{8.24}$$

Then $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$ (resp $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$) is a subgradient of the function $\mathrm{FGW}^2_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \cdot, \overline{\boldsymbol{h}})$ (resp $\mathrm{FGW}^2_2(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \cdot)$).

Proof of the Theorem 6.

Step 1. Let $\boldsymbol{T}^*$ be an optimal solution of the FGW problem *i.e.* $\mathrm{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \overline{\boldsymbol{h}}) = \langle \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}), \boldsymbol{T}^*\rangle_F$. We define $\boldsymbol{M}_\alpha(\boldsymbol{T}^*) := \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}})$. We consider the problem:

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}), \boldsymbol{T}\rangle_F \tag{8.25}$$

We will first show that the optimal coupling for the Fused Gromov-Wasserstein problem is also an optimal coupling for the problem (8.25), *i.e* $\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})}\langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F$. Applying Theorem 12 to our case gives exactly that:

$$\boldsymbol{T}^* \in \underset{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})}{\arg\min} \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F \tag{8.26}$$

since $\boldsymbol{T}^*$ is an optimal solution of the FGW problem, we do have the desired relation $\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})}\langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F$.

Now let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be an optimal solution to the dual problem of (8.4). Then by strong duality it implies that:

$$\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle = \langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F \tag{8.27}$$

Since $\langle \boldsymbol{M}_\alpha(\boldsymbol{T}^*), \boldsymbol{T}^*\rangle_F = \mathrm{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \overline{\boldsymbol{h}})$ we have:

$$\mathrm{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \overline{\boldsymbol{h}}) = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \tag{8.28}$$

Step 2. To prove the Theorem 7 the objective is to show that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \boldsymbol{q})$ (by symmetry the result will be true for $\boldsymbol{\mu}^*(\boldsymbol{T}^*)$). In other words we want to prove that:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle - \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \leq F(\boldsymbol{q}) - F(\overline{\boldsymbol{h}}) \tag{8.29}$$

Following the same development than for the step 2 of the proof of theorem 5, we have the following result for FGW, which is equivalent to (8.12) for GW. The dual variable $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \boldsymbol{q})$ if and only if:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle \leq F(\boldsymbol{q}) \tag{8.30}$$

Step 3. In order to prove Proposition 7 we have to prove that the equivalent condition given by equation (8.30) is satisfied. To this end we will prove an analog result for FGW than the one stated for the Lagrangian of GW in Lemma 1.

Let us define for any vectors $\boldsymbol{\mu} \in \mathbb{R}^n, \boldsymbol{\nu} \in \mathbb{R}^m$,

$$\begin{aligned}\mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\nu}) &:= \min_{\boldsymbol{T}\geq 0} \langle \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) - \boldsymbol{\mu}\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^\top, \boldsymbol{T}\rangle \\ &= \min_{\boldsymbol{T}\geq 0} \langle \boldsymbol{M}_\alpha(\boldsymbol{T}) - \boldsymbol{\mu}\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^\top, \boldsymbol{T}\rangle\end{aligned}$$

Let $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ be the dual variables of the problem in (8.26). We will the following implication: If $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ then $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of $F : \boldsymbol{q} \to \mathrm{FGW}_{2,\alpha}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{h}, \boldsymbol{q})$.

Let $\boldsymbol{q} \in \Sigma_m$ be any weights vector be fixed. The Lagrangian associated to the primal problem giving $F(\boldsymbol{q})$ reads

$$F(\boldsymbol{q}) \min_{\boldsymbol{T}\geq 0} \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \tag{8.31}$$

where

$$\begin{aligned}\mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) &:= \langle \alpha\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}), \boldsymbol{T}\rangle + \langle \boldsymbol{h} - \boldsymbol{T}\mathbf{1}_m, \boldsymbol{\mu}\rangle + \langle \boldsymbol{q} - \boldsymbol{T}^\top\mathbf{1}_n, \boldsymbol{\nu}\rangle \\ &= \langle \boldsymbol{M}_\alpha(\boldsymbol{T}), \boldsymbol{T}\rangle + \langle \boldsymbol{h} - \boldsymbol{T}\mathbf{1}_m, \boldsymbol{\mu}\rangle + \langle \boldsymbol{q} - \boldsymbol{T}^\top\mathbf{1}_n, \boldsymbol{\nu}\rangle\end{aligned} \tag{8.32}$$

Moreover by weak Lagrangian duality:

$$\min_{\boldsymbol{T}\geq 0} \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \geq \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \min_{\boldsymbol{T}\geq 0} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) \tag{8.33}$$

However:

$$\begin{aligned}\max_{\boldsymbol{\mu},\boldsymbol{\nu}} \min_{\boldsymbol{T}\geq 0} \mathtt{L}(\boldsymbol{T}, \boldsymbol{\mu}, \boldsymbol{\nu}) &= \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \langle \boldsymbol{\mu}, \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}, \boldsymbol{q}\rangle + \min_{\boldsymbol{T}\geq 0} \langle \boldsymbol{M}_\alpha(\boldsymbol{T}) - \boldsymbol{\mu}\mathbf{1}_m^\top - \mathbf{1}_n\boldsymbol{\nu}^\top, \boldsymbol{T}\rangle \\ &= \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \langle \boldsymbol{\mu}, \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}, \boldsymbol{q}\rangle + \mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\nu})\end{aligned}$$

So by considering the dual variable $\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)$ defined previously we have:

$$\max_{\boldsymbol{\mu},\boldsymbol{\nu}} \min_{\boldsymbol{T}\geq 0} \mathtt{L}(\boldsymbol{T},\boldsymbol{\mu},\boldsymbol{\nu}) \geq \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \tag{8.34}$$

Now combining (8.33) and (8.34) we have:

$$\min_{\boldsymbol{T}\geq 0} \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T},\boldsymbol{\mu},\boldsymbol{\nu}) \geq \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \tag{8.35}$$

Since $F(\boldsymbol{q}) = \min_{\boldsymbol{T}\geq 0} \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \mathtt{L}(\boldsymbol{T},\boldsymbol{\mu},\boldsymbol{\nu})$ we have proven that:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \leq F(\boldsymbol{q}) \tag{8.36}$$

However equation 8.30 states that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of F if and only if:

$$\forall \boldsymbol{q} \in \Sigma_m, \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \boldsymbol{q}\rangle + \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle \leq F(\boldsymbol{q}) \tag{8.37}$$

So combining (8.36) with (8.30) proves:

$$\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \geq 0 \implies \boldsymbol{\nu}^*(\boldsymbol{T}^*) \text{ is a subgradient of } F \tag{8.38}$$

However using the definition of F and the relation in 8.28, we have

$$F(\overline{\boldsymbol{h}}) = \langle \boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{h}\rangle + \langle \boldsymbol{\nu}^*(\boldsymbol{T}^*), \overline{\boldsymbol{h}}\rangle \tag{8.39}$$

So injecting (8.39) in (8.36) with $\boldsymbol{q} = \overline{\boldsymbol{h}}$, we have $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) \leq 0$. So we can only hope to have $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$. □

Step 4. The previous lemma states that it is sufficient to look at the quantity $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*))$ in order to prove that $\boldsymbol{\nu}^*(\boldsymbol{T}^*)$ is a subgradient of F. Interestingly the condition $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$ is satisfied as proven in the following:

We want to study:

$$\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = \min_{\boldsymbol{T}\geq 0} \langle \alpha \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top, \boldsymbol{T}\rangle$$

We define $H(\boldsymbol{T}) := \langle \alpha \mathrm{Ł}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top, \boldsymbol{T}\rangle$. Since $\boldsymbol{T}^*$ is optimal coupling for $\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \langle \boldsymbol{M}(\boldsymbol{T}^*), \boldsymbol{T}\rangle_F$ by (8.26) then for all i, j we have $T^*_{ij}(\boldsymbol{M}(\boldsymbol{T}^*)_{ij} - \mu^*_i(\boldsymbol{T}^*) - \nu^*_j(\boldsymbol{T}^*)) = 0$ by the property of the optimal couplings for the Wasserstein problems. Equivalently:

$$\forall (i,j) \in [n] \times [m],\ T^*_{ij}(\alpha[\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*]_{ij} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}})_{ij} - \mu^*_i(\boldsymbol{T}^*) - \nu^*_j(\boldsymbol{T}^*)) = 0 \tag{8.40}$$

Then:

$$\begin{aligned} H(\boldsymbol{T}^*) &= \mathrm{Tr}\left(\boldsymbol{T}^{*\top}(\alpha \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top)\right) \\ &= \sum_{ij} T^*_{ij}(\alpha \mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^* + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}}) - \boldsymbol{\mu}^*(\boldsymbol{T}^*)\mathbf{1}_m^\top - \mathbf{1}_n \boldsymbol{\nu}^*(\boldsymbol{T}^*)^\top)_{ij} \\ &= \sum_{ij} T^*_{ij}(\alpha[\mathcal{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T}^*]_{ij} + (1-\alpha)\boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}})_{ij} - \mu^*_i(\boldsymbol{T}^*) - \nu^*_j(\boldsymbol{T}^*)) = 0 \end{aligned} \tag{8.41}$$

Which proves $\mathcal{F}(\boldsymbol{\mu}^*(\boldsymbol{T}^*), \boldsymbol{\nu}^*(\boldsymbol{T}^*)) = 0$.
□

8.2 Proofs and additional results of Chapter 4

8.2.1 Notations and preliminaries

Notations An undirected attributed graph $\mathcal{G}$ with n nodes can be modeled in the OT context as a tuple $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$, where $\boldsymbol{C} \in \mathbb{S}_n(\mathbb{R})$ is a matrix encoding relationships between nodes, $\boldsymbol{F} = (\boldsymbol{f}_1, ..., \boldsymbol{f}_n)^\top \in \mathbb{R}^{n \times d}$ is a node feature matrix and $\boldsymbol{h} \in \Sigma_n$ is a vector of weights modeling the relative importance of the nodes within the graph (Figure 1 of the main paper). *We always assume in the following that values in $\boldsymbol{C}$ and $\boldsymbol{F}$ are finite.* Let us now consider two such graphs $\mathcal{G} = (\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and $\overline{\mathcal{G}} = (\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$, of respective sizes n and $\overline{n}$ (with possibly $n \neq \overline{n}$). The Fused Gromov-Wasserstein (FGW) distance is defined for $\alpha \in [0, 1]$ as Vayer et al. (2020; 2019a):

$$\mathrm{FGW}_\alpha(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) = \min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}) \tag{8.42}$$

with $\mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}}) := \{\boldsymbol{T} \in \mathbb{R}_+^{n \times \overline{n}} | \boldsymbol{T}\mathbf{1}_{\overline{n}} = \boldsymbol{h}, \boldsymbol{T}^\top \mathbf{1}_n = \overline{\boldsymbol{h}}\}$, the set of admissible coupling between $\boldsymbol{h}$ and $\overline{\boldsymbol{h}}$. For any $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$, the FGW cost $\mathcal{E}_\alpha^{FGW}$ can be decomposed as

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}) = \alpha \mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) + (1 - \alpha)\mathcal{E}^{W}(\boldsymbol{F}, \overline{\boldsymbol{F}}, \boldsymbol{T}) \tag{8.43}$$

which respectively refers to a Gromov-Wasserstein matching cost $\mathcal{E}^{GW}$ between graph structures $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ reading as

$$\mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) = \sum_{ijkl} (C_{ij} - \overline{C}_{kl})^2 T_{ik} T_{jl} \tag{8.44}$$

and a Wasserstein matching cost $\mathcal{E}^W$ between nodes features $\boldsymbol{F}$ and $\overline{\boldsymbol{F}}$,

$$\mathcal{E}^{W}(\boldsymbol{F}, \overline{\boldsymbol{F}}, \boldsymbol{T}) = \sum_{ik} \|\boldsymbol{f}_i - \overline{\boldsymbol{f}}_k\|_2^2 T_{ik} \tag{8.45}$$

Preliminaries Given two graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$, we first provide a reformulation of each matching costs $\mathcal{E}^{GW}$ and $\mathcal{E}^W$ through matrix operations which will facilitate the readability of our proof.

By first expanding the GW matching cost given in 8.44 and using the marginal constraints over $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$, $\mathcal{E}^{GW}$ can be expressed as

$$\begin{aligned}
\mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}) &= \sum_{ij} C_{ij}^2 h_i h_j + \sum_{kl} \overline{C}_{kl}^2 \overline{h}_k \overline{h}_l - 2 \sum_{ijkl} C_{ij} \overline{C}_{kl} T_{ik} T_{jl} \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{T}^\top \boldsymbol{C}^2 \boldsymbol{T}, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{T} \overline{\boldsymbol{C}}^2 \boldsymbol{T}^\top, \mathbf{1}_{n \times n} \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle
\end{aligned} \tag{8.46}$$

where power operations are applied element-wise and $\mathbf{1}^{p \times q}$ is the matrix of ones of size $p \times q$ for any integers p and q.

Then through similar operations $\mathcal{E}^W$ can be expressed as

$$\begin{aligned}
\mathcal{E}^{W}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}, \boldsymbol{T}) &= \sum_{i} \|\boldsymbol{f}_i\|_2^2 h_i + \sum_{k} \|\overline{\boldsymbol{f}}_k\|_2^2 \overline{h}_k - 2 \sum_{ik} \langle \boldsymbol{f}_i, \overline{\boldsymbol{f}}_k \rangle T_{ik} \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h} \rangle + \langle \overline{\boldsymbol{F}}^2 \mathbf{1}_d, \overline{\boldsymbol{h}} \rangle - 2\langle \boldsymbol{F} \overline{\boldsymbol{F}}^\top, \boldsymbol{T} \rangle \\
&= \langle \boldsymbol{T}^\top \boldsymbol{F}^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{T} \overline{\boldsymbol{F}}^2, \mathbf{1}_{n \times d} \rangle - 2\langle \boldsymbol{F}^\top \boldsymbol{T}, \overline{\boldsymbol{F}}^\top \rangle
\end{aligned} \tag{8.47}$$

8.2.2 Proof of Lemma 3: TFGW invariance to strong isomorphism

Lemma 2 (TFGW invariance to strong isomorphism) *The* TFGW *embeddings are invariant to strong isomorphism.*

Proof of Lemma 2. First, as our TFGW embeddings can operate after embedding the nodes feature of any graph, let us also introduce such an application. Given any feature matrix $\boldsymbol{F} = (\boldsymbol{f}_1, ..., \boldsymbol{f}_n)^\top \subset \mathbb{R}^{n \times d}$, we denote by $\phi : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d'}$ an application such that $\phi(\boldsymbol{F}) = (\varphi(\boldsymbol{f}_1), ..., \varphi(\boldsymbol{f}_n))^\top$ with $\varphi : \mathbb{R}^d \to \mathbb{R}^{d'}$.

Let us now consider any pair of graphs $\mathcal{G}_1 = (\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1)$ and $\mathcal{G}_2 = (\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}_2)$ defined as in Subsection 8.2.1. Assume that $\mathcal{G}_1$ and $\mathcal{G}_2$ are *strongly isomorphic*. This is equivalent to assuming that they have the same number of nodes n and there exists a permutation matrix $\boldsymbol{P} \in \{0,1\}^{n \times n}$ such that $\boldsymbol{C}_2 = \boldsymbol{P}\boldsymbol{C}_1\boldsymbol{P}^\top$, $\boldsymbol{F}_2 = \boldsymbol{P}\boldsymbol{F}_1$ and $\boldsymbol{h}_2 = \boldsymbol{P}\boldsymbol{h}_1$ (Vayer et al., 2019a; Chowdhury & Mémoli, 2019).

First observe that the application ϕ preserves the relation of strong isomorphism. Indeed, as ϕ operates on each node independently through φ, we have $\phi(\boldsymbol{F}_2) = \boldsymbol{P}\phi(\boldsymbol{F}_1)$ *i.e*,

$$\phi(\boldsymbol{F}_2) = (\varphi(\boldsymbol{F}_{2,1}), ..., \varphi(\boldsymbol{F}_{2,n})) = \boldsymbol{P}(\varphi(\boldsymbol{F}_{1,1}), ..., \varphi(\boldsymbol{F}_{1,n})) = \boldsymbol{P}\phi(\boldsymbol{F}_1) \tag{8.48}$$

Therefore the embedded graphs $(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1)$ and $(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2)$ are also strongly isomorphic and are associated by the same permutation $\boldsymbol{P}$ linking $\mathcal{G}_1$ and $\mathcal{G}_2$.

Let us consider any graph template $\overline{\mathcal{G}} = (\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. We will prove now that the FGW cost from $(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1)$ to $\overline{\mathcal{G}}$ applied in $\boldsymbol{T}$ is the same than the FGW cost from $(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2)$ to $\overline{\mathcal{G}}$ applied in $\boldsymbol{P}\boldsymbol{T}$. To this end we will prove that analog relations hold for the Gromov-Wasserstein and the Wasserstein matching costs independently (in this generic scenario), then we will conclude thanks the equation (8.43) which expresses FGW as a linear combination between both aforementioned costs.

First using the reformulation of $\mathcal{E}^{GW}$ of equation (8.46), we have

$$\begin{aligned}
\mathcal{E}^{GW}(\boldsymbol{C}_1, \overline{\boldsymbol{C}}, \boldsymbol{T}) &= \langle \boldsymbol{T}^\top \boldsymbol{C}_1^2 \boldsymbol{T}, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{T}\overline{\boldsymbol{C}}^2 \boldsymbol{T}^\top, \mathbf{1}_{n \times n} \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C}_1 \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{T}^\top \boldsymbol{P}^\top \boldsymbol{C}_2^2 \boldsymbol{P}\boldsymbol{T}, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{T}\overline{\boldsymbol{C}}^2 \boldsymbol{T}^\top, \boldsymbol{P}^\top \mathbf{1}_{n \times n} \boldsymbol{P} \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{P}^\top \boldsymbol{C}_2 \boldsymbol{P}\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \\
&= \langle (\boldsymbol{P}\boldsymbol{T})^\top \boldsymbol{C}_2^2 \boldsymbol{P}\boldsymbol{T}, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{P}\boldsymbol{T}\overline{\boldsymbol{C}}^2 (\boldsymbol{P}\boldsymbol{T})^\top, \mathbf{1}_{n \times n} \rangle - 2\langle (\boldsymbol{P}\boldsymbol{T})^\top \boldsymbol{C}_2 \boldsymbol{P}\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \\
&= \mathcal{E}^{GW}(\boldsymbol{C}_2, \overline{\boldsymbol{C}}, \boldsymbol{P}\boldsymbol{T})
\end{aligned} \tag{8.49}$$

where we used $\boldsymbol{C}_1^2 = (\boldsymbol{P}^\top \boldsymbol{C}_2 \boldsymbol{P})^2 = \boldsymbol{P}^\top \boldsymbol{C}_2^2 \boldsymbol{P}$ and the invariance to permutations of $\mathbf{1}_{n \times n}$.

Then, for $\mathcal{E}^W$ similar operations using equation (8.47) and $\phi(\boldsymbol{F}_2)^2 = (\boldsymbol{P}\phi(\boldsymbol{F}_1))^2 = \boldsymbol{P}\phi(\boldsymbol{F}_1)^2$ lead to,

$$\begin{aligned}
\mathcal{E}^{W}(\phi(\boldsymbol{F}_1), \overline{\boldsymbol{F}}, \boldsymbol{T}) &= \langle \boldsymbol{T}^\top \phi(\boldsymbol{F}_1)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{T}\overline{\boldsymbol{F}}^2, \mathbf{1}_{n \times d} \rangle - 2\langle \phi(\boldsymbol{F}_1)^\top \boldsymbol{T}, \overline{\boldsymbol{F}}^\top \rangle \\
&= \langle \boldsymbol{T}^\top \boldsymbol{P}^\top \phi(\boldsymbol{F}_2)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{T}\overline{\boldsymbol{F}}^2, \boldsymbol{P}\mathbf{1}_{n \times d} \rangle - 2\langle \phi(\boldsymbol{F}_2)^\top \boldsymbol{P}\boldsymbol{T}, \overline{\boldsymbol{F}}^\top \rangle \\
&= \langle (\boldsymbol{P}\boldsymbol{T})^\top \phi(\boldsymbol{F}_2)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{P}\boldsymbol{T}\overline{\boldsymbol{F}}^2, \mathbf{1}_{n \times d} \rangle - 2\langle \phi(\boldsymbol{F}_2)^\top \boldsymbol{P}\boldsymbol{T}, \overline{\boldsymbol{F}}^\top \rangle \\
&= \mathcal{E}^{W}(\phi(\boldsymbol{F}_2), \overline{\boldsymbol{F}}, \boldsymbol{P}\boldsymbol{T})
\end{aligned} \tag{8.50}$$

Therefore, the same result holds for *FGW* using (8.43) and equations (8.49)-(8.50) as

$$\begin{aligned}\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}) &= \alpha\mathcal{E}^{GW}(\boldsymbol{C}_1, \overline{\boldsymbol{C}}, \boldsymbol{T}) + (1-\alpha)\mathcal{E}^{W}(\phi(\boldsymbol{F}_1), \overline{\boldsymbol{F}}, \boldsymbol{T}) \\ &= \alpha\mathcal{E}^{GW}(\boldsymbol{C}_2, \overline{\boldsymbol{C}}, \boldsymbol{P}\boldsymbol{T}) + (1-\alpha)\mathcal{E}^{W}(\phi(\boldsymbol{F}_2), \overline{\boldsymbol{F}}, \boldsymbol{P}\boldsymbol{T}) \\ &= \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{P}\boldsymbol{T})\end{aligned} \tag{8.51}$$

Following an analog derivation than above, one can easily prove for $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}_2, \overline{\boldsymbol{h}})$ that

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}) = \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{P}^\top\boldsymbol{T}) \tag{8.52}$$

Using the relations (8.51) and (8.52), we will now prove the following equality

$$\mathrm{FGW}_\alpha(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) = \mathrm{FGW}_\alpha(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \tag{8.53}$$

First of all, the existence of optimal solutions for both FGW problems is ensured by the Weierstrass theorem (Santambrogio, 2015). We denote an optimal coupling $\boldsymbol{T}_1^\star \in \mathcal{U}(\boldsymbol{h}_1, \overline{\boldsymbol{h}})$ for $\mathrm{FGW}_\alpha(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. Assume there exists an optimal coupling $\boldsymbol{T}_2^\star$ for $\mathrm{FGW}_\alpha(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$ such that

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_2^\star) < \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{P}\boldsymbol{T}_1^\star) \tag{8.54}$$

then using the equalities (8.52) for the l.h.s and (8.51) for the r.h.s, we have

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{P}^\top\boldsymbol{T}_2^\star) < \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_1^\star) \tag{8.55}$$

which contradicts the optimality of $\boldsymbol{T}_1^\star$. Therefore such $\boldsymbol{T}_2^\star$ can not exist and necessarily $\boldsymbol{P}\boldsymbol{T}_1^\star$ is an optimal coupling for $\mathrm{FGW}_\alpha(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. Finally, we can conclude using the optimality of $\boldsymbol{T}_1^\star$ and $\boldsymbol{P}\boldsymbol{T}_1^\star$ for their respective FGW matching problems and the equality (8.51):

$$\begin{aligned}&\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_1^\star) = \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{P}\boldsymbol{T}_1^\star) \\ \Leftrightarrow\ &\mathrm{FGW}_\alpha(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) = \mathrm{FGW}_\alpha(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})\end{aligned} \tag{8.56}$$

□

8.2.3 Proof of Lemma 4: TFGW invariance to weak isomorphism

Lemma 3 (TFGW invariance to weak isomorphism) *The* TFGW *embeddings are invariant to weak isomorphism.*

Proof of Lemma 3. Let us consider any pair of graphs $\mathcal{G}_1 = (\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1)$ and $\mathcal{G}_2 = (\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}_2)$ which are now assumed to be *weakly isomorphic*. According to the notion of weak isomorphism, two nodes of a graph, e.g the nodes x_i and x_j ($x_i \neq x_j$) of $\mathcal{G}_1$, are informally the same if they have the same "internal perception", i.e $C_1(x_i, x_j) = C_1(x_i, x_i) = C_1(x_j, x_j) = C_1(x_j, x_i)$ and $F_1(x_i) = F_1(x_j)$, and the same "external perception", i.e all the incoming and outgoing edge weights are the same. Therefore there exists a *canonical representation* of $\mathcal{G}_1$ and $\mathcal{G}_2$ referred as $\mathcal{G}_c = (\boldsymbol{C}_c, \boldsymbol{F}_c, \boldsymbol{h}_c)$ composed of $C = |\operatorname{supp}(\boldsymbol{h}_c)|$ nodes such that

- $\mathcal{G}_1$ and $\mathcal{G}_2$ are weakly isomorphic to $\mathcal{G}_c$
- $\mathcal{G}_c$ does not admit any pair of nodes which have the same internal and external perceptions referred as canonical nodes. Moreover the masses assigned to these canonical

nodes correspond to the sum of masses of the corresponding nodes which have the same internal and external perceptions in $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively.

