N° National de Thèse : 2021LYSEN090

THESE DE DOCTORAT DE L'UNIVERSITE DE LYON

opérée par
l'École Normale Supérieure de Lyon

École Doctorale N°512
École doctorale en Informatique et Mathématiques de Lyon

Discipline : Informatique

Soutenue publiquement le 10/12/2021 par :

DOMINIQUE BARBE

Diffusion-Wasserstein Distances for Attributed Graphs

Distances de Diffusion-Wasserstein pour les graphes attribués

Devant le jury composé de :

M.	Rémi Flamary	MCF	*Ecole Polytechnique*	Rapporteur
Mme.	Christine Guillemot	DR	*IRISA*	Rapporteure
Mme.	Julie Digne	CR	*CNRS, LIRIS*	Examinatrice
M.	Philippe Ciblat	PU	*Telecom Paris*	Examinateur
Mme	Florence Forbes	DR	*Inria*	Examinatrice
M.	Paulo Gonçalves	DR	*ENS Lyon, Inria, LIP*	Directeur de thèse
M.	Pierre Borgnat	DR	*ENS Lyon, CNRS, IXXI*	Co-encadrant
M.	Marc Sebban	PU	*UJM, Laboratoire Hubert Curien*	Co-encadrant

Laboratoire de l'Informatique et
du Parallélisme
ENS de Lyon
46 allée d'Italie
69364 Lyon cedex 07

ED InfoMaths (ED 512)
Bât. Blaise Pascal
7 avenue Jean Capelle
69621 Villeurbanne Cedex

Remerciements

Je tiens à remercier mes encadrants pour ces 3 années à travailler ensemble, ainsi que Titouan Vayer, Rémi Gribonval et Rémi Flamary pour leurs précieux conseils.

Je tiens aussi à remercier ma famille, mes amis, mon groupe de jeu de rôles et toutes les personnes formidable que j'ai rencontrées à Lyon pour tout le temps passé ensemble.

Finallement, je veux remercier la personne avec laquelle je vais partager une vie de voyage et de découverte :)

Contents

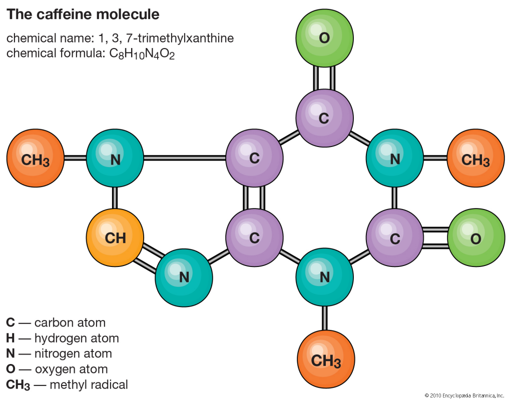
Introduction

A molecule, a social network, a road network, the internet, all these objects can be modelled as *attributed graphs*: a collection of identifiable objects (atoms, persons, towns, IPv4/6 addresses), some being linked to others. An attributed graph combines two modalities of information: structure and feature. Structure is a set of links between nodes, also called a *graph*. For instance the atoms are linked by covalent bounds in a molecule, people by willingly linking themselves in a social network, towns by roads, IPv4/6 addresses by cables. Feature is a description of each node. For instance an atom in a molecule has a weight, a charge, a boiling point, etc; a person has an age, a gender, a height, a self-description, interests, etc; a town has a number of inhabitants, an entry speed limit, etc; an IPv4/6 address has an ISP, an associated terminal, etc. Examples of attributed graphs are given in Figure 1.

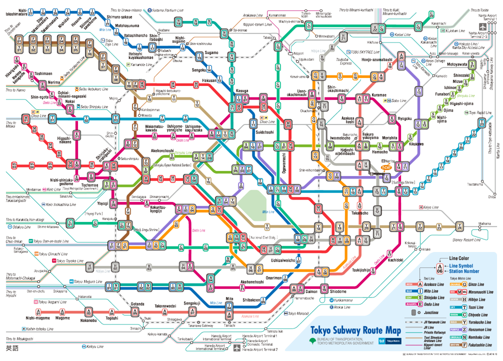
Studying attributed graphs is a hard task. Because they lack regularity in their description, they have been the topic of less study than other structured data. In comparison, images are almost figured out nowadays for instance. While they can be seen through the prism of attributed graphs (an image is a set of pixels, linked together in a grid fashion, each pixel being described by a RGB triplet), specific methods can take advantage of their regularity, namely a rectangular grid shape, with close pixels expected to be often similar. Powerful image-specific methods include SIFT descriptors [1] or convolutional neural networks [2, 3], and very large datasets [4, 5] have been created.

Sadly a large variety of data simply does not possess such regularity that images can display. But this lack of regularity does not prevent examination. The study of graphs by themselves is a whole active field of study [6]. For attributed graphs, one of the main theoretical framework is Graph Signal Processing [7]. Indeed, an attributed graph can be seen as a signal on a graph, i.e. a function defined on each node of a graph. This framework is especially relevant considering that the features can vary: for instance people change with time. So it makes sense to detach the attributes from the graph and regard them as a signal.

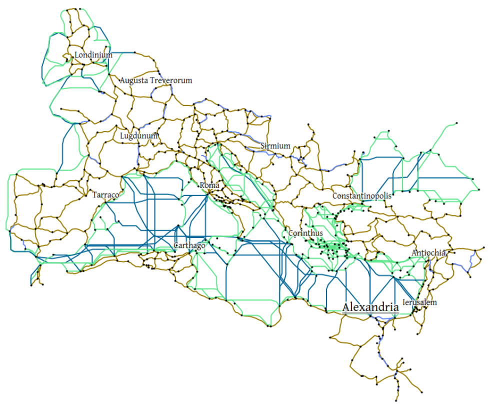
One of the challenges in handling attributed graphs is comparing them, measuring their (dis-)similarity. Comparing graphs allows plugging them in a lot of machine learning algorithms, thus enabling easy classification, regression, clustering, etc. Comparisons can be direct, or indirect through kernel-based


The caffeine molecule
chemical name: 1, 3, 7-trimethylxanthine
chemical formula: $C_8H_{10}N_4O_2$
C — carbon atom
H — hydrogen atom
N — nitrogen atom
O — oxygen atom
CH_3 — methyl radical
© 2010 Encyclopædia Britannica, Inc.

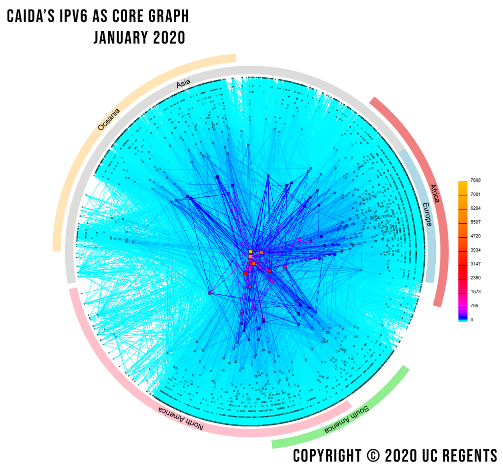
(a) The caffeine molecule: nodes are atoms, links are covalent bounds, atoms are described by their name, atomic number, etc. Image from Encyclopedia Britannica, Inc.



(b) The Tokyo subway map: nodes are stations, links are railways, stations are described by their name, frequentation, etc. Image from Bureau of Transportation, Tokyo Metropolitan Government.



(c) The ORBIS model (the Standford Geospatial Network Model of the Roman World): nodes are cities, links are roads, river sections and naval connections, cities are described by their number of inhabitants, their production, etc.



(d) The AS Core graph 2020: nodes are IPv6 addresses, links encode connectivity between addresses; addresses are ranked by their transit degree, ISP, prefix, etc. Image from Center for Applied Internet Data Analysis

Figure 1: Examples of attributed graphs.

metrics or euclidean embeddings for instance. Numerous distances between simple graphs have been proposed: for instance the Graph Edit Distance [8] counts the number of modifications required to transform a graph into another. The Graph Diffusion Distance [9] looks at the maximum of difference between the diffusion operators of two graphs of the same sizes, or the Graph Spectral Distance [10] that computes the $\mathcal{L}^1$ distance between the graph's spectra. Graphs kernels that take into account both structural and feature informations have achieved a tremendous success during the past years to address graph classification tasks (see the most recent survey [11]), *e.g.* using Support Vector Machines. As an example, the Weisfeiler-Lehman kernel and its variations [12–14] works repeatedly aggregating and compressing features around each node. Finally, the Fused Gromov-Wasserstein distance [15] works by merging two transport distances on graphs and vector-valued data; because the information it provides is much richer than a distance, and because it relates to our work, it will be studied later in this manuscript (see Section 2.2.2).

In our work, we looked at a more detailed way of comparing attributed graphs. More than a single number (to get a notion of distance or similarity), we considered the notion of *mapping*: a node-to-node relation between two such objects. A mapping gives insights as to which nodes are similar between the two graphs. It can also be used to transport a quantity from one graph to another. For instance, a classification of the nodes of one graph could be transported to the other. This would save the task of labelling from scratch the nodes of the other graph; this has applications in situations when, for instance, a graph changes with time, as a new classification of the nodes could be produced automatically from a previous one. Such graphs include the Wikipedia encyclopedia, a country's road network or a scientific paper citation network for instance.

We studied this problem through the lens of *Transportation Theory* [16], the study of optimally moving distributions onto others. Attributed graphs are considered as discrete distributions, each node being given a weight or probability.

Our contributions cover the following algorithmic, theoretical and experimental aspects:

- We introduce the notion of *Diffusion-Wasserstein Distance* (`DW`), a modification of the standard transport distance that incorporates structure information.

- We discuss the implementation of the diffusion procedure, including a new bound on the approximation error of real functions by Chebychev polynomials.

- We present and analyse a scheme to optimise the hyper-parameter of `DW`, inspired by the notion of triplet-loss.

- We perform an extensive comparison of transport methods that can handle attributed graphs.

The rest of this manuscript is organised as follows:

- In Chapter 1, we lay down the theoretical foundations necessary for the understanding of the contributions.
- In Chapter 2, we review the literature on distances for (attributed) graphs, mostly transport-based ones.
- In Chapter 3, we define the *Diffusion-Wasserstein Distance* (`DW`), discuss its theoretical properties and variants. This chapter corresponds to works published at the CAP conference [17] and the ECML/PKDD conference [18].
- In Chapter 4, we detail the implementation of `DW`. More precisely, we detail a Chebychev approximation of the Diffusion process, and a procedure to select the diffusion time. This chapter corresponds to two papers: one written with the help of Sibylle Marcotte during her internship currently at the submission stage [19], and one accepted at the ICTAI'21 conference [20].
- In Chapter 5, we perform various experiments to study `DW`, and compare it to other methods.

Publications

[21] D. Barbe, P. Borgnat, P. Gonçalves, and M. Sebban, "Transport optimal sous contrainte de régularité pour l'adaptation de domaines entre graphes avec attributs," in *GRETSI 2019-XXVIIème Colloque francophone de traitement du signal et des images*, pp. 1–4, 2019, held from August 23 to 26 in 2019.

[17] A. Barbe, M. Sebban, P. Gonçalves, P. Borgnat, and R. Gribonval, "Transport Optimal entre Graphes exploitant la Diffusion de la Chaleur," in *CAP 2020-Conférence sur l'Apprentissage Automatique*, 2020, held from June 23 to 26 in 2020.

[18] A. Barbe, M. Sebban, P. Gonçalves, P. Borgnat, and R. Gribonval, "Graph diffusion Wasserstein distances," in *ECML/PKDD (2)*, vol. 12458 of *Lecture Notes in Computer Science*, pp. 577–592, Springer, 2020, held from September 14 to 18 in 2020.

[19] S. Marcotte, A. Barbe, R. Gribonval, T. Vayer, M. Sebban, P. Borgnat, and P. Gonçalves, "Fast multiscale diffusion on graphs," in *CoRR, ArXiV preprint*, vol. abs/2104.14652, 2021.

[20] A. Barbe, P. Gonçalves, M. Sebban, P. Borgnat, R. Gribonval, and T. Vayer, "Optimization of the Diffusion Time in Graph Diffused-Wasserstein Distances: Application to Domain Adaptation," in *IEEE International Conference on Tools with Artificial Intelligence*, 2021.

Notations

For attributed graphs:

- $\mathcal{G}$ is a graph.
- V is the (finite) set of vertices (also called nodes) of a graph.
- $E \in \mathcal{P}(V^2)$ is the set of edges (also called links) of a graph.
- $\mathbf{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix of a graph of size N.
- $\mathbf{D} \in \mathbb{N}^{N \times N}$ is the degree matrix of a graph of size N.
- $\mathbf{L} \in \mathbb{R}^{N \times N}$ is the Laplacian matrix of a graph of size N.
- $\mathbf{C} \in \mathbb{R}_+^{N \times N}$ is the shortest-path matrix of a graph of size N.
- $X \in \mathbb{R}^{N \times r}$ is the matrix representation of the r-dimensional feature vectors of a graph of size N.
- $l \in [1, K]^N$ is a vector containing one of K possible labels (or class) for each node of a graph of size N.
- Any mathematical quantity related to a graph can be superscripted to denote the specific graph they refer to. In particular, $.^s$ and $.^t$ are used to denote respectively the source and target graph.

For Optimal Transport:

- δ_x is the Dirac function in x.
- μ and ν are discrete probability distributions.
- $\mathcal{X}$ and $\mathcal{Y}$ are the feature spaces where the distributions μ and ν are supported on.
- $a \in \mathbb{R}_+^m$ and $b \in \mathbb{R}_+^n$ are the weights of the points μ and ν are supported on.
- $M \in \mathbb{R}_+^{m \times n}$ is the matrix representation of the pairwise distances between any two sets of points $\{x_i\}$ and $\{y_j\}$: $\forall i \in [\![1, m]\!], \forall j \in [\![1, n]\!], M_{i,j} = \|x_i - y_j\|$.
- $\gamma \in \mathbb{R}_+^{m \times n}$ is a transport map between two discrete distributions of respective sizes m and n.
- $\Pi(a, b)$ is the set of all possible transport maps between two discrete distributions with point weights a and b.
- $\mathtt{W}$ is the Wasserstein distance.
- $\mathtt{GW}$ is the Gromov-Wasserstein distance.

- `FGW` is the Fused-Gromov-Wasserstein distance.
- `DW` is the Diffusion-Wasserstein distance.

Others:

- $\mathbb{1}_n$ is a vector of size n where every coordinate is 1.
- $\tau \in \mathbb{R}_+$ is the diffusion time of a diffusion process.
- $\langle \cdot | \cdot \rangle_F$ is the Frobenius inner product: $\forall x \in \mathbb{R}^r, \forall y \in \mathbb{R}^r, \langle x|y \rangle_F = \sum_{i=1}^r x_i y_i$. This notation naturally extends to matrices, by considering the sum along all possible dimensions.

Chapter 1

Preliminaries

Read the label before use. Keep close of reach of children.

In this chapter, we lay down the theoretical foundations related to our contributions and required for the remainder of this manuscript. Section 1.1 describes *attributed graphs*. They are a very generic model of data that captures a lot of different items. We also give a short introduction to *Graph Signal Processing*, the common theoretical framework used to study these data. Section 1.2 describes the *Heat Diffusion* process in graphs. Section 1.3 gives a simple overview of the field of *Optimal Transport*.

1.1 Graphs, Attributed Graphs, Graph Signal Processing

1.1.1 Graphs

A *graph* $\mathcal{G}$ is a model to represent structured data. It is a collection of *vertices* V and *edges* $E \subset V^2$ between them (also called *nodes* and *links*).

A graph of size N nodes can be conveniently represented by its *adjacency matrix* $\mathbf{A} \in \{0,1\}^{N\times N}$. Its elements $\mathbf{A}_{i,j}$ indicates the presence (≥ 0) or absence ($= 0$) of a link from node i to node j. A graph is said to by *undirected* if its adjacency matrix is symmetric, i.e. if for every link from node i to j there is also a link from node j to i, *directed* otherwise. A graph is said to be *unweighted* if its adjacency matrix is made of only 0 or 1 elements, *weighted* otherwise. In the following, we will only look at unweighted, undirected graphs.

From the adjacency matrix, the *degree matrix* $\mathbf{D}$ can be derived. It counts the number of edges adjacent to each vertex. It has value 0 outside the main diagonal, and each diagonal element $\mathbf{D}_{i,i} = d_i$ is the sum of the i-th row (or equivalently column if undirected) of $\mathbf{A}$.

Finally the *combinatorial Laplacian Matrix* $\mathbf{L}$ of a graph is defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$. It is a central tool in graph study, especially in the case of graph

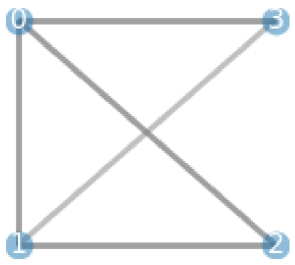


Figure 1.1: A simple graph, consisting of 4 nodes and 5 links.

signal processing as we will see in Section 1.1.3. Note that there exist other notions of Laplacian, such as the normalized Laplacian $\mathbf{L}^{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ or the random walk Laplacian $\mathbf{L}^{rw} = \mathbf{D}^{-1}\mathbf{L}$. They can also be used to find useful properties of a given graph.

As an illustration, a toy graph is represented in Figure 1.1. Its associated matrices are:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{pmatrix} \tag{1.1}$$

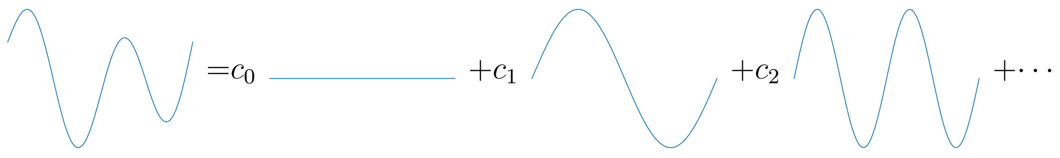
1.1.2 Attributed Graphs

A *signal* x on a graph is a function defined on the nodes of a graph [7, 22]. A pair consisting of a graph and a signal is called an *attributed graph* [23], and in this case the value of the signal at a node is called the *attribute* of the node. These two terms, graph signal and attributed graphs, are two different sides of the same coin. The term signal tends to be used more when multiple signals on the same graph will be considered, like time-dependent signals; the term attributes tends to be used more when this function on nodes is fixed. For instance, when considering the Wikipedia encyclopedia as a graph, with vertices being pages and edges hyperlinks, the text content of a page can be viewed as a signal on the graph. This model is appropriate because the text content of a webpage changes over time (especially with the Wikipedia project!).

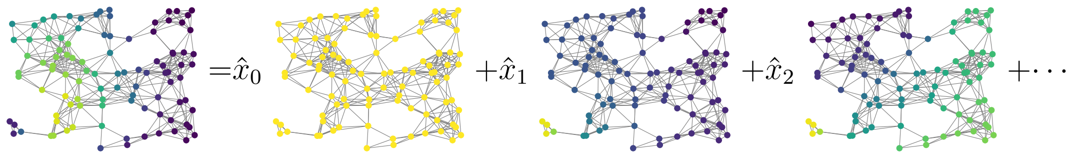
In the following, we will only consider scalar or vector-valued signals, but in practice signals can be defined on any space. It is also possible to consider signals defined on edges, but this falls outside the scope of this manuscript.

1.1.3 Graph Signal Processing

Graph Signal Processing [7, 22] is a recent field of study. As graphs are not regular domains, standard signal processing techniques cannot be employed directly on the graph signals, and basic operations have to be redefined: how



(a) Fourier decomposition of a 1d signal.



(b) Fourier decomposition of a graph signal. The signal's value at each node is color-coded.

Figure 1.2: Comparison of 1d Fourier transform and graph Fourier transform.

to shift, dilate or subsample a signal? What does it mean for a signal to be smooth, localized or noisy?

An important tool in the study of graph signals is the graph's Laplacian. It is used to extend the notion of *Fourier basis*, *frequencies* on graphs.

Let $\mathcal{G}$ be a graph of size N nodes, represented by its Laplacian matrix $\mathbf{L}$. As we are only considering undirected graph, $\mathbf{L}$ is a real symmetric matrix, and thus can be diagonalized $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{\Lambda} = \mathtt{diag}(\lambda_i)_{1\leq i\leq N}$ are the eigenvalues of $\mathbf{L}$, and the columns of $\mathbf{U} = (u_1 \cdots u_N)$ are its eigenvectors. The first eigenvector u_0 is constant, the second one u_1 varies slowly (positive on one "block", negative on the other), and the next ones vary quicker and quicker. Thus, these eigenvectors mimic the trigonometric functions that form the usual Fourier basis. The eigenvalues are then the analogous of the *frequencies* [24].

The eigenvalues verify a number of properties [24]. They are all nonnegative: $\forall i \in [\![1, n]\!] \lambda_i \geq 0$. The first K eigenvalues are zero, where K is the number of connected components in the graph (so a connected graph has exactly 1 null eigenvalue). For the specific case of the normalised Laplacian $\mathbf{L}^{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, the largest eigenvalue is always 2: $\lambda_n = 2$.

Let $x \in \mathbb{R}^N$ be a real-valued signal on that graph; its *graph Fourier transform* is the projection of x on the eigenbasis of $\mathbf{L}$:

$$\hat{x} = \mathbf{U}^T x = \left(\sum_{i=1}^{N} \mathbf{U}_{i,j} x_i\right)_{1\leq j\leq N} \tag{1.2}$$

Figure 1.2 illustrates this concept of graph Fourier transform: like 1d-signals, graph signals are expressed as a linear combination in the (graph) Fourier basis.

Likewise, the *inverse graph Fourier transform* of $\hat{x}$ reconstructs the signal:

$$x = \mathbf{U}\hat{x} = \left(\sum_{j=1}^{N} \mathbf{U}_{i,j} \hat{x}_j\right)_{1\leq i\leq N} = (\langle u_i^* | \hat{x}\rangle)_{1\leq i\leq N}\,. \tag{1.3}$$

Graph filters can be defined by lifting real functions $f : \mathbb{R} \to \mathbb{R}$ to symmetric matrices through the eigen-decomposition: $f(\mathbf{L}) := \mathbf{U}\mathtt{diag}(f(\lambda_i))\mathbf{U}^T$. When

$f(t) = t^k$ for some integer k, this yields $f(\mathbf{L}) = \mathbf{L}^k$, hence the definition matches with the intuition when f is polynomial or analytic.

1.1.4 Graph Convolutional Neural Networks

Graph Convolutional Neural Networks (abbreviated GCN) are a set of methods that try to adapt standard convolutional neural networks architectures on attributed graphs. Butt while traditional media enjoy a very regular structure (an image can be represented as a grid, and sound as a 1D signal for instance), graphs do not. The difficulty therefore lies in the definition of a meaningful *convolution* operator, capable of aggregating features in a meaningful fashion.

Most GCN follow the same general structure [25–29]. A single layer of the networks takes as input the node features $X \in \mathbb{R}^{n \times d^{in}}$ (where X_i is the attribute of node i) and a structural representation of the graph $\mathbf{G}$ (such as the Laplacian) used to aggregate the features in a proper neighbourhood around each node. It outputs new features $Y \in \mathbb{R}^{n \times d^{out}}$:

$$Y = f(X, \mathbf{G}). \tag{1.4}$$

A multi-layer GCN simply composes such layers:

$$\mathbf{H}^L = f(\mathbf{H}^{L-1}, \mathbf{G}) = \cdots = f(f(...f(\mathbf{H}^{\mathbf{0}}, \mathbf{G}), \mathbf{G}), \mathbf{G}) \tag{1.5}$$

An example of layer is the following one, introduced in [25]:

$$Y = \sigma(\bar{\mathbf{D}}^{-\frac{1}{2}} \bar{\mathbf{A}} \bar{\mathbf{D}}^{-\frac{1}{2}} X \mathbf{W}), \tag{1.6}$$

where σ is an activation function such as $\mathrm{ReLU} = \max(\cdot, 0)$, $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ is the adjacency matrix of the graph augmented with self-loops, $\bar{\mathbf{D}}$ is the corresponding degree matrix and $\mathbf{W} \in \mathbb{R}^{d^{in} \times d^{out}}$ are the layer's trainable weights.