Therefore there exist membership matrices $\boldsymbol{M}_1 \in \{0,1\}^{n_1 \times C}$ and $\boldsymbol{M}_2 \in \{0,1\}^{n_2 \times C}$, such that for

$$\forall k \in \{1,2\}, \quad \boldsymbol{C}_k = \boldsymbol{M}_k \boldsymbol{C}_c \boldsymbol{M}_k^\top, \quad \boldsymbol{F}_k = \boldsymbol{M}_k \boldsymbol{F}_c \tag{8.57}$$

but we cannot deduce $\boldsymbol{h}_k$ from $\boldsymbol{h}_c$ as the operation of summing masses of weakly isomorphic nodes of $\mathcal{G}_k$ to form $\boldsymbol{h}_c$ is not bijective. However it is easy to check that $\boldsymbol{T}^*_{kc} = diag(\boldsymbol{h}_k)\boldsymbol{M}_k \in \mathcal{U}(\boldsymbol{h}_k, \boldsymbol{h}_c)$ is an optimal coupling from $\mathcal{G}_k$ to $\mathcal{G}_c$.

One can first observe that the application Φ preserves the relation of weak isomorphism applied over $\boldsymbol{F}_k$ described in (8.57). Indeed as Φ operates on each node independently though ϕ, we have $\Phi(\boldsymbol{F}_k) = M_k \Phi(\boldsymbol{F}_c)$ for $k \in \{1,2\}$ as,

$$\Phi(\boldsymbol{F}_k) = (\phi(\boldsymbol{F}_{k,1}), ..., \phi(\boldsymbol{F}_{k,n_k})) = \boldsymbol{M}_k(\phi(\boldsymbol{F}_{c,1}, ..., \phi(\boldsymbol{F}_{c,C}))) = \boldsymbol{M}_k \Phi(\boldsymbol{F}_c) \tag{8.58}$$

Let us consider any graph template $\overline{\mathcal{G}} = (\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$. We will prove now that the FGW cost from $(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1)$ to $\overline{\mathcal{G}}$ is the same than the ones from $(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2)$ to $\overline{\mathcal{G}}$, i.e.

$$\mathrm{FGW}_\alpha(\boldsymbol{C}_1, \phi(\boldsymbol{F}_1), \boldsymbol{h}_1, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) = \mathrm{FGW}_\alpha(\boldsymbol{C}_2, \phi(\boldsymbol{F}_2), \boldsymbol{h}_2, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \tag{8.59}$$

Let us consider any $k \in \{1,2\}$ and $\boldsymbol{T}_k \in \mathcal{U}(\boldsymbol{h}_k, \overline{\boldsymbol{h}})$. We will now aim at expressing any FGW cost between $(\boldsymbol{C}_k, \Phi(\boldsymbol{F}_k), \boldsymbol{h}_k)$ and $\overline{\mathcal{G}}$, as a FGW cost between $(\boldsymbol{C}_c, \Phi(\boldsymbol{F}_c), \boldsymbol{h}_c)$ and $\overline{\mathcal{G}}$. First using the reformulation of $\mathcal{E}^{GW}$ of equation (8.46), we have

$$\begin{aligned}
\mathcal{E}^{GW}(\boldsymbol{C}_k, \overline{\boldsymbol{C}}, \boldsymbol{T}_k) &= \langle \boldsymbol{T}_k^\top \boldsymbol{C}_k^2 \boldsymbol{T}_k, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{T}_k \overline{\boldsymbol{C}}^2 \boldsymbol{T}_k^\top, \mathbf{1}_{n_k \times n_k} \rangle - 2\langle \boldsymbol{T}_k^\top \boldsymbol{C}_1 \boldsymbol{T}_k, \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{T}_k^\top \boldsymbol{M}_k \boldsymbol{C}_c^2 \boldsymbol{M}_k^\top \boldsymbol{T}_k, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{T}_k \overline{\boldsymbol{C}}^2 \boldsymbol{T}_k^\top, \boldsymbol{M}_k \mathbf{1}_{C \times C} \boldsymbol{M}_k^\top \rangle \\
&\quad - 2\langle \boldsymbol{T}_k^\top \boldsymbol{M}_k \boldsymbol{C}_c \boldsymbol{M}_k^\top \boldsymbol{T}_k, \overline{\boldsymbol{C}} \rangle \\
&= \langle (\boldsymbol{M}_k^\top \boldsymbol{T}_k)^\top \boldsymbol{C}_c^2 \boldsymbol{M}_k^\top \boldsymbol{T}_k, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{M}_k^\top \boldsymbol{T}_k \overline{\boldsymbol{C}}^2 (\boldsymbol{M}_k^\top \boldsymbol{T}_k)^\top, \mathbf{1}_{C \times C} \rangle \\
&\quad - 2\langle (\boldsymbol{M}_k^\top \boldsymbol{T}_k)^\top \boldsymbol{C}_c \boldsymbol{M}_k^\top \boldsymbol{T}_k, \overline{\boldsymbol{C}} \rangle \\
&= \mathcal{E}^{GW}(\boldsymbol{C}_c, \overline{\boldsymbol{C}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k)
\end{aligned} \tag{8.60}$$

Where the second equality comes from the relations, $\boldsymbol{C}_k^2 = (\boldsymbol{M}_k \boldsymbol{C}_c \boldsymbol{M}_k^\top)^2 = \boldsymbol{M}_k \boldsymbol{C}_c^2 \boldsymbol{M}_k^\top$ and $\mathbf{1}_{n_k \times n_k} = \boldsymbol{M}_k \mathbf{1}_{C \times C} \boldsymbol{M}_k^\top$.

Then, for $\mathcal{E}^W$ similar operations using equation (8.47), $\phi(\boldsymbol{F}_k)^2 = (\boldsymbol{M}_k \phi(\boldsymbol{F}_c))^2 = \boldsymbol{M}_k \phi(\boldsymbol{F}_c)^2$ and $\boldsymbol{M}_k^\top \mathbf{1}_{n_k \times d} = \mathbf{1}_{C \times d}$ lead to,

$$\begin{aligned}
\mathcal{E}^{W}(\phi(\boldsymbol{F}_k), \overline{\boldsymbol{F}}, \boldsymbol{T}_k) &= \langle \boldsymbol{T}_k^\top \phi(\boldsymbol{F}_k)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{T}_k \overline{\boldsymbol{F}}^2, \mathbf{1}_{n_k \times d} \rangle - 2\langle \phi(\boldsymbol{F}_k)^\top \boldsymbol{T}_k, \overline{\boldsymbol{F}}^\top \rangle \\
&= \langle \boldsymbol{T}_k^\top \boldsymbol{M}_k \phi(\boldsymbol{F}_c)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{T}_k \overline{\boldsymbol{F}}^2, \boldsymbol{M}_k^\top \mathbf{1}_{n \times d} \rangle - 2\langle \phi(\boldsymbol{F}_c)^\top \boldsymbol{M}_k^\top \boldsymbol{T}_k, \overline{\boldsymbol{F}}^\top \rangle \\
&= \langle (\boldsymbol{M}_k^\top \boldsymbol{T}_k)^\top \phi(\boldsymbol{F}_c)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{M}_k^\top \boldsymbol{T}_k \overline{\boldsymbol{F}}^2, \mathbf{1}_{C \times d} \rangle - 2\langle \phi(\boldsymbol{F}_c)^\top \boldsymbol{M}_k^\top \boldsymbol{T}_k, \overline{\boldsymbol{F}}^\top \rangle \\
&= \mathcal{E}^{W}(\phi(\boldsymbol{F}_c), \overline{\boldsymbol{F}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k)
\end{aligned} \tag{8.61}$$

Therefore, the same result holds for FGW using (8.43) and equations (8.60)-(8.61) as

$$\begin{aligned}
\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_k) &= \alpha \mathcal{E}^{GW}(\boldsymbol{C}_k, \overline{\boldsymbol{C}}, \boldsymbol{T}_k) + (1-\alpha)\mathcal{E}^{W}(\phi(\boldsymbol{F}_k), \overline{\boldsymbol{F}}, \boldsymbol{T}_k) \\
&= \alpha \mathcal{E}^{GW}(\boldsymbol{C}_c, \overline{\boldsymbol{C}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k) + (1-\alpha)\mathcal{E}^{W}(\phi(\boldsymbol{F}_c), \overline{\boldsymbol{F}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k) \\
&= \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k)
\end{aligned} \tag{8.62}$$

Now our goal will be to express any FGW cost between $(\boldsymbol{C}_c, \Phi(\boldsymbol{F}_c), \boldsymbol{h}_c)$ and $\overline{\mathcal{G}}$, as a FGW cost between $(\boldsymbol{C}_k, \Phi(\boldsymbol{F}_k), \boldsymbol{h}_k)$ and $\overline{\mathcal{G}}$. The procedure will be rather equivalent to the one to get the opposite relation stated in equation (8.62), however to proceed we need to express reciprocal relations than the one in (8.57) giving $\boldsymbol{C}_k$ (resp. $\boldsymbol{F}_k$) depending on $\boldsymbol{C}_c$ (resp. $\boldsymbol{F}_c$), i.e we need to express $\boldsymbol{C}_c$ (resp. $\boldsymbol{F}_c$) based on $\boldsymbol{C}_k$ (resp. $\boldsymbol{F}_k$).

We can show that simple developments of the following products of matrices lead to the following relations:

$$\forall k \in \{1,2\}, \quad \boldsymbol{C}_c = \widetilde{\boldsymbol{M}}_k^\top \boldsymbol{C}_k \widetilde{\boldsymbol{M}}_k, \quad \boldsymbol{F}_c = \widetilde{\boldsymbol{M}}_k^\top \boldsymbol{F}_k \tag{8.63}$$

where $\widetilde{\boldsymbol{M}}_k = diag(\boldsymbol{h}_k)\boldsymbol{M}_k diag(\frac{1}{\boldsymbol{h}_c})$ is a reweighed membership matrix.

First observe that for any coupling $\boldsymbol{T}_c \in \mathcal{U}(\boldsymbol{h}_c, \overline{\boldsymbol{h}})$, we have $\widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c \in \mathcal{U}(\boldsymbol{h}_k, \overline{\boldsymbol{h}})$. Moreover simple computations give $\widetilde{\boldsymbol{M}}_k^\top \boldsymbol{C}_k^2 \widetilde{\boldsymbol{M}}_k = \boldsymbol{C}_c^2$ and $\widetilde{\boldsymbol{M}}_k^\top \mathbf{1}_{n_k \times n_k} \widetilde{\boldsymbol{M}}_k = \mathbf{1}_{C \times C}$. The latter relations and the equation (8.63) lead to the following relation between the GW cost of $\boldsymbol{C}_c$ and $\boldsymbol{C}_k$ to $\overline{\boldsymbol{C}}$:

$$\begin{aligned}
\mathcal{E}^{GW}(\boldsymbol{C}_k, \overline{\boldsymbol{C}}, \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c) &= \langle \boldsymbol{T}_c^\top \widetilde{\boldsymbol{M}}_k^\top \boldsymbol{C}_k^2 \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c \overline{\boldsymbol{C}}^2 \boldsymbol{T}_c^\top \widetilde{\boldsymbol{M}}_k^\top, \mathbf{1}_{n_k \times n_k} \rangle \\
&\quad - 2\langle \boldsymbol{T}_c^\top \widetilde{\boldsymbol{M}}_k^\top \boldsymbol{C}_k \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_k, \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{T}_c^\top \boldsymbol{C}_c^2 \boldsymbol{T}_c, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c \overline{\boldsymbol{C}}^2 \boldsymbol{T}_c^\top \widetilde{\boldsymbol{M}}_k^\top, \mathbf{1}_{n_k \times n_k} \rangle - 2\langle \boldsymbol{T}_c^\top \boldsymbol{C}_c \boldsymbol{T}_k, \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{T}_c^\top \boldsymbol{C}_c^2 \boldsymbol{T}_c, \mathbf{1}_{\overline{n} \times \overline{n}} \rangle + \langle \boldsymbol{T}_c \overline{\boldsymbol{C}}^2, \mathbf{1}_{C \times C} \rangle - 2\langle \boldsymbol{T}_c^\top \boldsymbol{C}_c \boldsymbol{T}_k, \overline{\boldsymbol{C}} \rangle \\
&= \mathcal{E}^{GW}(\boldsymbol{C}_c, \overline{\boldsymbol{C}}, \boldsymbol{T}_c)
\end{aligned} \tag{8.64}$$

Then, for $\mathcal{E}^W$ similar operations using equation (8.47), equation (8.63), $\Phi(\boldsymbol{F}_c)^2 = \widetilde{\boldsymbol{M}}_k^\top \Phi(\boldsymbol{F}_k)^2$ and $\widetilde{\boldsymbol{M}}_k^\top \mathbf{1}_{n_k \times d} = \mathbf{1}_{C \times d}$ lead to,

$$\begin{aligned}
\mathcal{E}^{W}(\phi(\boldsymbol{F}_k), \overline{\boldsymbol{F}}, \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c) &= \langle \boldsymbol{T}_c^\top \widetilde{\boldsymbol{M}}_k^\top \phi(\boldsymbol{F}_k)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c \overline{\boldsymbol{F}}^2, \mathbf{1}_{n_k \times d} \rangle - 2\langle \phi(\boldsymbol{F}_k)^\top \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c, \overline{\boldsymbol{F}}^\top \rangle \\
&= \langle \boldsymbol{T}_c^\top \phi(\boldsymbol{F}_c)^2, \mathbf{1}_{\overline{n} \times d} \rangle + \langle \boldsymbol{T}_c \overline{\boldsymbol{F}}^2, \mathbf{1}_{C \times d} \rangle - 2\langle \phi(\boldsymbol{F}_c)^\top \boldsymbol{T}_c, \overline{\boldsymbol{F}}^\top \rangle \\
&= \mathcal{E}^{W}(\phi(\boldsymbol{F}_c), \overline{\boldsymbol{F}}, \boldsymbol{T}_c)
\end{aligned} \tag{8.65}$$

Therefore, the same result holds for the FGW cost using (8.43) and equations (8.64) and (8.65),

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_c) = \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c) \tag{8.66}$$

Therefore denoting $\boldsymbol{T}_k^*$ an optimal coupling from $(\boldsymbol{C}_k, \Phi(\boldsymbol{F}_k), \boldsymbol{h}_k)$ to $\overline{\mathcal{G}}$ and $\boldsymbol{T}_c^*$ an optimal coupling from $(\boldsymbol{C}_c, \Phi(\boldsymbol{F}_c), \boldsymbol{h}_c)$ to $\overline{\mathcal{G}}$, we have from equations (8.62) and (8.66),

$$\begin{aligned}
\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_k^*) &= \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k^*) \\
\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_c^*) &= \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c^*)
\end{aligned} \tag{8.67}$$

Thus, as $\boldsymbol{M}_k^\top \boldsymbol{T}_k^*$ is a suboptimal admissible coupling in $\mathcal{U}(\boldsymbol{h}_c, \overline{\boldsymbol{h}})$, we have

$$\begin{aligned}
\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_c^*) &\leq \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{M}_k^\top \boldsymbol{T}_k^*) \\
&= \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_k^*)
\end{aligned} \tag{8.68}$$

and similarly as $\widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c^*$ is a suboptimal admissible coupling in $\mathcal{U}(\boldsymbol{h}_k, \overline{\boldsymbol{h}})$, we have

$$\begin{aligned} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_k^*) &\leq \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \widetilde{\boldsymbol{M}}_k \boldsymbol{T}_c^*) \\ &= \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_c^*) \end{aligned} \tag{8.69}$$

Therefore for any $k \in \{1, 2\}$,

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_k, \phi(\boldsymbol{F}_k), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_k^*) = \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}_c, \phi(\boldsymbol{F}_c), \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}_c^*) \tag{8.70}$$

which implies the desired result i.e

$$\mathrm{FGW}_\alpha(\boldsymbol{C}_1, \Phi(\boldsymbol{F}_1), \boldsymbol{h}_1, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) = \mathrm{FGW}_\alpha(\boldsymbol{C}_2, \Phi(\boldsymbol{F}_2), \boldsymbol{h}_2, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}}) \tag{8.71}$$

□

8.2.4 Complements on graphs classification benchmark

Description of datasets. We report in Table 8.1 some statistics on the datasets used for various graphs classification or clustering benchmark.

TABLE 8.1: Statistics on real datasets considered in some classification and clustering benchmarks.

datasets	features	#graphs	#classes	mean #nodes	min #nodes	max #nodes	median #nodes
MUTAG	{0..6}	188	2	17.93	10	28	17.5
PTC-MR	{0, .., 17}	344	2	14.29	2	64	13
NCI1	{0, ..., 36}	4110	2	29.87	3	111	27
BZR	$\mathbb{R}^3$	405	2	35.75	13	57	35
COX2	$\mathbb{R}^3$	467	2	41.23	32	56	41
ENZYMES	$\mathbb{R}^{18}$	600	6	32.63	2	126	32
PROTEIN	$\mathbb{R}^{29}$	1113	2	29.06	4	620	26
IMDB-B	None	1000	2	19.77	12	136	17
IMDB-M	None	1500	3	13.00	7	89	10
COLLAB	None	5000	3	74.5	32	492	52

Learning with an alpha per template.

TABLE 8.2: Classification results from 10-fold cross-validation of our TFGW models in various scenarios: for $L \in \{0, 1, 2\}$ GIN layers, we either force the trade-off parameter α to be shared by all templates (*shared*, default for the method), or we learn an α for each template (*indep*). We learn the templates weights $\overline{\boldsymbol{h}}_k$ by default. The first and second best performing method are respectively highlighted in bold and underlined.

model	inputs	α	MUTAG	PTC	PROTEIN
TFGW (L=0)	ADJ	shared	94.2(3.0)	64.9(4.1)	78.8(2.2)
	ADJ	indep	94.1(3.4)	65.2(4.3)	79.1(2.6)
	SP	shared	<u>95.9</u>(<u>4.1</u>)	67.9(5.8)	79.5(2.9)
	SP	indep	94.7(2.9)	67.3(5.2)	79.9(3.2)
TFGW(L=1)	ADJ	shared	94.8(3.1)	68.7(5.8)	81.5(2.8)
	ADJ	indep	95.4(3.5)	69.5(5.6)	80.9(2.4)
	SP	shared	95.4(3.5)	<u>70.9</u>(<u>5.5</u>)	<u>82.1</u>(<u>3.4</u>)
	SP	indep	<u>95.9</u>(<u>4.1</u>)	70.2(5.6)	81.5(2.8)
TFGW (L=2)	ADJ	shared	**96.4(3.3)**	**72.4(5.7)**	**82.9(2.7)**
	ADJ	indep	94.2(3.8)	69.0(5.2)	80.5(3.1)
	SP	shared	94.8(3.5)	70.8(6.3)	82.0(3.0)
	SP	indep	94.1(3.4)	69.8(5.5)	80.3(2.5)

8.3 Proofs and additional results of Chapter 5

8.3.1 Proof of Lemma 5: convexity of the unmixing problem w.r.t the embeddings

Lemma 4 (FGW unmixing sub-problem w.r.t embeddings) *For any input graph* $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$*, any dictionary* $\{(\overline{\boldsymbol{C}}_k, \overline{\boldsymbol{F}}_k, \overline{\boldsymbol{h}})\}_{k \in [\![K]\!]}$ *and any admissible coupling* $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$*, the unmixing sub-problem*

$$\min_{\boldsymbol{w} \in \Sigma_K} \mathcal{E}_\alpha^{FGW}\left(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \overline{\boldsymbol{h}}\right) \tag{8.72}$$

is a convex *problem which is equivalent to the following canonical quadratic program*

$$\min_{\boldsymbol{w} \in \Sigma_K} \boldsymbol{w}^\top \boldsymbol{Q}_\alpha \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{c} \tag{8.73}$$

where $\boldsymbol{Q}_\alpha = 2(\alpha \boldsymbol{M}^{GW} + (1-\alpha)\boldsymbol{M}^W)$ *is PSD as a convex combination of the PSD matrices*

$$\boldsymbol{M}^{GW} = \left(\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \overline{\boldsymbol{C}}_p, \overline{\boldsymbol{C}}_q \boldsymbol{D}_{\overline{\boldsymbol{h}}} \rangle_F\right)_{pq} \qquad \text{and} \qquad \boldsymbol{M}^W = \left(\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{F}}_p, \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{F}}_q \rangle_F\right)_{pq \in [S]} \tag{8.74}$$

denoting $\boldsymbol{D}_{\overline{\boldsymbol{h}}} = diag(\overline{\boldsymbol{h}})$.
Finally, $\boldsymbol{c} = (c_k)_{k \in [\![K]\!]}$ *satisfies* $c_k = -2\alpha \langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}}_k \rangle - 2(1-\alpha)\langle \boldsymbol{T}^\top \boldsymbol{F}, \overline{\boldsymbol{F}}_k \rangle$.

Proof of Lemma 4. We start by developing the objective function

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T}) = \alpha \mathcal{E}^{GW}(\boldsymbol{C}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \boldsymbol{T}) + (1-\alpha)\mathcal{E}^W(\boldsymbol{F}, \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T}) \tag{8.75}$$

for any embedding $\boldsymbol{w} \in \Sigma_K$ and any admissible transport matrix $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$. We do so by first handling the GW cost $\mathcal{E}^{GW}$ and then the Wasserstein cost $\mathcal{E}^{W}$:

$$\begin{aligned}\mathcal{E}^{GW}(\boldsymbol{C}, \sum_s w_s \overline{\boldsymbol{C}}_s, \boldsymbol{T}) &= \sum_{ij} C_{ij}^2 h_i h_j + \sum_{kl} (\sum_s w_s \overline{C}_{s,kl})^2 \overline{h}_k \overline{h}_l - 2 \sum_{ijkl} C_{ij} (\sum_s w_s \overline{C}_{s,kl}) T_{ik} T_{jl} \\ &= \sum_{ij} C_{ij}^2 h_i h_j + \sum_{kl} \sum_{pq} w_p w_q \overline{C}_{p,kl} \overline{C}_{q,kl} \overline{h}_k \overline{h}_l - 2 \sum_{ijkl,s} w_s C_{ij} \overline{C}_{s,kl} T_{ik} T_{jl}\end{aligned} \tag{8.76}$$

Therefore the first derivates w.r.t $\boldsymbol{w}$ of the GW cost in the unmixing problem satisfies

$$\begin{aligned}\frac{\partial \mathcal{E}^{GW}}{\partial w_i} &= 2 \sum_{kl,p} w_p \overline{C}_{i,kl} \overline{C}_{p,kl} \overline{h}_k \overline{h}_l - 2 \sum_{ijkl} C_{ij} \overline{C}_{i,kl} T_{ik} T_{jl} \\ &= 2\{\langle \overline{\boldsymbol{C}}_i \odot \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \overline{\boldsymbol{h}\boldsymbol{h}}^\top \rangle - \langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}}_i \rangle\}\end{aligned} \tag{8.77}$$

Then the second derivatives read as

$$\frac{\partial^2 \mathcal{E}^{GW}}{\partial w_j \partial w_i} = 2 \langle \overline{\boldsymbol{C}}_i \odot \overline{\boldsymbol{C}}_j, \overline{\boldsymbol{h}\boldsymbol{h}}^\top \rangle \tag{8.78}$$

Using $\operatorname{Tr}\left((\boldsymbol{C}_1 \odot \boldsymbol{C}_2)^\top \boldsymbol{x}\boldsymbol{x}^\top\right) = \operatorname{Tr}\left((\boldsymbol{C}_1 \odot \boldsymbol{C}_2)\boldsymbol{x}\boldsymbol{x}^\top\right) = \operatorname{Tr}\left(\boldsymbol{C}_1^\top \operatorname{diag}(\boldsymbol{x}) \boldsymbol{C}_2 \operatorname{diag}(\boldsymbol{x})\right)$, we have

$$\frac{\partial^2 \mathcal{E}^{GW}}{\partial w_j \partial w_i} = 2 \operatorname{Tr}\{\boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{C}}_i^\top \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{C}}_j \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2}\} \tag{8.79}$$

Then using the relation $\operatorname{Tr}\{\boldsymbol{A}^\top \boldsymbol{B}\} = \operatorname{vec}(\boldsymbol{A})^\top \operatorname{vec}(\boldsymbol{B})$ (see *e.g* Petersen et al. (2008), equation 521), we have the following factorization $M_{ij}^{GW} = 2\operatorname{vec}(\boldsymbol{B}_i)^\top \operatorname{vec}(\boldsymbol{B}_j)$, where $\forall k \in [K], \boldsymbol{B}_k = \boldsymbol{D}_{\boldsymbol{h}}^{1/2} \overline{\boldsymbol{C}}_k \boldsymbol{D}_{\boldsymbol{h}}^{1/2}$ and vec is the vectorization operator of a matrix which converts the matrix into a column vector by stacking the columns of the matrix A on top of one another.
Hence with $\boldsymbol{B} = (\boldsymbol{B}_k)_{k \in [\![K]\!]} \subset \mathbb{R}^{N^2 \times K}$, the Hessian associated to the GW cost, denoted here $\boldsymbol{M}^{GW}$, admits for any $\boldsymbol{w} \in \Sigma_K$, the following PSD factorization $\boldsymbol{M}^{GW}(\boldsymbol{w}) = 2\boldsymbol{B}^T \boldsymbol{B}$. Therefore $\boldsymbol{M}^{GW}$ is a PSD matrix.

$$\begin{aligned}\mathcal{E}^{W}(\boldsymbol{F}, \sum_s w_s \overline{\boldsymbol{F}}_s, \overline{\boldsymbol{h}}) &= \sum_i \|\boldsymbol{F}_i\|_2^2 h_i + \sum_j \| \sum_s w_s \overline{\boldsymbol{F}}_s \|_2^2 \overline{h}_j - 2 \sum_{ij} \langle \boldsymbol{F}_i, \sum_s w_s \overline{\boldsymbol{F}}_{s,j} \rangle T_{ij} \\ &= \sum_i \|\boldsymbol{F}_i\|_2^2 h_i + \sum_{pq} w_p w_q \sum_j \langle \overline{\boldsymbol{F}}_{p,j}, \overline{\boldsymbol{F}}_{q,j} \rangle \overline{h}_j - 2 \sum_s w_s \sum_{ij} \langle \boldsymbol{F}_i, \overline{\boldsymbol{F}}_{s,j} \rangle T_{ij}\end{aligned} \tag{8.80}$$

Therefore the first derivates w.r.t $\boldsymbol{w}$ of the Wasserstein cost involved in the FGW unmixing reads for any $q \in [\![K]\!]$ as

$$\begin{aligned}\frac{\partial \mathcal{E}^{W}}{\partial w_q} &= 2 \sum_p w_p \sum_j \langle \overline{\boldsymbol{F}}_{q,j}, \overline{\boldsymbol{F}}_{p,j} \rangle \overline{h}_j - 2 \sum_{ij} \langle \boldsymbol{F}_i, \overline{\boldsymbol{F}}_{q,j} \rangle T_{ij} \\ &= 2 \langle \boldsymbol{D}_{\overline{\boldsymbol{h}}} \widetilde{\boldsymbol{F}}(\boldsymbol{w}) - \boldsymbol{T}^\top \boldsymbol{F}, \overline{\boldsymbol{F}}_q \rangle\end{aligned} \tag{8.81}$$

Then the second derivates read as

$$\frac{\partial^2 \mathcal{E}^W}{\partial w_p \partial w_q} = 2\sum_j \langle \overline{\boldsymbol{F}}_{q,j}, \overline{\boldsymbol{F}}_{p,j} \rangle \overline{h}_j = 2\langle \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{F}}_p, \boldsymbol{D}_{\overline{\boldsymbol{h}}}^{1/2} \overline{\boldsymbol{F}}_q \rangle \tag{8.82}$$

Therefore the Hessian associated to the Wasserstein cost reads as $\boldsymbol{M}^W(\boldsymbol{w}) = 2\left(\langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2} \overline{\boldsymbol{F}}_p, \boldsymbol{D}_{\boldsymbol{h}}^{1/2} \overline{\boldsymbol{F}}_q \rangle_F\right)_{p,q \in [\![K]\!]}$. $\boldsymbol{M}^W$ is also a PSD matrix as it can be factorized as $\boldsymbol{M}^W = 2\boldsymbol{B}^\top \boldsymbol{B}$ with $\boldsymbol{B} = \left(\mathrm{vec}(\boldsymbol{D}_{\boldsymbol{h}}^{1/2} \overline{\boldsymbol{F}}_k)\right)_{k \in [\![K]\!]} \in \mathbb{R}^{Nd \times K}$.

Therefore the Hessian $\boldsymbol{M}$ associated to the FGW cost reads as

$$\boldsymbol{M}(\boldsymbol{w}) = \alpha \boldsymbol{M}^{GW}(\boldsymbol{w}) + (1-\alpha)\boldsymbol{M}^W(\boldsymbol{w}) \tag{8.83}$$

is also a PSD matrix as the convex combination of the PSD matrices $\boldsymbol{M}^{GW}$ and $\boldsymbol{M}^W$. Thus we can conclude that $\boldsymbol{w} \to \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}), \boldsymbol{T})$ is a convex function.
□

8.3.2 Line-search of the Conditional Gradient solver for the unmixing subproblem

For completeness, we detail here the factorization used in the line-search sub-problem of Algorithm 7. As explicited in (5.10), the exact line-search comes down to solve for:

$$\gamma^\star = \underset{\gamma \in (0,1)}{\arg\min}\, \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) \tag{8.84}$$

where $\boldsymbol{z}(\gamma) = \boldsymbol{w} + \gamma(\boldsymbol{x}^\star - \boldsymbol{w}) = \boldsymbol{w} + \gamma \Delta \boldsymbol{w}$. This problem can be easily solved by observing that the FGW loss $\mathcal{E}_\alpha^{FGW}$ can be factored as a second-order polynomial function in γ, $f(\gamma) = a\gamma^2 + b\gamma + c$.

Indeed, following equations (8.46) and (8.47) we have

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \widetilde{\boldsymbol{F}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) = \alpha \mathcal{E}^{GW}(\boldsymbol{C}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) + (1-\alpha)\mathcal{E}^W(\boldsymbol{F}, \widetilde{\boldsymbol{F}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) \tag{8.85}$$

where

$$\begin{aligned}
&\mathcal{E}^{GW}(\boldsymbol{C}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma))^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C}\boldsymbol{T}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)) \rangle \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle - 2\sum_p (w_p + \gamma \Delta w_p) \langle \boldsymbol{T}^\top \boldsymbol{C}\boldsymbol{T}, \overline{\boldsymbol{C}}_p \rangle + \sum_{pq} (w_p + \gamma \Delta w_p)(w_q + \gamma \Delta w_q) \langle \overline{\boldsymbol{C}}_p \odot \overline{\boldsymbol{C}}_q, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C}\boldsymbol{T}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}) \rangle - 2\gamma \langle \boldsymbol{T}^\top \boldsymbol{C}\boldsymbol{T}, \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w}) \rangle \\
&+ \sum_{pq} \left(w_p w_q + \gamma(w_p \Delta w_q + \Delta w_p w_q) + \gamma^2 \Delta w_p \Delta w_q \right) \langle \overline{\boldsymbol{C}}_p \odot \overline{\boldsymbol{C}}_q, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C}\boldsymbol{T}, \widetilde{\boldsymbol{C}}(\boldsymbol{w}) \rangle + \langle \widetilde{\boldsymbol{C}}(\boldsymbol{w})^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle \\
&+ 2\gamma \{ \langle \widetilde{\boldsymbol{C}}(\boldsymbol{w}) \odot \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w}), \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - \langle \boldsymbol{T}^\top \boldsymbol{C}\boldsymbol{T}, \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w}) \rangle \} \\
&+ \gamma^2 \langle \widetilde{\boldsymbol{C}}(\Delta \boldsymbol{w})^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle
\end{aligned} \tag{8.86}$$

where power operations are applied element-wise. So one can define this GW cost as

$$\mathcal{E}^{GW}(\boldsymbol{C}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) := a^{GW}\gamma^2 + b^{GW}\gamma + c^{GW} \tag{8.87}$$

where coefficients are given in equation (8.86).

Then through similar operations $\mathcal{E}^W$ can be expressed as

$$\begin{aligned}
&\mathcal{E}^W(\boldsymbol{F}, \widetilde{\boldsymbol{F}}(\boldsymbol{z}_\gamma), , \boldsymbol{T}) \\
&= \sum_i \|\boldsymbol{f}_i\|_2^2 h_i + \sum_j \|\widetilde{\boldsymbol{f}}_j(\boldsymbol{z}(\gamma))\|_2^2 \overline{h}_j - 2\sum_{ij} \langle \boldsymbol{f}_i, \widetilde{\boldsymbol{f}}_j(\boldsymbol{z}(\gamma))\rangle T_{ij} \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h}\rangle + \langle \left(\sum_p (w_p + \gamma \Delta w_p)\overline{\boldsymbol{F}}_p\right)^2 \mathbf{1}_d, \overline{\boldsymbol{h}}\rangle - 2\langle \boldsymbol{F}\left(\sum_p (w_p + \gamma\Delta w_p)\overline{\boldsymbol{F}}_p\right)^\top, \boldsymbol{T}\rangle \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h}\rangle + \sum_{pq} (w_p + \gamma\Delta w_p)(w_q + \gamma \Delta w_q)\langle \overline{\boldsymbol{F}}_p \odot \overline{\boldsymbol{F}}_q \mathbf{1}_d, \overline{\boldsymbol{h}}\rangle - 2\sum_p (w_p + \gamma\Delta w_p)\langle \boldsymbol{F}\overline{\boldsymbol{F}}_p^\top, \boldsymbol{T}\rangle \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h}\rangle + \langle \widetilde{\boldsymbol{F}}(\boldsymbol{w})^2 \mathbf{1}_d, \overline{\boldsymbol{h}}\rangle - 2\langle \boldsymbol{F}\widetilde{\boldsymbol{F}}(\boldsymbol{w})^\top, \boldsymbol{T}\rangle \\
&+ 2\gamma \left\{ \langle \left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}) \odot \widetilde{\boldsymbol{F}}(\Delta \boldsymbol{w})\right)\mathbf{1}_d, \overline{\boldsymbol{h}}\rangle - \langle \boldsymbol{F}\widetilde{\boldsymbol{F}}(\Delta \boldsymbol{w})^\top, \boldsymbol{T}\rangle \right\} \\
&+ \gamma^2 \langle \widetilde{\boldsymbol{F}}(\Delta \boldsymbol{w})^2 \mathbf{1}_d, \overline{\boldsymbol{h}}\rangle
\end{aligned} \tag{8.88}$$

So one can define this Wasserstein cost as

$$\mathcal{E}^W(\boldsymbol{F}, \widetilde{\boldsymbol{F}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) := a^W \gamma^2 + b^W \gamma + c^W \tag{8.89}$$

where coefficients are given in equation (8.88).

Therefore, using equations (8.87) and (8.89), the FGW cost given in equation (8.85) can be written as

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \widetilde{\boldsymbol{C}}(\boldsymbol{z}(\gamma)), \widetilde{\boldsymbol{F}}(\boldsymbol{z}(\gamma)), \boldsymbol{T}) := a\gamma^2 + b\gamma + c \tag{8.90}$$

with for all $x \in \{a, b, c\}, x = \alpha x^{GW} + (1-\alpha)x^W$.

8.3.3 Proof of Theorem 9: FGW upper-bound in the embedding

Theorem 8 (FGW upper-bound in the embedding) *For two embedded graphs with node features, with embeddings $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ over the set of pairwise relation matrices $\{(\overline{\boldsymbol{C}}_s, \overline{\boldsymbol{F}}_s)\}_{s\in[S]} \subset \mathbb{R}^{N\times N} \times \mathbb{R}^{N\times d}$, and a shared masses vector $\boldsymbol{h}$, the following inequality holds $\forall \alpha \in [0,1]$,*

$$\mathrm{FGW}_{2,\alpha}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)})\right) \leq \|\boldsymbol{w}^{(1)} - \boldsymbol{w}^{(2)}\|_{\alpha \boldsymbol{M}^{GW} + (1-\alpha)\boldsymbol{M}^W} \tag{8.91}$$

with,

$$\widetilde{\boldsymbol{C}}(\boldsymbol{w}) = \sum_s w_s \overline{\boldsymbol{C}}_s \quad \text{and} \quad \widetilde{\boldsymbol{F}}(\boldsymbol{w}) = \sum_s w_s \overline{\boldsymbol{F}}_s \tag{8.92}$$

Where $\boldsymbol{M}^{GW} = \left(\langle \boldsymbol{D}_{\boldsymbol{h}}\overline{\boldsymbol{C}}_p, \overline{\boldsymbol{C}}_q \boldsymbol{D}_{\boldsymbol{h}}\rangle_F\right)_{pq}$ and $\boldsymbol{M}^W = (\langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_p, \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_q\rangle_F)_{pq\in[S]}$, and $\boldsymbol{D}_{\boldsymbol{h}} = diag(\boldsymbol{h})$, are PSD matrices and therefore their linear combinations being PSD engender Mahalanobis distances between embeddings.

Didactic proof of Theorem 8. First, let us recall the decomposition of the FGW cost applied in any $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$ between any pair of graphs $(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \overline{\boldsymbol{h}})$,

$$\mathcal{E}_\alpha^{FGW}\left(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{T}\right) = \alpha \mathcal{E}^{GW}\left(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}\right) + (1-\alpha)\mathcal{E}^W\left(\boldsymbol{F}, \overline{\boldsymbol{F}}, \boldsymbol{T}\right) \tag{8.93}$$

for any $\alpha \in [0,1]$, where the GW cost $\mathcal{E}^{GW}$ and the Wasserstein cost $\mathcal{E}^{W}$ are respectively given in equations (8.44) and (8.45).
To bound by above the FGW distance between two embedded graphs $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}), \boldsymbol{h})$ and $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}), \boldsymbol{h})$ with embeddings $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ in Σ_S, we will consider both costs $\mathcal{E}^{GW}$ and $\mathcal{E}^{W}$ independently.

Bounding the Gromov-Wasserstein distance.

Let us recall the formulation of the GW cost $\mathcal{E}^{GW}$ previously given in equation (8.44), as a Frobenius inner product (see *e.g* Peyré et al. (2016)). Let any pair of graphs $(\boldsymbol{C}, \boldsymbol{h})$ and $(\overline{\boldsymbol{C}}, \overline{\boldsymbol{h}})$ and an admissible coupling $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}})$, we have

$$\begin{aligned}\mathcal{E}^{GW}\left(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{T}\right) &= \sum_{ij} C_{ij}^2 h_i h_j + \sum_{kl} \overline{C}_{kl}^2 \overline{h}_k \overline{h}_l - 2\sum_{ijkl} C_{ij}\overline{C}_{kl} T_{ik} T_{jl} \\ &= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{C}\boldsymbol{T}\overline{\boldsymbol{C}}\boldsymbol{T}^\top, \boldsymbol{T} \rangle\end{aligned} \tag{8.94}$$

Then denoting any admissible coupling $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \boldsymbol{h})$, the GW cost between these embedded graph structures applied in $\boldsymbol{T}$ reads as

$$\begin{aligned}\mathcal{E}^{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{T}\right) &= \langle (\sum_p w_p^{(1)} \overline{\boldsymbol{C}}_p)^2 + (\sum_p w_p^{(2)} \overline{\boldsymbol{C}}_p)^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \\ &\quad - 2\langle (\sum_p w_p^{(1)} \overline{\boldsymbol{C}}_p) \boldsymbol{T} (\sum_q w_p^{(2)} \overline{\boldsymbol{C}}_q)^\top, \boldsymbol{T} \rangle \\ &= \sum_{pq} \{w_p^{(1)} w_q^{(1)} + w_p^{(2)} w_q^{(2)}\} \langle \overline{\boldsymbol{C}}_p \odot \overline{\boldsymbol{C}}_q, \boldsymbol{h}\boldsymbol{h}^\top \rangle \\ &\quad - 2\sum_{pq} w_p^{(1)} w_q^{(2)} \langle \overline{\boldsymbol{C}}_p \boldsymbol{T} \overline{\boldsymbol{C}}_q^\top, \boldsymbol{T} \rangle\end{aligned} \tag{8.95}$$

With the following property of the trace operator:

$$\mathrm{Tr}\left((\boldsymbol{C}_1 \odot \boldsymbol{C}_2)\boldsymbol{x}\boldsymbol{x}^\top\right) = \mathrm{Tr}\left(\boldsymbol{C}_1^\top \,\mathrm{diag}(\boldsymbol{x}) \boldsymbol{C}_2 \,\mathrm{diag}(\boldsymbol{x})\right) \tag{8.96}$$

For conciseness let us introduce the diagonal operator $\boldsymbol{x} \in \Sigma_K \to \boldsymbol{D}_{\boldsymbol{x}} = \mathrm{diag}(\boldsymbol{x})$. Then using equation 8.96 we have the following relations,

$$\langle \overline{\boldsymbol{C}}_p \odot \overline{\boldsymbol{C}}_q, \boldsymbol{h}\boldsymbol{h}^\top \rangle = \mathrm{Tr}\{(\overline{\boldsymbol{C}}_p \odot \overline{\boldsymbol{C}}_q)^\top \boldsymbol{h}\boldsymbol{h}^\top\} = \mathrm{Tr}\{(\overline{\boldsymbol{C}}_p^\top \odot \overline{\boldsymbol{C}}_q^\top)\boldsymbol{h}\boldsymbol{h}^\top\} = \mathrm{Tr}\{\overline{\boldsymbol{C}}_p^\top \boldsymbol{D}_{\boldsymbol{h}} \overline{\boldsymbol{C}}_q \boldsymbol{D}_{\boldsymbol{h}}\} \tag{8.97}$$

So we can rewrite the expression of the GW cost given in equation 8.95 as

$$\begin{aligned}\mathcal{E}^{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{T}\right) &= \sum_{pq} \{w_p^{(1)} w_q^{(1)} + w_p^{(2)} w_q^{(2)}\} \,\mathrm{Tr}\{\overline{\boldsymbol{C}}_p^\top \boldsymbol{D}_{\boldsymbol{h}} \overline{\boldsymbol{C}}_q \boldsymbol{D}_{\boldsymbol{h}}\} \\ &\quad - 2\sum_{pq} w_p^{(1)} w_q^{(2)} \,\mathrm{Tr}\{\overline{\boldsymbol{C}}_q \boldsymbol{T}^\top \overline{\boldsymbol{C}}_p^\top \boldsymbol{T}\} \\ &= \sum_{pq} \{w_p^{(1)} w_q^{(1)} + w_p^{(2)} w_q^{(2)}\} \,\mathrm{Tr}\{\overline{\boldsymbol{C}}_p^\top \boldsymbol{D}_{\boldsymbol{h}} \overline{\boldsymbol{C}}_q \boldsymbol{D}_{\boldsymbol{h}}\} \\ &\quad - 2\sum_{pq} w_p^{(1)} w_q^{(2)} \,\mathrm{Tr}\{\overline{\boldsymbol{C}}_p^\top \boldsymbol{T} \overline{\boldsymbol{C}}_q \boldsymbol{T}^\top\}\end{aligned} \tag{8.98}$$

As the equation 8.98 is true for any admissible coupling $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \boldsymbol{h})$, it is also true for the optimal coupling $\boldsymbol{T}^*$ which *minimizes* $\boldsymbol{T} \to \mathcal{E}^{GW}(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{T})$ and provides the GW cost between both embedded graphs $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \boldsymbol{h})$ and $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{h})$, *if we omit for now their*

node features. Moreover as $\boldsymbol{D_h}$ is an admissible and sub-optimal coupling it satisfies,

$$\mathrm{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{h}, \boldsymbol{h}\right) \leq \mathcal{E}^{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{D_h}\right) \tag{8.99}$$