The node-level information produced by a GCN can be used to perform a variety of tasks, such as node classification by trying to predict node labels [25–29]. Alternatively, the features for all nodes from all layers can be combined with a pooling/readout operation to produce a new information at the graph level (or *graph embedding*) that can be used for other tasks, such as graph classification [27, 28, 30, 31].

1.2 The Heat Diffusion in Graphs

At the core of our work is the *heat diffusion* process in graphs [32]. It is an analogous of the heat diffusion process in a physical system, where heat transfers over time from high temperature to low temperature points. It has application in a very wide array of domains, from materials physics [33] or economics [34] to quantum mechanic [35]. It has also found uses in computer science. In image analysis, it can be used to perform edge detection [36], and is one of the building blocks of the scale space theory [37] that handles images at multiple scales. In machine learning, it can be used to improved Graph

Neural Networks [38], or to learn an underlying graph structure that explains well a dataset [39] for instance.

Consider a graph $\mathcal{G} = (V, E)$ of laplacian $\mathbf{L}$, and a 1-d signal x defined on its nodes. Diffusion is a dynamical process by which information travels in the graph by following its structure. The heat diffusion process of x in $\mathcal{G}$ is described by the following first-order homogeneous differential equation:

$$\forall i, \frac{dx_i}{dt} = \sum_{j \text{s.t.} (i,j)\in E} (x_i - x_j) \qquad \text{(rate of change)} \qquad (1.7)$$

$$x(0) = v \qquad \text{(initial condition)} \qquad (1.8)$$

The rate of change at a node is equal to the sum for every adjacent node of the difference between the node's values. That is "colder" nodes (lower value than their neighbours) will be "filled in" (the signal's value will increase) and vice-versa. The speed of diffusion at a node being proportional to the difference with its neighbours, it slows down over time.

This equation can be rewritten in a simpler fashion using the Laplacian matrix:

$$\frac{dx}{dt} = \mathbf{L}x \qquad (1.9)$$

$$x(0) = x. \qquad (1.10)$$

This formulation mirrors the heat equation in a physical system. Consider an open subset $U \subset \mathbb{R}^n$ and a function $u : U \times \mathbb{R}_+ \mapsto \mathbb{R}$ describing the temperature at any point in an object (represented by U) at some time $t \geq 0$. We then have that u is a solution of the heat equation if and only if it verifies:

$$\frac{du}{dt} = \Delta u, \qquad (1.11)$$

where Δ is the Laplacian of the function $u(,t) : U \mapsto \mathbb{R}$.

The heat diffusion equation for graphs 1.9 admits a closed-form solution, which uses the *matrix exponential*:

$$x(\tau) = \exp(-\tau\mathbf{L})x. \qquad (1.12)$$

The matrix exponential is defined through the eigen-decomposition like a graph filter: the signal is multiplied by $\exp(-\tau\lambda_j)$ at each frequency. Thus all but the first frequencies are attenuated as τ increases, with higher frequencies decaying quicker. The effect of the exponential filter on the frequencies is visible in Figure 1.3, illustrating this low-pass behaviour. When $\tau \to \infty$ the only frequency remaining is the first one ($\lambda_1 = 0$), which corresponds to a constant eigenvector, and is associated to the average signal value ($\hat{x}_0 = \sum_i x_i$).

Two illustrations of the diffusion process are given below. Figure 1.4 displays the diffusion of a Dirac function in a graph. The graph is a road network, and nodes are positioned according to the geographical positions of the intersections. We can see that with τ increasing, the signal diffuses farther in the network.

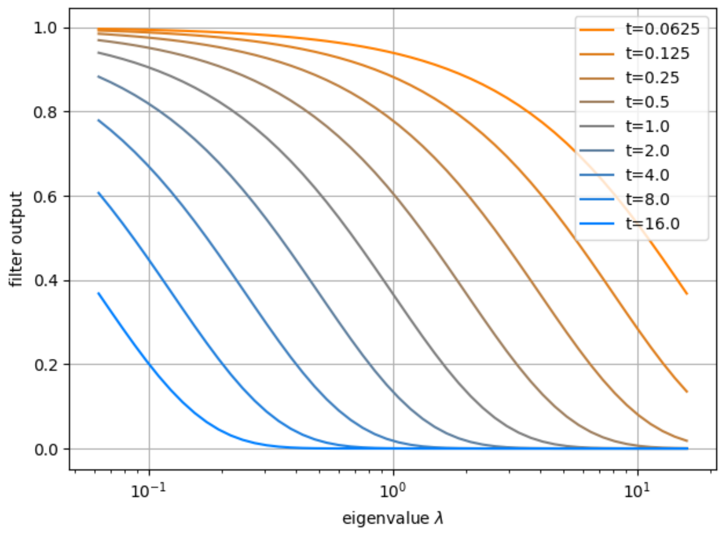


Figure 1.3: Frequency response of the exponential filter for various values of τ.

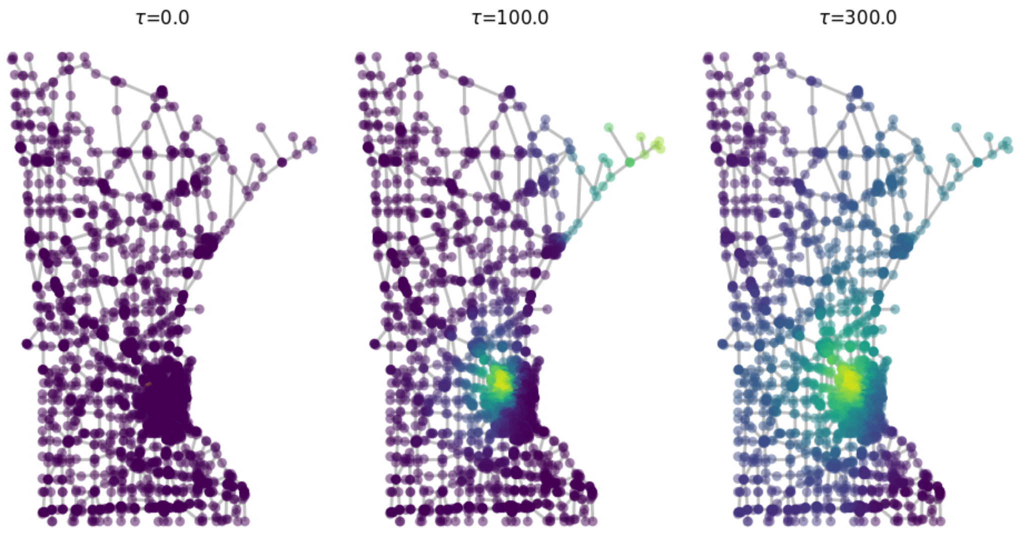


Figure 1.4: Diffusion of a signal centred on two nodes, in the Minnesota Road Network graph [40].

Figure 1.5 displays diffusion operators as images: for 3 synthetic graphs of 100 nodes, and 3 values of τ, the matrix $\exp(-\tau\mathbf{L})$ is displayed. The graphs are:

1. an Erdös-Renyi graph: every pair of nodes has a 10% probability of having an edge between them, and edges are sampled independently.
2. a 10×10 2D Grid: nodes are arranged in a grid pattern, and connected with their 4 (or less on the edges) neighbours.
3. a Community graph: nodes belong to one of three groups, supported on 2D points sampled following a Gaussian distribution whose center depends on the group. Centres are equidistant on the unit circle. Nodes have a 50% chance of being connected if they belong to the same group, 1% chance otherwise.

We can see that the scale τ is reflected in the operator $\exp(-\tau\mathbf{L})$. For instance with the 2D grid, as τ increases the entries corresponding to the closest neighbours increase first, and the farthest neighbours later.

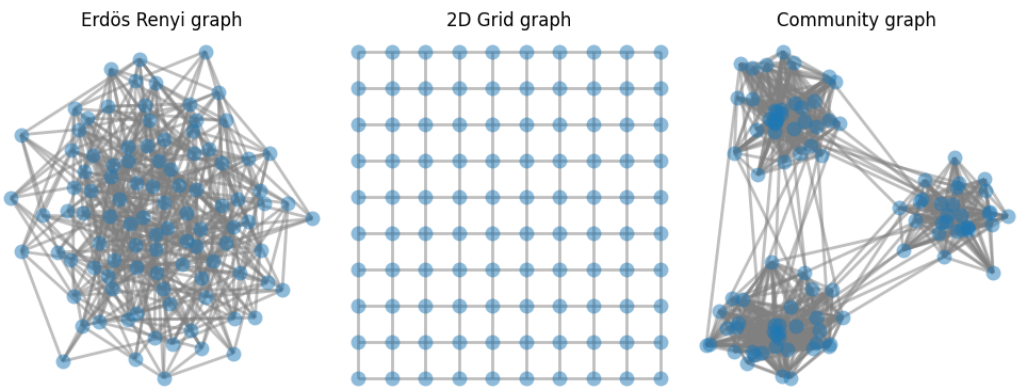
Fast and accurate computation of the diffusion in graphs is studied in Section 4.1.

1.3 An Introduction to Optimal Transport

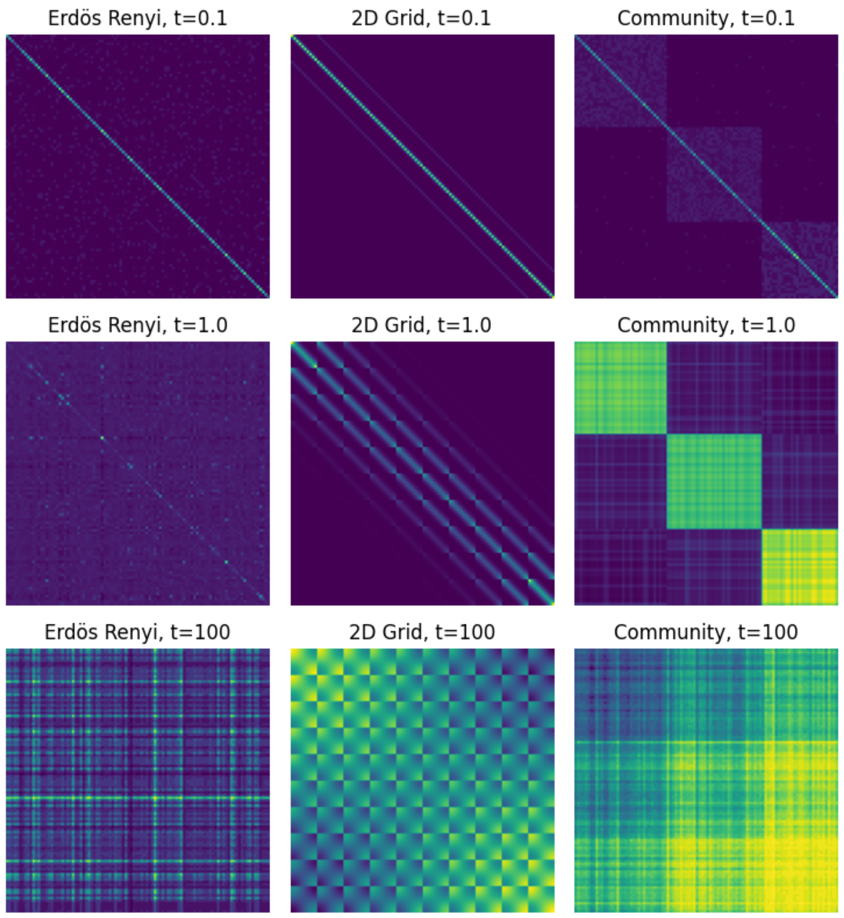
Optimal transport [41, 42] (abbreviated OT) is a tool that can be used to define distances between distributions. It works by computing a joint distribution (or transport map) that minimizes a cost criterion with respect to a distance between the individual samples. Applications are numerous. Initial studies have practical concerns with resource allocation [43, 44]. Its capacity to measure a meaningful notion of distance between distributions can be used for error measurement (for instance with neural network's resistance to adversarial attacks [45]). In machine learning, it can be used in Generative Adversarial Networks [46] to measure the error while training [47], in Domain Adaptation [48, 49] to correct shifts between the training and test distributions [50]. Later in Section 2.2, we will introduce various Optimal Transport methods that extend the basic framework presented here.

1.3.1 A toy example

Optimal transport emerged literally from the question *how to transport things in an optimal fashion [43]*, and was later extended with a more formal approach [51]. We illustrate this initial problem with a toy example, reported on Figure 1.6. On the left, in blue, 10 heaps of size a_i and positions x_i (regularly spaced here) are represented; they are called the *source (distribution)*. On the right, in red, 10 holes of size b_j and positions y_j are represented; they are called the *target (distribution)*. Knowing that the cost of moving some material from a heap to a hole can be defined as the Euclidean distance $\|x_i - y_j\|_2$ between



(a) Graphs used to plot diffusion operators.



(b) Diffusion operators for three graphs, for 3 values of t, plotted as images. Darker values correspond to lower entries in the matrix $\exp(-t\mathbf{L})$.

Figure 1.5: Examples of diffusion operators and associated graphs.

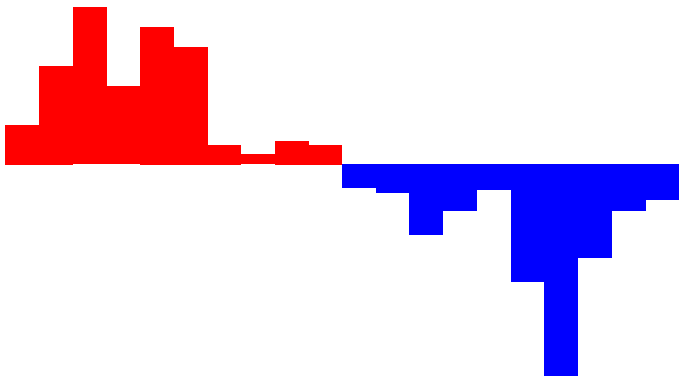


Figure 1.6: A toy optimal transport problem. On the left are blue heaps, representing the source distribution. On the right are red holes, representing the target distribution. Solving the corresponding optimal transport problem consists in finding how to fill the holes with the heaps in a way that minimizes the total transport cost.

the two, an interesting question is to find the least costly way to do so. This example uses heaps of material and holes; a more concrete example would be moving products from factories to warehouses while minimizing fuel cost.

The solution of this discrete problem is called a *transport map*. It is a matrix γ^* ($\in \mathbb{R}_+^{10\times 10}$ here) where each entry $\gamma_{i,j}$ indicates the quantity of material transported from a heap to a hole. A visualisation of this matrix is given on Figure 1.7. This matrix satisfies the following equation (with $n = m = 10$):

$$\gamma^* = \underset{\substack{\gamma\in\mathbb{R}_+^{n\times m} \\ \gamma\mathbb{1}_n = a \\ \gamma^T\mathbb{1}_m = b}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n}\sum_{j=1}^{m} \gamma_{i,j}\cdot \|x_i - y_j\|_2 \right\}. \tag{1.13}$$

This equation states that the optimal transport map minimizes the total cost of transport $\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_{i,j}\cdot\|x_i - y_j\|_2$, while conserving the masses ($\gamma\mathbb{1}_n = a$ and $\gamma^T\mathbb{1}_m = b$).

In this example, some quantity of mass (either present [heaps] or required [holes]) is present at various points on a 1D line. The OT framework abstracts this, simply working with *distributions* in some probability measure space; the problem remains moving a distribution onto another, while minimizing the global transportation cost. Of importance is the fact that this minimal cost defines a distance between distributions, effectively lifting a distance between points in a given space to distributions in that space.

1.3.2 Definition and properties

We now give a more formal definition of Optimal Transport in the discrete case, and highlight some interesting properties.

Definition 1.3.1. *Let us consider two empirical probability measures μ and ν, called* source *and* target *distributions, and supported on two sample sets*

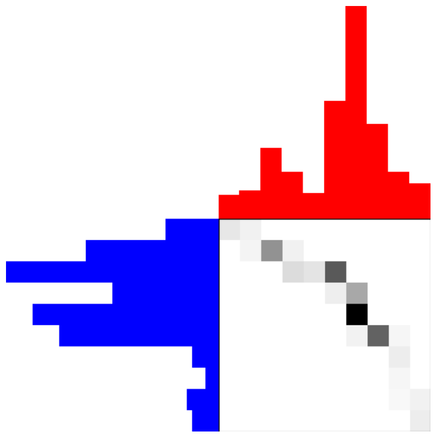


Figure 1.7: Optimal transport map solving the OT problem of Figure 1.6. Low matrix entries are lighter, and high matrix entries are darker.

$X = \{x_i\}_{i=1}^m$ and $Y = \{y_j\}_{j=1}^n$, respectively, lying in some feature space $\mathcal{X}$ and with weights $a = (a_i)_{i=1}^m$, $b = (b_j)_{j=1}^n$ such that $\mu = \sum_{i=1}^m a_i\delta_{x_i}$ and $\nu = \sum_{j=1}^n b_j\delta_{y_j}$, where δ is the Dirac measure. If $\mathcal{X} = \mathbb{R}^r$ for some integer $r \geq 1$, a matrix representation of X (resp. of Y) is the matrix $\mathbf{X} \in \mathbb{R}^{m\times r}$ (resp. $\mathbf{Y} \in \mathbb{R}^{n\times r}$) whose rows are $x_i^\top, 1 \leq i \leq m$ (resp. $y_j^\top, 1 \leq j \leq n$). Let $M = M(X,Y) \in \mathbb{R}_+^{m\times n}$ be a cost matrix, where $M_{ij} \overset{def}{=} [d(x_i, y_j)]_{ij}$ is the cost (w.r.t. to some distance function d) of moving x_i on top of y_j. Let $\Pi(a,b)$ be a transportation polytope defined as the set of admissible coupling matrices γ:

$$\Pi(a,b) = \{\gamma \in \mathbb{R}_+^{m\times n} \text{ s.t. } \gamma 1_n = a, \gamma^T 1_m = b\}, \tag{1.14}$$

*where γ_{ij} is the mass transported from x_i to y_j and 1_k is the vector of dimension k with all entries equal to one. The p-*Wasserstein distance* $\mathtt{W}_p^p(\mu,\nu)$ between the source and target distributions is defined as follows:*

$$\mathtt{W}_p^p(\mu,\nu) = \min_{\gamma\in\Pi(a,b)} \langle \gamma, M^p(X,Y)\rangle_F, \tag{1.15}$$

where $\langle .,.\rangle_F$ is the Frobenius inner product and $M^p(X,Y) := (M_{ij}^p)_{ij}$ is the entry-wise p-th power of $M(X,Y)$ with an exponent $p \geq 1$.

Theorem 1.3.1. *[16] For two empirical probability distributions μ and ν, supported on sample sets $X = \{x_i\}_{i=1}^m$ and $Y = \{y_j\}_{j=1}^n$ respectively lying in some feature space $\mathcal{X}$ and with weights $a = (a_i)_{i=1}^m$, $b = (b_j)_{j=1}^n$ such that $\mu = \sum_{i=1}^m a_i\delta_{x_i}$ and $\nu = \sum_{j=1}^n b_j\delta_{y_j}$, $\mathtt{W}_p$ defines a distance between a and b.*

Remark 1.3.1. *The implicit condition for this theorem is that M is itself a distance matrix. This holds true in our context, because the entries of M come from a distance on $\mathcal{X}$.*

The Wasserstein distance offers a natural and useful way to measure the discrepancy between two probability distributions on a metric space. It has applications in content retrieval, for instance in Natural Language Processing where texts are seen as distributions on words [52] or Computer Vision where images are seen as distributions through their histograms [53]. It has applications in machine learning too, for instance in generative models [47,54].

1.3.3 Solving the underlying optimisation problem

Problem 1.15 is a linear constrained optimisation problem. It can easily be solved with a variety of solvers. But its complexity is polynomial in the sample set's sizes (super cubic in practice [55]), and thus is quickly computationally prohibitive.

There exist many variants of the OT problem. In particular *entropic regularisation* helps in solving this scalability issue. It transforms the original problem, which is ill-defined because of the non-unicity of the solution, in a well-defined problem thanks to the strongly convex nature of the entropy. Define $h : \gamma \to -\sum_{i,j} \gamma_{i,j} \log(\gamma_{i,j} - 1)$ the Shannon entropy, and consider the following problem:

$$\mathtt{W}_p^p(\mu, \nu; \varepsilon) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, M^p(X,Y) \rangle_F - \varepsilon h(\gamma). \tag{1.16}$$

The solution of Equation 1.16 is different from the one of Equation 1.15, but converges to it when $\varepsilon \to 0$ [41]. However, the underlying optimisation problem is much easier to solve, with an iterative scheme that is orders of magnitude faster than standard constrained linear optimisation solving. It can be shown that the optimal coupling has the form $\operatorname{diag}(u) K \operatorname{diag}(v)$ with $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$, $K = \exp(-\frac{1}{\varepsilon} M^p)$ the entry-wise exponential of M^p scaled by $-\frac{1}{\varepsilon}$, and $\operatorname{diag} : \mathbb{R}^r \mapsto \mathbb{R}^{r \times r}$ the operator that builds a diagonal matrix from a vector. The vectors u and v are found by repeatedly computing the iterations $u_{n+1} = \frac{a}{K v_n}$ and $v_{n+1} = \frac{b}{K^T u_n}$. Algorithm 1 describes the matrix-scaling algorithm used to compute the Sinkhorn distance between two distributions. The reader can refer to [42] or [41] for a detailed analysis of the algorithm and its convergence.

The parameter ε offers a trade-off between closeness to the initial problem, and computation speed. When $\varepsilon \to 0$, the solution of this problem 1.16 tends to the one of the initial problem 1.15, while when $\varepsilon \to \infty$ it tends to the limit distribution $a^T b$ (which is the uniform distribution $\left(\frac{1}{n \times m}\right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}}$ when a and b are both uniform too). An illustration is visible on Figure 1.8, where the effect of increasing the parameter ε is shown: entropic regularisation "blurs" the original transport map.

Note that transport maps resulting from computing the Sinkhorn distance are not sparse, which is a direct consequence of the "interpolation" between the optimal transport map and the uniform map. When $\varepsilon \to 0$, the optimal transport map tends to the (sparse) solution of the original problem. As ε increases, the transport map becomes a smoother version of the original one, which can prevent overfitting. At the extreme ($\varepsilon \to \infty$), the transport map tends to the naïve solution $a^T b$, which does not take into account the transport cost anymore.

1.3.4 Transporting the samples

Once the transport map γ has been computed, it can be used to transport the source samples into the target domain. One way to do so is to compute the barycentric mapping of the source samples.

```
Input: $a \in \mathbb{R}_+^m$ and $b \in \mathbb{R}_+^n$ two histograms, $M \in \mathbb{R}_+^{m\times n}$ a cost matrix,
       $\varepsilon > 0$
Output: $\gamma$ minimizer of Equation 1.16
// Initialisation
$u \leftarrow a$
$v \leftarrow b$
$K \leftarrow e^{-\frac{1}{\varepsilon}M}$ // Entry-wise exponential

// Iterations
while convergence not achieved do
    $u \leftarrow \frac{a}{Kv}$ // Update left scaling

    $v \leftarrow \frac{b}{K^T u}$ // Update right scaling

end

// Final value
return $\mathrm{diag}(u) \cdot K \cdot \mathrm{diag}(v)$
```

Algorithm 1: Matrix scaling algorithm to compute the Sinkhorn distance between two distributions.