Observed now that using the equation (8.98), we can expressive the GW cost applied in $\boldsymbol{D_h}$ as

$$\begin{aligned}\mathcal{E}^{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}), \widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}), \boldsymbol{D_h}\right) &= \sum_{pq}\{w_p^{(1)}w_q^{(1)} + w_p^{(2)}w_q^{(2)} - 2w_p^{(1)}w_q^{(2)}\}\,\mathrm{Tr}\{\overline{\boldsymbol{C}}_p^{\top}\boldsymbol{D_h}\overline{\boldsymbol{C}}_q\boldsymbol{D_h}\} \\ &= \sum_{pq}\{w_p^{(1)}w_q^{(1)} + w_p^{(2)}w_q^{(2)} - 2w_p^{(1)}w_q^{(2)}\}M_{pq}^{GW} \\ &= \boldsymbol{w}^{(1)^T}\boldsymbol{M}^{GW}\boldsymbol{w}^{(1)} + \boldsymbol{w}^{(2)^\top}\boldsymbol{M}^{GW}\boldsymbol{w}^{(2)} - 2\boldsymbol{w}^{(1)^\top}\boldsymbol{M}^{GW}\boldsymbol{w}^{(2)}\end{aligned} \tag{8.100}$$

where $\boldsymbol{M}^{GW} = (M_{pq}^{GW})_{p,q\in[\![K]\!]}$ satisfies $M_{pq}^{GW} = \mathrm{Tr}\{\overline{\boldsymbol{C}}_p^{\top}\boldsymbol{D_h}\overline{\boldsymbol{C}}_q\boldsymbol{D_h}\} = \langle \boldsymbol{D_h}\overline{\boldsymbol{C}}_p, \overline{\boldsymbol{C}}_q\boldsymbol{D_h}\rangle$.
It suffices to prove that the matrix $\boldsymbol{M}^{GW}$ is a PSD matrix to conclude that it defines a Mahalanobis distance over the set of embeddings $\boldsymbol{w}$ which bounds by above the GW distance between corresponding embedded graphs.
Let us reformulate any entry (p,q) of $\boldsymbol{M}^{GW}$ as

$$\begin{aligned}M_{pq}^{GW} &= \langle \boldsymbol{D_h}\overline{\boldsymbol{C}_p}, \overline{\boldsymbol{C}_q}\boldsymbol{D_h}\rangle = \mathrm{Tr}\{\overline{\boldsymbol{C}}_p^{\top}\boldsymbol{D_h}\overline{\boldsymbol{C}}_q\boldsymbol{D_h}\} = \mathrm{Tr}\{\boldsymbol{D_h}^{1/2}\overline{\boldsymbol{C}}_p^{\top}\boldsymbol{D_h}^{1/2}\boldsymbol{D_h}^{1/2}\overline{\boldsymbol{C}}_q\boldsymbol{D_h}^{1/2}\} \\ &= \mathrm{Tr}\{\left(\boldsymbol{D_h}^{1/2}\overline{\boldsymbol{C}}_p\boldsymbol{D_h}^{1/2}\right)^{\top}\boldsymbol{D_h}^{1/2}\overline{\boldsymbol{C}}_q\boldsymbol{D_h}^{1/2}\}\end{aligned} \tag{8.101}$$

Then using the relation $\mathrm{Tr}\{\boldsymbol{A}^{\top}\boldsymbol{B}\} = \mathrm{vec}(\boldsymbol{A})^{\top}\mathrm{vec}(\boldsymbol{B})$ (see *e.g* Petersen et al. (2008), equation 521), we have the following factorization $M_{pq}^{GW} = \mathrm{vec}(\boldsymbol{B}_p)^{\top}\mathrm{vec}(\boldsymbol{B}_q)$, where $\forall n \in [S], \boldsymbol{B}_n = \boldsymbol{D_h}^{1/2}\overline{\boldsymbol{C}}_n\boldsymbol{D_h}^{1/2}$ and vec is the vectorization operator of a matrix which converts the matrix into a column vector by stacking the columns of the matrix A on top of one another. Hence with $\boldsymbol{B} = (\boldsymbol{B}_n)_n \subset \mathbb{R}^{N^2\times S}$, $\boldsymbol{M}^{GW}$ admits the PSD factorization $\boldsymbol{B}^T\boldsymbol{B}$ and therefore is a PSD matrix.

Bounding the Wasserstein distance.

We will now focus on the wasserstein cost $\mathcal{E}^W$ involved in FGW and show that omitting the graph structures, we can bound the Wasserstein distance between embedded node features $(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}), \boldsymbol{h})$ and $(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}), \boldsymbol{h})$ which can be seen as point clouds. Let us first highlight a suitable factorization of the Wasserstein cost between any pair of point clouds $(\boldsymbol{F}, \boldsymbol{h})$ and $(\overline{\boldsymbol{F}}, \boldsymbol{h})$. For any admissible coupling $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, \boldsymbol{h})$ we have

$$\begin{aligned}\mathcal{E}^W(\boldsymbol{F}, \overline{\boldsymbol{F}}, \boldsymbol{T}) &= \sum_{ij}\|\boldsymbol{F}_i - \overline{\boldsymbol{F}}_j\|_2^2 T_{ij} \\ &= \sum_i \|\boldsymbol{F}_i\|_2^2 h_i + \sum_j \|\overline{\boldsymbol{F}}_j\|_2^2 h_j - 2\sum_{ij}\langle \boldsymbol{F}_i, \overline{\boldsymbol{F}}_j\rangle T_{ij} \\ &= \langle \boldsymbol{D_h}^{1/2}\boldsymbol{F}, \boldsymbol{D_h}^{1/2}\boldsymbol{F}\rangle_F + \langle \boldsymbol{D_h}^{1/2}\overline{\boldsymbol{F}}, \boldsymbol{D_h}^{1/2}\overline{\boldsymbol{F}}\rangle_F - 2\langle \boldsymbol{F}\overline{\boldsymbol{F}}^{\top}, \boldsymbol{T}\rangle_F\end{aligned} \tag{8.102}$$

Therefore using equation (8.102), we can rewrite the Wasserstein cost between any two embedded feature matrices as,

$$\begin{aligned}\mathcal{E}^W\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{T}\right) &= \langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\boldsymbol{D}_{\boldsymbol{h}}^{1/2}\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)})\rangle + \langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{D}_{\boldsymbol{h}}^{1/2}\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)})\rangle \\ &\quad - 2\langle \widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)})\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)})^\top,\boldsymbol{T}\rangle \\ &= \sum_{pq}\{w_p^{(1)}w_q^{(1)} + w_p^{(2)}w_q^{(2)}\}\langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_p,\boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_q\rangle \\ &\quad - 2\sum_{pq} w_p^{(1)}w_q^{(2)}\langle \overline{\boldsymbol{F}}_p\overline{\boldsymbol{F}}_q^\top,\boldsymbol{T}\rangle_F\end{aligned} \tag{8.103}$$

As the equation 8.103 is true for any admissible coupling $\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\boldsymbol{h})$, it is also true for the optimal coupling $\boldsymbol{T}^*$ which *minimizes* $\boldsymbol{T}\to\mathcal{E}^W\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{T}\right)$ and provides the Wasserstein cost between both embedded point clouds $(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\boldsymbol{h})$ and $(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{h})$, *if we omit for now their graph structure*. Moreover as $\boldsymbol{D}_{\boldsymbol{h}}$ is an admissible and sub-optimal coupling it satisfies,

$$\mathrm{W}\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{h},\boldsymbol{h}\right) \leq \mathcal{E}^W\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{D}_{\boldsymbol{h}}\right) \tag{8.104}$$

Observed now that using the equation (8.103), we can expressive $\mathcal{E}^W$ applied in $\boldsymbol{D}_{\boldsymbol{h}}$ as

$$\begin{aligned}\mathcal{E}^W\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{D}_{\boldsymbol{h}}\right) &= \sum_{pq}\{w_p^{(1)}w_q^{(1)} + w_p^{(2)}w_q^{(2)} - 2w_p^{(1)}w_q^{(2)}\}\langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_p,\boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_q\rangle \\ &= \boldsymbol{w}^{(1)^T}\boldsymbol{M}^W\boldsymbol{w}^{(1)} + \boldsymbol{w}^{(2)^\top}\boldsymbol{M}^W\boldsymbol{w}^{(2)} - 2\boldsymbol{w}^{(1)^\top}\boldsymbol{M}^W\boldsymbol{w}^{(2)}\end{aligned} \tag{8.105}$$

where $\boldsymbol{M}^W = \left(\langle \boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_p,\boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_q\rangle_F\right)_{pq\in[\![S]\!]}$ which is also a PSD matrix as it can be factorized as $\boldsymbol{B}^\top\boldsymbol{B}$ with $\boldsymbol{B} = \left(\mathrm{vec}(\boldsymbol{D}_{\boldsymbol{h}}^{1/2}\overline{\boldsymbol{F}}_s)\right)_{s\in[S]} \in \mathbb{R}^{Nd\times S}$, hence defines a Mahalonobis distance between embeddings.

Bounding the Fused Gromov-Wasserstein distance.

As the equations (8.98) and (8.103) hold for any admissible coupling $\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\boldsymbol{h})$, it is also true for the optimal coupling $\boldsymbol{T}^*$ which *minimizes* $\boldsymbol{T}\to\alpha\mathcal{E}^{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}),\boldsymbol{T}\right) + (1-\alpha)\mathcal{E}^W\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{T}\right)$ and provides the Fused Gromov-Wasserstein cost between both embedded graphs $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\boldsymbol{h})$ and $(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{h})$.

Moreover as $\boldsymbol{D}_{\boldsymbol{h}}$ is an admissible and sub-optimal coupling it satisfies,

$$\begin{aligned}\mathrm{FGW}_{2,\alpha}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)})\right) &\leq \alpha\mathcal{E}^{GW}\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}),\boldsymbol{D}_{\boldsymbol{h}}\right) \\ &\quad + (1-\alpha)\mathcal{E}^W\left(\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{F}}(\boldsymbol{w}^{(2)}),\boldsymbol{D}_{\boldsymbol{h}}\right)\end{aligned} \tag{8.106}$$

Let us denote $\forall\alpha\in(0,1)$, $\boldsymbol{M}_\alpha = \alpha\boldsymbol{M}^{GW} + (1-\alpha)\boldsymbol{M}^W$ where M^{GW} and M^W are respectively defined in equations (8.100) and (8.105). M_α defines a PSD matrix as a convex combination of PSD matrices M^{GW} and M^W, hence engender a Mahalanobis distance in the embedding

space. Moreover following equations (8.100) and (8.105), we have for all $\alpha \in [0,1]$,

$$\begin{aligned}\mathrm{FGW}_{2,\alpha}^2\left(\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{A}}(\boldsymbol{w}^{(1)}),\widetilde{\boldsymbol{C}}(\boldsymbol{w}^{(2)}),\widetilde{\boldsymbol{A}}(\boldsymbol{w}^{(2)})\right) &\leq {\boldsymbol{w}^{(1)}}^\top \boldsymbol{M}_\alpha \boldsymbol{w}^{(1)} + {\boldsymbol{w}^{(2)}}^\top \boldsymbol{M}_\alpha \boldsymbol{w}^{(2)} - 2{\boldsymbol{w}^{(1)}}^\top \boldsymbol{M}_\alpha \boldsymbol{w}^{(2)} \\ &= \|\boldsymbol{w}^{(1)} - \boldsymbol{w}^{(2)}\|_{\boldsymbol{M}_\alpha}\end{aligned} \tag{8.107}$$

which concludes the proof of the theorem 8.
□

8.3.4 Clustering benchmark : additional results

TABLE 8.3: GDL configurations leading to best clustering performances using spectral clustering on embeddings for benchmarked real-world datasets. K refers to the number of atoms and α to the trade-off parameter inherent to the FGW distance (set to - if the GW distance is used). If several configurations lead to the same performances, the one with the less atoms is reported.

	GDL		GDL_λ		
datasets	K	α	K	α	λ
IMDB-B	16	-	16	-	0.01
IMDB-M	12	-	16	-	{0.01, 0.001}
MUTAG	4	0.9997	4	0.9997	0.01
PTC	4	0.9997	4	0.9997	0.01
BZR (norm)	4	{0.5, ..., 0.9997}	4	{0.5, ..., 0.9997}	{0.01, 0.001}
COX2 (norm)	8	0.9997	8	0.9997	0.01

TABLE 8.4: FGW configurations leading to best clustering performances using spectral clustering on input graphs with node features.

datasets	α
MUTAG	$\{0.995, 0.9997\}$
PTC	$\{0.9\}$
BZR (norm)	0.1
BZR (raw)	0.9997
COX (norm)	0.9997
COX (raw)	0.995
ENZYMES (norm)	0.995
ENZYMES (raw)	0.995

8.4 Proofs and additional results of Chapter 6

8.4.1 Proof of Proposition 2: srFGW equivalent problems

Proposition 2 (srFGW equivalent problem) *Problem (6.1) is equivalent to the following optimization problem:*

$$\text{srFGW}^2_{2,\alpha}(\boldsymbol{C},\boldsymbol{F},\boldsymbol{h},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}}) = \min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \sum_{ijkl} \left\{\alpha|C_{ij}-\overline{C}_{kl}|^2 + (1-\alpha)\|\boldsymbol{F}_i-\overline{\boldsymbol{F}}_k\|_2^2\right\} T_{ik}T_{jl} \tag{8.108}$$

where $\mathcal{U}_{\overline{n}}(\boldsymbol{h})$ denotes the set of admissible coupling with first marginal $\boldsymbol{h}$ and relaxed second marginal:

$$\mathcal{U}_{\overline{n}}(\boldsymbol{h}) = \left\{\boldsymbol{T}\in\mathbb{R}_+^{n\times\overline{n}} | \boldsymbol{T}\boldsymbol{1}_{\overline{n}} = \boldsymbol{h}\right\} \tag{8.109}$$

Proof of Proposition 2. Consider a solution of problem (6.1) denoted $(\overline{\boldsymbol{h}}_1^\star, \boldsymbol{T}_1)$ and note that the definition implies that $\boldsymbol{T}_1 \in \mathcal{U}(\boldsymbol{h}, \overline{\boldsymbol{h}}_1^\star)$. Another observation is that given $\overline{\boldsymbol{h}}_1^\star$, the transport plan $\boldsymbol{T}_1$ also belongs to $\arg\min_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}}_1^\star)} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T})$ hence is an optimal solution of $\text{FGW}_{2,\alpha}(\boldsymbol{C},\boldsymbol{F},\boldsymbol{h},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\overline{\boldsymbol{h}}_1^\star)$.

Now consider a solution of problem (8.108) denoted $\boldsymbol{T}_2$ with second marginal $\overline{\boldsymbol{h}}_2^\star$. By definition the couple $(\overline{\boldsymbol{h}}_2^\star, \boldsymbol{T}_2)$ is suboptimal for problem (6.1) i.e

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}_1) \leq \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}_2) \tag{8.110}$$

And the symmetric also holds as $\boldsymbol{T}_1$ is a suboptimal admissible coupling for problem 8.108 i.e ,

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}_2) \leq \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}_1) \tag{8.111}$$

These inequalities imply that $\mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}_1) = \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}_2)$. Therefore we necessarily have

$$\boldsymbol{T}_1 \in \arg\min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}) \tag{8.112}$$

and

$$(\overline{\boldsymbol{h}}_2^\star, \boldsymbol{T}_2) \in \arg\min_{\overline{\boldsymbol{h}}\in\Sigma_{\overline{n}}, \boldsymbol{T}\in\mathcal{U}(\boldsymbol{h}_1,\overline{\boldsymbol{h}})} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}) \tag{8.113}$$

Hence the equality, $\text{FGW}_{2,\alpha}(\boldsymbol{C},\boldsymbol{F},\boldsymbol{h},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\overline{\boldsymbol{h}}_1^\star) = \text{FGW}_{2,\alpha}(\boldsymbol{C},\boldsymbol{F},\boldsymbol{h},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\overline{\boldsymbol{h}}_2^\star)$, holds true. Therefore by double inclusion we have

$$\arg\min_{\boldsymbol{T}\in\mathcal{U}_{\overline{n}}(\boldsymbol{h})} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}) = \arg\min_{\overline{\boldsymbol{h}}\in\Sigma_m, \boldsymbol{T}\in\mathcal{U}(\boldsymbol{h},\overline{\boldsymbol{h}})} \mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{T}). \tag{8.114}$$

Which is enough to prove that both problems are equivalent. □

8.4.2 Proofs of Lemma 6: sr(F)GW properties

Lemma 5 (srFGW properties) *For any attributed graphs $\mathcal{G}_1 = (\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1)$ and $\mathcal{G}_2 = (\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}_2)$, the following propositions hold true for any $\alpha \in [0,1]$*

i) $\text{srFGW}^2_{2,\alpha}(\boldsymbol{C}_1,\boldsymbol{F}_1,\boldsymbol{h}_1,\boldsymbol{C}_2,\boldsymbol{F}_2) = 0$ *iff there exists a reweighed sub-graph of $\mathcal{G}_2$ which is weakly isomorphic to $\mathcal{G}_1$ (Definition 4).*

ii) If $\boldsymbol{C}_1$ *and* $\boldsymbol{C}_2$ *are distance matrices then* $\text{srFGW}^2_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2) = 0$ *iff there exists a reweighed sub-graph of* $\mathcal{G}_2$ *which is strongly isomorphic to* $\mathcal{G}_1$ *(Definition 3)*

Then considering any third attributed graph $\mathcal{G}_3 = (\boldsymbol{C}_3, \boldsymbol{F}_3, \boldsymbol{h}_3)$*, and* $\alpha \in]0, 1[$*,*

iii) For any optimal reweighing $\boldsymbol{h}^\star_{(1,2)}$ *from* $\text{srFGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2)$*, we have*

$$\begin{aligned}\text{srFGW}^2_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_3, \boldsymbol{F}_3) \leq 2(&\,\text{srFGW}^2_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2) \\ &+ \text{srFGW}^2_{2,\alpha}(\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}^\star_{(1,2)}, \boldsymbol{C}_3, \boldsymbol{F}_3))\end{aligned} \tag{8.115}$$

iv) For any optimal reweighing $\boldsymbol{h}^\star_{(1,2)}$ *from* $\text{srGW}(\boldsymbol{C}_1, \boldsymbol{h}_1, \boldsymbol{C}_2)$*, we have*

$$\text{srGW}^2_2(\boldsymbol{C}_1, \boldsymbol{h}_1, \boldsymbol{C}_3) \leq 2\left\{\text{srGW}^2_2(\boldsymbol{C}_1, \boldsymbol{h}_1, \boldsymbol{C}_2) + \text{srGW}^2_2(\boldsymbol{C}_2, \boldsymbol{h}^\star_{(1,2)}, \boldsymbol{C}_3)\right\} \tag{8.116}$$

Proof of Lemma 5. Let us denote any attributed graphs $\mathcal{G}_1 = (\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1)$ and $\mathcal{G}_2 = (\boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{h}_2)$, and $\alpha \in [0, 1]$. We prove in the following each assertion of Lemma 5.

proof of assertions i)-ii) The reasoning involved in this proof mostly relates on the definition of srFGW as $\min_{\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}} \text{FGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2, \overline{\boldsymbol{h}})$.

($\Rightarrow$) Assume that $\text{srFGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2) = 0$. Then we have $\text{FGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2, \overline{\boldsymbol{h}}^*) = 0$, for some $\overline{\boldsymbol{h}}^* \in \Sigma_{\overline{n}}$. By virtue of the Fused Gromov-Wasserstein properties [ref theo], $(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1)$ and $(\boldsymbol{C}_2, \boldsymbol{F}_2, \overline{\boldsymbol{h}}^*)$ are *weakly* isomorphic. Moreover, if $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are distance matrices, then both graphs are *strongly* isomorphic.

($\Leftarrow$) Assume there exists reweighing $\overline{\boldsymbol{h}} \in \Sigma_{\overline{n}}$ of $\mathcal{G}_2$, such that $(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1)$ and $(\boldsymbol{C}_2, \boldsymbol{F}_2, \overline{\boldsymbol{h}})$ are weakly (resp. strongly) isomorphic. Then $\text{FGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2, \overline{\boldsymbol{h}}) = 0$, so there exists $\boldsymbol{T}^* \in \mathcal{U}(\boldsymbol{h}_1, \overline{\boldsymbol{h}})$ such that $\mathcal{E}^{FGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{C}_2, \boldsymbol{F}_2, \boldsymbol{T}^*) = 0$. Moreover as $T^* \in \mathcal{U}_{\overline{n}}(\boldsymbol{h}_1)$, since $\mathcal{U}(\boldsymbol{h}_1, \overline{\boldsymbol{h}}) \subset \mathcal{U}_{\overline{n}}(\boldsymbol{h}_1)$, and the same FGW cost is involved in both transport problems, we have

$$0 \leq \text{srFGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2) \leq \text{FGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2, \overline{\boldsymbol{h}}) = 0 \tag{8.117}$$

which implies $\text{srFGW}_{2,\alpha}(\boldsymbol{C}_1, \boldsymbol{F}_1, \boldsymbol{h}_1, \boldsymbol{C}_2, \boldsymbol{F}_2) = 0$.

We now prove the assertion iv) before iii) as the proof follows analog steps with simpler computations.

proof of assertion iv) To emphasize the action of srGW as a projection from a graph $\mathcal{G}_i$ to another $\mathcal{G}_j$, we denote here by $\overline{\boldsymbol{h}}^{(i|j)}$ an optimal reweighing of the nodes of $\mathcal{G}_j$ in the GW sense. Moreover, as the following proof requires manipulation over indices of the matrices composing a graph, the i^{th} graph $(\boldsymbol{C}_i, \boldsymbol{F}_i, \boldsymbol{h}_i)$ is denoted here $(\boldsymbol{C}^{(i)}, \boldsymbol{F}^{(i)}, \boldsymbol{h}^{(i)})$.

We consider now the following solutions to the corresponding srGW problems:

- $(\boldsymbol{T}^{(1|2)}, \overline{\boldsymbol{h}}^{(1|2)})$ as optimal solutions of $\text{srGW}(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(2)})$
- $(\boldsymbol{T}^{(1|3)}, \overline{\boldsymbol{h}}^{(1|3)})$ as optimal solutions of $\text{srGW}(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(3)})$
- $(\boldsymbol{T}^{(2|3)}, \overline{\boldsymbol{h}}^{(2|3)})$ as optimal solutions of $\text{srGW}(\boldsymbol{C}^{(2)}, \boldsymbol{h}^{(2)}, \boldsymbol{C}^{(3)})$

We introduce a padded version $\overline{\boldsymbol{h}} \in \Sigma_{n^{(2)}}$ of $\overline{\boldsymbol{h}}^{(1|2)}$ satisfying for all $i \in [\![n^{(2)}]\!]$,

$$\overline{h}_i = \begin{cases} \overline{h}_i^{(1|2)} & \text{if } \overline{h}_i^{(1|2)} \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{8.118}$$

Note that the constant 1 is arbitrary and could be set to any other scalar value $c \neq 0$.

Let us consider now the matrix $\boldsymbol{S} \in \mathbb{R}^{n^{(1)} \times n^{(3)}}$ defined as

$$\boldsymbol{S} = \boldsymbol{T}^{(1|2)} \text{diag}(1/\overline{\boldsymbol{h}}) \boldsymbol{T}^{(2|3)} = \left(\sum_k \frac{T_{ik}^{(1|2)} T_{kj}^{(2|3)}}{\overline{h}_k} \right) \tag{8.119}$$

Let us check the marginals of $\boldsymbol{S}$, starting for its right marginal

$$\boldsymbol{S}\mathbf{1}_{n^{(3)}} = \boldsymbol{T}^{(1|2)} \text{diag}\left(\overline{\boldsymbol{h}}^{-1}\right) \overline{\boldsymbol{h}}^{(1|2)} = \boldsymbol{T}^{(1|2)} \left(\mathbb{1}_{\overline{h}_j^{(1|2)} \neq 0} \right)_{j \in [\![n^{(2)}]\!]} = \boldsymbol{h}^{(1)} \tag{8.120}$$

These equalities come from the respective right marginals of $\boldsymbol{T}^{(1|2)}$ and $\boldsymbol{T}^{(2|3)}$, and that $\forall j \in [\![n^{(2)}]\!], \overline{h}_j^{(1|2)} = 0 \implies \forall i \in [\![n^{(1)}]\!], T_{ij}^{(1|2)} = 0$. Then using analog arguments for its left marginal:

$$\boldsymbol{S}^\top \mathbf{1}_{n^{(1)}} = \boldsymbol{T}^{(2|3)T} \text{diag}(\overline{\boldsymbol{h}}^{-1}) \boldsymbol{T}^{(1|2)T} \mathbf{1}_{n^{(1)}} = \boldsymbol{T}^{(2|3)T} \text{diag}(\overline{\boldsymbol{h}}^{-1}) \overline{\boldsymbol{h}}^{(1|2)} = \overline{\boldsymbol{h}}^{(2|3)} \tag{8.121}$$

So $\boldsymbol{S} \in \mathcal{U}_{n^{(3)}}(\boldsymbol{h}_1)$ is a *sub-optimal admissible coupling* to the srGW projection of $(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)})$ onto $\boldsymbol{C}^{(3)}$. So we have:

$$\begin{aligned} \text{srGW}_2(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(3)}) &= \mathcal{E}^{GW}(\boldsymbol{C}_1, \boldsymbol{C}_3, \boldsymbol{T}^{(1|3)}) \\ &\leq \mathcal{E}^{GW}(\boldsymbol{C}_1, \boldsymbol{C}_3, \boldsymbol{S}) \\ &= \sum_{ijkl} (C_{ij}^{(1)} - C_{kl}^{(3)})^2 S_{ik} S_{jl} \\ &= \sum_{ijkl} (C_{ij}^{(1)} - C_{kl}^{(3)})^2 \left(\sum_p \frac{T_{ip}^{(1|2)} T_{pk}^{(2|3)}}{\overline{h}_p} \right) \left(\sum_q \frac{T_{jq}^{(1|2)} T_{ql}^{(2|3)}}{\overline{h}_q} \right) \\ &= \sum_{ijkl,pq} (C_{ij}^{(1)} - C_{pq}^{(2)} + C_{pq}^{(2)} - C_{kl}^{(3)})^2 \left(\frac{T_{ip}^{(1|2)} T_{pk}^{(2|3)}}{\overline{h}_p} \right) \left(\frac{T_{jq}^{(1|2)} T_{ql}^{(2|3)}}{\overline{h}_q} \right) \end{aligned} \tag{8.122}$$

Then by using the convexity of $x \to x^2$ or equivalently by using Holder inequality, we have

$$\begin{aligned}
\mathrm{srGW}_2(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(3)}) &\leq 2 \sum_{ijkl,pq} (C^{(1)}_{ij} - C^{(2)}_{pq})^2 \left(\frac{T^{(1|2)}_{ip} T^{(2|3)}_{pk}}{\overline{h}_p} \right) \left(\frac{T^{(1|2)}_{jq} T^{(2|3)}_{ql}}{\overline{h}_q} \right) \\
&+ 2 \sum_{ijkl,pq} (C^{(2)}_{pq} - C^{(3)}_{kl})^2 \left(\frac{T^{(1|2)}_{ip} T^{(2|3)}_{pk}}{\overline{h}_p} \right) \left(\frac{T^{(1|2)}_{jq} T^{(2|3)}_{ql}}{\overline{h}_q} \right) \\
&= 2 \sum_{ijpq} (C^{(1)}_{ij} - C^{(2)}_{pq})^2 \left(\sum_k \frac{T^{(1|2)}_{ip} T^{(2|3)}_{pk}}{\overline{h}_p} \right) \left(\sum_l \frac{T^{(1|2)}_{jq} T^{(2|3)}_{ql}}{\overline{h}_q} \right) \\
&+ 2 \sum_{klpq} (C^{(2)}_{pq} - C^{(3)}_{kl})^2 \left(\sum_i \frac{T^{(1|2)}_{ip} T^{(2|3)}_{pk}}{\overline{h}_p} \right) \left(\sum_j \frac{T^{(1|2)}_{jq} T^{(2|3)}_{ql}}{\overline{h}_q} \right) \\
&= 2 \sum_{ijpq} (C^{(1)}_{ij} - C^{(2)}_{pq})^2 T^{(1|2)}_{ip} T^{(1|2)}_{jq} + 2 \sum_{klpq} (C^{(2)}_{pq} - C^{(3)}_{kl})^2 T^{(2|3)}_{pk} T^{(2|3)}_{ql} \\
&= 2\mathcal{E}^{GW}(\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}, \boldsymbol{T}^{(1|2)}) + 2\mathcal{E}^{GW}(\boldsymbol{C}^{(2)}, \boldsymbol{C}^{(3)}, \boldsymbol{T}^{(2|3)})
\end{aligned} \tag{8.123}$$

So we can conclude that

$$\mathrm{srGW}(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(3)}) \leq 2 \left\{ \mathrm{srGW}(\boldsymbol{C}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(2)}) + \mathrm{srGW}(\boldsymbol{C}^{(2)}, \overline{\boldsymbol{h}}^{(1|2)}, \boldsymbol{C}^{(3)}) \right\} \tag{8.124}$$

proof of assertion iv) For any $\alpha \in]0, 1[$, we consider now the following solutions to the corresponding srFGW problems:

- $(\boldsymbol{T}^{(1|2)}, \overline{\boldsymbol{h}}^{(1|2)})$ as optimal solutions of $\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(1)}, \boldsymbol{F}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(2)}, \boldsymbol{F}^{(2)})$
- $(\boldsymbol{T}^{(1|3)}, \overline{\boldsymbol{h}}^{(1|3)})$ as optimal solutions of $\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(1)}, \boldsymbol{F}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(3)}, \boldsymbol{F}^{(3)})$
- $(\boldsymbol{T}^{(2|3)}, \overline{\boldsymbol{h}}^{(2|3)})$ as optimal solutions of $\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(2)}, \boldsymbol{F}^{(2)}, \boldsymbol{h}^{(2)}, \boldsymbol{C}^{(3)}, \boldsymbol{F}^{(3)})$

In the same way as for the proof of Assertion iv), we introduce a padded version $\overline{\boldsymbol{h}} \in \Sigma_{n^{(2)}}$ of $\overline{\boldsymbol{h}}^{(1|2)}$ defined in equation (8.118), and the matrix $\boldsymbol{S} = \boldsymbol{T}^{(1|2)} \mathrm{diag}(1/\overline{\boldsymbol{h}}) \boldsymbol{T}^{(2|3)} \in \mathcal{U}_{n^{(3)}}(\boldsymbol{h}_1)$ (see equation (8.119)).

So $\boldsymbol{S} \in \mathcal{U}_{n^{(3)}}(\boldsymbol{h}_1)$ is a *sub-optimal admissible coupling* to the srFGW projection of $(\boldsymbol{C}^{(1)}, \boldsymbol{F}^{(1)}, \boldsymbol{h}^{(1)})$ onto $(\boldsymbol{C}^{(3)}, \boldsymbol{F}^{(3)})$. So we have:

$$\begin{aligned}
&\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(1)}, \boldsymbol{F}^{(1)}, \boldsymbol{h}^{(1)}, \boldsymbol{C}^{(3)}, \boldsymbol{F}^{(3)}) \\
&= \mathcal{E}^{FGW}_{\alpha}(\boldsymbol{C}^{(1)}, \boldsymbol{F}^{(1)}, \boldsymbol{C}^{(3)}, \boldsymbol{F}^{(3)}, \boldsymbol{T}^{(1|3)}) \\
&\leq \mathcal{E}^{FGW}_{\alpha}(\boldsymbol{C}^{(1)}, \boldsymbol{F}^{(1)}, \boldsymbol{C}^{(3)}, \boldsymbol{F}^{(3)}, \boldsymbol{S}) \\
&= \alpha \mathcal{E}^{GW}(\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(3)}, \boldsymbol{S}) + (1 - \alpha) \mathcal{E}^{W}(\boldsymbol{F}^{(1)}, \boldsymbol{F}^{(3)}, \boldsymbol{S})
\end{aligned} \tag{8.125}$$

Using exactly the same inequalities than in equation (8.123) of the proof of assertion iv), one has

$$\mathcal{E}^{GW}(\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(3)}, \boldsymbol{S}) \leq 2 \left\{ \mathcal{E}^{GW}(\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}, \boldsymbol{T}^{(1|2)}) + \mathcal{E}^{GW}(\boldsymbol{C}^{(2)}, \boldsymbol{C}^{(3)}, \boldsymbol{T}^{(2|3)}) \right\} \tag{8.126}$$

Next to handle the Wasserstein cost over feature matrices, we have by definition

$$\begin{aligned}\mathcal{E}^W(\boldsymbol{F}^{(1)},\boldsymbol{F}^{(3)},\boldsymbol{S}) &= \sum_{ij}\|\boldsymbol{F}_i^{(1)}-\boldsymbol{F}_j^{(3)}\|_2^2 S_{ij}\\ &= \sum_{ij}\|\boldsymbol{F}_i^{(1)}-\boldsymbol{F}_j^{(3)}\|_2^2\left(\sum_k \frac{T_{ik}^{(1|2)}T_{kj}^{(2|3)}}{\overline{h}_k}\right)\\ &= \sum_{ijk}\|\boldsymbol{F}_i^{(1)}-\boldsymbol{F}_k^{(2)}+\boldsymbol{F}_k^{(2)}-\boldsymbol{F}_j^{(3)}\|_2^2\frac{T_{ik}^{(1|2)}T_{kj}^{(2|3)}}{\overline{h}_k}\end{aligned} \tag{8.127}$$

Then by applying the triangle inequality with the Euclidean norm of $\mathbb{R}^d$, and using the convexity of $x \to x^2$, we have

$$\begin{aligned}\mathcal{E}^W(\boldsymbol{F}^{(1)},\boldsymbol{F}^{(3)},\boldsymbol{S}) &\leq 2\sum_{ijk}\|\boldsymbol{F}_i^{(1)}-\boldsymbol{F}_k^{(2)}\|_2^2\frac{T_{ik}^{(1|2)}T_{kj}^{(2|3)}}{\overline{h}_k} + 2\sum_{ijk}\|\boldsymbol{F}_k^{(2)}-\boldsymbol{F}_j^{(3)}\|_2^2\frac{T_{ik}^{(1|2)}T_{kj}^{(2|3)}}{\overline{h}_k}\\ &= 2\sum_{ik}\|\boldsymbol{F}_i^{(1)}-\boldsymbol{F}_k^{(2)}\|_2^2 T_{ik}^{(1|2)}2 + \sum_{jk}\|\boldsymbol{F}_k^{(2)}-\boldsymbol{F}_j^{(3)}\|_2^2 T_{kj}^{(2|3)}\\ &= 2\mathcal{E}^W(\boldsymbol{F}^{(1)},\boldsymbol{F}^{(2)},\boldsymbol{T}^{(1|2)}) + 2\mathcal{E}^W(\boldsymbol{F}^{(2)},\boldsymbol{F}^{(3)},\boldsymbol{T}^{(2|3)})\end{aligned} \tag{8.128}$$

Therefore bounding each term in the rhs of (8.125), using the bounds on the GW cost and the Wasserstein cost, given respectively in equations (8.126) and (8.128), we have

$$\begin{aligned}&\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(1)},\boldsymbol{F}^{(1)},\boldsymbol{h}^{(1)},\boldsymbol{C}^{(3)},\boldsymbol{F}^{(3)})\\ &\leq 2\alpha\left\{\mathcal{E}^{GW}(\boldsymbol{C}^{(1)},\boldsymbol{C}^{(2)},\boldsymbol{T}^{(1|2)}) + \mathcal{E}^{GW}(\boldsymbol{C}^{(2)},\boldsymbol{C}^{(3)},\boldsymbol{T}^{(2|3)})\right\}\\ &+ 2(1-\alpha)\left\{\mathcal{E}^{W}(\boldsymbol{F}^{(1)},\boldsymbol{F}^{(2)},\boldsymbol{T}^{(1|2)}) + \mathcal{E}^{W}(\boldsymbol{F}^{(2)},\boldsymbol{F}^{(3)},\boldsymbol{T}^{(2|3)})\right\}\\ &= 2\left\{\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}^{(1)},\boldsymbol{F}^{(1)},\boldsymbol{C}^{(2)},\boldsymbol{F}^{(2)},\boldsymbol{T}^{(1|2)}) + \mathcal{E}_\alpha^{FGW}(\boldsymbol{C}^{(2)},\boldsymbol{F}^{(2)},\boldsymbol{C}^{(3)},\boldsymbol{F}^{(3)},\boldsymbol{T}^{(2|3)})\right\}\\ &= 2\left\{\mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(1)},\boldsymbol{F}^{(1)},\boldsymbol{h}^{(1)},\boldsymbol{C}^{(2)},\boldsymbol{F}^{(2)}) + \mathrm{srFGW}_{2,\alpha}(\boldsymbol{C}^{(2)},\boldsymbol{F}^{(2)},\overline{\boldsymbol{h}}^{(1|2)},\boldsymbol{C}^{(3)},\boldsymbol{F}^{(3)})\right\}\end{aligned} \tag{8.129}$$

□

8.4.3 Line-search of the Conditional Gradient solver for srFGW

For completeness, we detail here the factorization used in the line-search step of Algorithm 12. The exact line-search comes down to find:

$$\gamma^\star = \underset{\gamma\in(0,1)}{\arg\min}\,\mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{Z}(\gamma)) \tag{8.130}$$

where $\boldsymbol{Z}(\gamma) = \boldsymbol{T} + \gamma(\boldsymbol{X}^\star - \boldsymbol{T}) = \boldsymbol{T} + \gamma\Delta\boldsymbol{T}$. This problem can be easily solved by observing that the FGW loss $\mathcal{E}_\alpha^{FGW}$ can be factored as a second-order polynomial function in γ, $f(\gamma) = a\gamma^2 + b\gamma + c$.

Indeed, following equations (8.46) and (8.47) we have

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C},\boldsymbol{F},\overline{\boldsymbol{C}},\overline{\boldsymbol{F}},\boldsymbol{Z}(\gamma)) = \alpha\mathcal{E}^{GW}(\boldsymbol{C},\overline{\boldsymbol{C}},\boldsymbol{Z}(\gamma)) + (1-\alpha)\mathcal{E}^{W}(\boldsymbol{F},\overline{\boldsymbol{F}},\boldsymbol{Z}(\gamma)) \tag{8.131}$$

Then, denoting $\overline{\boldsymbol{h}}(\gamma) = \boldsymbol{Z}(\gamma)^\top \mathbf{1}_n = (\boldsymbol{T}+\gamma\Delta\boldsymbol{T})^\top \mathbf{1}_n = \overline{\boldsymbol{h}}+\gamma\Delta\overline{\boldsymbol{h}}$, the GW cost can be expressed as

$$\begin{aligned}
&\mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{Z}(\gamma)) \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, \boldsymbol{Z}(\gamma)^\top \mathbf{1}_n \mathbf{1}_n^\top \boldsymbol{Z}(\gamma) \rangle - 2\langle \boldsymbol{Z}(\gamma)^\top \boldsymbol{C} \boldsymbol{Z}(\gamma), \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, (\overline{\boldsymbol{h}} + \gamma\Delta\overline{\boldsymbol{h}})(\overline{\boldsymbol{h}} + \gamma\Delta\overline{\boldsymbol{h}})^\top \rangle - 2\langle (\boldsymbol{T} + \gamma\Delta\boldsymbol{T})^\top \boldsymbol{C} (\boldsymbol{T} + \gamma\Delta\boldsymbol{T}), \overline{\boldsymbol{C}} \rangle \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle + \gamma \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\Delta\overline{\boldsymbol{h}}^\top + \Delta\overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle + \gamma^2 \langle \overline{\boldsymbol{C}}^2, \Delta\overline{\boldsymbol{h}}\Delta\overline{\boldsymbol{h}}^\top \rangle \\
&- 2\left\{ \langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle + \gamma \langle \boldsymbol{T}^\top \boldsymbol{C} \Delta\boldsymbol{T} + \Delta\boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle + \gamma^2 \langle \Delta\boldsymbol{T}^\top \boldsymbol{C} \Delta\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \right\} \\
&= \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \\
&+ \gamma \left\{ \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\Delta\overline{\boldsymbol{h}}^\top + \Delta\overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C} \Delta\boldsymbol{T} + \Delta\boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \right\} \\
&+ \gamma^2 \left\{ \langle \overline{\boldsymbol{C}}^2, \Delta\overline{\boldsymbol{h}}\Delta\overline{\boldsymbol{h}}^\top \rangle - 2\langle \Delta\boldsymbol{T}^\top \boldsymbol{C} \Delta\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \right\}
\end{aligned} \tag{8.132}$$

where power operations are applied element-wise. Then using properties of the Trace operator, we have the relations

$$\langle \boldsymbol{A}, \boldsymbol{x}\boldsymbol{y}^\top \rangle = \mathrm{Tr}\{\boldsymbol{A}^\top \boldsymbol{x}\boldsymbol{y}^\top\} = \mathrm{Tr}\{\boldsymbol{y}\boldsymbol{x}^\top \boldsymbol{A}\} = \mathrm{Tr}\{\boldsymbol{A}\boldsymbol{y}\boldsymbol{x}^\top\} = \langle \boldsymbol{A}^\top, \boldsymbol{y}\boldsymbol{x}^\top \rangle \tag{8.133}$$

and

$$\langle \boldsymbol{T}^\top \boldsymbol{C} \Delta\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle = \mathrm{Tr}\{\Delta\boldsymbol{T}^\top \boldsymbol{C}^\top \boldsymbol{T} \overline{\boldsymbol{C}}\} = \mathrm{Tr}\{\overline{\boldsymbol{C}} \Delta\boldsymbol{T}^\top \boldsymbol{C}^\top \boldsymbol{T}\} = \langle \Delta\boldsymbol{T}^\top \boldsymbol{C}^\top \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \tag{8.134}$$

one has

$$\begin{aligned}
\mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{Z}(\gamma)) = \langle \boldsymbol{C}^2, \boldsymbol{h}\boldsymbol{h}^\top \rangle + \langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{T}^\top \boldsymbol{C} \boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \\
+ \gamma \left\{ \langle \overline{\boldsymbol{C}}^2 + \overline{\boldsymbol{C}}^{2T}, \overline{\boldsymbol{h}}\Delta\overline{\boldsymbol{h}}^\top \rangle - 2\langle \boldsymbol{T}^\top (\boldsymbol{C} + \boldsymbol{C}^\top) \Delta\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \right\} \\
+ \gamma^2 \left\{ \langle \overline{\boldsymbol{C}}^2, \Delta\overline{\boldsymbol{h}}\Delta\overline{\boldsymbol{h}}^\top \rangle - 2\langle \Delta\boldsymbol{T}^\top \boldsymbol{C} \Delta\boldsymbol{T}, \overline{\boldsymbol{C}} \rangle \right\}
\end{aligned} \tag{8.135}$$

So one can define this GW cost as

$$\mathcal{E}^{GW}(\boldsymbol{C}, \overline{\boldsymbol{C}}, \boldsymbol{Z}(\gamma)) := a^{GW}\gamma^2 + b^{GW}\gamma + c^{GW} \tag{8.136}$$

where coefficients are given in equation (8.135).

Then through similar operations $\mathcal{E}^W$ can be expressed as

$$\begin{aligned}
&\mathcal{E}^W(\boldsymbol{F}, \overline{\boldsymbol{F}}, \boldsymbol{Z}(\gamma)) \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h} \rangle + \langle \overline{\boldsymbol{F}}^2 \mathbf{1}_d, \overline{\boldsymbol{h}}(\gamma) \rangle - 2\langle \boldsymbol{F}\overline{\boldsymbol{F}}^\top, \boldsymbol{Z}(\gamma) \rangle \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h} \rangle + \langle \overline{\boldsymbol{F}}^2 \mathbf{1}_d, \overline{\boldsymbol{h}} + \gamma\Delta\overline{\boldsymbol{h}} \rangle - 2\langle \boldsymbol{F}\overline{\boldsymbol{F}}^\top, \boldsymbol{T} + \gamma\Delta\boldsymbol{T} \rangle \\
&= \langle \boldsymbol{F}^2 \mathbf{1}_d, \boldsymbol{h} \rangle + \langle \overline{\boldsymbol{F}}^2 \mathbf{1}_d, \overline{\boldsymbol{h}} \rangle - 2\langle \boldsymbol{F}\overline{\boldsymbol{F}}^\top, \boldsymbol{T} \rangle + \gamma \left\{ \langle \overline{\boldsymbol{F}}^2 \mathbf{1}_d, \Delta\overline{\boldsymbol{h}} \rangle - 2\langle \boldsymbol{F}\overline{\boldsymbol{F}}^\top, \Delta\boldsymbol{T} \rangle \right\}
\end{aligned} \tag{8.137}$$

So one can define this Wasserstein cost as

$$\mathcal{E}^W(\boldsymbol{F}, \overline{\boldsymbol{F}}, \boldsymbol{Z}(\gamma)) := b^W\gamma + c^W \tag{8.138}$$

where coefficients are given in equation (8.137).

Notice that only the GW cost leads to a second order polynom in γ. Therefore, using equations (8.136) and (8.138), the FGW cost given in equation (8.131) can be written as

$$\mathcal{E}_\alpha^{FGW}(\boldsymbol{C}, \boldsymbol{F}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{F}}, \boldsymbol{Z}(\gamma)) := \alpha a^{GW}\gamma^2 + b\gamma + c \tag{8.139}$$

with for all $x \in \{b, c\}, x = \alpha x^{GW} + (1-\alpha)x^{W}$.

8.4.4 Proof of Proposition 3: Convergence of srFGW Mirror-Descent algorithm

Even if the objective function of the srGW problem and its variants is not convex, we prove in the following the non-asymptotic stationary convergence of the MD algorithms for the generic objective which encompasses every aforementioned ones:

$$\min_{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, m)} \langle a\boldsymbol{L}(\boldsymbol{C}, \overline{\boldsymbol{C}}) \otimes \boldsymbol{T} + b\boldsymbol{D}, \boldsymbol{T} \rangle, \tag{8.140}$$

where $(a, b) \in \mathbb{R}_+^2$ denote any scalar constants.