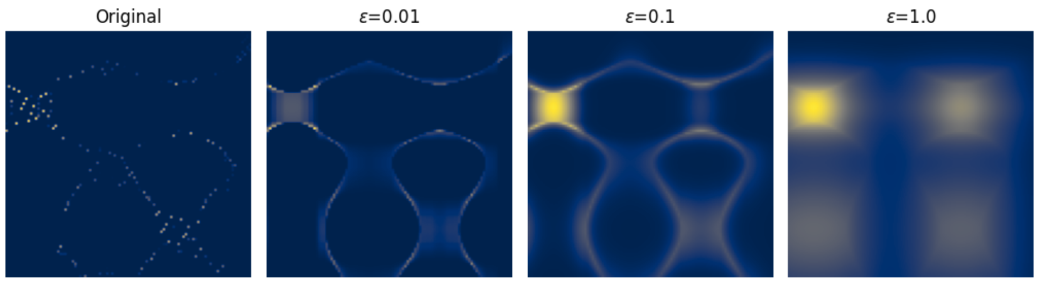


Figure 1.8: Effect of entropic regularisation on transport. Transport maps are plotted as images, with darker values indicating lower entries (closer to 0). On the left, the original transport map is displayed (i.e. without entropic regularisation), and following are transport maps with entropic regularisation (the strength being indicated on top of each image).

The *barycentric mapping* [56, 57] of a source sample x_i^s is defined by:

$$\hat{x}_i^s = \underset{x \in \mathbb{R}^r}{\text{argmin}} \left\{ \sum_{j=1}^{n} \gamma_{i,j} d(x, x_j^t) \right\}. \tag{1.17}$$

In other words, $\hat{x}_i^s$ is the barycentre of these points weighted by the mass transferred to them. When the cost function is the $\mathcal{L}^2$ distance, this equation can be expressed in matrix form for all the source samples:

$$\hat{X}^s = \text{diag}(\gamma \mathbb{1}_n)^{-1} \gamma X^t. \tag{1.18}$$

Finally, when the weights a and b are uniform, we have:

$$\hat{X}^s = m \gamma X^t. \tag{1.19}$$

Note that it is possible to define a barycentric mapping of the target samples onto the source domain by considering γ^T. Following the assumptions for 1.19, we have $\hat{X}^t = n \gamma^T X^s$.

A more general notion of Wasserstein barycentre is:

Definition 1.3.2 ([58]). *Consider N measures μ_i with respective supports X_i and weights a_i. A barycentre ν with support Y and weights b is a minimizer of:*

$$f(b, Y) = \frac{1}{N} \sum_{i=1}^{N} \mathtt{W}(\mu_i, \nu) = \frac{1}{N} \sum_{i=1}^{N} \min_{\gamma \in \Pi(a_i, b)} \{ \langle \gamma, M(X_i, Y) \rangle_F \} \tag{1.20}$$

over a relevant feasible set for either b or Y (not both).

Remark 1.3.2. *There are therefore two types of barycentres:* fixed-support barycentres*, where the support Y is fixed and minimization is done on the weights b, and* free-support barycentres*, where b is fixed and minimization is done on Y. If f is minimized on both quantities at the same time, this minimization problem becomes equivalent to finding the k-means [59].*

Remark 1.3.3. *One can show that the barycentric mapping is a first-order approximation of the real Wasserstein barycentre [58]. In fact, minimizing f over Y can be achieved by alternating between (1) computing the transport maps γ_i between the X_i and Y and (2) updating Y by minimizing a local quadratic approximation approximation of f at Y which yields the iteration $Y \leftarrow \sum_{i=1}^{N} \text{diag}(\gamma_i \mathbb{1}_{n_i})^{-1} \gamma_i X_i$.*

1.4 Domain Adaptation

The contributions of this manuscript have applications in supervised learning, in particular in Domain Adaptation on graphs, a difficult study topic that emerged recently [15, 60]. In this section we introduce the notion of supervised learning through the angle of (empirical) risk minimization [61]. We then define Domain Adaptation, the machine learning scenario studied through this manuscript. We conclude with theoretical generalization guarantees for Domain Adaptation, justifying the use of our approach for graph-structured data.

1.4.1 Supervised learning

Consider a *supervised learning* scenario. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input space and $Y = \mathbb{R}$ or $Y = [\![1, K]\!]$ for some $K \in \mathbb{N}$ be the output space. Let $\mathcal{D}$ be an unknown distribution on $\mathcal{X} \times Y$. The goal is to find a hypothesis h from a hypothesis space $\mathcal{H} \subseteq \mathcal{X}^Y$, for instance the parameters of a linear model, that best predicts the output from the input. Prediction is measured via a loss function $l : Y \times Y \to [0, 1]$ that measures for $(x, y) \sim \mathcal{D}$ the deviation of the output $h(x)$ from y.

Definition 1.4.1. *The* true risk *for a loss function $l : Y \times Y \to [0, 1]$, a given hypothesis $h \in \mathcal{H}$ and a distribution $\mathcal{D}$ over $\mathcal{X} \times Y$ is defined as:*

$$R_l^{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\, l(h(x), y). \tag{1.21}$$

When dealing with real-world problems, the distribution $\mathcal{D}$ is generally not available. Instead, it is know through a finite number of labelled examples, pairs (x, y) that are realizations of the distribution. For a given pair x is called the *instance* and y the *label*.

Definition 1.4.2. *A* training sample *of size n is a set of n i.i.d. samples of a distribution $\mathcal{D}$ over $\mathcal{X} \times Y$.*

Lacking access to the original distribution $\mathcal{D}$ prevents the use of the true risk to select a hypothesis h. Instead, the empirical risk is minimized:

Definition 1.4.3. *The* empirical risk *for a loss function $l : Y \times Y \to [0, 1]$, a given hypothesis $h \in \mathcal{H}$ and a training sample $\mathcal{T} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{1\leq i\leq n}$ of size n is defined as:*

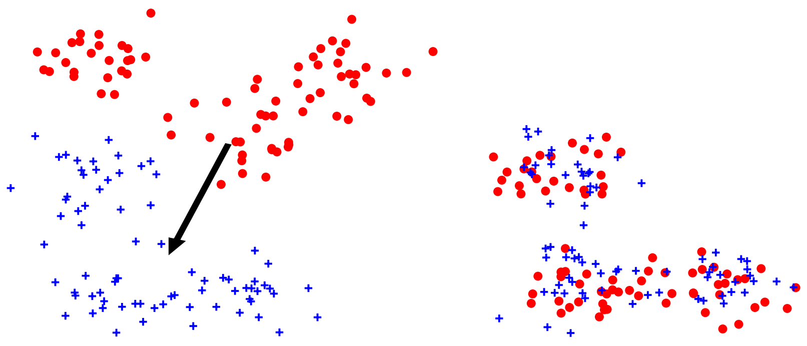
$$R_l^{\mathcal{T}}(h) = \frac{1}{n}\sum_{i=1}^{n} l(h(x_i), y_i). \tag{1.22}$$

Finding a hypothesis h that minimizes the empirical risk $R_l^{\mathcal{T}}$ does not guarantee that this hypothesis minimizes the true risk $R_l^{\mathcal{D}}$ too. However, with various assumptions on the hypothesis space $\mathcal{H}$ or the distribution $\mathcal{D}$, it is possible to prove that the true risk does not deviate too much from the empirical risk on average. These theorems are called *generalization guarantees*. We give a simple one for the case where $\mathcal{H}$ is finite as an example:

Theorem 1.4.1 ([62]). *Let $\mathcal{D}$ be an unknown distribution on $\mathcal{X} \times Y$. Let $\mathcal{T}$ be a training sample of size n drawn i.i.d. from $\mathcal{D}$. Let $\mathcal{H}$ be a finite hypothesis space.*

Let $\delta \in (0, 1]$. For any $h \in \mathcal{H}$, with probability $1 - \delta$ over the random sample $\mathcal{T}$, we have:

$$R_l^{\mathcal{D}}(h) \leq R_l^{\mathcal{T}}(h) + \sqrt{\frac{\ln(|\mathcal{H}|) + \ln(1/\delta)}{2n}}. \tag{1.23}$$



(a) Distributions before alignment. (b) Distributions after alignment.

Figure 1.9: Example of a toy DA problem. Two point distributions are given: a source in red circles and a target in blue crosses. Both are similar up to a displacement, indicated with a black arrow. A DA problem consists in finding this displacement, looking only at the two distributions.

This theorem states that, provided the training sample is large enough (n large), the true risk can be bounded arbitrarily close to the empirical risk with arbitrarily large probability. For a more complete introduction, we refer the reader to [61].

1.4.2 Definition of Domain Adaptation

A *Domain Adaptation* (DA) scenario arises in machine learning when we observe a change of distribution (a.k.a. *domain shift*) between the training data (the *source* distribution) and the samples used at test time with the deployed model (the *target* distribution). To cite a few examples, DA can occur in *image processing*, when changing the lighting or camera lens while acquiring images, in *demography* with social mobility of people or in *fraud detection*, with fraudsters trying to adapt over time to better mimic genuine behaviours. Most of the time, training a new model from the target distribution is not desirable for several reasons: (i) the algorithmic complexity required for optimizing from scratch the parameters of a new model; (ii) the lack of target training examples; (iii) the lack (or absence) of supervision (i.e. no labelled target data available), etc. In such a setting, the domain adaptation theory [63, 64] suggests to reduce the divergence between the source and the target distributions while learning an efficient model from the labelled source data.

A visual illustration of a DA problem is given on Figure 1.9. It features two similar point clouds. The red one represents the source data; they have to be re-aligned with the target data. Because the goal is to align the *distributions*, there is no one-to-one correspondence between source and target points to be found; instead, here, a rotation and a translation are used to align them.

One way to solve DA problems is to use *Optimal Transport* [16, 42] (OT), as presented earlier in Section 1.3. As illustrated in Figure 1.9, OT provides a natural geometry for comparing and aligning two distributions in the space of probability measures. In the discrete case, when dealing with point clouds, it looks for the coupling matrix (and its corresponding Wasserstein distance) that minimizes the global cost of transporting the individual masses from the source to the target distribution. In [50], the authors introduced OTDA, the first DA algorithm based on OT which moves the source on the top of the target by preventing - using a group-sparse regularisation - two source data of different labels from being transported on the same target example. Once the alignment is achieved, a model is learned from the labelled source data and applied on the target distribution. The mathematical formulation of OTDA is described in Section 2.2.4.

The exists other methods to perform Domain Adaptation using Optimal Transport. The Joint Distribution Optimal Transport framework [65] define a new minimization problem that yields both a transport map γ and a prediction function $f : \mathcal{X} \mapsto \mathcal{C}$ that assign to every target point some quantity, such as labels in the case of Domain Adaptation. Their method covers many families of prediction functions f such as neural networks or kernel machines. Wasserstein Distance Guided Representation Learning [66] is another method inspired by Generative Adversarial Networks [46]. It alternatively learns three neural networks: (1) $f_g : \mathbb{R}^d \mapsto \mathbb{R}^{d'}$ that extracts features for source and target points, (2) $f_w : \mathbb{R}^{d'} \mapsto \mathbb{R}$ for the domain critic [47] used to minimize the Wasserstein distance between the source and target data in the extracted features space, and (3) $f_c : \mathbb{R}^{d'} \mapsto \mathcal{C}$ a task-dependant function such as a classifier.

1.4.3 Generalization guarantees in Domain Adaptation

Consider a *Domain Adaptation* scenario. There are two unknown distributions: the source distribution and the target distribution. The goal is now to find how much a hypothesis learned for the source distribution is applicable for the target distribution. The following theorem relates the true risks R^s and R^t (for the source and target respectively):

Theorem 1.4.2. *([49], theorem 33) Let us consider two empirical probability measures μ and ν supported on two samples sets $\{x_i^s\}_{i=1}^m$ and $\{x_j^t\}_{j=1}^n$ respectively, lying in some feature space R^d. Consider a loss function with parametric form $l_q = |y - h(x)|^q$ for some $q > 0$. Then for any $d' > d$, and $\zeta' < \sqrt{2}$, there exists some constant N_0 depending on d' such that for any $\delta > 0$ and $\min(m, n) \geq N_0 \max(\delta^{-(d'+2)}, 1)$ we have with probability of at least $1 - \delta$ for all h:*

$$R^t_{l_q}(h) \leq R^s_{l_q}(h) + W_1(\mu, \nu) + \sqrt{2 \log(1/\delta)/\zeta'}\,(1/m + 1/n) + \lambda, \tag{1.24}$$

where λ is the combined error of the ideal hypothesis h^* that minimizes the combined error of $R^t_{l_q}(h)$ and $R^s_{l_q}(h)$.

Theorem 1.4.2 bounds the risk for the target domain with the sum of four terms: (i) the risk for the source, (ii) the $\mathtt{W}_1$ distance between the source and target distributions, (iii) a term depending on the samples sizes N^s and N^t and (iv) the joint error λ. Therefore, provided the joint error is small (i.e. the distributions are not too different) and the samples numerous enough, Optimal Transport is a good method to use an hypothesis h learned on the source domain to deal with the target domain, as the risks will stay close. This justifies the use of Optimal Transport for Domain Adaptation, as done in the rest of this manuscript in the context of graph-structured data.

Chapter 2

Graph Distances

On est trop souvent imprécis lorsqu'on fait une citation.

Quelqu'un, un jour.

In this chapter, we present a variety of discrepancy measures between graphs. A special attention is paid to OT-based graph distances.

2.1 The Graph Diffusion Distance (GDD)

The *Graph Diffusion Distance* [67] is designed to measure a similarity between two weighted graphs whose nodes are already in correspondence.

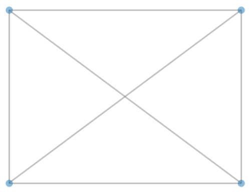
Definition 2.1.1. *Given two graphs $\mathcal{G}^1$ and $\mathcal{G}^2$ with the same n nodes V but possibly different edges E^1 and E^2, represented by their respective Laplacian matrices $\mathbf{L}^1$ and $\mathbf{L}^2$, and a diffusion time $\tau > 0$, the Graph Diffusion Distance is:*

$$\texttt{GDD}(\mathcal{G}^1, \mathcal{G}^2, \tau) = \max_{\tau \geq 0} \{\xi(\tau)\} = \max_{\tau \geq 0} \left\{ \| \exp(-\tau \mathbf{L}^1) - \exp(-\tau \mathbf{L}^2) \|_F^2 \right\}. \quad (2.1)$$

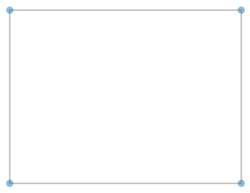
An illustration of the computation of the `GDD` is given in Figure 2.1. Two simple graphs are given in Figures 2.1a and 2.1b; the function to maximize to obtain the `GDD` is visible on Figure 2.1c.

Complexity Every evaluation of $\xi(\tau)$ costs $O(n^3)$ (where n is the size of the graphs in nodes). The maximization of ξ is done with a line-search, using for instance Brent's algorithm [68].

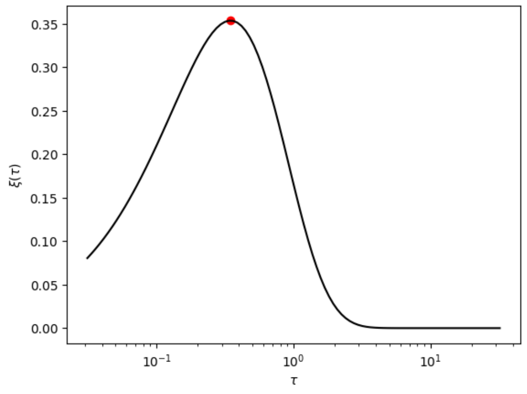
Remark 2.1.1. *The `GDD` is restricted to comparing graphs whose nodes have already been matched (for instance for two algorithms estimating connection graphs between the same sets of regions in the brain) which is limiting. Additionally, the need to explicitly compute the exponential of the Laplacian of each graph prevents its use with large graphs, as such an operation does not scale well (cubic complexity).*



(a) First graph



(b) Second graph



(c) Plot of $\xi(\tau) = \| \exp(-\tau \mathbf{L}^1) - \exp(-\tau \mathbf{L}^2) \|_F^2$ in black, and maximum of the function as a red point.

Figure 2.1: Illustration of the `GDD`.

Despite its limitations, the `GDD` does define a distance between graphs (in the mathematical sense), and has applications in studying edges importance. Also, the idea of studying *diffusion patterns* is something we will re-use for our distance based on Optimal Transport.

2.2 OT-based graph distances

As the optimal transport framework allows lifting distances between points to distributions, many extensions and variants try to do the same with graphs. Note that the following methods will not systematically define an actual distance in the mathematical sense.

In order to be able to apply the OT setting on structured data, we need now to formally define the notion of probability measure on graphs and adapt the previous notations. Following [15], let us consider undirected and connected attributed graphs as tuples of the form $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{S})$, where $\mathcal{V}$ and $\mathcal{E}$ are the classic sets of vertices (also called nodes) and edges of the graph, respectively. $\mathcal{F} : \mathcal{V} \to \mathcal{X}$ is a function which assigns a feature vector $x_i \in \mathcal{X}$ (also called a graph signal in [7]) to each vertex v_i of the graph (given an arbitrary ordering of the vertices). $\mathcal{S} : \mathcal{V} \to \mathcal{Z}$ is a function which associates each vertex v_i with some structural representation $z_i \in \mathcal{Z}$, *e.g.* a local description of the graph, the vertex and a list of its neighbours, etc. We can further define a cost function $C : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+$ which measures the dissimilarity $C(z, z')$ between

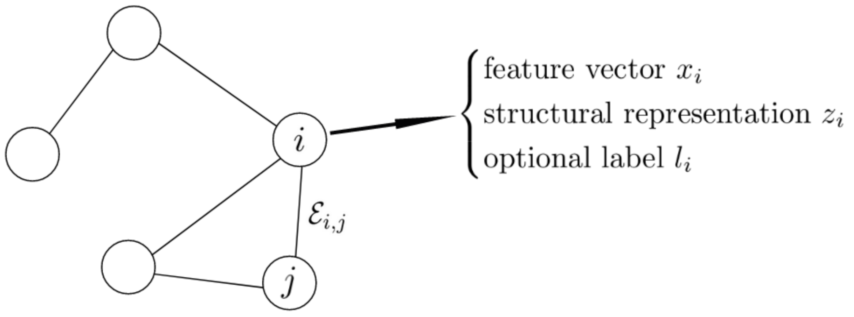


Figure 2.2: Example of a structured object. The i-th node has an associated feature vector $x_i \in \mathcal{X}$, a structural representation $z_i \in \mathcal{Z}$ and an optional label $l_i \in \mathcal{L}$. The nodes i and j share an edge $\mathcal{E}_{i,j}$.

two structural representations z, z'. Typically, $C(z, z')$ can capture the length of the shortest path between two nodes. Additionally, if the graph $\mathcal{G}$ is *labelled*, each vertex v_i is also assigned a label from some label space $\mathcal{L}$.

When each vertex of the graph is weighted according to its relative importance, the source and target graphs can be seen as probability distributions,

$$\mu = \sum_{i=1}^{m} a_i \delta_{(x_i, z_i)}, \quad \nu = \sum_{j=1}^{n} b_j \delta_{(y_j, z'_j)} \tag{2.2}$$

where x_i, z_i are the features / structural representations associated to the vertices of the source graph while y_j,z'_j are those associated to the target one.

2.2.1 The Gromov-Wasserstein distance $\texttt{GW}$

The *Gromov-Wasserstein* distance [60,69] is an extension of the OT framework, that allows the comparison of distributions living in different spaces. Instead of minimizing the total transport cost, it tries to preserve distances between pairs of source points and pairs of target points. It does so by using two cost matrices instead of one: one for the source, and one for the target.

Definition 2.2.1. *Let μ and ν be two discrete distributions of size N^s and N^t. Let $C^s \in \mathbb{R}_+^{N^s \times N^s}$ and $C^t \in \mathbb{R}_+^{N^t \times N^t}$ be two cost matrices, associated to the source and the target respectively. Let L be a loss function, typically $L(a, b) = {}^1\!/_2 |a - b|^2$. The* Gromov-Wasserstein distance *between the two distributions is:*

$$\texttt{GW}_2(\mu, \nu) = \min_{\gamma \in \Pi(a,b)} \left\{ \sum_{i,j,k,l} L(C^s_{i,k}, C^t_{j,l}) \gamma_{i,j} \gamma_{k,l} \right\}^{\frac{1}{2}}. \tag{2.3}$$

Like $\texttt{W}$, $\texttt{GW}_2$ too defines a distance in the mathematical sense:

Theorem 2.2.1 ([69]). $\texttt{GW}_2$ *defines a distance on the collection of all isomorphisms of metric measure spaces, i.e. between triplets (X, d_X, μ), where (X, d_X) is a compact metric space and μ a Borel probability measure on X, up to a permutation. Note that in our context we only look at discrete distributions.*

The matrices C can be any distance matrices in the source and target spaces, so it is possible, for instance, to have the source and target data live in euclidean spaces of different dimensions. In particular for graphs, any notion of similarity between nodes can be used for this method; the matrices C could be adjacency matrices, shortest-path matrices or diffusion operators for instance. However, the design of a cost matrix that could incorporate both structure and feature information and produce meaningful distances/transport maps is an open problem. Another drawback of this method is its computational cost. Entropic regularisation can similarly be used, producing similar results: a simple iterative scheme, orders of magnitude faster, at the cost of a tradeoff (between the sparse solution with a risk of overfitting, and a uniform transport map that doesn't take into account the transport costs).

Complexity Computing GW requires solving a non-convex problem, which can be recast as a quadratic assignment problem [70]. It is NP-hard in full generality. Like W, an entropy-regularized version is possible; it can be solved in an iterative fashion, where each step requires the computation of a loss in $O(n^2m^2)$ ($O(n^2m + nm^2)$ with the $\mathcal{L}^2$ loss) and a Sinkhorn distance.

2.2.2 The Fused-Gromov-Wasserstein distance FGW

We now present the Fused Gromov-Wasserstein (FGW) distance introduced in [15] as the first attempt to define a distance that takes into account both structural and feature information in an OT problem. It is defined via the minimization of a convex combination between (i) the cost for the Wasserstein distance (see Definition 1.3.1) which considers the features x_i, y_j associated with the nodes and (ii) the cost for the Gromov-Wasserstein distance (see Definition 2.2.1) which takes into account the structure of both graphs.

Definition 2.2.2. *Let $\mathcal{G}^s$ (resp. $\mathcal{G}^t$) be a source (resp. target) graph described by its discrete probability measure μ (resp. ν). Let $X^s \in \mathbb{R}^{m\times r}$ (resp. $X^t \in \mathbb{R}^{n\times r}$) be the r-dimensional features associated with the nodes of $\mathcal{G}^s$ (resp. $\mathcal{G}^t$). Let $C^s \in \mathbb{R}^{m\times m}$ and $C^t \in \mathbb{R}^{n\times n}$ be the structure matrices associated with the source and target graphs respectively. For $\alpha \in [0,1]$, the Fused-Gromov-Wasserstein distance is defined as follows:*

$$\mathtt{FGW}_p^p(\mu,\nu) = \min_{\gamma\in\Pi(a,b)} \left\{ \sum_{i,j,k,l} \left((1-\alpha)M_{ij}^p + \alpha|C_{ik}^s - C_{jl}^t|^p\right)\gamma_{ij}\gamma_{kl} \right\}, \tag{2.4}$$

where the summation indices are $1 \leq i,k \leq m$ and $1 \leq j,l \leq n$, and the dependency on α is omitted from the notation $\mathtt{FGW}_p(\mu,\nu)$ for the sake of concision.