The proof relies on Proposition 2 of Scetbon et al. (2021a). The following steps consist in proving that the objective function in problem (8.140) satisfies the sufficient conditions of Proposition 2, to conclude on the convergence of our MD algorithm. The main challenge, addressed in Lemma 6, is to prove the relative smoothness of our objective *w.r.t.* the KL divergence. This lemma proves a sufficient condition, following Zhang et al. (2020), to such a relative smoothness.

Let us first recall some notions linked to the relative smoothness with respect to a convex function. Let $\mathcal{X}$ a closed convex subset in an Euclidean space $\mathbb{R}^d$. Given a convex function $H : \mathcal{X} \to \mathbb{R}$ continuously differentiable, one can define the *prox-function* associated to H as

$$D_H(\boldsymbol{x}, \boldsymbol{z}) := H(\boldsymbol{x}) - H(\boldsymbol{z}) - \langle \nabla H(\boldsymbol{z}), \boldsymbol{x} - \boldsymbol{z} \rangle. \tag{8.141}$$

Then the definition of the relative smoothness with respect to H is formally expressed as follow:

Definition 6 (Relative smoothness) *Let $L > 0$ and f continuously differentiable on $\mathcal{X}$. f is said to be L-smooth relatively to H if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,*

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + L D_H(\boldsymbol{y}, \boldsymbol{x}). \tag{8.142}$$

Another definition of use for the following is:

Definition 7 (relative strong convexity) *Let $\alpha > 0$ and f continuously differentiable on $\mathcal{X}$. f is said to be α-strongly convex relatively to h if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$,*

$$f(y) \geq f(x) + \langle \nabla f(\boldsymbol{x}), y - x \rangle + \alpha D_h(y, x). \tag{8.143}$$

We will use in the following this general result on the non-asymptotic stationary convergence of the mirror-descent scheme defined by the following recursion:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} \rangle + \frac{1}{\gamma_k} D_H(\boldsymbol{x}, \boldsymbol{x}_k) \tag{8.144}$$

where (γ_k) is a sequence of positive step-size. Which is expressed for srFGW by equation (6.10), for $b = (1-a)$ with $a \in [0, 1]$ and $\boldsymbol{D} = \boldsymbol{D}(\boldsymbol{F}, \overline{\boldsymbol{F}})$ the pairwise Euclidean distance matrix between node features $\boldsymbol{F}$ and $\overline{\boldsymbol{F}}$. The latter can be easily adapted to include the additional sparsity promoting regularization discussed in Section 6.2.2.

Proposition 2 (Scetbon et al. (2021a)) *Let $N \geq 1$, f continuously differentiable on $\mathcal{X}$ and L-smooth relatively to H. By considering for all $k \in [\![N]\!], \gamma_k = 1/2L$ and by*

denoting $D_0 = f(x_0) - \min_{x\in\mathcal{X}} f(x)$, *we have*

$$\min_{k\in[\![N-1]\!]} \Delta_k \leq \frac{4LD_0}{N}, \tag{8.145}$$

where for all $k \in [\![N]\!]$,

$$\Delta_k := \frac{1}{\gamma_k^2}\{D_H(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) + D_H(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k)\}. \tag{8.146}$$

Let us now show that the generic objective function given in equation (8.140) is relatively smooth with respect to the KL divergence, so we will conclude on the convergence of our MD scheme directly using proposition 2.
Note that, without loss of generality, we omit in the definition of $F_{a,b,\varepsilon}$ the term of the srFGW objective that involves the constant vector $\boldsymbol{h}$.

Lemma 6 *Let* $(a, b, \varepsilon) \in \mathbb{R}_+^3$. *We denote for any* $\boldsymbol{T} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h})$ *with second marginal* $\overline{\boldsymbol{h}}$ *and* $\varepsilon \in \mathbb{R}_+^*$, *the reduced objective function:*

$$F_{a,b,\varepsilon}(\boldsymbol{T}) = a\langle \overline{\boldsymbol{C}}^2, \overline{\boldsymbol{h}}\overline{\boldsymbol{h}}^\top\rangle_F - 2a\langle \boldsymbol{T}\overline{\boldsymbol{C}}\boldsymbol{T}^\top, \boldsymbol{C}\rangle_F + b\langle \boldsymbol{D}, \boldsymbol{T}\rangle_F + \varepsilon H(\boldsymbol{T}) \tag{8.147}$$

where $H(\boldsymbol{T}) = \langle \boldsymbol{T}, \log \boldsymbol{T} - \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top\rangle_F$ *denotes the* negative entropy *of* $\boldsymbol{T}$.
Then for all $\boldsymbol{T}_1$ *and* $\boldsymbol{T}_2$ *in* $\mathcal{U}(\boldsymbol{h}, m)$, *we have*

$$\|\nabla F_{a,b,\varepsilon}(\boldsymbol{T}_1) - \nabla F_{a,b,\varepsilon}(\boldsymbol{T}_2)\|_F \leq L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon}\|\nabla H(\boldsymbol{T}_1) - \nabla H(\boldsymbol{T}_2)\|_F \tag{8.148}$$

where $L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon} = a\{n\|\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2\|_F + 4\|\boldsymbol{C}\|_F\|\overline{\boldsymbol{C}}\|_F\} + \varepsilon$.

Proof of Lemma 6. First of all we have for any $\boldsymbol{T} \in \mathcal{U}(\boldsymbol{h}, m)$,

$$\begin{aligned}
\nabla_{\boldsymbol{T}} F_{a,b,\varepsilon}(\boldsymbol{T}) &= \nabla_{\boldsymbol{T}} \operatorname{Tr}\{a\boldsymbol{T}^\top\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{T}\overline{\boldsymbol{C}}^{2\top} - 2a\boldsymbol{T}^\top\boldsymbol{C}\boldsymbol{T}\overline{\boldsymbol{C}}^\top + b\boldsymbol{D} - \varepsilon\boldsymbol{T}^\top(\log\boldsymbol{T} - \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top)\} \\
&= a\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{T}(\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2) - 2a\{\boldsymbol{C}\boldsymbol{T}\overline{\boldsymbol{C}}^\top + \boldsymbol{C}^\top\boldsymbol{T}\overline{\boldsymbol{C}}\} + b\boldsymbol{D} - \varepsilon\{\log\boldsymbol{T} + \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top - \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top\} \\
&= a\mathbf{1}_n\mathbf{1}_n^\top\boldsymbol{T}(\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2) - 2a\{\boldsymbol{C}\boldsymbol{T}\overline{\boldsymbol{C}}^\top + \boldsymbol{C}^\top\boldsymbol{T}\overline{\boldsymbol{C}}\} + b\boldsymbol{D} - \varepsilon\log\boldsymbol{T}.
\end{aligned} \tag{8.149}$$

Therefore we have,

$$\begin{aligned}
&\|\nabla F_{a,b,\varepsilon}(\boldsymbol{T}_1) - \nabla F_{a,b,\varepsilon}(\boldsymbol{T}_2)\|_F \\
&= \|a\mathbf{1}_n\mathbf{1}_n^\top(\boldsymbol{T}_1 - \boldsymbol{T}_2)(\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2) - 2a\{\boldsymbol{C}(\boldsymbol{T}_1 - \boldsymbol{T}_2)\overline{\boldsymbol{C}}^\top + \boldsymbol{C}^\top(\boldsymbol{T}_1 - \boldsymbol{T}_2)\overline{\boldsymbol{C}}\} - \varepsilon(\log\boldsymbol{T}_1 - \log\boldsymbol{T}_2)\|_F \\
&\leq a\|\mathbf{1}_n\mathbf{1}_n^\top(\boldsymbol{T}_1 - \boldsymbol{T}_2)(\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2)\|_F + 2a\{\|\boldsymbol{C}(\boldsymbol{T}_1 - \boldsymbol{T}_2)\overline{\boldsymbol{C}}^\top\|_F + \|\boldsymbol{C}^\top(\boldsymbol{T}_1 - \boldsymbol{T}_2)\overline{\boldsymbol{C}}\|_F\} \\
&+ \varepsilon\|\log\boldsymbol{T}_1 - \log\boldsymbol{T}_2\|_F
\end{aligned} \tag{8.150}$$

By applying the triangle inequality. Then using the sub-multiplicativity of the Frobenius norm, we have

$$\begin{aligned}
&a\|\mathbf{1}_n\mathbf{1}_n^\top(\boldsymbol{T}_1 - \boldsymbol{T}_2)(\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2)\|_F + 2a\{\|\boldsymbol{C}(\boldsymbol{T}_1 - \boldsymbol{T}_2)\overline{\boldsymbol{C}}^\top\|_F + \|\boldsymbol{C}^\top(\boldsymbol{T}_1 - \boldsymbol{T}_2)\overline{\boldsymbol{C}}\|_F\} \\
&\leq a\{\|\mathbf{1}_n\mathbf{1}_n^\top\|_F\|\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2\|_F + 4\|\boldsymbol{C}\|_F\|\overline{\boldsymbol{C}}\|_F\}\|\boldsymbol{T}_1 - \boldsymbol{T}_2\|_F
\end{aligned} \tag{8.151}$$

Now we seek for a comparison between H and the Frobenius norm to conclude. Note that H is equivalent to the sum of negative entropy functions over columns *i.e* $\boldsymbol{T} \in \mathcal{U}_{\overline{n}}(\boldsymbol{h}) \to$

$\sum_j \widetilde{H}(\boldsymbol{T}_{:j}) = H(\boldsymbol{T})$. For any $\boldsymbol{x} \in \Sigma_n$, denoting $g : \boldsymbol{x} \to -\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} - \widetilde{H}(\boldsymbol{x})$, we have

$$\begin{aligned} &\nabla g(\boldsymbol{x}) = -\boldsymbol{x} + \log \boldsymbol{x} + \mathbf{1}_n - \mathbf{1}_n = -\boldsymbol{x} + \log \boldsymbol{x} \\ \implies &\nabla^2 g(\boldsymbol{x}) = \operatorname{diag}(-1 + 1/\boldsymbol{x}) \end{aligned}$$

which is a PSD matrix onto the probability simplex leading to $\nabla^2_{\boldsymbol{x}} \widetilde{H}_{|\Sigma_n} < \nabla^2_{\boldsymbol{x}} \left(-\frac{1}{2}\|.\|_2^2\right)_{|\Sigma_n}$. This property is equivalent to the 1-strong relative convexity of $\left(-\frac{1}{2}\|.\|_2^2\right)$ with respect to $\widetilde{H}$ as proven in Lu et al. (2018). Then we can conclude by linearity that $\left(-\frac{1}{2}\|.\|_F^2\right)$ is 1-strongly convex relatively to H, which provides from $\boldsymbol{T}_1$ to $\boldsymbol{T}_2$ the following inequalities resulting from the aforementioned definition:

$$\begin{aligned} &\frac{-\langle \boldsymbol{T}_2, \boldsymbol{T}_2\rangle_F}{2} \geq \frac{-\langle \boldsymbol{T}_1, \boldsymbol{T}_1\rangle_F}{2} - \langle \boldsymbol{T}_1, \boldsymbol{T}_2 - \boldsymbol{T}_1\rangle_F - \langle \boldsymbol{T}_2, \log \boldsymbol{T}_2 - \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top\rangle_F + \langle \boldsymbol{T}_1, \log \boldsymbol{T}_1 - \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top\rangle_F \\ &\qquad + \langle \log \boldsymbol{T}_1, \boldsymbol{T}_2 - \boldsymbol{T}_1\rangle_F \\ \Leftrightarrow &\frac{-\langle \boldsymbol{T}_2, \boldsymbol{T}_2\rangle_F}{2} \geq \frac{\langle \boldsymbol{T}_1, \boldsymbol{T}_1\rangle_F}{2} - \langle \boldsymbol{T}_1, \boldsymbol{T}_2\rangle_F + \langle \boldsymbol{T}_2, \log \boldsymbol{T}_1 - \log \boldsymbol{T}_2\rangle_F \end{aligned} \tag{8.152}$$

using that $\langle \boldsymbol{T}_1, \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top\rangle_F = \langle \boldsymbol{T}_2, \mathbf{1}_n\mathbf{1}_{\overline{n}}^\top\rangle_F = 1$. Then a simple rearrangement gives

$$\frac{1}{2}\|\boldsymbol{T}_2 - \boldsymbol{T}_1\|_F^2 \leq \langle \boldsymbol{T}_2, \log \boldsymbol{T}_2 - \log \boldsymbol{T}_1\rangle_F. \tag{8.153}$$

The analog computation from $\boldsymbol{T}_2$ to $\boldsymbol{T}_1$ gives that

$$\frac{1}{2}\|\boldsymbol{T}_2 - \boldsymbol{T}_1\|_F^2 \leq \langle \boldsymbol{T}_1, \log \boldsymbol{T}_1 - \log \boldsymbol{T}_2\rangle_F. \tag{8.154}$$

Adding both last inequalities, we obtain

$$\begin{aligned} \|\boldsymbol{T}_1 - \boldsymbol{T}_2\|_F^2 &\leq \langle \boldsymbol{T}_1 - \boldsymbol{T}_2, \log \boldsymbol{T}_1 - \log \boldsymbol{T}_2\rangle_F \\ &\leq \|\boldsymbol{T}_1 - \boldsymbol{T}_2\|_F \|\log \boldsymbol{T}_1 - \log \boldsymbol{T}_2\|_F \end{aligned} \tag{8.155}$$

by Cauchy-Schwarz inequality. Which is equivalent to

$$\|\boldsymbol{T}_1 - \boldsymbol{T}_2\|_F \leq \|\log \boldsymbol{T}_1 - \log \boldsymbol{T}_2\|_F. \tag{8.156}$$

Finally, Putting equations 8.150, 8.151 and 8.156 all together, we have

$$\begin{aligned} \|\nabla F_{a,b,\varepsilon}(\boldsymbol{T}_1) - \nabla F_{a,b,\varepsilon}(\boldsymbol{T}_2)\|_F &\leq a\{n\|\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2\|_F + 4\|\boldsymbol{C}\|_F\|\overline{\boldsymbol{C}}\|_F + \varepsilon\}\|\log \boldsymbol{T}_1 - \log \boldsymbol{T}_2\|_F \\ &= L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon}\|\nabla H(\boldsymbol{T}_1) - \nabla H(\boldsymbol{T}_2)\|_F \end{aligned} \tag{8.157}$$

where $L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon} = a\{n\|\overline{\boldsymbol{C}}^{2\top} + \overline{\boldsymbol{C}}^2\|_F + 4\|\boldsymbol{C}\|_F\|\overline{\boldsymbol{C}}\|_F\} + \varepsilon$, as $\nabla H(\boldsymbol{T}) = -\log \boldsymbol{T}$ (see (8.149)). This final result provides the desired inequality of Lemma 6.
□

Proposition 3 (Mirror-Descent convergence) *For any attributed graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$, the Mirror-Descent algorithm 13 converges to a stationary point non-asymptotically.*

Proof of Proposition 3. Using Lemma 6, we can conclude that the objective function involved in equation (8.140) and Proposition 3 is $L_{\boldsymbol{C},\overline{\boldsymbol{C}},a,\varepsilon}$-smooth relatively to H. Therefore all assumptions from Proposition 2 are satisfied. This allows to conclude on the convergence

of our MD algorithm stated in Proposition 3, which then only requires to set appropriate parameters in the generic equation (8.140) to match those of srFGW.
□

8.4.5 Complements on graphs clustering experiments

TABLE 8.5: Clustering performances on real datasets measured by Adjusted Rand Index(ARI. In bold (resp. italic) we highlight the first (resp. second) best method.

	NO ATTRIBUTE		DISCRETE ATTRIBUTES		REAL ATTRIBUTES			
MODELS	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
srGW (ours)	3.14(0.19)	2.26(0.08)	41.12(0.93)	2.71(0.16)	6.24(1.49)	5.98(1.26)	3.74(0.22)	16.67(0.19)
srGW_g	**5.03(0.90)**	**3.09(0.11)**	43.27(1.20)	3.28(0.76)	**16.50(2.06)**	**7.78(1.46)**	*4.12(0.12)*	**18.52(0.28)**
srGW_e	3.51(1.10)	2.18(0.05)	*48.32(1.65)*	*4.60(0.91)*	15.44(2.46)	5.71(0.93)	3.36(0.39)	16.81(0.17)
srGW_{e+g}	*4.56(1.62)*	*2.71(0.24)*	**48.77(1.47)**	**4.97(0.83)**	*16.38(2.15)*	6.15(1.24)	3.98(0.62)	18.03(0.32)
GDL	2.67(0.52)	2.26(0.13)	39.62(0.49)	2.72(0.48)	6.43(1.42)	5.12(1.37)	3.39(0.31)	17.08(0.21)
GDL_{reg}	3.44(1.09)	2.17(0.19)	40.75(0.23)	3.59(0.71)	14.83(2.88)	*6.27(1.89)*	3.57(0.44)	*18.25(0.37)*
GWF-r	2.09(0.61)	2.03(0.15)	37.09(1.13)	2.92(0.92)	2.89(2.66)	5.18(1.17)	**4.27(0.31)**	17.34(0.14)
GWF-f	0.85(0.57)	1.74(0.13)	18.14(3.09)	1.54(1.24)	2.78(2.41)	4.03(0.96)	3.69(0.28)	15.89(0.20)
GW-k	0.66(0.07)	1.23(0.04)	15.09(2.48)	0.66(0.43)	4.56(0.83)	4.19(0.58)	2.34(0.96)	0.43(0.06)

TABLE 8.6: Clustering performances on real datasets measured by Adjusted Mutual Information(AMI). In bold (resp. italic) we highlight the first (resp. second) best method.

	NO ATTRIBUTE		DISCRETE ATTRIBUTES		REAL ATTRIBUTES			
MODELS	IMDB-B	IMDB-M	MUTAG	PTC-MR	BZR	COX2	ENZYMES	PROTEIN
srGW (ours)	3.31(0.25)	2.63(0.33)	32.97(0.57)	3.21(0.23)	8.20(0.75)	2.64(0.40)	6.99(0.18)	12.69(0.32)
srGW_g	**4.65(0.33)**	**2.95(0.24)**	33.82(1.58)	**5.47(0.55)**	9.25(1.66)	3.08(0.61)	7.48(0.24)	13.75(0.18)
srGW_e	3.58(0.25)	*2.57(0.26)*	*35.01(0.96)*	2.53(0.56)	**10.28(1.03)**	3.01(0.78)	7.71(0.29)	12.51(0.35)
srGW_{e+g}	*4.20(0.17)*	2.49(0.61)	**35.13(2.10)**	2.80(0.64)	*10.09(1.19)*	**3.76(0.63)**	*8.27(0.34)*	**14.11(0.30)**
GDL	2.78(0.20)	*2.57(0.39)*	32.25(0.95)	3.81(0.46)	8.14(0.84)	2.02(0.89)	6.86(0.32)	12.06(0.31)
GDL_{reg}	3.42(0.41)	2.52(0.27)	32.73(0.98)	*4.93(0.49)*	8.76(1.25)	2.56(0.95)	7.39(0.40)	*13.77(0.49)*
GWF-r	2.11(0.34)	2.41(0.46)	32.94(1.96)	2.39(0.79)	5.65(1.86)	*3.28(0.71)*	**8.31(0.29)**	12.82(0.28)
GWF-f	1.05(0.15)	1.85(0.28)	15.03(0.71)	1.27(0.96)	3.89(1.62)	1.53(0.58)	7.56(0.21)	11.05(0.33)
GW-k	0.68(0.08)	1.39(0.19)	9.68(1.04)	0.80(0.18)	6.91(0.48)	1.51(0.17)	4.99(0.63)	3.94(0.09)

Bibliography

Yonathan Aflalo, Alexander Bronstein, and Ron Kimmel. On convex relaxation of graph isomorphism. *Proceedings of the National Academy of Sciences*, 112(10):2942–2947, 2015. (Cited on page 50).

SN Afriat. Theory of maxima and the method of lagrange. *SIAM Journal on Applied Mathematics*, 20(3):343–357, 1971. (Cited on page 69).

Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011. (Cited on pages 37 and 38).

Nesreen K Ahmed, Theodore L Willke, and Ryan A Rossi. Estimation of local subgraph counts. In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 586–595. IEEE, 2016. (Cited on page 21).

Fabio Aiolli, Michele Donini, Nicolò Navarin, and Alessandro Sperduti. Multiple graph-kernel learning. In *2015 IEEE Symposium Series on Computational Intelligence*, pp. 1607–1614. IEEE, 2015. (Cited on page 20).

David Alvarez-Melis, Stefanie Jegelka, and Tommi S Jaakkola. Towards optimal transport with global invariances. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1870–1879. PMLR, 2019. (Cited on page 49).

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2005. (Cited on pages 33 and 49).

Paul Appell. Mémoire sur les déblais et les remblais des systemes continus ou discontinus. *Mémoires présentes par divers Savants à l'Académie des Sciences de l'Institut de France*, 29:1–208, 1887. (Cited on page 30).

Ernesto Araya Valdivia and De Castro Yohann. Latent distance estimation for random geometric graphs. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 142).

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. (Cited on pages 14 and 86).

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950. (Cited on page 18).

James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. (Cited on page 26).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. (Cited on page 24).

Lu Bai, Peng Ren, Xiao Bai, and Edwin R Hancock. A graph kernel from the depth-based representation. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 1–11. Springer, 2014. (Cited on page 21).

Lu Bai, Luca Rossi, Zhihong Zhang, and Edwin Hancock. An aligned subtree kernel for weighted graphs. In *International Conference on Machine Learning*, pp. 30–39. PMLR, 2015. (Cited on page 21).

Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive graph-level representation learning via graph-graph proximity. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 1988–1994, 2019. (Cited on page 26).

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1):89–112, 2008. (Cited on pages 15, 19, and 68).

Muhammet Balcilar, Renton Guillaume, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. (Cited on page 71).

Abhijit Banerjee, Arun G Chandrasekhar, Esther Duflo, and Matthew O Jackson. The diffusion of microfinance. *Science*, 341(6144), 2013. (Cited on pages 12 and 125).

Amélie Barbe, Marc Sebban, Paulo Gonçalves, Pierre Borgnat, and Rémi Gribonval. Graph diffusion wasserstein distances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 577–592. Springer, 2021. (Cited on page 140).

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. (Cited on pages 11, 12, and 66).

Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011. (Cited on page 36).

François Bavaud. Euclidean distances, soft and spectral clustering on weighted graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 103–118. Springer, 2010. (Cited on page 42).

Valérie Beaudouin, Isabelle Bloch, David Bounie, Stéphan Clémençon, Florence d'Alché Buc, James Eagan, Winston Maxwell, Pavlo Mozharovskyi, and Jayneel Parekh. Flexible and context-specific ai explainability: A multidisciplinary approach. *arXiv preprint arXiv:2003.07703*, 2020. (Cited on page 13).

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. (Cited on page 38).