Theorem 2.2.2 ([15]). *If C^s and C^t are distance matrices, $\mathtt{FGW}_p$ defines a metric for $p = 1$ and a semi-metric for $p \geq 1$.*

Remark 2.2.1. *For the case $p \geq 1$, the triangle inequality is relaxed by a factor 2^{p-1}.*

Roughly speaking, the optimal coupling matrix $\gamma^\star$ will tend to associate two source and target nodes if both their features and structural representations are similar. Note that α can be seen as a hyper-parameter which will allow `FGW`, given the underlying task and data, to find a good compromise between the features and the structures of the graphs.

Remark 2.2.2. *In the special case $\alpha = 0$, we recover the Wasserstein distance $\mathtt{FGW}_p(\mu, \nu \mid \alpha = 0) = \mathtt{W}_p(\mu, \nu)$. In the case $\alpha = 1$, we recover the Gromov-Wasserstein distance $\mathtt{FGW}_p(\mu, \nu \mid \alpha = 1) = \mathtt{GW}_p(\mu, \nu)$.*

Complexity The authors suggest using a Conditional Gradient algorithm. It alternates between computing a gradient G (in $O(n^2m + nm^2)$ when $p = 2$ and $O(n^2m^2)$ otherwise) and solving an OT problem with cost matrix G.

2.2.3 Laplacian regularisation

The authors of [71] introduce two possible regularisation terms to the optimal transport problem 1.15. They only consider uniform distributions $a = ({}^1/_{N_s})$ and $b = ({}^1/_{N_t})$.

Both use the *barycentric projection* of the samples: once a transport map is computed, the source samples can be transported onto the target domain (and vice-versa). For a cost using the $\mathcal{L}^2$ norm, the barycentric mapping of a source sample x_i^s is:

$$\tilde{x_i^s} = \underset{x \in \mathbb{R}^r}{\operatorname{argmin}} \left\{ \sum_j \gamma_{i,j} \|x - x_j^t\|_2^2 \right\}. \tag{2.5}$$

In other words, $\tilde{x_i^s}$ is the barycenter of the target points, weighted with the mass transported from it. The barycentric mapping of the target points onto the source domain is defined similarly with γ^T. When the distributions μ and ν are uniform, these barycenters can be computed in closed form:

$$\tilde{X^s} = m\gamma X^t \quad \text{and} \quad \tilde{X^t} = n\gamma^T X^s. \tag{2.6}$$

Their first regularisation term regularizes the transported samples' positions: two adjacent nodes should have their transported features close. The optimisation problem becomes:

$$\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, M(\mathbf{X}_s, \mathbf{X}_t) \rangle + \frac{\lambda_s}{N_s^2} \sum_{(i,j) \in \mathcal{E}_s} \|\tilde{x}_i^s - \tilde{x}_j^s\|^2 + \frac{\lambda_t}{N_t^2} \sum_{(i,j) \in \mathcal{E}_t} \|\tilde{x}_i^t - \tilde{x}_j^t\|^2 \right\}. \tag{2.7}$$

Their second one regularizes the transported sample's displacements: two adjacent samples should have a similar displacement. The corresponding optimisation problem is:

$$\min_{\gamma\in\Pi(a,b)} \left\{ \langle \gamma, M(\mathbf{X}_s, \mathbf{X}_t)\rangle + \frac{\lambda_s}{N_s^2} \sum_{(i,j)\in\mathcal{E}_s} \|(\tilde{x}_i^s - x_i^s) - (\tilde{x}_j^s - x_j^s)\|^2 \right.$$
$$\left. + \frac{\lambda_t}{N_t^2} \sum_{(i,j)\in\mathcal{E}_t} \|(\tilde{x}_i^t - x_i^t) - (\tilde{x}_j^t - x_j^t)\|^2 \right\}. \quad (2.8)$$

Both of these problems can be rewritten as quadratic programs:

$$\min_{\gamma\in\Pi(a,b)} \left\{ \langle \gamma, M(\mathbf{X}_s, \mathbf{X}_t)\rangle + \lambda_s \mathrm{Tr}(\mathbf{X}_t^T \gamma^T \mathbf{L}_s \gamma \mathbf{X}_t) + \lambda_t \mathrm{Tr}(\mathbf{X}_s^T \gamma \mathbf{L}_t \gamma^T \mathbf{X}_s) \right\} \quad (2.9)$$

$$\min_{\gamma\in\Pi(a,b)} \left\{ \langle \gamma, M(\mathbf{X}_s, \mathbf{X}_t + \lambda_s C_s + \lambda_t C_t)\rangle + \lambda_s \mathrm{Tr}(\mathbf{X}_t^T \gamma^T \mathbf{L}_s \gamma \mathbf{X}_t) + \lambda_t \mathrm{Tr}(\mathbf{X}_s^T \gamma \mathbf{L}_t \gamma^T \mathbf{X}_s) \right\}, \quad (2.10)$$

with $C_s = -{}^1\!/\!_{N_s}(\mathbf{L}_s + \mathbf{L}_s^T)\mathbf{X}_s\mathbf{X}_t^T$ and $C_t = -{}^1\!/\!_{N_t}\mathbf{X}_s\mathbf{X}_t^T(\mathbf{L}_t + \mathbf{L}_t^T)$. The authors then provide an optimization scheme based on the Frank-Wolfe algorithm [72] (also known as the Conditional Gradient algorithm). The Frank-Wolfe algorithm can be used to optimize a convex differentiable real function over a convex compact set, which is the case here. Note f the objective function to be minimized. The algorithm works by alternating two steps: (i) find the transport map minimizing a linear approximation of the problem $\gamma^* = \operatorname*{argmin}_{\gamma\in\Pi(a,b)} \left\{ \langle \gamma, \nabla_\gamma f(\gamma_k)\rangle_F \right\}$, and (ii) find the optimal step size $s_k = \operatorname*{argmin}_{0\leq s\leq 1} \{f(y_k + s(\gamma^* - \gamma_k))\}$ to update $\gamma_{k+1} = \gamma_k + s_k(\gamma^* - \gamma_k)$. These two steps are repeated until a convergence criteria is reached.

While this method provides transport maps that take into account both the features and the structure of the source and target data, it does not yield a distance between attributed graphs. Nonetheless, it provides insights as to how to incorporate both modalities into a single optimisation problem.

Complexity Like for FGW, the authors suggest a Conditional Gradient algorithm. The gradient for the first step of the algorithm can be computed in $O(nm)$.

2.2.4 Label regularisation

A class structure (each data point has an associated label) is more restrictive than a graph structure. Nevertheless, because we study attributed graphs that are potentially labelled, it makes sense to review methods that exploit the attributes. The authors of OTDA [73] present two possibles regularisation terms to be added to the optimization problem 1.15. The first one is the one presented in Section 2.2.3, where the Laplacian $\mathbf{L}$ relates to a graph encoding class structure ($\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{A}_{i,j} = 1 \iff i$ and j share the same label). The second one is a ℓ_1-ℓ_2 group-lasso regularizer, that prevents source points from different classes to be mapped on the same target point.

Assume each source point i has a associated label l_i^s. Define $\mathcal{I}_{cl} = \{i \mid l_i^s = cl\}$ the set of indexes corresponding to source points of class cl. For a transport map γ, $\gamma(\mathcal{I}_{cl}, j)$ is then a vector containing the weights transported from class cl to the sample j. The regularizer is then:

$$\Omega_c(\gamma) = \sum_j \sum_{cl} \|\gamma(\mathcal{I}_{cl}, j)\|_2. \tag{2.11}$$

This way of cutting γ into vectors is illustrated in Equation 2.12:

$$\gamma = \begin{array}{r} cl = 1 \;\; \Bigg\{ \\ 2 \;\; \Bigg\{ \\ \vdots \;\; \Bigg\{ \end{array} \left(\begin{array}{c|c|c|c} \mathcal{I}_1(0) & \mathcal{I}_1(1) & \cdots & \mathcal{I}_1(n) \\ \mathcal{I}_2(0) & \mathcal{I}_2(1) & \cdots & \mathcal{I}_2(n) \\ \vdots & \vdots & \cdots & \vdots \end{array} \right) \tag{2.12}$$

The goal of this regularisation term is to promote a label-related sparsity in the transport map. The final optimization problem to solve is then:

$$\min_{\gamma \in \Pi(a,b)} \langle \gamma, M^p(X,Y) \rangle_F - \varepsilon h(\gamma) + \eta \Omega_c(\gamma). \tag{2.13}$$

The convexity of their terms allows then to design an efficient optimization algorithm to solve 2.13, with a generalized conditional gradient algorithm [74]. It works by alternating a minimization and a line-search, but here the minimization step requires solving an entropy-regularized optimal transport problem.

Chapter 3

The Diffusion-Wasserstein Distance DW

On est trop souvent imprécis lorsqu'on fait une citation.

Quelqu'un, un jour.

In this chapter, we introduce the *Diffusion-Wasserstein distance* (DW). We provide insight into its design and its properties. Its experimental study is done in Chapter 5. This chapter corresponds to works published in 2020 at the ECML/PKDD conference [18].

Our goal with the Diffusion-Wasserstein distance was to provide an alternative to FGW that would be computationally more efficient, with easier to interpret hyperparameter(s), and with a wide range of applications. Our design is related to [67] where the diffusion operator of two graphs is used to compare them. The key difference is that it is the actions of these diffusion operators on the graph's features that are compared, using the OT framework.

In this chapter, we start by giving a formal definition of the DW distance in section 3.1; we then analyse theoretically its properties in Section 3.2; we finally discuss various questions raised by this notion.

3.1 Definition

We first give some insight about the design of our method. Lets assume that we are given two attributed graphs to compare (either via a number (a distance) or a node-to-node mapping). One could ignore the structure, and simply compare the attributes; for this task, the OT framework is a natural choice. This measurement is incomplete, but a possible solution could be to represent each attributed graph by a set of attributes that would merge the information of the original features and the graphs. This way, the OT framework could still be leveraged with all its advantages: computing both a distance and a mapping, fast computations via entropic regularisation, a theoretical framework and multiple extensions and regularisation terms for various scenarios. The goal of our strategy is *to let the attributes diffuse in the graph.* These diffused

$$\begin{tikzcd}
\mathbf{Y} \arrow[r, "\exp(-\tau^s\mathbf{L}^s)\cdot"] & \tilde{\mathbf{X}} \arrow[dr] & & \\
& & \tilde{M}(\tau^s,\tau^t) \arrow[r, "\min\limits_{\gamma\in\Pi(a,b)}\{\langle\gamma,\cdot\rangle\}"] & \mathtt{DW}_p^p(\mu,\nu) \\
\mathbf{X} \arrow[r, "\exp(-\tau^t\mathbf{L}^t)\cdot"] & \tilde{\mathbf{Y}} \arrow[ur] & &
\end{tikzcd}$$

Figure 3.1: Diagram of the steps involved in computing $\mathtt{DW}$.

attributes differ from the original ones in a way unique to each graph, that hopefully successfully merges the two modalities in a way OT can exploit. We introduce in the following the Diffusion-Wasserstein distance:

Definition 3.1.1. *Consider a source graph $\mathcal{G}^s$, a target graph $\mathcal{G}^t$ represented through two discrete probability measures μ and ν with weights vectors $a \in \mathbb{R}^m$, $b \in \mathbb{R}^n$ and Laplacian matrices $\mathbf{L}^s \in \mathbb{R}^{m\times m}$ and $\mathbf{L}^t \in \mathbb{R}^{n\times n}$. Let $\mathbf{X} \in \mathbb{R}^{m\times r}$, $\mathbf{Y} \in \mathbb{R}^{n\times r}$ represent the sample sets associated to the features on the vertices of the two graphs. Note that features for both graphs live in the same space $\mathbb{R}^r$.*

Given parameters $0 \le \tau^s, \tau^t < \infty$, consider the diffused sample sets $\tilde{X}$, $\tilde{Y}$ represented by the matrices $\tilde{\mathbf{X}} = \exp(-\tau^s\mathbf{L}^s)\mathbf{X} \in \mathbb{R}^{m\times r}$, $\tilde{\mathbf{Y}} = \exp(-\tau^t\mathbf{L}^t)\mathbf{Y} \in \mathbb{R}^{n\times r}$ and define $\tilde{M}(\tau^s,\tau^t) := M(\tilde{X},\tilde{Y}) \in \mathbb{R}^{m\times n}$, a cost matrix between features that takes into account the structure of the graphs through diffusion operators. We define the Diffusion Wasserstein distance ($\mathtt{DW}$) between μ and ν as:

$$\mathtt{DW}_p^p(\mu,\nu \mid \tau^s,\tau^t) = \min_{\gamma\in\Pi(a,b)} \left\{\langle\gamma,\tilde{M}^p\rangle\right\}. \tag{3.1}$$

Here again $\tilde{M}^p$ is the entrywise p-th power of $\tilde{M}$. The underlying distance is implicit in $M(\cdot,\cdot)$. For the sake of concision, the dependency on τ^s and τ^t will be omitted from the notation $\mathtt{DW}_p^p(\mu,\nu)$ if not specifically required.

Note that the $\mathtt{DW}$ distance can be broken down in two parts. A pre-processing step, in which the graph features are allowed to diffuse for some time τ, and a distance computation step, which uses the Wasserstein distance between two distributions supported on those diffused attributes. A diagram of the computation of $\mathtt{DW}_p^p(\mu,\nu)$ is given in Figure 3.1.

3.2 Properties

3.2.1 $\mathtt{DW}$ limits and bounds

We first study the asymptotic behaviour of $\mathtt{DW}$ with respect to the diffusion times $\tau^{s/t}$.

Denote $D^s = \exp(-\tau^s\mathbf{L}^s) \in \mathbb{R}^{m\times m}$, $D^t = \exp(-\tau^t\mathbf{L}^t) \in \mathbb{R}^{n\times n}$ the diffusion matrices, which depend on the (symmetric) Laplacians $\mathbf{L}^s \in \mathbb{R}^{m\times m}$, $\mathbf{L}^t \in \mathbb{R}^{n\times n}$ and the diffusion parameters $0 \le \tau^s,\tau^t < \infty$. Given $1 \le i \le m, 1 \le j \le n$ let $x_i, y_j \in \mathbb{R}^r$ be the features on nodes i on $\mathcal{G}^s$ and j on $\mathcal{G}^t$, i.e. respectively the i-th row of $\mathbf{X} \in \mathbb{R}^{m\times r}$ and the j-th row of $\mathbf{Y} \in \mathbb{R}^{n\times r}$, and similarly for

$\tilde{x}_i, \tilde{y}_j \in \mathbb{R}^r$ built from $\tilde{\mathbf{X}} = D^s\mathbf{X}$ and $\tilde{\mathbf{Y}} = D^t\mathbf{Y}$. Observe that $\tilde{M}(\tau^s, \tau^t)$ and $\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t)$ depend on the diffusion parameters τ^s, τ^t.

Limit of `DW` in 0

Proposition 3.2.1. *When $\tau^s = \tau^t = 0$, since $D^s = I_m$ and $D^t = I_n$ we have $\tilde{M}(0,0) = M$ hence*

$$\mathtt{DW}_p^p(\mu, \nu \mid 0, 0) = \mathtt{W}_p^p(\mu, \nu), \tag{3.2}$$

i.e., `DW` *generalizes the Wasserstein distance* `W`*.*

From now on we focus on `DW` defined using a cost matrix $\tilde{M}$ based on the Euclidean distance and $p = 2$. Denote

$$\begin{cases} M_{ij}^2 = \|x_i - y_j\|_2^2 \\ \tilde{M}_{ij}^2 = \|\tilde{x}_i - \tilde{y}_j\|_2^2 \end{cases} \tag{3.3}$$

the squared entries of the cost matrices associated to the Wasserstein ($\mathtt{W}_2$) and Diffusion Wasserstein ($\mathtt{DW}_2$) distances. The next proposition establishes the asymptotic behavior of $\mathtt{DW}_2^2(\mu, \nu)$ with respect to τ^s and τ^t as well as an upper bound expressed in terms of a uniform coupling matrix. Denote $\bar{\gamma} \in \Pi(a, b) \subset \mathbb{R}_+^{m\times n}$ this (uniform) transport plan such that $\bar{\gamma}_{i,j} = {}^1\!/_{nm}, \forall i, j$.

Limit of `DW` in ∞

Proposition 3.2.2. *Consider Laplacians $\mathbf{L}^s \in \mathbb{R}^{m\times m}$, $\mathbf{L}^t \in \mathbb{R}^{n\times n}$ associated to two undirected connected graphs ($\mathcal{G}^s$ and $\mathcal{G}^t$) and two matrices $\mathbf{X} \in \mathbb{R}^{m\times r}, \mathbf{Y} \in \mathbb{R}^{n\times r}$ representing the sample sets $x_i \in \mathbb{R}^r, 1 \le i \le m$ and $y_j \in \mathbb{R}^r, 1 \le j \le n$ (associated to their vertices). Consider the associated measures μ, ν with flat weight vectors $a = 1_m/m$, $b = 1_n/n$. We have*

$$\lim_{\tau^s, \tau^t \to \infty} \mathtt{DW}_2^2(\mu, \nu \mid \tau^s, \tau^t) = \|\frac{1}{m}\sum_i x_i - \frac{1}{n}\sum_j y_j\|_2^2. \tag{3.4}$$

Moreover, the function $(\tau^s, \tau^t) \mapsto \langle \bar{\gamma}, \tilde{M}^2(\tau^s, \tau^t)\rangle$ is non-increasing with respect to τ^s and with respect to τ^t and also satisfies for each $0 \le \tau^s, \tau^t < \infty$

$$\|\frac{1}{m}\sum_i x_i - \frac{1}{n}\sum_j y_j\|_2^2 \le \mathtt{DW}_2^2(\mu, \nu \mid \tau^s, \tau^t) \le \langle \bar{\gamma}, \tilde{M}^2(\tau^s, \tau^t)\rangle \le \langle \bar{\gamma}, M^2\rangle \tag{3.5}$$

$$\lim_{\tau^s, \tau^t \to \infty} \langle \bar{\gamma}, \tilde{M}^2(\tau^s, \tau^t)\rangle = \|\frac{1}{m}\sum_i x_i - \frac{1}{n}\sum_j y_j\|_2^2. \tag{3.6}$$

Equation 3.4 formalizes an intuitive behaviour. In a physical system, without border constraints, diffusion happens until a steady-state is reached where the temperature is constant, equal to the average initial temperature. Here, if diffusion is arbitrarily long, the graph attributes converge toward their respective average value, and comparing them with OT is just comparing these averages.

Proof. Denoting U a matrix associated to an orthonormal basis of eigenvectors of a Laplacian L of a connected graph, $0 = \lambda_1 < \lambda_2 \leq \ldots$ the corresponding eigenvalues, and $\hat{x}_i \in \mathbb{R}^r$ the rows of $\hat{\mathbf{X}} = U^\top \mathbf{X}$, we have $\| \exp(-\tau L)\mathbf{X}\|_F^2 = \sum_i \exp(-2\tau\lambda_i)\|\hat{x}_i\|_2^2$ hence $\tau \mapsto \| \exp(-\tau L)\mathbf{X}\|_F^2$ is non-increasing w.r.t. τ, and we have $\lim_{\tau\to\infty} \| \exp(-\tau L)\mathbf{X}\|_F^2 = \|\hat{x}_1\|_2^2$. Moreover, since $D = \exp(-\tau L)$ is a diffusion operator, it preserves the mean hence $\sum_i \tilde{x}_i = \sum_i x_i$ and $\sum_j \tilde{y}_j = \sum_j y_j$. As a result we have:

$$\langle \tilde{M}^2, \bar{\gamma}\rangle = \frac{1}{nm}\sum_{i=1}^{m}\sum_{j=1}^{n} \tilde{M}_{ij}^2 = \frac{1}{nm}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\|\tilde{x}_i\|_2^2 + \|\tilde{y}_j\|_2^2 - 2\langle \tilde{x}_i, \tilde{y}_j\rangle\right] \tag{3.7}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\|\tilde{x}_i\|_2^2 + \frac{1}{n}\sum_{j=1}^{n}\|\tilde{y}_j\|_2^2 - \frac{2}{nm}\left\langle \sum_{i=1}^{m}\tilde{x}_i, \sum_{j=1}^{n}\tilde{y}_j\right\rangle \tag{3.8}$$

$$= \frac{1}{m}\|D^s\mathbf{X}\|_F^2 + \frac{1}{n}\|D^t\mathbf{Y}\|_F^2 - 2\left\langle \frac{1}{m}\sum_{i=1}^{m}x_i, \frac{1}{n}\sum_{j=1}^{n}y_j\right\rangle. \tag{3.9}$$

Hence, $\langle \tilde{M}^2(\tau^s,\tau^t), \bar{\gamma}\rangle$ is a non-increasing function of τ^s and of τ^t, and:

$$\lim_{\tau^s,\tau^t\to\infty}\langle \tilde{M}^2(\tau^s,\tau^t), \bar{\gamma}\rangle = \frac{1}{m}\|\widehat{x}_1\|_2^2 + \frac{1}{n}\|\widehat{y}_1\|_2^2 - 2\left\langle \frac{1}{m}\sum_{i=1}^{m}x_i, \frac{1}{n}\sum_{j=1}^{n}y_j\right\rangle \tag{3.10}$$

$$= \|\frac{1}{m}\sum_{i=1}^{m}x_i - \frac{1}{n}\sum_{j=1}^{n}y_j\|_2^2 \tag{3.11}$$

where we used that $\widehat{x}_1 = \frac{1}{\sqrt{m}}\sum_{i=1}^{m}x_i$ and similarly for $\widehat{y}_1$. As the weight vectors a and b are flat, the uniform plan $\bar{\gamma}$ is admissible. Since $\tilde{M}(0,0) = M$ it follows that $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t) \leq \langle \tilde{M}^2(\tau^s,\tau^t), \bar{\gamma}\rangle \leq \langle M^2, \bar{\gamma}\rangle$. Moreover, [42, Remark 2.19] states that:

$$\mathtt{W}_2^2(\mu,\nu) = \mathtt{W}_2^2(\tilde{\mu},\tilde{\nu}) + \|\frac{1}{m}\sum_{i=1}^{m}x_i - \frac{1}{n}\sum_{j=1}^{n}y_j\|_2^2, \tag{3.12}$$

where $\tilde{\mu}$ and $\tilde{\nu}$ are the same distributions as μ and ν, but with their support centred $\left\{x_i - \frac{1}{m}\sum_{i=1}^{m}x_i \,|\, 1 \leq i \leq m\right\}$ and $\left\{y_j - \frac{1}{n}\sum_{j=1}^{n}y_i \,|\, 1 \leq j \leq n\right\}$. Therefore, we have:

$$\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t) \geq \|\frac{1}{m}\sum_{i=1}^{m}\tilde{x}_i - \frac{1}{n}\sum_{j=1}^{n}\tilde{y}_j\|_2^2 = \|\frac{1}{m}\sum_{i=1}^{m}x_i - \frac{1}{n}\sum_{j=1}^{n}y_j\|_2^2, \tag{3.13}$$

hence:

$$\lim_{\tau^s,\tau^t\to\infty}\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t) = \|\frac{1}{m}\sum_i x_i - \frac{1}{n}\sum_j y_j\|_2^2. \tag{3.14}$$

□

Remark 3.2.1. *We can also establish that:*

$$\langle \tilde{M}^2(\tau^s,\tau^t), \bar{\gamma}\rangle = \langle M^2, \bar{\gamma}\rangle + \left[\sum_{i=2}^{m}\left(e^{-2\tau^s\lambda_i^s} - 1\right)\|\widehat{x}_i\|_2^2 + \sum_{j=2}^{n}\left(e^{-2\tau^t\lambda_j^t} - 1\right)\|\widehat{y}_j\|_2^2\right]. \tag{3.15}$$

Indeed:

$$\langle\tilde{M}^2(\tau^s,\tau^t),\bar{\gamma}\rangle = \frac{1}{m}\|D^s\mathbf{X}\|_F^2 + \frac{1}{n}\|D^t\mathbf{Y}\|_F^2 - 2\left\langle \frac{1}{m}\sum_{i=1}^m x_i, \frac{1}{n}\sum_{j=1}^n y_j \right\rangle \tag{3.16}$$

$$= \frac{1}{m}\sum_{i=1}^m e^{-2\tau^s\lambda_i^s}\|\hat{x}_i\|_2^2 + \frac{1}{n}\sum_{j=1}^n e^{-2\tau^t\lambda_j^t}\|\hat{y}_j\|_2^2 - \frac{2}{nm}\left\langle \sum_{i=1}^m x_i, \sum_{j=1}^n y_j \right\rangle \tag{3.17}$$

$$\begin{aligned} =& \frac{1}{m}\sum_{i=1}^m (e^{-2\tau^s\lambda_i^s}-1)\|\hat{x}_i\|_2^2 + \frac{1}{m}\sum_{i=1}^m \|\hat{x}_i\|_2^2 \\ &+ \frac{1}{n}\sum_{j=1}^n (e^{-2\tau^t\lambda_j^t}-1)\|\hat{y}_j\|_2^2 + \frac{1}{n}\sum_{j=1}^n \|\hat{y}_j\|_2^2 - \frac{2}{nm}\left\langle \sum_{i=1}^m x_i, \sum_{j=1}^n y_j \right\rangle \end{aligned} \tag{3.18}$$

$$= \langle M^2,\bar{\gamma}\rangle + \left[\sum_{i=2}^m \left(e^{-2\tau^s\lambda_i^s}-1\right)\|\widehat{x}_i\|_2^2 + \sum_{j=2}^n \left(e^{-2\tau^t\lambda_j^t}-1\right)\|\widehat{y}_j\|_2^2\right]. \tag{3.19}$$

Remark 3.2.2. *The limit of* $\mathtt{DW}$ *in* ∞*, given in Equation 3.4, coincides with the Bures-Wasserstein distance between two Gaussian distributions with same covariance matrices and respective centres* $\frac{1}{m}\sum_i x_i$ *and* $\frac{1}{n}\sum_j y_j$*.*

In the general case, the Bures-Wasserstein distance [75, 76] is the $\mathtt{W}_2$ *distance in the space of Gaussian distributions of* $\mathbb{R}^d$*. For two such distributions, with respective centres* $c_1 \in \mathbb{R}^d$ *and* $c_2 \in \mathbb{R}^d$*, and respective covariance matrices* $U^1 \in \mathbb{R}^{d\times d}$ *and* $U^2 \in \mathbb{R}^{d\times d}$*, we have:*

$$\mathtt{W}_2^2(\mathcal{N}(c_1,U_1),\mathcal{N}(c_2,U_2)) = \|c_1-c_2\|_2^2 + \mathrm{tr}(U) + \mathrm{tr}(V) - 2\,\mathrm{tr}(U^{\frac{1}{2}}VU^{\frac{1}{2}})^{\frac{1}{2}} \tag{3.20}$$

Bound on expected value

Contrary to its non-increasing upper bound $\langle\bar{\gamma},\tilde{M}^2(\tau^s,\tau^t)\rangle$, the squared Diffusion Wasserstein distance $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t)$ may not behave monotonically with τ^s,τ^t. Even though $\mathtt{DW}_2^2(\mu,\nu \mid 0,0) = \mathtt{W}_2^2(\mu,\nu)$ we may have $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t) > \mathtt{W}_2^2(\mu,\nu)$ for some values of τ^s,τ^t. The following gives a sufficient condition to ensure that (in expectation) $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t)$ does not exceed $\mathtt{W}_2^2(\mu,\nu)$.

Proposition 3.2.3. *Consider integers* $m,n,r \geq 1$, $a \in \mathbb{R}_+^m$, $b \in \mathbb{R}_+^n$ *such that* $\sum_i a_i = 1 = \sum_j b_j$*, two random Laplacians* $\mathbf{L}^s \in \mathbb{R}^{m\times m}$, $\mathbf{L}^t \in \mathbb{R}^{n\times n}$ *drawn independently according to possibly distinct probability distributions, two random feature matrices* $\mathbf{X} \in \mathbb{R}^{m\times r}$, $\mathbf{Y} \in \mathbb{R}^{n\times r}$*, and* $0 \leq \tau^s,\tau^t < \infty$*.*

If $\mathbb{E}\,\tilde{M}_{ij}^2(\tau^s,\tau^t) \leq M_{ij}^2\ \forall(i,j)$*, then* $\mathbb{E}\,\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t) \leq \mathtt{W}_2^2(\mu,\nu)$*.*

Remark 3.2.3. *The case where the Laplacians and/or the features are deterministic is covered by considering probability distributions that are Diracs.*

Proof. For brevity we omit the dependency on μ,ν.

$$\mathbb{E}\mathtt{DW}_2^2 = \mathbb{E}\inf_{\gamma\in\Pi(a,b)}\langle\tilde{M}^2,\gamma\rangle \leq \inf_{\gamma}\mathbb{E}\langle\tilde{M}^2,\gamma\rangle = \inf_{\gamma}\langle\mathbb{E}\tilde{M}^2,\gamma\rangle \leq \inf_{\gamma}\langle M^2,\gamma\rangle = \mathtt{W}_2^2. \quad \square \tag{3.21}$$

□

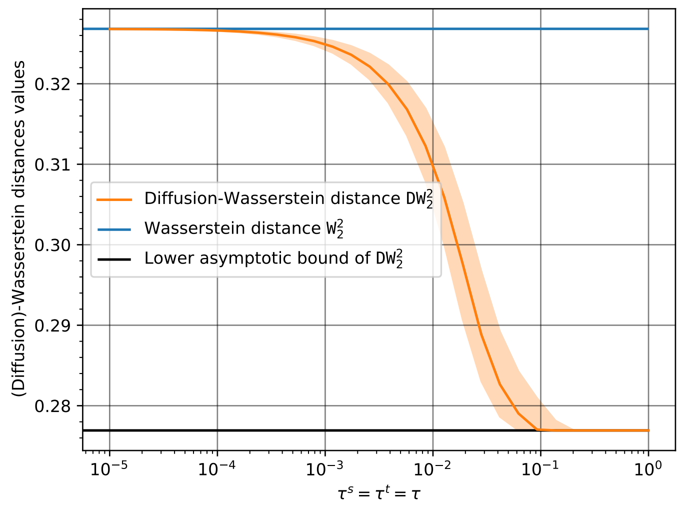


Figure 3.2: Numerical illustration of Proposition 3.2.3, with distance $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t)$ defined in Eq. (3.1). $\mathbb{E}\ \mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t)$ is empirically estimated from 2500 independent realisations of source and target graphs drawn from the same stochastic block model, with $p_{11} = 0.32$, $p_{22} = 0.32$, $p_{12} = p_{21} = 0.02$ and $n = m = 100$. The feature vectors $\mathbf{X} \in \mathbb{R}^m$ and $\mathbf{Y} \in \mathbb{R}^n$ are arbitrarily chosen and remain fixed across all realisations, to restrict randomness only to the structures. Empirical median (solid line) and quartiles 1 and 3 (strip) of $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s = \tau, \tau^t = \tau)$ are plotted against τ and compared to the Wasserstein distance $\mathtt{W}_2^2(\mu,\nu) = \mathtt{DW}_2^2(\mu,\nu \mid 0,0)$ (upper bound) and to the asymptotic regime given in Eq. (3.4), when $\tau \to +\infty$ (lower plateau).

Moreover, by [42, Remark 2.19] we have $\mathtt{W}_2^2(\mu,\nu) \geq \|\frac{1}{m}\sum_{i=1}^m x_i - \frac{1}{n}\sum_{j=1}^n y_j\|_2^2$. If $\mathbf{X}$ and $\mathbf{Y}$ are such that in fact $\mathtt{W}_2^2(\mu,\nu) > \|\frac{1}{m}\sum_{i=1}^m x_i - \frac{1}{n}\sum_{j=1}^n y_j\|_2^2$ then for sufficiently large τ^s, τ^t we must have $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t) < \mathtt{W}_2^2(\mu,\nu)$.

However we can find examples such that $\mathtt{DW}_2^2(\mu,\nu) > \mathtt{W}_2^2(\mu,\nu)$ and $\mathbb{E}\mathtt{DW}_2^2(\mu,\nu) > \mathtt{W}_2^2(\mu,\nu)$ for all $0 < \tau^s, \tau^t < \infty$. For this, it is sufficient to choose $\mathbf{X} = \mathbf{Y}$, so that $\mathtt{W}_2^2(\mu,\nu) = 0$, and deterministic or random graphs and parameters τ^s, τ^t such that $\exp(-\tau^s \mathbf{L}^s)\mathbf{X}$ is not equal (even up to permutation) to $\exp(-\tau^t \mathbf{L}^t)\mathbf{Y}$, so that (almost surely) $\mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau_t) > 0$.

Figure 3.2 illustrates the results of Propositions 3.2.2 and 3.2.3, where we empirically estimated $\mathbb{E}\ \mathtt{DW}_2^2(\mu,\nu \mid \tau^s,\tau^t)$, and plotted its evolution against $\tau = \tau^s = \tau^t$ (experimental conditions are detailed in the legend of Fig. 3.2). Trivially, we verify that $\mathtt{DW}_2^2(\mu,\nu \mid 0,0) = \mathtt{W}_2^2(\mu,\nu)$. But, more importantly, we observe that $\mathbb{E}\ \mathtt{DW}_2^2$ systematically stands below $\mathtt{W}_2^2$, confirming thus the prediction of Proposition 3.2.3, and converges towards the theoretical bound given in Eq. (3.4) of Proposition 3.2.2, when $\tau \to \infty$. Interestingly also, although we know from the counter-example $\mathbf{X} = \mathbf{Y}$ above, that it is not true in general, the trend of $\mathbb{E}\ \mathtt{DW}_2^2$ in Fig. 3.2 seems to validate the conjecture whereby it is often a non-increasing function of the diffusion scale τ. However, we still lack the theoretical conditions that warrant the result of Prop. 3.2.2 on $(\tau^s,\tau^t) \mapsto \langle \bar{\gamma}, \tilde{M}^2(\tau^s,\tau^t)\rangle$ to extend to $\min_{\gamma\in\Pi(a,b)}\langle \gamma, \tilde{M}^2(\tau^s,\tau^t)\rangle$.

3.2.2 Metric property of DW

Recall that DW can be seen as a generalization of the Wasserstein distance W which leverages the diffusion operator over the features. Moreover, it is known that when the cost matrix $M_{ij} \overset{def}{=} [d(x_i, y_j)]$ is associated to a distance d, the Wasserstein distance defines a metric. The next proposition shows that the diffusion does not change this metric property up to a natural condition.

Proposition 3.2.4. *Let $\mathcal{G}^s$ and $\mathcal{G}^t$ be two attributed graphs of respective sizes m and n nodes. Let $X^s = (x_i^s)_{1\leq i\leq m}$ and $X^t = (x_j^t)_{1\leq j\leq n}$ be the associated node features. Let D^s and D^t be the associated diffusion operators.*

Note $\tilde{x}_i^s$ and $\tilde{x}_j^t$ the features of $\mathcal{G}^s$ and $\mathcal{G}^t$ after time τ^s and τ^t (the i-th and j-th rows of D^sX^s and D^tX^t). Let $\mathcal{T}_\tau$ be the operator that maps a distribution supported on a graph's node's features to the same distribution but with support altered by the diffusion: $\mathcal{T}(\mu) = \mathcal{T}(\sum a_i\delta_{x_i}) = \sum a_i\delta_{\tilde{x}_i}$.

For $p \in [1, \infty)$ and $0 \leq \tau^s, \tau^t < \infty$, the Diffusion Wasserstein $\mathtt{DW}_p(\cdot,\cdot \mid \tau^s, \tau^t)$ defines a pseudo-metric: it satisfies all the axioms of a metric, except that $\mathtt{DW}_p(\mu, \nu) = 0$ if, and only if, $\mathcal{T}(\mu) = \mathcal{T}(\nu)$.

Proof. According to Definition 3.1.1, DW is defined between two probability measures $\mu = \sum_{i=1}^m a_i\delta_{(x_i,z_i)}$ and $\nu = \sum_{j=1}^n b_j\delta_{(y_j,z'_j)}$ with (x_i, z_i) and (y_j, z'_j) lying in some joint space $\mathcal{X} \times \mathcal{Z}$ encoding both the feature and the structure information of two source and target vertices, respectively. Since $\mathtt{DW}_p(\mu, \nu) = \mathtt{W}_p(\tilde{\mu}, \tilde{\nu}) = \mathtt{W}_p(\mathcal{T}(\mu), \mathcal{T}(\nu))$, the proposition follows from the metric property of $\mathtt{W}_p(\cdot, \cdot)$. □

Remark 3.2.4. *The condition $\mathcal{T}(\mu) = \mathcal{T}(\nu)$ almost always implies $\mu = \nu$. Consider the following function:*

$$f(\tau^s, \tau^t) = \sum_{i=1}^m \sum_{j=1}^m \|\tilde{x}_i^s - \tilde{x}_j^t\| \tag{3.22}$$

$$= \sum_{i=1}^m \sum_{j=1}^m \|(\exp(-\tau^s\mathbf{L}^s)X^s)_i - (\exp(-\tau^t\mathbf{L}^t)X^t)_j\| \tag{3.23}$$

The zeros of f correspond to couples (τ^s, τ^t) where two atoms $\delta_{\tilde{x}_i^s}$ and $\delta_{\tilde{x}_j^t}$ overlap, and $\tilde{M}$ is not a metric anymore (two distinct nodes have the same features, and therefore distance 0). However f is an analytical function, as a composition of analytical functions (see Equation 3.23). Therefore, all its zeros are isolated, i.e. if $f(\tau^{s\star}, \tau^{t\star}) = 0$, then $\exists r > 0$ such that for all τ^s, τ^t is a ball of radius r around $\tau^{s\star}, \tau^{t\star}$ where $f(\tau^s, \tau^t) \neq 0$.

3.2.3 Algorithmic complexity

The initial diffusion step can be computed with cubic complexity. Computing the diffusion operator $\exp(-\tau\mathbf{L})$ is achieved by diagonalizing $\mathbf{L}$ (cubic complexity) and exponentiation of the diagonal entries. Then computing the diffusion is a simple matrix-vector multiplication (quadratic complexity). Therefore, DW retains the super-cubic complexity of W, as the added diffusion step will not increase its complexity class. If needed, this complexity can be lowered by computing an approximation: we presented in Section 1.3.3 how entropic regularisation can be used for the distance computation step, and we will show in Section 4.1 a polynomial approximation of the diffusion step.

Notice that compared to FGW, our new distance DW allows us to get free from the costly term in $\mathcal{O}(m^2n^2)$ corresponding to the Gromov part of FGW (even though when $p = 2$ one can compute this term more efficiently in $\mathcal{O}(m^2n+n^2m)$ [60]), while still accounting for both the structure and the features of the graphs. Our study on the computational time of the state of the art methods in Chapter 5 will give evidence that DW is a cheaper way to compute a distance encompassing both sources of information.

3.3 Variants

3.3.1 A note on going beyond diffusion

The DW distance can be interpreted as a two-steps process: a pre-processing step (computing a new set of features $Y^{s/t}$ from the graphs $\mathcal{G}^{s/t}$ and their features $X^{s/t}$ using a diffusion process) and a distance computation step (using the W Distance). Sections 3.3.2 and 3.3.3 will look into changing the distance computation step. But the diffusion process could be swapped for another one too. In fact, one could define a whole family of distances, using any graph filter $f : (X, \mathcal{G}) \to Y$ to solve:

$$\min_{\gamma \in \Pi(a,b)} \left\{ \left\langle \gamma, \tilde{C} \right\rangle \right\}, \tag{3.24}$$

$$\text{where } \tilde{C}_{i,j} = d(f(X^s, \mathcal{G}^s)_i, f(X^t, \mathcal{G}^t)_j)^p. \tag{3.25}$$

This is similar to the kernel trick.

Overall, this possibility remains open for future research. Nonetheless, it can be noted that a few other methods fit this paradigm. In particular, the authors of the Wasserstein Weisfeiler-Lehman [14] have a similar approach. They compute a fixed-size embedding for each node using the Weisfeiler-Lehman scheme [12], and compare the graphs using the $\mathtt{W}_1$ distance between uniform distributions supported on these embeddings.

This also resembles the general mechanism of Graph Neural Networks (outlines in Section 1.1.4). It hints at the possibility of using the diffusion operator as the feature aggregator in GCN, though the selection of the diffusion time τ may be a challenge (see Section 4.2 for our approach). To the best of our knowledge this has not been explicitly tried for GCN. Despite the similar

3.3.2 Using entropy regularisation

It is worth noticing that the heat diffusion operator can be seen as a reweighing scheme applied over the node features leading to the new cost matrix $\tilde{M}(\tau^s, \tau^t)$. Therefore the design of our cost matrix can be used for any optimal transport method that uses a pairwise distance matrix. This includes any OT method that adds a regularizer to the original OT problem (Equation 1.15). In particular, entropy regularisation can be used: we note $\mathtt{DW}_\varepsilon$ the entropy-regularized version of $\mathtt{DW}$:

$$\mathtt{DW}_\varepsilon(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle + \varepsilon \sum_{i,j} \gamma_{i,j} \log(\gamma_{i,j}) \right\}. \tag{3.26}$$

This comes in handy when the task of solving the original transport problem 1.15 becomes computationally prohibitive.

However, doing so comes at the loss of some properties of the $\mathtt{DW}$ distance. The limit in 0 of $\mathtt{DW}_\varepsilon$ coincides with the entropy-regularized Wasserstein distance between the distributions. Proof of Propositions 3.2.2 and 3.2.3 cannot be made with entropy regularisation; in particular, we cannot use [42, Remark 2.19] anymore.

Remark 3.3.1. *Any regularisation term designed for* $\mathtt{W}$ *can be used for* $\mathtt{DW}$ *instead, entropic regularisation being just one possible example.*

3.3.3 Extending FGW

It is also possible to apply our cost matrix design to $\mathtt{FGW}$. We call this method the DifFused Gromov Wasserstein distance $\mathtt{DFGW}$:

Definition 3.3.1.

$$\mathtt{DFGW}_p^p(\mu, \nu) = \min_{\gamma \in \Pi(a,b)} \left\{ \sum_{i,j,k,l} \left((1-\alpha)\tilde{M}_{ij}^p + \alpha |C_{ik}^s - C_{jl}^t|^p \right) \gamma_{ij}\gamma_{kl} \right\}, \tag{3.27}$$

where the summation indices are $1 \leq i, k \leq m$ *and* $1 \leq j, l \leq n$ *for a source graph* $\mathcal{G}^s$ *(resp. a target graph* $\mathcal{G}^t$*) of size* m *(resp.* n*), and the dependency on* τ^s,τ^t *and the considered distance* d *is implicit in* $\tilde{M}$*.*

While this method loses any computational advantage over $\mathtt{DW}$ and $\mathtt{FGW}$, it captures in an additional way the structure.

We recall all OT-based methods presented so far in Table 3.1.

Table 3.1: List of OT-based DA methods on attributed graphs used in the manuscript. Also provided are a reference in the bibliography to their definition, the equation number of their definition in this manuscript, and their specificities.

Method	Reference	Eq.	Complexity	Notes
W	[51]	1.15	super-cubic	Only considers attributes.
GW	[60]	2.3	NP-hard	Only considers structure.
FGW	[15]	2.4	NP-hard	$O(n^2m^2)$ to compute tensor.
OT_LAPLACE	[71]	2.7	NP-hard	Does not define a distance.
L1L2_GL	[50]	2.13	NP-hard	Does not use the structure but source labels instead.
DW	[18]	3.1	super-cubic	

Chapter 4

Algorithmic implementation of DW

Computing the DW distance between two attributed graphs is done in two steps. First, the attributes are diffused for a given time τ in the graph. Then, these diffused attributes are used to get a cost matrix to compare the graphs using the Wasserstein distance. In this Chapter, we examine two key parts of this process.

In Section 4.1, we look at how to implement efficiently the diffusion process. It corresponds to work whose ArXiv preprint is available [19]. This work was done in conjunction with Sybille Marcotte during her internship in the DANTE team. We worked on a Chebychev approximation of the diffusion process. We proved a new bound on the approximation error, improving upon the state of the art. Moreover, we showed how part of the computations can be factorized, so that computing the diffusion at multiple times τ is almost as fast as for just one time τ.

In Section 4.2, we look at how to select the hyper-parameter τ. We present a new method inspired by the notion of triplet-loss, where we compare attributed graph to artificial impostors. We show that this method improves upon the circular validation criterion usually employed in Domain Adaptation. It corresponds to work that have been accepted to the international conference ICTAI'21 [20].

4.1 Fast Multiscale Diffusion on Graphs

DW makes heavy use of the diffusion procedure in a graph. We recall that the latter is described by the equation $\frac{dw}{d\tau} = -\mathbf{L}w$ with $w(0) = x$ [24], and admits a closed-form solution $w(\tau) = \exp(-\tau\mathbf{L})x$ involving the *heat kernel* $\tau \to \exp(-\tau\mathbf{L})$, which features the matrix exponential.

Applying the exponential of a matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ to a vector $x \in \mathbb{R}^n$ can be achieved by computing the matrix $\mathbf{B} = \exp(\mathbf{M})$ to compute then the matrix-vector product $\mathbf{B}x$. However, this becomes quickly computationally prohibitive in high dimension, as storing and computing $\mathbf{B}$, as well as the matrix-vector product $\mathbf{B}x$, have cost at least quadratic in n. However computing a matrix such as $\exp(-\tau\mathbf{L})$ is rarely required, as one rather needs to compute its *action* on a given vector. Dropping the requirement to compute the exponential enables faster methods.

Polynomial approximations are a family of methods suited to this task: given a square symmetric matrix $\mathbf{L}$ and a univariate function f, a suitable univariate polynomial p is used to approximate $f(\mathbf{L})$ with $p(\mathbf{L})$. Such a polynomial can depend on both f and $\mathbf{L}$. When the function f admits a Taylor expansion, a natural choice for p is a truncated version of the Taylor series [77]. Other polynomial bases can be used, such as the Padé polynomials. We settled on a truncated Chebyshev polynomial approximation [78, 79] (see [80] for a survey), leading to approximation errors that decay exponentially with the polynomial order K. We build on the fact that polynomial approximations [81] can significantly reduce the computational burden of approximating $\exp(\mathbf{M})x$ with good precision when $\mathbf{M} = -\tau\mathbf{L}$ where $\mathbf{L}$ is sparse positive semi-definite (PSD); this is often the case when $\mathbf{L}$ is the Laplacian of a graph when each node is connected to a limited number of neighbors.

Here, the principle will be to approximate the exponential as a low-degree polynomial in $\mathbf{M}$, $\exp(\mathbf{M}) \approx p(\mathbf{M}) := \sum_{k=0}^{K} a_k \mathbf{M}^k$. Note that several methods use such a decomposition, some requiring the explicit computation of coefficients associated with a particular choice of polynomial basis, others, including Krylov-based techniques, not requiring explicit evaluation of the coefficients but relying on an iterative determination [82] of the polynomial approximation on the subspace spanned by $\left\{x, \mathbf{M}x, \cdots, \mathbf{M}^K x\right\}$.