Asa Ben-Hur, David Horn, Hava T Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001. (Cited on page 18).

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015. (Cited on pages 36, 38, and 118).

Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Advances in Neural Information Processing Systems*, 16, 2003. (Cited on page 76).

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013. (Cited on pages 13 and 17).

Alexander C Berg, Tamara L Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pp. 26–33. IEEE, 2005. (Cited on page 50).

Florian Bernard, Christian Theobalt, and Michael Moeller. Ds*: Tighter lifting-free convex relaxations for quadratic matching problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4310–4319, 2018. (Cited on pages 50 and 112).

Dimitri P Bertsekas and Jonathan Eckstein. Dual coordinate step methods for linear network flow problems. *Mathematical Programming*, 42(1):203–243, 1988. (Cited on page 35).

Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997. (Cited on page 31).

Gregory Beylkin. The inversion problem and applications of the generalized radon transform. *Communications on pure and applied mathematics*, 37(5):579–599, 1984. (Cited on page 34).

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pp. 874–883. PMLR, 2020a. (Cited on pages 26 and 112).

Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE Transactions on Neural Networks and Learning Systems*, 2020b. (Cited on page 26).

Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucuman, Ser. A*, 5: 147–154, 1946. (Cited on page 31).

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008. (Cited on page 126).

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013. (Cited on page 85).

Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 1998. (Cited on page 17).

J. Bonnans and Alexander Shapiro. *Perturbation Analysis of Optimization Problems*. 01 2000. ISBN 978-1-4612-7129-1. doi: 10.1007/978-1-4612-1394-9. (Cited on pages 45, 55, 64, 96, and 123).

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*, pp. 1–12, 2011. (Cited on pages 35 and 51).

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1): 22–45, 2015. (Cited on page 33).

Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: Histogram regression using optimal transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)*, 35(4), 2016. (Cited on page 86).

Christian Borgs, Jennifer T Chayes, Henry Cohn, and Nina Holden. Sparse exchangeable graphs and their limits via graphon processes. *arXiv preprint arXiv:1601.07134*, 2016. (Cited on page 142).

Christian Borgs, Jennifer Chayes, Henry Cohn, and Yufei Zhao. An lp theory of sparse graph convergence i: Limits, sparse random graph models, and power law distributions. *Transactions of the American Mathematical Society*, 372(5):3019–3062, 2019. (Cited on page 142).

Christian Borgs, Jennifer T Chayes, Souvik Dhara, and Subhabrata Sen. Limits of sparse configuration models and beyond: Graphexes and multigraphexes. *The Annals of Probability*, 49(6):2830–2873, 2021. (Cited on page 142).

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005. (Cited on pages 12, 22, 66, 99, 104, and 129).

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 (suppl_1):i47–i56, 2005. (Cited on page 75).

Gecia Bravo Hermsdorff and Lee Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 26).

Kristian Bredies, Dirk A Lorenz, and Peter Maass. A generalized conditional gradient method and its connection to an iterative shrinkage method. *Computational Optimization and applications*, 42(2):173–193, 2009. (Cited on page 37).

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967. (Cited on page 36).

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991. (Cited on pages 30 and 41).

Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Counting graphlets: Space vs time. In *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 557–566, 2017. (Cited on page 21).

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. (Cited on page 52).

Luc Brogat-Motte, Rémi Flamary, Celine Brouard, Juho Rousu, and Florence D'Alché-Buc. Learning to predict graphs with fused gromov-Wasserstein barycenters. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2321–2335. PMLR, 17–23 Jul 2022. (Cited on pages 43 and 53).

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. (Cited on pages 12, 13, 17, 23, 25, 66, and 85).

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021a. (Cited on pages 15, 18, 23, and 25).

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velivcković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, abs/2104.13478, 2021b. (Cited on page 66).

Céline Brouard, Florence d'Alché Buc, and Marie Szafranski. Semi-supervised penalized output kernel regression for link prediction. In *28th International Conference on Machine Learning (ICML 2011)*, pp. 593–600, 2011. (Cited on page 85).

Richard A Brualdi. *Combinatorial matrix classes*, volume 13. Cambridge University Press, 2006. (Cited on page 31).

Rickard Brüel Gabrielsson. Universal function approximation on graphs. *Advances in Neural Information Processing Systems*, 33:19762–19772, 2020. (Cited on page 71).

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. (Cited on pages 25 and 26).

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013. (Cited on page 105).

Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009. (Cited on page 12).

Horst Bunke and Gudrun Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983. (Cited on page 142).

Donald Bures. An extension of kakutani's theorem on infinite product measures to the tensor product of semifinite w*-algebras. *Transactions of the American Mathematical Society*, 135:199–212, 1969. (Cited on page 34).

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020. (Cited on page 141).

Guillermo Canas and Lorenzo Rosasco. Learning probability measures with respect to optimal transport metrics. *Advances in Neural Information Processing Systems*, 25, 2012. (Cited on page 38).

Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011. (Cited on page 85).

Mathieu Carriere, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *International conference on machine learning*, pp. 664–673. PMLR, 2017. (Cited on page 33).

Elsa Cazelles, Vivien Seguy, Jérémie Bigot, Marco Cuturi, and Nicolas Papadakis. Geodesic pca versus log-pca of histograms in the wasserstein space. *SIAM Journal on Scientific Computing*, 40(2):B429–B456, 2018. (Cited on page 39).

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011. (Cited on page 19).

Laetitia Chapel, Mokhtar Z Alaya, and Gilles Gasso. Partial gromov-wasserstein with applications on positive-unlabeled learning. *arXiv preprint arXiv:2002.08276*, 2019. (Cited on pages 113 and 115).

Benson Chen, Gary Bécigneul, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Optimal transport graph neural networks. *arXiv preprint arXiv:2006.04804*, 2020a. (Cited on pages 14, 66, 67, 70, 72, 73, 75, and 86).

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3438–3445, 2020b. (Cited on page 140).

Samantha Chen, Sunhyuk Lim, Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Weisfeiler-lehman meets gromov-wasserstein. *arXiv preprint arXiv:2202.02495*, 2022. (Cited on page 49).

Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001. (Cited on page 90).

Xiaowei Chen and John CS Lui. Mining graphlet counts in online social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):1–38, 2018. (Cited on page 21).

Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7239–7248, 2018. (Cited on page 12).

Yudong Chen, Sujay Sanghavi, and Huan Xu. Improved graph clustering. *IEEE Transactions on Information Theory*, 60(10):6440–6455, 2014. (Cited on page 112).

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019. (Cited on page 73).

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020c. (Cited on page 25).

Mark Cheung, John Shi, Oren Wright, Lavendar Y Jiang, Xujin Liu, and José MF Moura. Graph signal processing and deep learning: Convolution, pooling, and topology. *IEEE Signal Processing Magazine*, 37(6):139–149, 2020. (Cited on page 25).

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018. (Cited on page 33).

KR1442 Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pp. 603–649, 2020. (Cited on page 12).

Samir Chowdhury and Facundo Mémoli. The Gromov-Wasserstein distance between networks and stable network invariants. *arXiv:1808.04337 [cs, math]*, September 2019. arXiv: 1808.04337. (Cited on pages 14, 40, 42, 47, 48, 53, 59, 60, 67, 71, and 150).

Samir Chowdhury and Tom Needham. Gromov-wasserstein averaging in a riemannian framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 842–843, 2020. (Cited on pages 43 and 53).

Samir Chowdhury and Tom Needham. Generalized spectral clustering via gromov-wasserstein learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 712–720. PMLR, 2021. (Cited on pages 67, 113, 124, 125, and 138).

Samir Chowdhury, David Miller, and Tom Needham. Quantized gromov-wasserstein. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 811–827. Springer, 2021. (Cited on pages 52 and 62).

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997. (Cited on page 17).

Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Lió, Marco Maggini, and Stefano Melacci. Logic explained networks. *Artificial Intelligence*, 314:103822, 2023. (Cited on page 13).

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004. (Cited on page 126).

Adam Coates and Andrew Ng. Selecting receptive fields in deep networks. *Advances in neural information processing systems*, 24, 2011. (Cited on page 26).

Laurent Condat. Fast projection onto the simplex and the l1 ball. *Mathematical Programming*, 158(1):575–585, 2016. (Cited on page 69).

Padraig Corcoran. Function space pooling for graph convolutional networks. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pp. 473–483. Springer, 2020. (Cited on page 26).

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995. (Cited on page 18).

Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *ICML*, 2010. (Cited on page 21).

Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 274–289. Springer, 2014. (Cited on pages 14, 37, and 119).

Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. Learning wasserstein embeddings. *arXiv preprint arXiv:1710.07457*, 2017a. (Cited on page 39).

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *Advances in Neural Information Processing Systems*, 30, 2017b. (Cited on pages 13 and 14).

Imre Csiszár. Information-type measures of difference of probability distributions and indirect observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318, 1967. (Cited on page 32).

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013. (Cited on pages 35 and 117).

Marco Cuturi and David Avis. Ground metric learning. *The Journal of Machine Learning Research*, 15(1):533–564, 2014. (Cited on pages 39, 45, and 55).

Marco Cuturi and Mathieu Blondel. Soft-DTW: a Differentiable Loss Function for Time-Series. *arXiv:1703.01541 [stat]*, February 2018. arXiv: 1703.01541. (Cited on page 85).

Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *International conference on machine learning*, pp. 685–693. PMLR, 2014. (Cited on pages 38 and 63).

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. (Cited on pages 71 and 72).

William HE Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of classification*, 2(1):7–28, 1985. (Cited on pages 12 and 85).

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 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. (Cited on pages 12, 99, and 129).

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016. (Cited on page 25).

Ishan Deshpande, Ziyu Zhang, and Alexander G Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3483–3491, 2018. (Cited on page 33).

Arnaud Dessein, Nicolas Papadakis, and Jean-Luc Rouas. Regularized optimal transport and the rot mover's distance. *The Journal of Machine Learning Research*, 19(1):590–642, 2018. (Cited on page 36).

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29 (11):1944–1957, 2007. (Cited on page 26).

Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*, 2022. (Cited on page 141).

Persi Diaconis and Svante Janson. Graph limits and exchangeable random graphs. *arXiv preprint arXiv:0712.2749*, 2007. (Cited on pages 112 and 142).

G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *Computational Intelligence Magazine, IEEE*, 10:12–25, 11 2015. (Cited on pages 12 and 85).

Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 0210–0215. IEEE, 2018. (Cited on page 13).

Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996. (Cited on page 18).

Richard Mansfield Dudley. The speed of mean glivenko-cantelli convergence. *The Annals of Mathematical Statistics*, 40(1):40–50, 1969. (Cited on page 33).

Theo Dumont, Théo Lacombe, and François-Xavier Vialard. ON THE EXISTENCE OF MONGE MAPS FOR THE GROMOV-WASSERSTEIN DISTANCE. working paper or preprint, October 2022. (Cited on page 41).

Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *International conference on machine learning*, pp. 1367–1376. PMLR, 2018. (Cited on page 51).

Nadav Dym, Haggai Maron, and Yaron Lipman. Ds++: A flexible, scalable and provably tight relaxation for matching problems. *arXiv preprint arXiv:1705.06148*, 2017. (Cited on page 50).

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. (Cited on page 12).

Christian Fabian, Kai Cui, and Heinz Koeppl. Learning sparse graphon mean field games. *arXiv preprint arXiv:2209.03880*, 2022. (Cited on page 142).

Kilian Fatras, Bharath Bhushan Damodaran, Sylvain Lobry, Remi Flamary, Devis Tuia, and Nicolas Courty. Wasserstein adversarial regularization for learning with label noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021a. (Cited on page 142).

Kilian Fatras, Younes Zine, Szymon Majewski, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. Minibatch optimal transport distances; analysis and applications. *arXiv preprint arXiv:2101.01792*, 2021b. (Cited on page 52).

Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten Borgwardt. Scalable kernels for graphs with continuous attributes. *Advances in neural information processing systems*, 26, 2013. (Cited on pages 66 and 104).

Aasa Feragen, Francois Lauze, and Soren Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3032–3042, 2015. (Cited on pages 18 and 19).

Sira Ferradans, Nicolas Papadakis, Gabriel Peyré, and Jean-François Aujol. Regularized discrete optimal transport. *SIAM Journal on Imaging Sciences*, 7(3):1853–1882, 2014. (Cited on pages 50 and 61).

Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009. (Cited on page 39).

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 869–877, 2018. (Cited on page 26).

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690. PMLR, 2019. (Cited on page 36).

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2): 298–305, 1973. (Cited on page 113).

R Flamary, N Courty, D Tuia, and A Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 1, 2016a. (Cited on page 37).

Rémi Flamary, Nicolas Courty, Alain Rakotomamonjy, and Devis Tuia. Optimal transport with laplacian regularization. In *NIPS 2014, Workshop on Optimal Transport and Machine Learning*, 2014. (Cited on pages 50 and 61).

Rémi Flamary, Cédric Févotte, Nicolas Courty, and Valentin Emiya. Optimal spectral transportation with application to music transcription. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 703–711, Red Hook, NY, USA, 2016b. Curran Associates Inc. ISBN 9781510838819. (Cited on page 115).

Rémi Flamary, Karim Lounici, and André Ferrari. Concentration bounds for linear monge mapping estimation and optimal transport domain adaptation. *arXiv preprint arXiv:1905.10155*, 2019. (Cited on page 34).

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. (Cited on pages 35, 51, 69, 96, 125, and 126).

P Thomas Fletcher, Conglin Lu, Stephen M Pizer, and Sarang Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE transactions on medical imaging*, 23(8):995–1005, 2004. (Cited on page 39).

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956. (Cited on pages 37 and 50).

Maurice Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pp. 215–310, 1948. (Cited on page 37).

Holger Fröhlich, Jörg K Wegner, Florian Sieker, and Andreas Zell. Optimal assignment kernels for attributed molecular graphs. In *Proceedings of the 22nd international conference on Machine learning*, pp. 225–232, 2005. (Cited on page 20).

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019. (Cited on page 26).

A Garcez, M Gori, LC Lamb, L Serafini, M Spranger, and SN Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–632, 2019. (Cited on page 13).

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pp. 3419–3430. PMLR, 2020. (Cited on page 71).

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003. (Cited on pages 22, 66, and 104).

Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic optimization for large-scale optimal transport. *Advances in neural information processing systems*, 29, 2016. (Cited on page 37).

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617. PMLR, 2018. (Cited on pages 14, 36, and 119).

Aude Genevay, Lénaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of sinkhorn divergences. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1574–1583. PMLR, 2019. (Cited on page 36).

James E Gentle. *Numerical linear algebra for applications in statistics*. Springer Science & Business Media, 1998. (Cited on page 22).

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017. (Cited on pages 13 and 23).

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017. (Cited on page 12).

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. (Cited on pages 12, 18, and 23).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. (Cited on page 142).

Marco Gori, Marco Maggini, and Lorenzo Sarti. Exact and approximate graph matching using random walks. *IEEE transactions on pattern analysis and machine intelligence*, 27 (7):1100–1111, 2005a. (Cited on page 142).

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, volume 2, pp. 729–734, 2005b. (Cited on page 66).

Daniele Grattarola, Daniele Zambon, Lorenzo Livi, and Cesare Alippi. Change detection in graph streams by learning graph embeddings on constant-curvature manifolds. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–14, 07 2019. (Cited on pages 85 and 86).

Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. (Cited on page 26).

Edouard Grave, Armand Joulin, and Quentin Berthet. Unsupervised alignment of embeddings with wasserstein procrustes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1880–1890. PMLR, 2019. (Cited on pages 14 and 49).

Marvin J Greenberg. *Euclidean and non-Euclidean geometries: Development and history*. Macmillan, 1993. (Cited on page 17).

Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, pp. 377–384, 2006. (Cited on page 20).

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006. (Cited on pages 33 and 36).

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. (Cited on pages 25 and 66).

William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020. (Cited on pages 13, 15, 18, 23, and 24).

Zaïd Harchaoui and Francis Bach. Image classification with segmentation graph kernels. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2007. (Cited on pages 12, 66, and 85).

David Haussler et al. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California ..., 1999. (Cited on page 19).

S. Heitmann and M. Breakspear. Putting the "dynamic" back into dynamic functional connectivity. *Network Neuroscience*, 2(2):150–174, 2018. (Cited on pages 12 and 85).

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. (Cited on page 23).

Frank L Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of mathematics and physics*, 20(1-4):224–230, 1941. (Cited on page 31).

Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014. (Cited on page 21).

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983. (Cited on pages 45, 56, 96, 124, and 142).

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. (Cited on page 71).

Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 158–167, 2004. (Cited on page 21).

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. (Cited on page 39).

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016. (Cited on page 13).

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. Supervised word mover's distance. *Advances in neural information processing systems*, 29, 2016. (Cited on page 14).

David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004. (Cited on pages 37 and 119).

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000. (Cited on page 26).

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pp. 427–435. PMLR, 2013. (Cited on pages 89, 90, and 116).

Tony Jebara and Risi Kondor. Bhattacharyya and expected likelihood kernels. In *Learning theory and kernel machines*, pp. 57–71. Springer, 2003. (Cited on page 22).

Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11313–11320, 2019. (Cited on page 70).

Fredrik Johansson, Vinay Jethava, Devdatt Dubhashi, and Chiranjib Bhattacharyya. Global graph kernels using geometric embeddings. In *International Conference on Machine Learning*, pp. 694–702. PMLR, 2014. (Cited on page 22).

Fredrik D Johansson and Devdatt Dubhashi. Learning with similarity functions on graphs using matchings of geometric embeddings. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 467–476, 2015. (Cited on pages 19, 22, and 68).

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. (Cited on pages 12 and 66).

Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pp. 199–201, 1942. (Cited on pages 30 and 31).

Hermann Karcher. Riemannian center of mass and so called karcher mean. *arXiv preprint arXiv:1407.2087*, 2014. (Cited on page 37).

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 321–328, 2003. (Cited on page 22).

Tatsuro Kawamoto, Masashi Tsubaki, and Tomoyuki Obuchi. Mean-field theory of graph neural networks in graph partitioning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 4366–4376, Red Hook, NY, USA, 2018. Curran Associates Inc. (Cited on page 112).

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21(70):1–73, 2020. (Cited on page 23).

Damian J Kelly and Garrett M ONeill. *The minimum cost flow problem and the network simplex solution method*. PhD thesis, Citeseer, 1991. (Cited on page 35).

Tanguy Kerdoncuff, Rémi Emonet, and Marc Sebban. Sampled gromov wasserstein. *Machine Learning*, 110(8):2151–2186, 2021. (Cited on page 52).

Tanguy Kerdoncuff, Rémi Emonet, Michaël Perrot, and Marc Sebban. Optimal tensor transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7124–7132, 2022. (Cited on page 71).

Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *arXiv preprint arXiv:2205.12156*, 2022. (Cited on page 140).

Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. (Cited on page 75).

Itay Kezurer, Shahar Z Kovalsky, Ronen Basri, and Yaron Lipman. Tight relaxation of quadratic matching. In *Computer graphics forum*, volume 34, pp. 115–128. Wiley Online Library, 2015. (Cited on page 50).

Satyanad Kichenassamy, Arun Kumar, Peter Olver, Allen Tannenbaum, and Anthony Yezzi. Gradient flows and geometric active contour models. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 810–815. IEEE, 1995. (Cited on page 141).

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. (Cited on pages 45, 55, 96, and 122).

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. (Cited on pages 13, 25, and 66).

Donald E Knuth. Sorting and searching. 1973. (Cited on page 33).

Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32, 2019. (Cited on pages 66, 71, and 106).

Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE signal processing magazine*, 34(4):43–59, 2017. (Cited on page 14).

Soheil Kolouri, Phillip E Pope, Charles E Martin, and Gustavo K Rohde. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. (Cited on page 33).

Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. *Advances in neural information processing systems*, 32, 2019. (Cited on page 34).

Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2021. (Cited on pages 14, 66, and 75).

Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. *Advances in neural information processing systems*, 29, 2016. (Cited on page 22).

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pp. 2747–2755. PMLR, 2018. (Cited on page 12).

Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015. (Cited on page 35).

Walid Krichene, Syrine Krichene, and Alexandre Bayen. Efficient bregman projections onto the simplex. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 3291–3298. IEEE, 2015. (Cited on pages 12, 99, and 129).

Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012. (Cited on pages 22 and 75).

Nils M Kriege. Deep weisfeiler-lehman assignment kernels via multiple kernel learning. *arXiv preprint arXiv:1908.06661*, 2019. (Cited on page 23).

Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. *Advances in neural information processing systems*, 29, 2016. (Cited on pages 20 and 21).

Nils M. Kriege, Matthias Fey, Denis Fisseler, Petra Mutzel, and Frank Weichert. Recognizing Cuneiform Signs Using Graph Based Methods. *arXiv:1802.05908 [cs]*, March 2018. arXiv: 1802.05908. (Cited on page 85).

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020. (Cited on pages 13, 14, 17, 20, 66, and 76).

Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 469–477. Springer, 2017. (Cited on pages 12, 66, and 85).

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pp. 957–966. PMLR, 2015. (Cited on page 14).

Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016. (Cited on pages 37, 51, 90, and 116).

Luis C Lamb, Artur Garcez, Marco Gori, Marcelo Prates, Pedro Avelar, and Moshe Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. *arXiv preprint arXiv:2003.00330*, 2020. (Cited on page 13).

Tam Le, Truyen Nguyen, Dinh Phung, and Viet Anh Nguyen. Sobolev transport: A scalable metric for probability measures with graph metrics. In *International Conference on Artificial Intelligence and Statistics*, pp. 9844–9868. PMLR, 2022. (Cited on page 141).

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. (Cited on pages 12 and 24).

Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000. (Cited on page 39).

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. (Cited on page 39).

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. PMLR, 2019. (Cited on pages 26 and 66).

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018. (Cited on page 26).

Mengyu Li, Jun Yu, Hongteng Xu, and Cheng Meng. Efficient approximation of gromov-wasserstein distance using importance sparsification. *arXiv preprint arXiv:2205.13573*, 2022. (Cited on page 52).

Ping Li, Syama Sundar Rangapuram, and Martin Slawski. Methods for sparse and low-rank recovery under simplex constraints. *arXiv preprint arXiv:1605.00507*, 2016. (Cited on page 90).

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. (Cited on page 70).

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. (Cited on pages 25 and 26).

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 142).

Weijie Liu, Chao Zhang, Jiahao Xie, Zebang Shen, Hui Qian, and Nenggan Zheng. Partial gromov-wasserstein learning for partial graph matching. *arXiv preprint arXiv:2012.01252*, 2020. (Cited on page 115).

Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pp. 4104–4113. PMLR, 2019. (Cited on page 33).

Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.09.032. (Cited on page 50).

Gaëlle Loosli, Stéphane Canu, and Cheng Soon Ong. Learning svm in krein spaces. *IEEE transactions on pattern analysis and machine intelligence*, 38(6):1204–1216, 2015. (Cited on page 19).

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020. (Cited on page 73).

László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979. (Cited on page 22).

Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018. (Cited on page 171).

Yulong Lu and Jianfeng Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33:3094–3105, 2020. (Cited on page 71).

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. (Cited on page 140).