In this section, we present the Chebyshev approximation of the diffusion procedure, and the two improvements we made. First, we devise a new bound on the approximation error of truncated Chebyshev expansions of the exponential, that improves upon existing works [78, 83, 84]. This avoids unnecessary computations by determining a small truncation order K to achieve a prescribed error. Second, we propose to compute $\exp(-\tau\mathbf{L})$ at different scales $\tau \in \mathbb{R}$ faster, by reusing the calculations of the action of Chebyshev polynomials on x and combining them with adapted coefficients for each scale τ.

Remark 4.1.1. *Note that the work presented here has a wider range of application than diffusion in a graph. The matrix exponential operator has applications in numerous domains, ranging from time integration of Ordinary Differential Equations [85] or network analysis [86] to various simulation problems (like power grids [87] or nuclear reactions [88]) or machine learning [18, 89]. Moreover, multiscale graph representations such as graph wavelets [90] and spectral graph wavelets [91], graph-based machine learning methods [18], graph node embeddings such as GraphWave [92] rely on graph diffusion at different scales, thus implying applications of the matrix exponential of various multiples of the graph Laplacian. But in this section, we will keep the focus on graph diffusion.*

4.1.1 Chebyshev approximation of the exponential function

Here we define Chebyshev polynomials, series and coefficients. We apply this to the exponential function, which will be used later for matrix exponentials.

Chebyshev polynomials and series decomposition

The Chebyshev polynomials of the first kind are characterized by the identity $T_k(\cos(\theta)) = \cos(k\theta)$. They can be computed as $T_0(t) = 1$, $T_1(t) = t$ and using the following recurrence relation:

$$T_{k+2}(t) = 2tT_{k+1}(t) - T_k(t). \tag{4.1}$$

The *Chebyshev series decomposition* of a function $f : [-1, 1] \mapsto \mathbb{R}$ is: $f(t) = \frac{c_0}{2} + \sum_{k\geq 1} c_k \cdot T_k(t)$, where the *Chebyshev coefficients* are:

$$c_k = \frac{2}{\pi} \int_0^{\pi} \cos(k\theta) \cdot f(\cos(\theta))\mathrm{d}\theta. \tag{4.2}$$

Truncating this series yields an approximation of f. For theoretical aspects of the approximation by Chebyshev polynomials (and other polynomial basis) we refer the reader to [93].

Chebyshev series of the exponential

We focus on approximating the univariate transfer function $h_\tau : \lambda \in [0, 2] \mapsto \exp(-\tau\lambda)$, which will be useful to obtain low-degree polynomial approximations of the matrix exponential $\exp(-\tau\mathbf{L})$ for positive semi-definite matrices whose largest eigenvalue satisfies $\lambda_{\max} = 2$ (see Section 4.1.2). This condition is verified for the normalized Laplacian of any graph; in Section 4.1.3 we explain how to work with the combinatorial Laplacian, or any PSD matrix whose largest eigenvalue is not 2.

Using a change of variable: $t = (\lambda - 1) \in [-1, 1]$, $\tilde{h}_\tau(t) = h_\tau(t + 1)$ and the Chebyshev series of $f := \tilde{h}_\tau$ yields:

$$\begin{aligned}\tilde{h}_\tau(t) &= \frac{1}{2}c_0(\tau) + \sum_{k=1}^{\infty} c_k(\tau)T_k(t),\\ c_k(\tau) &= \frac{2}{\pi} \int_0^{\pi} \cos(k\theta) \exp(-\tau(\cos(\theta) + 1))\mathrm{d}\theta.\end{aligned} \tag{4.3}$$

This leads to the following expression for h_τ:

$$h_\tau(\lambda) = \frac{1}{2}c_0(\tau) + \sum_{k=1}^{\infty} c_k(\tau)\tilde{T}_k(\lambda), \tag{4.4}$$

where for any $k \in \mathbb{N}$: $\tilde{T}_k(\lambda) = T_k(\lambda - 1)$.

Truncating the series (4.4) to order K yields a polynomial approximation of h_τ of degree K whose quality can be controlled, leading to a control of the error in approximating the action of $\exp(-\tau\mathbf{L})$ as studied in Section 4.1.2. First we focus on how to evaluate the coefficients c_k defined in Equation (4.3).

Chebyshev coefficients of the exponential

Evaluating numerically the coefficients using the integral formulation (4.3) would be computationally costly, fortunately they are expressed using Bessel functions [94]:

$$c_k(\tau) = 2I_k(\tau) \cdot \exp(-\tau) = 2 \cdot Ie_k(-\tau), \tag{4.5}$$

with $I_k(\cdot)$ the modified Bessel function of the first kind and $Ie_k(\cdot)$ the exponentially scaled modified Bessel function of the first kind.

An alternative way of evaluating these coefficients (still in the particular case of the exponential function) makes use of a recurrence relation between coefficients (not to be confused with the recurrence relation (4.1) between Chebyshev polynomials). As h_τ is $\mathcal{C}^\infty$, we can write h_τ as a Chebyshev series where the k-th coefficient is function of the k-th derivative of h_τ for any $x \in [0, \lambda_{\max}]$. Then it can be proved that for any $k \in \mathbb{N}$ [93]:

$$c_k(\tau) = \frac{2k-2}{\tau} c_{k-1}(\tau) + c_{k-2}(\tau). \tag{4.6}$$

While this recurrence relation is unstable if used as is (the coefficients decay exponentially to zero with alternating signs), this stability issue can be solved by initializing with $c_K(\tau)$ and $c_{K-1}(\tau)$ (computed using the expression (4.5)) for a given order K and working backward [78].

The following lemma applied to $f = \tilde{h}_\tau$ yields another expression of the coefficients (4.3), which will be used to bound the error of the truncated Chebyshev expansion.

Lemma 4.1.1 ([93], Equation 2.91)**.** *Let f be a function expressed as an infinite power series $f(t) = \sum_{i=0}^{\infty} b_i t^i$ and assume that this series is uniformly convergent on $[-1, 1]$. Then, we can express the Chebyshev coefficients of f by:*

$$c_k = \frac{1}{2^{k-1}} \sum_{i=0}^{\infty} \frac{1}{2^{2i}} \binom{k+2i}{i} b_{k+2i}. \tag{4.7}$$

Corollary 4.1.1. *Consider $\tilde{h}_\tau(t) := \exp(-\tau(t+1))$, $t \in [-1, 1]$. The coefficients of its Chebyshev expansion satisfy:*

$$c_k = (-1)^k d_k \bar{c}_k \tag{4.8}$$

$$\bar{c}_k = 2\,(\tau/2)^k \exp(-\tau)(k!)^{-1} \tag{4.9}$$

$$d_k = \sum_{i=0}^{\infty} (\tau/2)^{2i} \frac{k!}{i!(k+i)!}. \tag{4.10}$$

Moreover we have:

$$1 \le d_k \le \min\left(\exp\left(\frac{(\tau/2)^2}{k+1}\right), \cosh(\tau)\right). \tag{4.11}$$

Proof. Denoting $C = \tau/2$, we expand $f(t) = \tilde{h}_\tau(t) = \exp(-2C(t+1)) = \exp(-2C)\exp(-2Ct)$ into a power series:

$$f(t) = \sum_{i=0}^{\infty} \exp(-2C) \frac{(-2C)^i}{i!} t^i.$$

Using Lemma 4.1.1, we obtain for each $k \in \mathbb{N}$:

$$c_k = (-1)^k C^k 2\exp(-2C) \sum_{i=0}^{\infty} C^{2i} \frac{1}{i!(k+i)!} = (-1)^k \bar{c}_k d_k.$$

For any integers k, i we have $k!/(k+i)! \leq \min(1/i!, 1/(k+1)^i)$ and $1/(i!)^2 = \binom{2i}{i}/(2i)! \leq 2^{2i}/(2i)!$ hence

$$\begin{aligned} d_k &= \sum_{i=0}^{\infty} \frac{C^{2i}}{i!} \frac{k!}{(k+i)!} \\ &\leq \min\left(\sum_{i=0}^{\infty} \frac{C^{2i}}{i!} \frac{1}{(k+1)^i}, \sum_{i=0}^{\infty} \frac{C^{2i}}{i!i!} \right) \\ &\leq \min\left(\exp\left(C^2/(k+1)\right), \sum_{i=0}^{\infty} \frac{C^{2i}2^{2i}}{(2i)!} \right) \\ &= \min\left(\exp\left(C^2/(k+1)\right), \cosh(2C) \right). \end{aligned}$$

□

4.1.2 Approximation of the matrix exponential

Chebyshev approximation of the matrix exponential

Consider $\mathbf{L}$ any *PSD matrix* of largest eigenvalue $\lambda_{\max} = 2$ (adaptations to matrices with arbitrary largest eigenvalue will be discussed in Section 4.1.3). To approximate the action of $\exp(-\tau\mathbf{L})$, where $\tau \geq 0$, we use the matrix polynomial $p_K(\mathbf{L})$ where $p_K(\lambda)$ is the polynomial obtained by truncating the series (4.4). The truncation order K offers a compromise between computational speed and numerical accuracy.

The recurrence relations (4.1) on Chebyshev polynomials yields recurrence relations to compute $\tilde{T}_k(\mathbf{L})x = T_k(\mathbf{L} - \mathbf{Id})x$. Given a polynomial order K, computing $p_K(\mathbf{L})x$ requires K matrix-vector products for the polynomials, and $K+1$ Bessel function evaluations for the coefficients. This cost is dominated by the K matrix-vector products, which can be very efficient if $\mathbf{L}$ is a sparse matrix.

Generic bounds on relative approximation errors

Denote p_K the polynomial obtained by truncation at order K of the Chebyshev expansion (4.4). For a given input vector $x \neq 0$, one can measure a relative error as:

$$\epsilon_K(x) := \frac{\|\exp(-\tau\mathbf{L})x - p_K(\mathbf{L})x\|_2^2}{\|x\|_2^2}. \tag{4.12}$$

Expressing $\exp(-\tau\mathbf{L})$ and $p_K(\mathbf{L})$ in an orthonormal eigenbasis of $\mathbf{L}$ yields a worst-case relative error:

$$\epsilon_K := \sup_{x \neq 0} \epsilon_K(x) = \max_i |h_\tau(\lambda_i) - p_K(\lambda_i)|^2 \leq \|h_\tau - p_K\|_\infty^2 \tag{4.13}$$

with $\lambda_i \in [0, \lambda_{\max}]$ the eigenvalues of $\mathbf{L}$ and $\|g\|_\infty := \sup_{\lambda \in [0,\lambda_{\max}]} |g(\lambda)|$.

Lemma 4.1.2. *Consider $\tau \geq 0$, h_τ and $\mathbf{L}$ a PSD matrix with largest eigenvalue $\lambda_{\max} = 2$. Consider p_K as above where $K > \tau/2 - 1$. With $C := {}^\tau\!/_2$ we have*

$$\|h_\tau - p_K\|_\infty \leq 2e^{\frac{(\tau/2)^2}{K+2} - \tau} \frac{(\tau/2)^{K+1}}{K!(K+1-\tau/2)} =: g(K,\tau). \tag{4.14}$$

Proof. Denote $C = \tau/2$. For $K > C - 1$ we have:

$$\begin{aligned}\sum_{k=K+1}^{\infty} \frac{C^k}{k!} &\leq \frac{1}{K!}\sum_{k=K+1}^{\infty} \frac{C^k}{(K+1)^{k-K}} = \frac{C^K}{K!}\sum_{\ell=1}^{\infty}\frac{C^\ell}{(K+1)^\ell} \\ &= \frac{C^{K+1}}{K!(K+1-C)}\end{aligned} \tag{4.15}$$

and $C^2/(K+1) < C$ hence for $k \geq K+1$ (4.11) yields:

$$1 \leq d_k \leq \exp(C^2/(K+2)) \leq \exp(C). \tag{4.16}$$

Since $|T_k(t)| \leq 1$ on $[-1,1]$ (recall that $T_k(\cos\theta) = \cos(k\theta)$), we obtain using Corollary 4.1.1:

$$\begin{aligned}\|h_\tau - p_K\|_\infty &\overset{(4.4)}{=} \sup_{\lambda \in [0,\lambda_{\max}]} \left|\sum_{k>K}^{\infty} c_k(\tau)\tilde{T}_k(\lambda)\right| \leq \sum_{k>K}^{\infty} |d_k \bar{c}_k| \\ &\overset{(4.9),(4.16)}{\leq} \exp\left(\tfrac{C^2}{K+2}\right) 2\exp(-2C)\sum_{k>K}^{\infty}\frac{C^k}{k!} \\ &\overset{(4.15)}{\leq} 2\exp\left(\tfrac{C^2}{K+2} - 2C\right)\frac{C^{K+1}}{K!(K+1-C)}.\end{aligned}$$

□

While (4.12) is the error of approximation of $\exp(-\tau\mathbf{L})x$, *relative to the input energy* $\|x\|_2^2$, an alternative is to measure this error w.r.t. *the output energy* $\|\exp(-\tau\mathbf{L})x\|_2^2$:

$$\eta_K(x) := \frac{\|\exp(-\tau\mathbf{L})x - p_K(\mathbf{L})x\|_2^2}{\|\exp(-\tau\mathbf{L})x\|_2^2}. \tag{4.17}$$

Since $\|\exp(-\tau\mathbf{L})x\|_2 \geq e^{-\tau\lambda_{\max}}\|x\|_2 = e^{-2\tau}\|x\|_2$ we have $\eta_K(x) \leq \|h_\tau - p_K\|_\infty^2 e^{4\tau}$. Using Lemma 4.1.2 we obtain for $K > \tau/2 - 1$ and any x:

$$\epsilon_K(x) \leq g^2(K,\tau); \tag{4.18}$$
$$\eta_K(x) \leq g^2(K,\tau)e^{4\tau}. \tag{4.19}$$

Specific bounds on relative approximation errors

As the bounds (4.18)-(4.19) are worst-case estimates, they may be improved for a specific input signal x by taking into account its properties. To illustrate this, let us focus on graph diffusion where $\mathbf{L}$ is a graph Laplacian, assuming that $a_1 := \sum_i x_i \neq 0$. Since $a_1/\sqrt{n}$ is the inner product between x and the unit constant vector $(1,\ldots,1)/\sqrt{n}$, which is an eigenvector of the graph Laplacian $\mathbf{L}$

associated to the zero eigenvalue $\lambda_1 = 0$, we have $\| \exp(-\tau \mathbf{L})x\|_2^2 \geq |a_1/\sqrt{n}|^2$. For $K > \tau/2 - 1$ this leads to the bound:

$$\eta_K(x) \leq \epsilon_K(x) \frac{\|x\|_2^2}{a_1^2/n} \leq g^2(K,\tau) \frac{n\|x\|_2^2}{a_1^2}. \tag{4.20}$$

This bound improves upon (4.19) if $e^{4\tau} \geq \frac{n\|x\|_2^2}{a_1^2}$, i.e. when

$$\tau \geq \frac{1}{4} \log \frac{n\|x\|_2^2}{a_1^2}. \tag{4.21}$$

Considering a graph with Laplacian $\mathbf{L}$, the diffusion of a graph signal x at scale τ is obtained by computing $\exp(-\tau\mathbf{L})x$. In the general case, the largest eigenvalue of $\mathbf{L}$ is not necessarily $\lambda_{\max} = 2$ (except for example if $\mathbf{L}$ is a so-called *normalized* graph Laplacian, instead of a *combinatorial* graph Laplacian). To handle this case with the polynomial approximations studied in the previous section, we first observe that $\exp(-\tau\mathbf{L}) = \exp(-\tau'\mathbf{L}')$ where $\mathbf{L}' = {}^1\!/\!{}_\Phi\mathbf{L}$, $\tau' = \Phi\tau$ and $\Phi = \lambda_{\max}/2$. Using Equation (4.21) with scale τ' allows to select which of the two bounds (4.19) or (4.20) is the sharpest. The selected bound is then used to find a polynomial order K that satisfies a given precision criterion. Then, we can use the recurrence relations (4.2) to compute the action of the polynomials $\tilde{T}_k(\mathbf{L}') = T_k(\mathbf{L}' - \mathbf{Id})$ on x [79], and combine them with the coefficients $c_k(\tau')$ given by (4.5).

The complete procedure is presented in Algorithm 2 in pseudo-code.

4.1.3 Experiments

We perform four experiments, to highlight the interest of our approach and bounds, and to discuss implementation details.

Bound tightness

Our new bounds can be illustrated by plotting the minimum truncated order K required to achieve a given precision. The new bounds can be compared to the tightest bound we could find in the literature [78]:

$$\eta_K(x) \leq 4E(K)^2 \frac{n\|x\|_2^2}{a_1^2} \tag{4.22}$$

where $a_1 = \sum_i x_i$, and:

$$E(K) = \begin{cases} e^{\frac{-b(K+1)^2}{2\tau}} \left(1 + \sqrt{\frac{\pi\tau/2}{b}}\right) + \frac{d^{2\tau}}{1-d} & \text{if } K \leq 2\tau \\ \frac{d^K}{1-d} & \text{if } K > 2\tau \end{cases} \tag{4.23}$$

with $b = \frac{2}{1+\sqrt{5}}$ and $d = \frac{\exp(b)}{2+\sqrt{5}}$. This bound can be made independent of x by using the same procedure as that of used to establish (4.19):

$$\eta_K(x) \leq 4E(K)^2 \exp(4\tau). \tag{4.24}$$

```
Input: $\mathbf{L} \in \mathbb{R}^{n\times n}$ a PSD matrix, $x \in \mathbb{R}^{n\times d}$ a vector, $T \in \mathbb{R}^m_+$ a set of
       diffusion times, $err$ a target maximum approximation error.
Output: Approximation of $\exp(-\tau\mathbf{L})x$ for all $\tau \in T$

// Define useful quantities
$\Phi \leftarrow {}^{\lambda_{max}}/_2$
$a_1 \leftarrow \sum_i x_i$
$n \leftarrow \text{length}(x)$
$m \leftarrow \text{length}(T)$
$\tau_{max} \leftarrow \max(T)$
$C \leftarrow {}^{\tau_{max}}/_2$

// Compute order $K$
$K \leftarrow \min\left\{K > C-1 \mid g^2(K, \frac{\tau_{max}\Phi}{2})\frac{n\|x\|_2^2}{a_1^2} - err \leq 0\right\}$

// Compute coefficients
for $i \leftarrow 0$ to $K$ do
    for $j \leftarrow 1$ to $m$ do
        $c[i][j] \leftarrow 2 \cdot Ie_i(-T[j]\Phi)$
    end
end

// Compute first two polynomials
$P[0] \leftarrow x$
$P[1] \leftarrow {}^1/_\Phi \mathbf{L}x - P[0]$
for $j \leftarrow 1$ to $m$ do
    $y[j] \leftarrow {}^1/_2 \cdot c[0][j] + c[1][j] \cdot P[1]$
end

// Compute approximation
for $i \leftarrow 2$ to $K$ do
    $P[i] \leftarrow {}^2/_\Phi \mathbf{L}P[i-1] - 2P[i-1] - P[i-2]$
    for $j \leftarrow 1$ to $m$ do
        $y[j] \leftarrow y[j] + c[i[[j] \cdot P[i]$
    end
end
return $y$
```

Algorithm 2: How to compute the diffusion for a matrix, a vector and a set of τ values at a given precision.

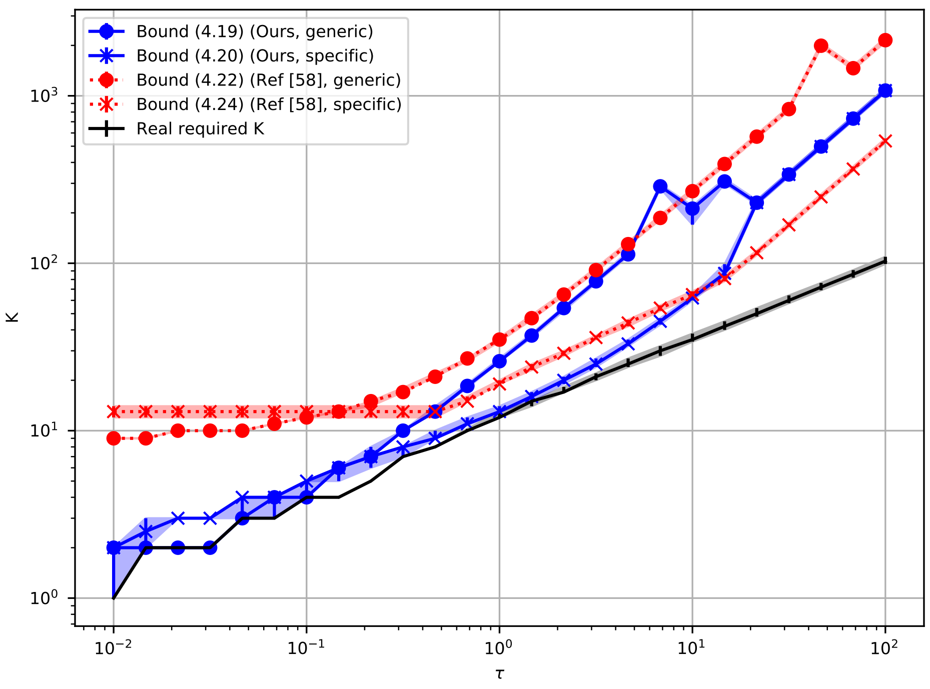


Figure 4.1: Minimum order K to achieve an error $\eta_K(x)$ below 10^{-5}, either real or according to each bound. Median values taken for 100 Erdos-Reyni graphs of size 200 with 5% connection probability, and a centered standard normal distributed signal.

An experiment was performed over 25 values of τ ranging from 10^{-2} to 10^2, 100 samplings of Erdos-Reyni graphs of size $n = 200$, with connection probability $p = 5\%$ (which yields $\lambda_{max} \simeq 20$), and coupled with a random signal with entries drawn i.i.d. from a centered standard normal distribution. For each set of experiment parameters, for each bound, generically noted $B(K, \tau, x)$, the minimum order K ensuring $\eta_K(x) \leq B(K, \tau, x) \leq 10^{-5}$ was computed, as well as the oracle minimum degree K guaranteeing MSE $\eta_K(x) \leq 10^{-5}$. The median values over graph generations are plotted on Fig 4.1 against τ, with errorbars using quartiles.

We observe that our new bounds (blue) follow more closely the true minimum K (black) achieving the targeted precision, up to $\tau \simeq 10$, thus saving computations over the one of [78] (red). Also of interest is the fact that the bounds (4.20)-(4.22) specific to the input signal are much tighter than their respective generic counterparts (4.19)-(4.24).

Comparing the coefficients formulas for speed

Through this section, we devised several ways to compute the Chebyshev coefficients, which we recall here:

- *Integral formulation:* (Eq. 4.3) $c_k(\tau) = \frac{2}{\pi} \int_0^\pi \cos(k\theta) \exp(-\tau\Phi(\cos(\theta) + 1)) \mathrm{d}\theta$.

- *Recurrence relation:* (Eq. 4.6) $c_k(\tau) = \frac{2k-2}{\tau\Phi} c_{k-1}(\tau) + c_{k-2}(\tau)$, whose backward version is $c_k(\tau) = -\frac{2k+2}{\tau\Phi} c_{k+1}(\tau) + c_{k+2}(\tau)$.
- *Bessel functions:*(Eq. 4.5) $c_k(\tau) = 2I_k(\tau) \cdot \exp(-\tau) = 2 \cdot Ie_k(-\tau\Phi)$.