Ronny Luss and Alexandre d'Aspremont. Support vector machine classification with indefinite kernels. *Advances in neural information processing systems*, 20, 2007. (Cited on page 19).

Enxhell Luzhnica, Ben Day, and Pietro Lio. Clique pooling for graph classification. *arXiv preprint arXiv:1904.00374*, 2019. (Cited on page 26).

Vince Lyzinski, Donniell E Fishkind, Marcelo Fiori, Joshua T Vogelstein, Carey E Priebe, and Guillermo Sapiro. Graph matching: Relax at your own risk. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):60–73, 2015. (Cited on page 50).

Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 723–731, 2019. (Cited on page 26).

Pierre Mahé and Jean-Philippe Vert. Graph kernels based on tree patterns for molecules. *Machine learning*, 75(1):3–35, 2009. (Cited on page 21).

Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Extensions of marginalized graph kernels. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 70, 2004. (Cited on page 22).

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pp. 689–696, 2009. (Cited on pages 85, 92, and 122).

Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. Got: an optimal transport framework for graph comparison. In *Advances in Neural Information Processing Systems*, pp. 13876–13887, 2019. (Cited on pages 66, 86, and 112).

Haggai Maron and Yaron Lipman. (probably) concave graph matching. In *Advances in Neural Information Processing Systems*, pp. 408–418, 2018. (Cited on pages 50 and 112).

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019a. (Cited on pages 25, 74, 76, and 77).

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019b. (Cited on pages 66 and 76).

Naoki Masuda and Renaud Lambiotte. *A Guide To Temporal Networks*, volume 6. World Scientific, 2020. (Cited on page 85).

Robert J McCann. A convexity principle for interacting gases. *Advances in mathematics*, 128 (1):153–179, 1997. (Cited on page 37).

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. (Cited on page 78).

Facundo Mémoli. On the use of gromov-hausdorff distances for shape comparison. 2007. (Cited on page 40).

Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011. (Cited on pages 14, 28, 40, 49, 52, and 138).

Facundo Mémoli and Guillermo Sapiro. Comparing point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 32–40, 2004. (Cited on page 49).

Facundo Mémoli and Guillermo Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5(3):313–347, 2005. (Cited on page 49).

Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. *Advances in Neural Information Processing Systems*, 33:2220–2231, 2020. (Cited on pages 25 and 66).

Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021. (Cited on page 12).

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. (Cited on page 142).

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018. (Cited on page 18).

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pp. 666–704, 1781. (Cited on page 29).

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124, 2017. (Cited on page 26).

Christopher Morris, Nils M Kriege, Kristian Kersting, and Petra Mutzel. Faster kernels for graphs with continuous attributes via hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1095–1100. IEEE, 2016. (Cited on page 21).

Christopher Morris, Kristian Kersting, and Petra Mutzel. Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 327–336. IEEE, 2017. (Cited on page 21).

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019. (Cited on page 25).

K.G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Sigma series in applied mathematics. Heldermann, 1988. ISBN 978-3-88538-403-8. (Cited on pages 44, 54, and 143).

Boris Muzellec and Marco Cuturi. Generalizing point embeddings using the wasserstein space of elliptical distributions. *Advances in Neural Information Processing Systems*, 31, 2018. (Cited on page 34).

Assaf Naor and Gideon Schechtman. Planar earthmover is not in l_1. *SIAM Journal on Computing*, 37(3):804–826, 2007. (Cited on page 20).

Praneeth Narayanamurthy and Namrata Vaswani. Nearly optimal robust subspace tracking. In *International Conference on Machine Learning*, pp. 3701–3709. PMLR, 2018. (Cited on page 86).

Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Universal readout for graph convolutional neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2019. (Cited on page 26).

Azade Nazi, Will Hang, Anna Goldie, Sujith Ravi, and Azalia Mirhoseini. Gap: Generalizable approximate graph partitioning framework. *arXiv preprint arXiv:1903.00614*, 2019. (Cited on page 112).

Arkadi Nemirovski and Uriel Rothblum. On complexity of matrix scaling. *Linear Algebra and its Applications*, 302:435–460, 1999. (Cited on page 35).

Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2): 209–245, 2016. (Cited on pages 21 and 104).

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002. (Cited on page 85).

Yin Cheng Ng, Nicolò Colombo, and Ricardo Silva. Bayesian semi-supervised learning with graph gaussian processes. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. (Cited on page 66).

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023. PMLR, 2016. (Cited on pages 13, 76, 85, and 86).

Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 2429–2435, 2017. (Cited on pages 14, 20, 66, and 112).

James R Norris and James Robert Norris. *Markov chains*. Number 2. Cambridge university press, 1998. (Cited on page 12).

Emmanuel Noutahi, Dominique Beaini, Julien Horwood, Sébastien Giguère, and Prudencio Tossou. Towards interpretable sparse graph representation learning with laplacian pooling. *arXiv preprint arXiv:1905.11577*, 2019. (Cited on page 26).

James B Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, 1997. (Cited on page 35).

Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018. (Cited on pages 17 and 25).

Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020. (Cited on page 12).

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. (Cited on page 13).

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. DropGNN: Random dropouts increase the expressiveness of graph neural networks. In *Advances in Neural Information Processing Systems*, 2021. (Cited on pages 73, 74, 76, and 77).

Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: A competitive, scalable and diverse community detection algorithm. In *International conference on complex networks and their applications*, pp. 229–240. Springer, 2017. (Cited on page 52).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. (Cited on page 69).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. (Cited on page 35).

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014. (Cited on pages 13 and 85).

Bruno-Edouard Perrin, Liva Ralaivola, Aurelien Mazurie, Samuele Bottani, Jacques Mallet, and Florence d'Alche Buc. Gene networks inference using dynamic bayesian networks. *Bioinformatics*, 19(suppl_2):ii138–ii148, 2003. (Cited on page 66).

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008. (Cited on pages 156 and 160).

Gabriel Peyré. Entropic approximation of wasserstein gradient flows. *SIAM Journal on Imaging Sciences*, 8(4):2323–2351, 2015. (Cited on page 36).

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11:355–607, 2019. (Cited on pages 14, 15, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 66, and 86).

Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pp. 2664–2672, 2016. (Cited on pages 14, 42, 43, 50, 51, 52, 53, 61, 62, 63, 64, 86, 89, 92, 99, 117, 121, 129, and 159).

Nikolaos Ploskas and Nikolaos Samaras. Gpu accelerated pivoting rules for the simplex algorithm. *Journal of Systems and Software*, 96:1–9, 2014. (Cited on page 35).

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018. (Cited on pages 18 and 23).

Julien Rabin, Sira Ferradans, and Nicolas Papadakis. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 4852–4856, 2014. doi: 10.1109/ICIP.2014.7025983. (Cited on page 115).

Johann Radon. On the determination of functions from their integral values along certain manifolds. *IEEE transactions on medical imaging*, 5(4):170–176, 1986. (Cited on page 33).

Alain Rakotomamonjy, Francis Bach, Stéphane Canu, et al. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008. (Cited on pages 18 and 23).

Alain Rakotomamonjy, Rémi Flamary, and Nicolas Courty. Generalized conditional gradient: analysis of convergence and applications. *arXiv preprint arXiv:1510.06567*, 2015. (Cited on page 37).

Alain Rakotomamonjy, Abraham Traoré, Maxime Berar, Rémi Flamary, and Nicolas Courty. Distance measure machines. *arXiv preprint arXiv:1803.00250*, 2018. (Cited on page 68).

Liva Ralaivola and Florence d'Alché Buc. Time series filtering, smoothing and learning using the kernel kalman filter. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 3, pp. 1449–1454. IEEE, 2005. (Cited on page 12).

Jan Ramon and Thomas Gärtner. Expressivity versus efficiency of graph kernels. In *Proceedings of the first international workshop on mining graphs, trees and sequences*, pp. 65–74, 2003. (Cited on page 21).

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. (Cited on pages 100 and 129).

Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5470–5477, 2020. (Cited on page 26).

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003. (Cited on page 18).

Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021. (Cited on page 21).

Frigyes Riesz. *Sur les opérations functionnelles linéaires*. Gauthier-Vllars, 1909. (Cited on page 18).

Antoine Rolet, Marco Cuturi, and Gabriel Peyré. Fast dictionary learning with a smoothed wasserstein loss. In *Artificial Intelligence and Statistics*, pp. 630–638. PMLR, 2016. (Cited on pages 39 and 86).

Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17(1): 4635–4666, 2016. (Cited on page 124).

Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. (Cited on page 126).

Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, pp. 3125–3132. ACM, 2020. (Cited on page 106).

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pp. 59–66. IEEE, 1998. (Cited on page 14).

Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. Chapman and Hall/CRC, 2005. (Cited on page 22).

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017. (Cited on page 12).

Asif Salim, SS Shiju, and S Sumitra. Design of multi-view graph embedding using multiple kernel learning. *Engineering Applications of Artificial Intelligence*, 90:103534, 2020. (Cited on page 23).

Roman Sandler and Michael Lindenbaum. Nonnegative matrix factorization with earth mover's distance metric for image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1590–1602, 2011. (Cited on page 39).

Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55 (58-63):94, 2015. (Cited on pages 14, 15, 28, 33, 41, and 151).

Filippo Santambrogio. {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017. (Cited on page 141).

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017. (Cited on page 12).

Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020. (Cited on pages 21, 25, and 71).

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 73).

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021. (Cited on page 73).

Akrati Saxena and Sudarshan Iyengar. Centrality measures in complex networks: A survey. *arXiv preprint arXiv:2011.07190*, 2020. (Cited on page 70).

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008. (Cited on pages 13, 24, and 85).

Meyer Scetbon, Marco Cuturi, and Gabriel Peyré. Low-rank sinkhorn factorization. In *International Conference on Machine Learning*, pp. 9344–9354. PMLR, 2021a. (Cited on pages 52, 62, 118, and 169).

Meyer Scetbon, Gabriel Peyré, and Marco Cuturi. Linear-time gromov wasserstein distances using low rank couplings and costs, 2021b. (Cited on page 52).

Morgan A Schmitz, Matthieu Heitz, Nicolas Bonneel, Fred Ngole, David Coeurjolly, Marco Cuturi, Gabriel Peyré, and Jean-Luc Starck. Wasserstein dictionary learning: Optimal transport-based unsupervised nonlinear dictionary learning. *SIAM Journal on Imaging Sciences*, 11(1):643–678, 2018. (Cited on pages 14, 39, 85, and 86).

Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019. (Cited on page 36).

Bernhard Schmitzer and Christoph Schnörr. Modelling convex shape priors and matching based on the gromov-wasserstein distance. *J. Math. Imaging Vis.*, 46(1):143–159, may 2013. ISSN 0924-9907. doi: 10.1007/s10851-012-0375-6. (Cited on page 115).

Isaac J Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, pp. 811–841, 1938. (Cited on page 19).

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pp. 583–588. Springer, 1997. (Cited on page 18).

Vivien Seguy and Marco Cuturi. Principal geodesic analysis for probability measures under the optimal transport metric. *Advances in Neural Information Processing Systems*, 28, 2015. (Cited on page 39).

Thibault Séjourné, François-Xavier Vialard, and Gabriel Peyré. The unbalanced gromov wasserstein distance: Conic formulation and relaxation. *Advances in Neural Information Processing Systems*, 34:8766–8779, 2021. (Cited on pages 113 and 115).

Thibault Séjourné, Gabriel Peyré, and François-Xavier Vialard. Unbalanced optimal transport, from theory to numerics. *arXiv preprint arXiv:2211.08775*, 2022. (Cited on pages 115 and 141).

Linda G Shapiro, George C Stockman, et al. *Computer vision*, volume 3. Prentice Hall New Jersey, 2001. (Cited on page 12).

Alexander Shekhovtsov and Václav Hlaváč. A distributed mincut/maxflow algorithm combining path augmentation and push-relabel. *International journal of computer vision*, 104(3): 315–342, 2013. (Cited on page 83).

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009. (Cited on pages 21, 66, and 104).

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011. (Cited on pages 20, 21, 66, and 75).

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. (Cited on page 99).

Kilho Shin and Tetsuji Kuboyama. A generalization of haussler's convolution kernel: mapping kernel. In *Proceedings of the 25th international conference on Machine learning*, pp. 944–951, 2008. (Cited on page 20).

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020. (Cited on page 12).

David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30:83–98, 2013. (Cited on page 66).

Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020. (Cited on page 104).

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018. (Cited on page 18).

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. (Cited on page 35).

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. (Cited on page 142).

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. (Cited on page 141).

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (ToG)*, 34 (4):1–11, 2015. (Cited on page 38).

Justin Solomon, Gabriel Peyré, Vladimir G Kim, and Suvrit Sra. Entropic metric alignment for correspondence problems. *ACM Transactions on Graphics (TOG)*, 35(4):1–13, 2016. (Cited on pages 14, 51, and 89).

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. (Cited on page 24).

Matthew Staib, Sebastian Claici, Justin M Solomon, and Stefanie Jegelka. Parallel streaming wasserstein barycenters. *Advances in Neural Information Processing Systems*, 30, 2017. (Cited on page 37).

Douglas Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004. (Cited on page 100).

X Yu Stella and Jianbo Shi. Multiclass spectral clustering. In *null*, pp. 313. IEEE, 2003. (Cited on page 99).

Todd Andrew Stephenson. An introduction to bayesian network theory and usage. Technical report, Idiap, 2000. (Cited on page 17).

Karl-Theodor Sturm. The space of spaces: curvature bounds and gradient flows on the space of metric measure spaces. *arXiv preprint arXiv:1208.0434*, 2012. (Cited on pages 14, 28, 40, 41, 48, 49, 59, and 138).

Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *Journal of chemical information and computer sciences*, 43(6):1906–1915, 2003. (Cited on pages 12, 99, and 129).

Robert E Tarjan. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Mathematical Programming*, 78(2):169–177, 1997. (Cited on pages 35 and 51).

Savannah Thais, Paolo Calafiura, Grigorios Chachamis, Gage DeZoort, Javier Duarte, Sanmay Ganguly, Michael Kagan, Daniel Murnane, Mark S Neubauer, and Kazuhiro Terao. Graph neural networks in particle physics: Implementations, innovations, and challenges. *arXiv preprint arXiv:2203.12852*, 2022. (Cited on page 12).

Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern recognition*. Elsevier, 2006. (Cited on pages 17 and 18).

Matthew Thorpe, Serim Park, Soheil Kolouri, Gustavo K Rohde, and Dejan Slepčev. A transportation lp distance for signal analysis. *Journal of mathematical imaging and vision*, 59(2):187–210, 2017. (Cited on page 14).

Alexis Thual, Huy Tran, Tatiana Zemskova, Nicolas Courty, Rémi Flamary, Stanislas Dehaene, and Bertrand Thirion. Aligning individual brains with fused unbalanced gromov-wasserstein. *arXiv preprint arXiv:2206.09398*, 2022. (Cited on pages 115 and 141).

Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler–lehman graph kernels. In *Advances in Neural Information Processing Systems*, pp. 6436–6446. Curran Associates, Inc., 2019. (Cited on pages 14, 21, 66, 75, 86, and 112).

Alexander Tong, Guillaume Huguet, Dennis Shung, Amine Natik, Manik Kuchroo, Guillaume Lajoie, Guy Wolf, and Smita Krishnaswamy. Embedding signals on graphs with unbalanced diffusion earth mover's distance. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5647–5651. IEEE, 2022. (Cited on page 141).

Alexander Y Tong, Guillaume Huguet, Amine Natik, Kincaid MacDonald, Manik Kuchroo, Ronald Coifman, Guy Wolf, and Smita Krishnaswamy. Diffusion earth mover's distance and distribution embeddings. In *International Conference on Machine Learning*, pp. 10336–10346. PMLR, 2021. (Cited on page 141).

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021. (Cited on page 140).

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001. (Cited on page 89).

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. (Cited on page 78).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. (Cited on page 24).

Titouan Vayer. A contribution to optimal transport on incomparable spaces. *arXiv preprint arXiv:2011.04447*, 2020. (Cited on pages 41, 42, and 60).

Titouan Vayer, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284. PMLR, 2019a. (Cited on pages 14, 50, 57, 59, 60, 62, 63, 64, 67, 69, 75, 76, 89, 92, 104, 129, 138, 149, and 150).

Titouan Vayer, Rémi Flamary, Romain Tavenard, Laetitia Chapel, and Nicolas Courty. Sliced gromov-wasserstein. In *NeurIPS 2019-Thirty-third Conference on Neural Information Processing Systems*, volume 32, 2019b. (Cited on page 52).

Titouan Vayer, Laetitia Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. Fused gromov-wasserstein distance for structured objects. *Algorithms*, 13(9):212, 2020. (Cited on pages 14, 15, 29, 39, 57, 58, 59, 60, 62, 67, 138, and 149).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. (Cited on pages 24, 78, 81, and 83).

Jean-Philippe Vert. The optimal assignment kernel is not positive definite. *arXiv preprint arXiv:0801.4061*, 2008. (Cited on page 20).

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009. (Cited on pages 14, 15, 28, 66, and 138).

Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021. (Cited on pages 31, 32, 33, and 48).

Cédric Vincent-Cuaz, Titouan Vayer, Rémi Flamary, Marco Corneli, and Nicolas Courty. Online graph dictionary learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10564–10574. PMLR, 18–24 Jul 2021. (Cited on pages 15, 16, 29, 43, 54, 62, 64, 67, 84, 122, and 129).

Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. Semi-relaxed gromov-wasserstein divergence and applications on graphs. In *International Conference on Learning Representations*, 2022a. (Cited on pages 16, 67, 83, and 111).

Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. Semi-relaxed gromov-wasserstein divergence for graphs classification. In *Colloque GRETSI 2022-XXVIIIème Colloque Francophone de Traitement du Signal et des Images*, 2022b. (Cited on pages 16, 83, and 111).

Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. Template based graph neural network with optimal transport distances. In *Advances in Neural Information Processing Systems*, 2022c. (Cited on pages 15, 65, and 105).

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010. (Cited on page 124).

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010. (Cited on pages 20, 22, and 104).

Stefan Vlaski, Hermina P Maretić, Roula Nassif, Pascal Frossard, and Ali H Sayed. Online graph learning from sequential data. In *2018 IEEE Data Science Workshop (DSW)*, pp. 190–194. IEEE, 2018. (Cited on page 85).

Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002. (Cited on page 22).

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007. (Cited on page 26).

Junshan Wang, G. Song, Y. Wu, and Liang Wang. Streaming graph neural networks via continual learning. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020. (Cited on page 85).

Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018. (Cited on page 13).

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018. (Cited on page 12).

Yuchung J Wang and George Y Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987. (Cited on page 96).

Jonathan Weed and Francis Bach. Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance. *Bernoulli*, 25(4A):2620–2648, 2019. (Cited on page 33).

Jonathan Daniel Weed. *Statistical problems in transport and alignment*. PhD thesis, Massachusetts Institute of Technology, 2019. (Cited on page 33).

William W.S. Wei. 458Time Series Analysis. In *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2: Statistical Analysis*. Oxford University Press, 2013. doi: 10.1093/oxfordhb/9780199934898.013.0022. (Cited on page 12).

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968. (Cited on page 20).

Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001. (Cited on page 17).

Alan Geoffrey Wilson. The use of entropy maximising models, in the theory of trip distribution, mode split and route split. *Journal of transport economics and policy*, pp. 108–126, 1969. (Cited on page 35).

Jun Wu, Jingrui He, and Jiejun Xu. Net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 406–415, 2019. (Cited on page 26).

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. (Cited on pages 12, 13, 15, 18, 23, 25, 66, 85, and 141).

Xinyue Xia, Gal Mishne, and Yusu Wang. Implicit graphon neural representation. *arXiv preprint arXiv:2211.03329*, 2022. (Cited on page 142).

Yu Xie, Chuanyu Yao, Maoguo Gong, Cheng Chen, and A Kai Qin. Graph convolutional networks with multi-level coarsening for graph classification. *Knowledge-Based Systems*, 194:105578, 2020a. (Cited on page 26).

Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for computing exact wasserstein distance. In Ryan P. Adams and Vibhav Gogate (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 433–453. PMLR, 22–25 Jul 2020b. (Cited on page 117).

Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable gromov-wasserstein learning for graph partitioning and matching. *Advances in neural information processing systems*, 32: 3052–3062, 2019a. (Cited on pages 43, 46, 52, 53, 57, 62, 63, 67, 86, 93, 113, 128, and 138).

Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*, pp. 6932–6941. PMLR, 2019b. (Cited on pages 52, 62, 117, and 125).

Hongteng Xu, Dixin Luo, Lawrence Carin, and Hongyuan Zha. Learning graphons via structured gromov-wasserstein barycenters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10505–10513, 2021a. (Cited on pages 43, 53, 112, and 142).

Hongteng Xu, Peilin Zhao, Junzhou Huang, and Dixin Luo. Learning graphon autoencoders for generative graph modeling. *arXiv preprint arXiv:2105.14244*, 2021b. (Cited on page 142).

Hongtengl Xu. Gromov-wasserstein factorization models for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6478–6485, 2020. (Cited on pages 43, 53, 67, 69, 78, 86, 89, 99, 105, 113, 121, and 129).

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018. (Cited on pages 24, 76, and 80).

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019c. (Cited on pages 24, 25, 66, 68, 71, 73, 76, 77, 80, 83, 105, and 140).

Junchi Yan, Xu-Cheng Yin, Weiyao Lin, Cheng Deng, Hongyuan Zha, and Xiaokang Yang. A short survey of recent advances in graph matching. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pp. 167–174, 2016. (Cited on page 112).

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015. (Cited on pages 12, 20, 66, 75, 85, 99, and 129).

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015. (Cited on pages 12 and 125).

Peng Yang, Peilin Zhao, and Xin Gao. Bandit online learning on graphs via adaptive optimization. International Joint Conferences on Artificial Intelligence, 2018. (Cited on page 85).

Wei Ye, Jiayi Yang, Sourav Medya, and Ambuj Singh. Incorporating heterophily into graph neural networks for graph classification. *arXiv preprint arXiv:2203.07678*, 2022. (Cited on page 140).

Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 555–564, 2017. (Cited on pages 12 and 125).

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018. (Cited on pages 26, 66, and 76).

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017. (Cited on page 25).

Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Concept drift and anomaly detection in graph streams. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 06 2017. (Cited on page 85).

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pp. 7354–7363. PMLR, 2019a. (Cited on page 142).

Kelvin Shuangjian Zhang, Gabriel Peyré, Jalal Fadili, and Marcelo Pereyra. Wasserstein control of mirror langevin monte carlo. In *Conference on Learning Theory*, pp. 3814–3841. PMLR, 2020. (Cited on pages 118 and 169).

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-second AAAI conference on artificial intelligence*, 2018. (Cited on pages 26 and 66).

Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019b. (Cited on page 70).

Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022. (Cited on pages 23 and 140).

Feng Zhou and Fernando De la Torre. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 127–134. IEEE, 2012. (Cited on page 112).

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. (Cited on page 23).

Hongmin Zhu, Fuli Feng, Xiangnan He, Xiang Wang, Yan Li, Kai Zheng, and Yongdong Zhang. Bilinear graph neural network with neighbor interactions. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165. (Cited on page 24).

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020. (Cited on page 140).