Each of these formulations has only 3 parameters: the order K (the highest k), the diffusion time τ and $\Phi = \lambda_{max}/2$. We will settle by default for $K = 20$, $\tau = 0.1$ and $\Phi = 50$, which are rather typical values for the experiments we performed. Then we will have each parameter vary in a range around these values, to determine which method performs faster. The results are visible on Figure 4.2.

This experiment produces unstable results. Still, we can see that the integral formulation is the slowest one, by a large margin. The two other methods can perform similarly; in the end we settled for the formula using Bessel functions, based on this experiment and others realized earlier.

Comparing each step of the algorithm for speed

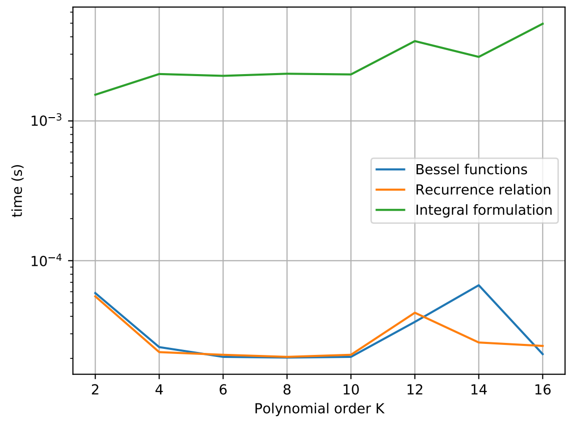
In its current version, the diffusion algorithm devised earlier (Algorithm 2) has 3 main steps: computing the minimal order K to achieve a desired error, computing the Chebyshev coefficients, and computing recursively the polynomials to combine them. Here we perform an experiment to have a sense of which of these 3 steps takes most of the computation time. We sample 100 Erdös-Reyni graphs of size 1000 nodes with connection probability 2%, and for each sample we perform the diffusion at scale $\tau = 0.1$ and for a relative error $\eta_K \leq err \in [2^{-16}, 2^{-3}]$, recording the time taken by each of these 3 steps.

The results are visible on Figure 4.3. We can see that computing the coefficients takes a negligible amount of time compared to the other steps. Reversing the bounds to get an order K sufficient to achieve an error takes a small portion of the time of the algorithm. Computing the polynomials and combining them to get the final values takes most of the time; as expected, this part grows as the desired error goes down, as higher orders K are necessary to ensure smaller errors.

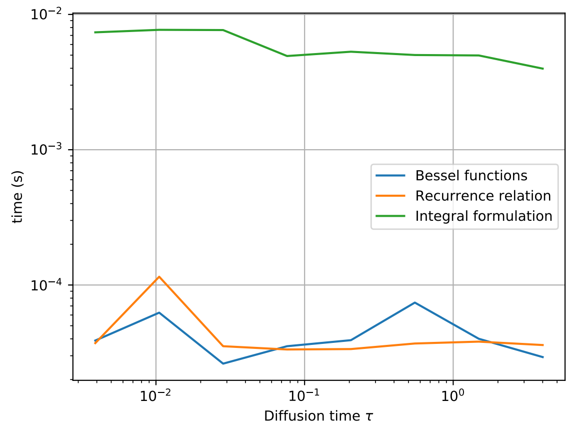
Acceleration of multiscale diffusion

When diffusing at multiple scales $\{\tau_1 \cdots \tau_m\}$, it is worth noting that computations can be factorized. The order K can be computed only once (using the largest τ_i'), as well as $\tilde{T}_k(\mathbf{L}')x$. Eventually, the coefficients can be evaluated for all values τ_i to generate the needed linear combinations of $\tilde{T}_k(\mathbf{L}')x$, $0 \leq k \leq K$. In order to illustrate this speeding-up phenomenon, our method is compared to `scipy.sparse.linalg.expm_multiply`, from the standard `SciPy` Python package, which uses a Taylor approximation combined with a squaring-and-scaling method. See [77] for details.

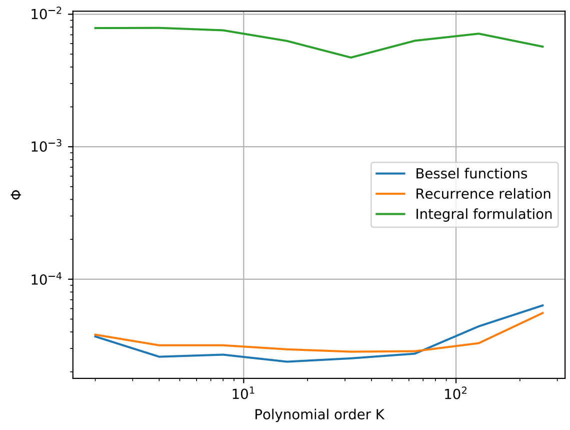
For a first experiment, we take the Standford bunny [95], a graph built from a rabbit ceramic scanning (2503 nodes and 65.490 edges, with $\lambda_{max} \simeq 78$). For the signal, we choose a Dirac located at a random node. We compute repeatedly the diffusion from 2 to 20 scales τ sampled in $[10^{-3}, 10^1]$. Our



(a) Computation time as a function of K.



(b) Computation time as a function of τ.



(c) Computation time as a function of Φ.

Figure 4.2: Computation time of each method to compute the coefficients, with varying parameters. For fixed parameters, we use $K = 20$, $\tau = 0.1$ and $\Phi = 50$

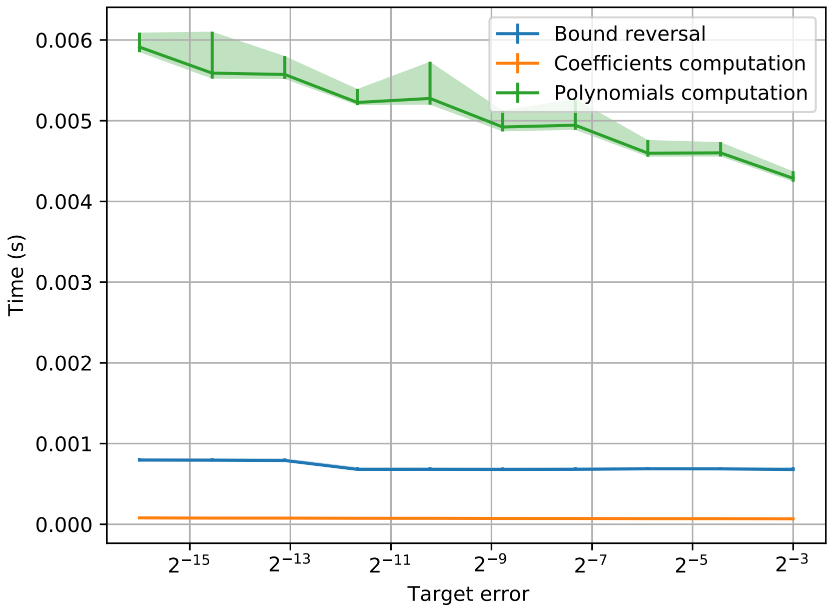


Figure 4.3: Time required to perform the diffusion in a graph, broken down into the three main steps, against the desired approximation error. Median and quartile times are reported. The graph is Erdös-Reyni with 1000 nodes, a 2% connection probability. The diffusion time is $\tau = 0.1$, and the approximation order K is decided by the desired error.

method is set with a target error $\eta_K \leq 10^{-5}$. When the τ values are linearly spaced, both methods can make use of their respective multiscale acceleration. In this context, our method is about twice faster than Scipy's; indeed, it takes 0.36 s plus 6.1×10^{-3} s per scale, while Scipy's takes 0.74 s plus 2.4×10^{-3} s per scale. On the other hand, when the τ values are uniformly sampled at random, SciPy cannot make use of its multiscale acceleration. Indeed, its computation cost increases linearly with the number of τ's, with an average cost of 0.39 s per scale. Whereas, the additional cost for repeating our method for each new τ is negligible (0.0094 s on average) compared to the necessary time to initialize once and for all, the $\tilde{T}_k(\mathbf{L}')x$ (0.30 s).

Figure 4.4 offers a visualization of the experimental setup. A Dirac centered on the bunny's back is diffused according to Scipy's method (top row) and ours with a target precision $\eta_K \leq 10^{-2}$ (bottom row), with $\tau' \in \{0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$. Both methods are indistinguishable to the eye, yet our method executes 60% faster in this context (0.0778s versus 0.220s).

The trend observed here holds for larger graphs as well. We run a similar experiment on the ogbn-arxiv graph from the OGB datasets [96]. Nodes correspond to ArXiV papers published; the structure correspond to citations, i.e. two nodes are linked if one of the articles cite the other; the features are 128-dimensional embeddings of the article's titles and abstracts. We take uniformly sampled scales in $[7.6 \times 10^{-2}, 2.4 \times 10^{-1}]$ (following recommendations of [97]), and set our method for $\eta_K \leq 10^{-3}$. We observe an average computation time of 504 s per scale (i.e. 1 hr and 24 min for 10 scales) for Scipy's

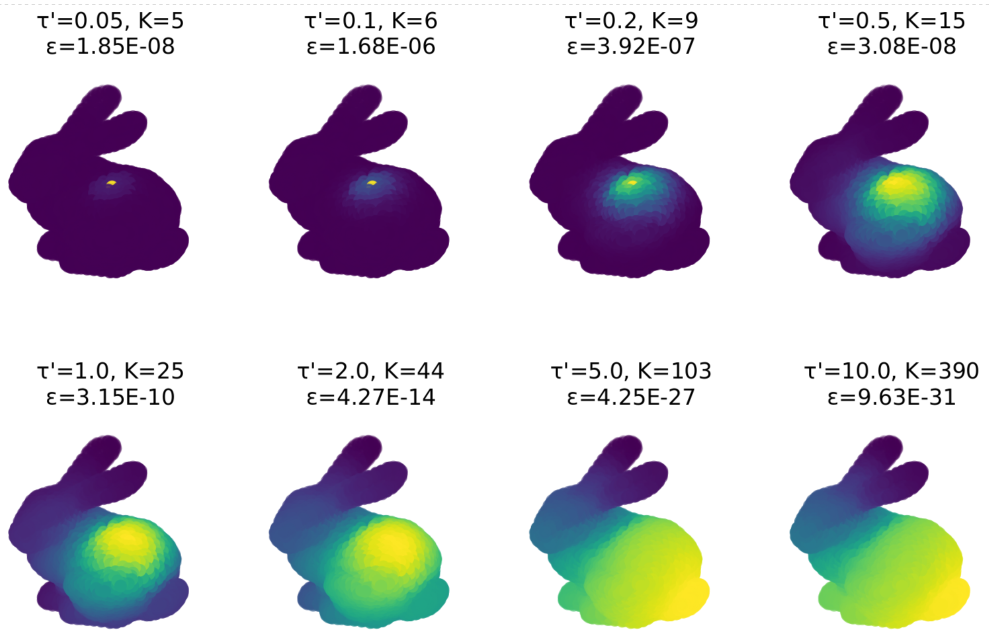


Figure 4.4: Diffusion of a Dirac in the Standford Bunny graph. Our method is set with an error $\eta_K \leq 10^{-2}$; diffusion scale τ', computed order K and error ϵ_K are reported on top of each subfigure.

method, and 87 s plus 50 s per scale for our method (i.e. around 9 min for 10 scales). If we impose a value $\eta_K \leq 2^{-24}$, comparable to the floating point precision achieved by Scipy, the necessary polynomial order K only increases by 6%, which does not jeopardise the computational gain of our method. This behavior gives insight into the advantage of using our fast approximation for addressing the multiscale diffusion on very large graphs.

In this Section 4.1, we presented a method to accelerate the computation of $\exp(-\tau\mathbf{L})$. We showed that a factorization of the computation was possible, which allows getting the diffusion at additional scales τ' much faster. This property makes it possible to solve the optimization problem of finding the optimal τ for DW, where diffusion would have to be repeatedly computed at multiple scales. This is the subject of the next Section 4.2.

4.2 Hyper-parameter τ selection of DW

In this section, we present our strategy to select the hyper-parameter τ in DW. We address this task in the complex setting of unsupervised Domain Adaptation, where we do not have access to labelled target data. This work corresponds to a paper accepted to the international conference ICTAI'21 [20]. We recall the definition of DW:

$$\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle \right\}. \tag{4.25}$$

Unlike the circular validation [98] that aims at tuning hyper-parameters in unsupervised Domain Adaptation by benefiting from pseudo-target labels, we suggest here to directly optimize the diffusion time τ in a self-supervised way.

In this section, we restrict ourselves to $\tau^s = \tau^t$. We propose two possible loss functions of τ, whose minimization yields a suitable parameter $\tau^\star$. They are based around the notion of "impostor", inspired by triplet-based loss functions (used in metric learning, SVM training [99] or image retrieval [100] for instance).

4.2.1 Circular validation

One peculiarity of unsupervised domain adaptation comes from the absence of target labels. In such a setting, a standard method to tune hyper-parameters is the *circular validation* [98], largely used in the past 10 years (see for instance [65,101,102] for examples of applications). The "circular" aspect is due to the fact that the labels go back-and-forth between the source and the target data. Let us detail the underlying principle in the context of an OT-based graph domain adaptation task. Given a transport map γ and a set of labels $l^s \in \mathcal{C}$ for the source graph, one can define pseudo-labels $\tilde{l}^t$ for the target graph by choosing, for each node, the label from which the maximum weight comes from:

$$\tilde{l}^t_j = l^s_{\underset{1\leq i\leq N^s}{\operatorname{argmax}}\{\gamma_{i,j}\}}. \tag{4.26}$$

Note that Like-wise, pseudo-labels for the source graph can be re-inferred in a similar fashion:

$$\tilde{l}^s_i = \tilde{l}^t_{\underset{1\leq j\leq N^t}{\operatorname{argmax}}\{\gamma_{i,j}\}}. \tag{4.27}$$

It is now possible to define an unsupervised score for ranking the transport maps obtained by different sets of hyper-parameters. This score measures the level of agreement between the original and pseudo source labels:

$$s(\gamma) = \frac{|\left\{i \,|\, l^s_i = \tilde{l}^s_i\right\}|}{N^s}. \tag{4.28}$$

It is important to note that while a low score of $s(\gamma)$ is an evidence that the considered hyper-parameter does not lead to a good model, a high score would not allow us to definitely conclude. Indeed, for two graphs of the same size, any permutation matrix γ would produce a perfect score of 1, that can make the circular validation unstable.

The procedure presented here to transport labels from the source domain to target one is not the only one possible. In [103] the authors compute a probability distribution on the target labels by defining $\mathbb{D}_1 \in \{0,1\}^{K\times m}$, where $\mathbf{D}_1(c,i) = 1$ iff $l^s_i = c$. They can then compute a probability vector on the target labels $\mathbb{D}_1\gamma$ that gives at each target node the proportion of mass coming from each class. Pseudo-labels could then be defined as the most likely label (as opposed to the one corresponding to the source point with the most mass transmitted as we do here).

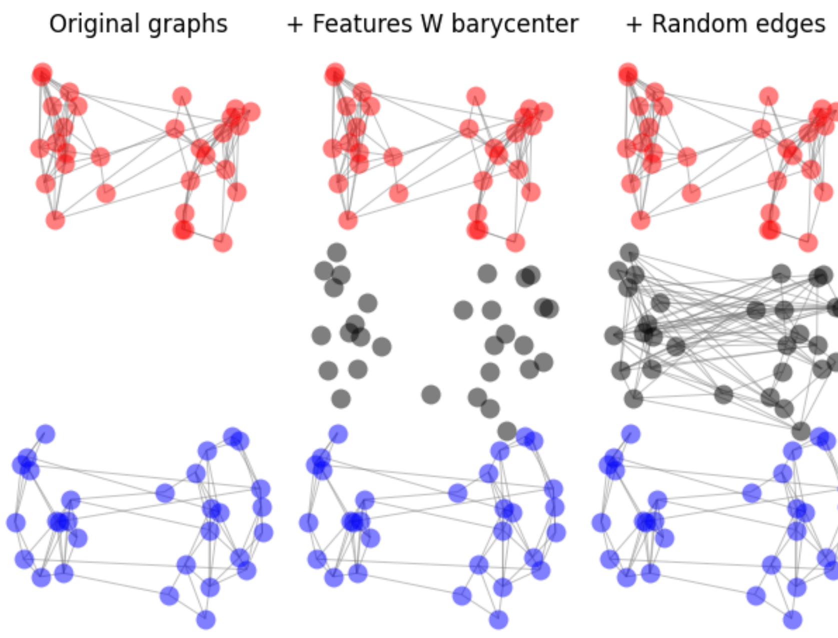


Figure 4.5: Illustration of the construction of the impostor (in grey) of two graphs (in red and blue). Features correspond to the 2D coordinates of the nodes. Impostor nodes are supported on the Wasserstein barycenter (eq. 4.29) of the original graphs' features. Impostor edges are drawn uniformly with probability equal to the average connection probability of the two original graphs.

4.2.2 A triplet-based loss function to learn τ

To address the limitation of the circular validation, we propose in the following to learn the diffusion time by minimizing a loss function that considers impostors attributed graphs. Inspired from the triplet-based constraints used, e.g., in metric learning [104,105], the impostor facilitates the choice of the diffusion time τ that brings $\mathcal{G}^s$ and $\mathcal{G}^t$ close together without suffering from degenerate phenomena.

Definition 4.2.1. *Let $\mathcal{G}^s$ and $\mathcal{G}^t$ be two source and target graphs of size m and n nodes respectively, with their associated probability distributions μ and ν over the nodes. The impostor $\mathcal{G}^0$ with respect to $\mathcal{G}^s$ and $\mathcal{G}^t$ is a graph with $k = \lceil \frac{m+n}{2} \rceil$ nodes whose features X^0 are defined as the minimizer of the following Wasserstein barycenter problem:*

$$X^0 = \underset{X \in \mathbb{R}^{k \times r}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left(\mathtt{W}(X^s, X) + \mathtt{W}(X^t, X) \right) \right\}. \tag{4.29}$$

The adjacency matrix A^0 of $\mathcal{G}^0$ is sampled according to an Erdös-Rényi model [106], with connection probability $p^0 = \frac{p^s + p^t}{2}$ the average connection probability of the two graphs.

The solution of the Wasserstein barycenter problem (Equation 1.3.2) is difficult in practice and is the subject of a rich literature (see [107]). Our problem is a free-support barycenter problem where the main obstacle is that we have to optimize on the support X of the barycenter [58, 108]. It has been recently proved that this problem can be solved in polynomial time when the number of points of each measure is fixed [1]. In our case, we chose to rely on the heuristic proposed in [107] which is reasonable as there are only 2 distributions involved and the weights of the barycenter are considered as fixed (uniform in our case). Overall it boils down to iterating over 1) solving two linear OT problems $\mathtt{W}(X^s, X)$ and $\mathtt{W}(X^t, X)$ 2) updating the support X which can be done in closed-form as detailed in [58] (Equation 8):

$$X \leftarrow (1-\theta)X + \theta {}^1\!/_2 \left(X^s \gamma^{sT} + X^t \gamma^{tT} \right) \mathrm{diag}(a^{-1}), \tag{4.30}$$

where $\theta \in (0,1)$ is fixed or chosen with a line search, γ^s and γ^t are the two transport maps computed at the first step, and $a = ({}^1\!/_k)_{1\leq}$ are the (uniform) weights of the impostor. The procedure for generating the impostor is illustrated in Figure 4.5 for two toy graphs with 2D features.

We now define a triplet-based loss function that uses the notion of impostor given in Definition 4.2.1.

Definition 4.2.2. *Let $\mathcal{G}^s$ and $\mathcal{G}^t$ be two attributed graphs. Define:*

$$\mathcal{L}_1(\tau) = \mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau) - \left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^0 \mid \tau) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^0 \mid \tau) \right). \tag{4.31}$$

The diffusion parameter can be now defined as the solution of the following optimization problem:

$$\tau^* = \underset{\tau \geq 0}{\mathrm{argmin}} \left\{ \mathcal{L}_1(\tau) \right\}, \tag{4.32}$$

Intuitively, like in metric learning, the idea is to learn the parameters of a model (here, a unique diffusion time τ) that (i) constrains $\mathcal{G}^s$ and $\mathcal{G}^t$ to get closer while (ii) preventing a scenario facilitating the bringing together of $\mathcal{G}^s$ and $\mathcal{G}^t$ with a “different” graph. However, unlike supervised metric learning where the impostors of a pair of points of the same class can be defined as the training samples of the opposite label in the close neighbourhood, the notion of “different” is here ill-defined because of the absence of target labels in DA tasks. By using $\mathcal{G}^0$ as defined above, we generate an impostor sufficiently close in terms of features (defined as the Wasserstein barycenter) and structure (mean structure of $\mathcal{G}^s$ and $\mathcal{G}^t$ in terms of vertices) forcing the identification of a parameter τ which aligns as well as possible $\mathcal{G}^s$ and $\mathcal{G}^t$ while ensuring a certain “margin” to a different but related graph.

4.2.3 A quadruplet-based loss function to learn τ

We introduce a second loss function, with a similar design to $\mathcal{L}_1$. Two impostors are used instead of one. Theoretical and practical comparison of these two losses is done in Section 4.2.4.

Definition 4.2.3. *Let $\mathcal{G}$ be a graph of size n nodes, with node features X and associated probability distributions μ over the nodes. The impostor $\mathcal{G}^\dagger$ with respect to $\mathcal{G}$ is a graph with $n^\dagger = n$ nodes. Its features are equal to the original graph $X^\dagger = X$. The adjacency matrix $A^\dagger$ of $\mathcal{G}^\dagger$ is sampled according to an Erdös-Rényi model [106], with connection probability $p^\dagger = p$ the average connection probability of $\mathcal{G}$.*

Similarly to $\mathcal{L}_1$, we define $\mathcal{L}_2$ with the second definition of impostor.

Definition 4.2.4. *Let $\mathcal{G}^1$ and $\mathcal{G}^2$ be two attributed graphs. Define:*

$$\mathcal{L}_2(\tau) = \mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau) - \frac{1}{2}\Big(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^{t\dagger} \mid \tau) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^{s\dagger} \mid \tau)\Big). \tag{4.33}$$

The diffusion time τ can be defined as the solution of the following optimization problem:

$$\tau^* = \underset{\tau \geq 0}{\operatorname{argmin}} \left\{\mathcal{L}_2(\tau)\right\}, \tag{4.34}$$

Remark 4.2.1. *With the two definitions of impostor, we defined two possible losses: $\mathcal{L}_1$ (in Equation 4.31) that uses 1 impostor, and $\mathcal{L}_2$ (in Equation 4.33) that uses 2 impostors. Note that their mathematical definitions have a similar shape, but use different definitions of impostor and $\mathcal{L}_2$ has a factor $1/2$.*

The intuition behind $\mathcal{L}_2$ is the same as $\mathcal{L}_1$. Minimizing this loss promotes closeness between $\mathcal{G}^s$ and $\mathcal{G}^t$, while keeping $\mathcal{G}^s$ away from a "false" version of $\mathcal{G}^t$ (and vice-versa). Because an impostor is a copy of a graph with "shuffled" links, minimizing the loss prevents ignoring the structure, as $\mathcal{G}^s$ has to get close to $\mathcal{G}^t$ but far from its copy with no underlying structure (random edges).

4.2.4 Analysis of the loss functions

In this section, we provide a thorough analysis of our loss functions. First, we illustrate the correlation between the minimizer of Equations 4.31 and 4.33 and the corresponding accuracy on a DA task. Then, after the derivation of some theoretical properties, we give evidence that they are very robust to some approximations aiming at reducing their algorithmic complexity.

Correlation between minimum of the loss and DA accuracy

The coupling matrix γ resulting from the calculation of $\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau)$ with τ the solution of Problems 4.31 or 4.33 can be directly used for addressing a graph DA task. Given γ, each target node is assigned the class corresponding to the most mass transported to it: $\underset{cl \in [1,K]}{\operatorname{argmax}} \{\sum_i \gamma_{i,j} | l_i = cl\}$. According to this classification rule, one can compute the DA accuracy measuring the level of agreement between the predicted and the expected labels.

In Figure 4.6, we illustrate the correlation between $\mathcal{L}_1(\tau)$ and $\mathcal{L}_2(\tau)$ and the DA accuracy from two source and target synthetic community graphs, made of

200 nodes and 3 classes. A community graph is built by (i) assigning a random class to each node, (ii) generating 2D features using a Gaussian distribution with a different center for each class, (iii) connecting all points of the same class that are closer than some radius r, and (iv) linking points of different classes at random with a small connection probability. The two resulting source and target graphs are presented in Figure 4.6a. In Figure 4.6b and Figure 4.6c, we plot the losses $\mathcal{L}_1(\tau)$ and $\mathcal{L}_2(\tau)$, and the accuracy obtained from a large range of τ. The most interesting point is that the global minimum of the losses (which are not convex in general with potentially several local minima) corresponds to the parameter τ yielding the maximum accuracy. On the other hand, the global maximum of $\mathcal{L}_1(\tau)$ and $\mathcal{L}_2(\tau)$ matches with the worst behaviour in DA. Therefore, this correlation between loss and accuracy confirms the interest of our losses for addressing graph-based optimal transport tasks with Diffusion Wasserstein distances.

Theoretical properties

Given the definition of the loss functions $\mathcal{L}_1(\tau)$ and $\mathcal{L}_2(\tau)$, we can derive the following properties.

Theorem 4.2.1. *Let $\mathcal{G}^s$ and $\mathcal{G}^t$ be two connected attributed graphs. The loss function $\mathcal{L}_1(\tau)$ of Eq. (4.31) is continuous, negative, and its limits are:*

$$\lim_{\tau\to 0}\mathcal{L}_1(\tau) = \lim_{\tau\to\infty}\mathcal{L}_1(\tau) = 0. \tag{4.35}$$

Proof. Continuity stems from continuity of all the functions involved.

Then, writing X^u_τ the distribution of the features of a graph $\mathcal{G}^u$ diffused for a time τ ($u \in \{0, s, t\}$), we have:

$$\begin{aligned}\mathcal{L}_1(\tau) &= \mathtt{DW}_p(\mathcal{G}^s,\mathcal{G}^t \mid \tau) - \Big(\mathtt{DW}_p(\mathcal{G}^s,\mathcal{G}^0 \mid \tau) + \mathtt{DW}_p(\mathcal{G}^t,\mathcal{G}^0 \mid \tau)\Big) &(4.36)\\ &= \mathtt{W}(X^s_\tau, X^t_\tau) - (\mathtt{W}(X^s_\tau, X^0_\tau) + \mathtt{W}(X^0_\tau, X^t_\tau)). &(4.37)\end{aligned}$$

$\mathtt{W}$ being a distance, negativity follows from triangle inequality.

Now, denote X^s and X^t the features of $\mathcal{G}^s$ and $\mathcal{G}^t$, and X^0 the features of the impostor $\mathcal{G}^0$. By definition of X^0 as a minimizer:

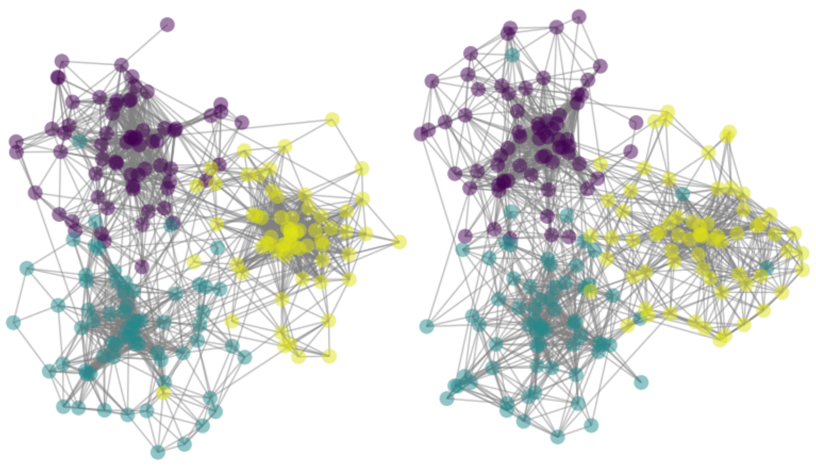
$$\begin{aligned}\mathtt{W}(X^s, X^0) + \mathtt{W}(X^t, X^0) &\le \mathtt{W}(X^s, X^t) + \mathtt{W}(X^t, X^t) &(4.38)\\ &\le \mathtt{W}(X^s, X^t), &(4.39)\end{aligned}$$

and by triangle inequality we have:

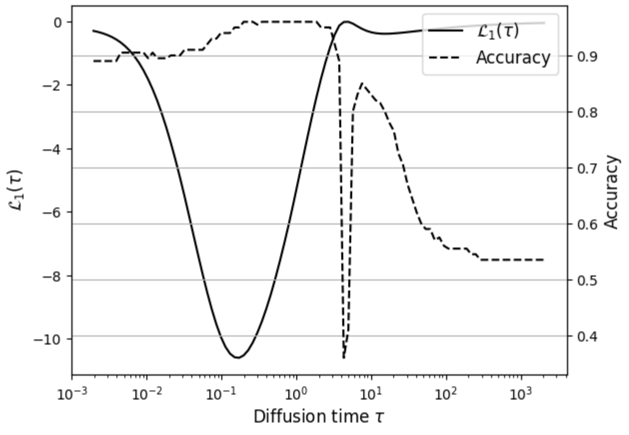
$$\mathtt{W}(X^s, X^0) + \mathtt{W}(X^t, X^0) \ge \mathtt{W}(X^s, X^t). \tag{4.40}$$

By combining Eq.(4.38) and (4.40), we get:

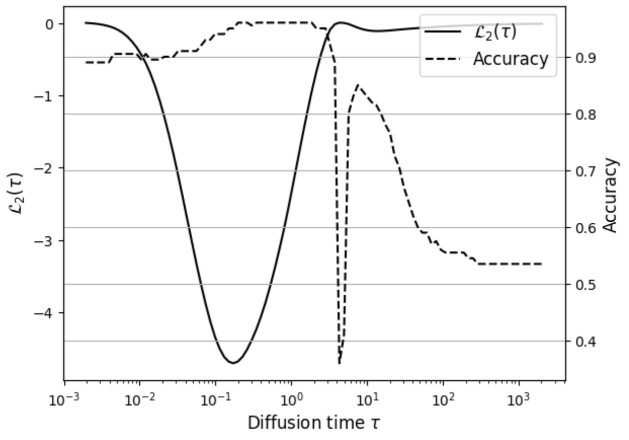
$$\mathcal{L}_1(0) = \mathtt{W}(X^s, X^t) - (\mathtt{W}(X^s, X^0) + \mathtt{W}(X^t, X^0)) = 0. \tag{4.41}$$



(a) Source (left) and target (right) community graphs



(b) $\mathcal{L}_1(\tau)$ and DA accuracy as a function of τ.



(c) $\mathcal{L}_2(\tau)$ and DA accuracy as a function of τ.

Figure 4.6: Correlation between $\mathcal{L}_1(\tau)$ and $\mathcal{L}_2(\tau)$ and the DA accuracy: (a) source and target community graphs $\mathcal{G}^s$ and $\mathcal{G}^t$; (b) the global minimum of $\mathcal{L}_1(\tau)$ corresponds to the maximum accuracy reachable in a DA task; (c) the global minimum of $\mathcal{L}_2(\tau)$ corresponds to the maximum accuracy reachable in a DA task.

Finally, [42], Remark 9.1, states that the expected value of the Wasserstein barycenter is the barycenter of the expected values, so we get:

$$\mathbb{E}[X^0] = \frac{1}{2}\left(\mathbb{E}[X^s] + \mathbb{E}[X^t]\right). \tag{4.42}$$

From [18], Proposition 1, we have the limit of $\texttt{DW}_2$:

$$\lim_{\tau\to\infty} \texttt{DW}_2(\mathcal{G}^s, \mathcal{G}^t) = \|\mathbb{E}[X^s] - \mathbb{E}[X^t]\|_2. \tag{4.43}$$

Therefore, we know that the limit of $\mathcal{L}_1(\tau)$ exists and is:

$$\begin{aligned}\lim_{\tau\to\infty} \mathcal{L}_1(\tau) &= \|\mathbb{E}[X^s] - \mathbb{E}[X^t]\|_2 \\ &- \|\mathbb{E}[X^s] - \mathbb{E}[X^0]\|_2 - \|\mathbb{E}[X^t] - \mathbb{E}[X^0]\|_2 \\ &= \|\mathbb{E}[X^s] - \mathbb{E}[X^t]\|_2 \\ &- {}^1\!/\!_2\|\mathbb{E}[X^s] - \mathbb{E}[X^t]\|_2 - {}^1\!/\!_2\|\mathbb{E}[X^s] - \mathbb{E}[X^t]\|_2 \\ &= 0. \end{aligned} \tag{4.44}$$

□

Theorem 4.2.2. *Let $\mathcal{G}^s$ and $\mathcal{G}^t$ be two connected attributed graphs. The loss function $\mathcal{L}_2(\tau)$ of Eq. (4.33) is continuous, and its limits are:*

$$\lim_{\tau\to 0} \mathcal{L}_2(\tau) = \lim_{\tau\to\infty} \mathcal{L}_2(\tau) = 0. \tag{4.45}$$

Proof. Like $\mathcal{L}_1$, continuity stems from the continuity of all the functions involved.

Next, denotes X^s, X^t, $X^{s\dagger}$ and $X^{t\dagger}$ the features of $\mathcal{G}^s$, $\mathcal{G}^t$, $\mathcal{G}^{s\dagger}$ and $\mathcal{G}^{t\dagger}$ respectively. By unrolling the definition of $\mathcal{L}_2$ in 0 we have:

$$\mathcal{L}_2(0) = \texttt{W}(X^s, X^t) - {}^1\!/\!_2(\texttt{W}(X^s, X^{t\dagger}) + \texttt{W}(X^t, X^{s\dagger})) \tag{4.46}$$

$$= \texttt{W}(X^s, X^t) - {}^1\!/\!_2(\texttt{W}(X^s, X^t) + \texttt{W}(X^t, X^s)) \tag{4.47}$$

$$= 0. \tag{4.48}$$

Finally, by writing $x^s = {}^1\!/\!_m \sum_i X^s_i = {}^1\!/\!_m \sum_i X^{s\dagger}_i$ and $x^t = {}^1\!/\!_n \sum_i X^t_i = {}^1\!/\!_n \sum_i X^{t\dagger}_i$ we have:

$$\lim_{\tau\to\infty} \mathcal{L}_2(\tau) = d(x^s, x^t) - {}^1\!/\!_2(d(x^s, x^t) + d(x^t, x^s)) \tag{4.49}$$

$$= 0. \tag{4.50}$$

□

Thanks to Theorems 4.2.1 and 4.2.2, the proposed losses prevent the trivial values $\tau = 0$ or $\tau \to \infty$ from being chosen.

Approximations of $\mathcal{L}_1(\tau)$ and $\mathcal{L}_2(\tau)$

The algorithmic complexity of the losses $\mathcal{L}_1$ and $\mathcal{L}_2$ depends on three main elements: (i) the exponential of the graph Laplacians, (ii) the Wasserstein distance W on the diffused source and target features and (iii) the size of the impostor. Different strategies can be used to simplify the overall complexity, including (but not exhaustively) the entropic regularisation [41], the reduction of the impostor sizes, or a Chebychev approximation of the diffusion [19] (see also Section 4.1). In the following, we study the capacity of our losses to resist these approximations and thus to provide a similar global minimum.

Entropic regularisation The entropic regularisation of W [41], as defined in Eq. (1.16), allows to overcome the limitations of the original problem due to the super-cubic complexity, the instability and the non uniqueness of the solution. The entropy-regularized version is several orders of magnitude faster (an η-approximation is computed in $O(n^2 \log(n)\eta^{-3})$ [100]). Using the same graphs as those of Figure 4.6a, Figures 4.7a and 4.7b show the magnitude of the losses $\mathcal{L}_1$ and $\mathcal{L}_2$ for different values of the regularisation parameter ε. We can see that whatever the value of ε the shape of the resulting loss does not change much, meaning that the global minimum stays very close to the one corresponding to the maximum accuracy (see the dashed-line in Fig. 4.6).

Chebychev approximation of the diffusion. Instead of calculating the exact heat kernel applied to the features, one can resort to polynomial approximations [93] of the diffusion. Following [19], and benefiting from our contributions detailed in Section 4.1, we applied a Chebychev approximation of the exponential operator where the truncation order (degree of the final approximating polynomial) offers a trade-off between accuracy and computational speed. An illustration of the effect of various truncation orders on the losses $\mathcal{L}_1$ and $\mathcal{L}_2$ is given in Figures 4.7c and 4.7d. Once again, the global minimum appears to be very robust to changes in the degree of the approximation.

Note that contrary to what was done in Section 4.1, we do not aim a good approximation by defining a reasonable approximation error, but instead show that even an aggressively low degree can still be enough to select τ.

Size of the impostor for $\mathcal{L}_1$. While Def. 4.2.1 suggests to set the size of $\mathcal{G}^0$ to $\lceil \frac{m+n}{2} \rceil$, we study here the impact of reducing the number of nodes of $\mathcal{G}^0$ before computing the Wasserstein barycenter. The behaviour of the loss $\mathcal{L}_1(\tau)$ is reported in Figure 4.7e for different reduction ratios (from 2 to 10). Interestingly, we can observe that even though the curves get smoother as the size of the impostor decreases, the global minimum changes very little, pointing to a relatively stable value for τ. Note that the connection probability of $\mathcal{G}^0$ has to adapt to the reduction ratio. For instance, if the size is reduced by a factor of 2, p^0 must be doubled.

Size of the impostor for $\mathcal{L}_2$. In Definition 4.2.3, the impostors have the same size as the initial graphs, as they are defined as copies with shuffled links. While the impostor construction procedure is not as time consuming as for $\mathcal{L}_1$, computing the loss $\mathcal{L}_2$ itself repeatedly may still be too computationally heavy. Similarly to $\mathcal{L}_1$, the impostor sizes can be decreased, by subsampling the nodes and increasing the connection probability proportionally. The behaviour of the loss $\mathcal{L}_2(\tau)$ is reported in Figure 4.7e for different reduction ratios (from 2 to 10). We observe again that the global minimum is not affected much by this technique, unless the reduction factor is too high. Note that an alternative technique is to compute the impostors at full size, and subsample nodes at computation time after diffusion. Computing a transport distance with a reduced randomized sample of the available points have been shown to yield good approximations for any transport distance [109].

Using two values of τ

While the definition of DW allows two diffusion times, τ^s for the source and τ^t for the target, both our losses $\mathcal{L}_1$ and $\mathcal{L}_2$ assume $\tau^s = \tau^t$. There are multiple ways to extend $\mathcal{L}_1$ and $\mathcal{L}_2$ to $\mathbb{R}_+^2 \mapsto \mathbb{R}$ such that minimizing them yields one value of τ per graph. We explored 5 variations of our losses, that diffuse for some time τ^s on $\mathcal{G}^s$ and τ^t on $\mathcal{G}^t$, and some combination/permutation of τ^s and τ^t for the impostors. They are:

$$\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau^s, \tau^t) - \left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^0 \mid \tau^s, \frac{\tau^s + \tau^t}{2}) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^0 \mid \tau^t, \frac{\tau^s + \tau^t}{2})\right) \tag{4.51}$$

$$\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau^s, \tau^t) - \left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^0 \mid \tau^s, \tau^s) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^0 \mid \tau^t, \tau^t)\right) \tag{4.52}$$

$$\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau^s, \tau^t) - \left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^0 \mid \tau^s, \tau^t) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^0 \mid \tau^t, \tau^s)\right) \tag{4.53}$$

$$\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau^s, \tau^t) - \frac{1}{2}\left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^{t\dagger} \mid \tau^s, \tau^s) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^{s\dagger} \mid \tau^t, \tau^t)\right) \tag{4.54}$$

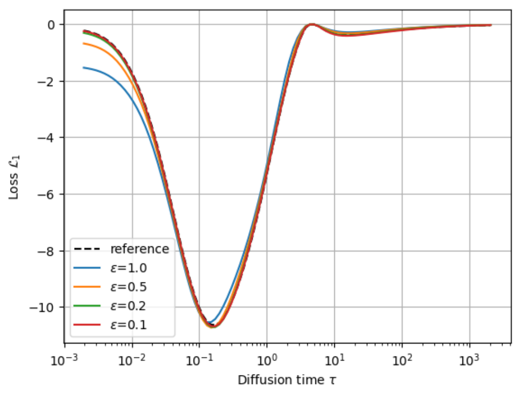
$$\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^t \mid \tau^s, \tau^t) - \frac{1}{2}\left(\mathtt{DW}_p(\mathcal{G}^s, \mathcal{G}^{t\dagger} \mid \tau^s, \tau^t) + \mathtt{DW}_p(\mathcal{G}^t, \mathcal{G}^{s\dagger} \mid \tau^s, \tau^t)\right) \tag{4.55}$$

However, we are not able to conclude at this time if any of these methods are a reliable way to select τ^s and τ^t. When plotting these functions on synthetic data, most of them do not present a clear global minimum that correlates with the maximum of DA accuracy.

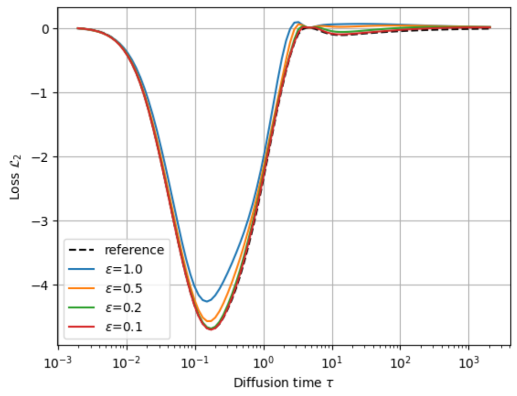
4.3 Conclusion

Computing DW is costly, as it inherits the super-cubic complexity of W and the cubic complexity of the diffusion process. It also introduces a new hyperparameter τ to control the trade-off between the features and the structure. In this chapter, we presented two contributions:

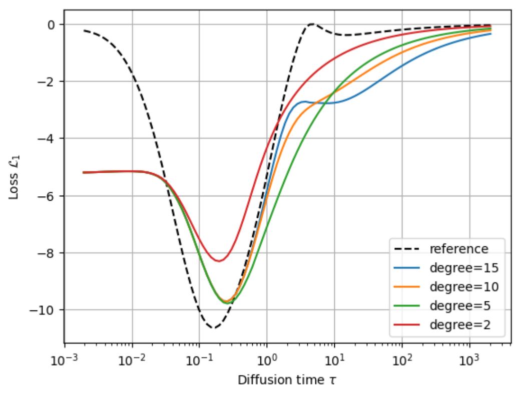
1. We proposed a Chebychev approximation of the diffusion process. We proved a new bound on the approximation error that improves upon



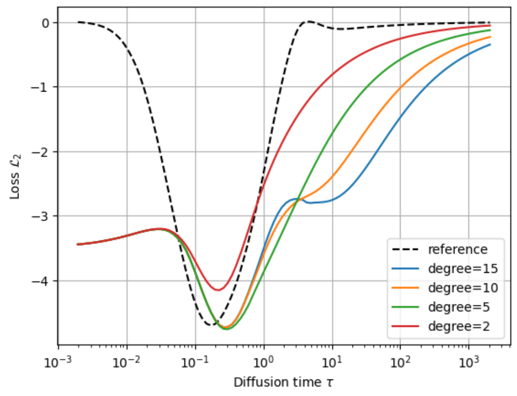
(a) Entropy regularisation for $\mathcal{L}_1$.



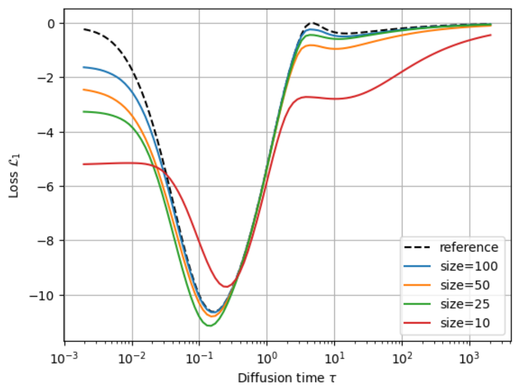
(b) Entropy regularisation for $\mathcal{L}_2$.



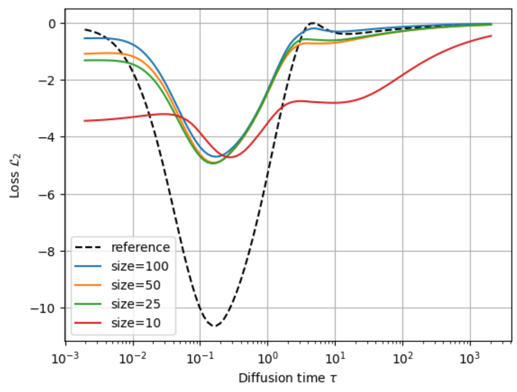
(c) Chebychev polynomials for $\mathcal{L}_1$.



(d) Chebychev polynomials for $\mathcal{L}_2$.



(e) Impostor downsizing for $\mathcal{L}_1$.



(f) Impostor downsizing for $\mathcal{L}_2$.

Figure 4.7: Behavior of the losses $\mathcal{L}_1$ and $\mathcal{L}_2$ computed from the two source and target Community Graphs of Fig. 4.6a in three approximation scenarios: (a) effect of the entropic regularisation used in the inner `DW` distance for various values of the regularisation parameter ε; (b) effect of downsizing the number of nodes of the impostor; (c) effect of approximating the diffusion process with different Chebychev polynomials. The dashed line represents the exact $\mathcal{L}_1$ or $\mathcal{L}_2$.

the state of the art, allowing to select a smaller polynomial order and thus saving more computation time. We also showed how part of the computations can be reused when computing the diffusion for the same attributed graph at different diffusion times. This factorization enables optimizing `DW` over τ, which is crucial for the next contribution.

2. We designed a new scheme to select the hyperparameter τ in a Domain Adaptation context. It is based on the notion of impostor, and consists in minimizing a triplet-based or quadruplet-based loss function. We showed on synthetic data how the proposed losses coincide with the maximum of the DA accuracy. We also proposed various methods to alleviate the additional cost of having to repeatedly compute `DW` distances to minimize the proposed losses.

Chapter 5

Experimental study

On est trop souvent imprécis lorsqu'on fait une citation.

Quelqu'un, un jour.

This chapter gathers most of the experimental studies we performed. In Section 5.1 we present the synthetic data model often used in our experiments (the Contextual Stochastic Block Model), and analyse the circular validation criterion to show its limits. In Section 5.2, we perform Domain Adaption experiments on synthetic and real data, comparing a wide variety of OT-based methods. We show that DW outperforms the other OT-based methods when using our heuristics to select the hyperparameter τ (defined in Section 4.2).

5.1 Preliminary analysis

5.1.1 Synthetic Data: Contextual Stochastic Block Model

In the following experiments, we use a Contextual Stochastic Block Model [110] (CSBM) for both the source and target data. These graphs have an underlying block structure: at generation, each node is assigned a label, and the labels are only used to generate connections and attributes. This way, nodes in the same block (with the same underlying label) are statistically more closely related that nodes in different blocks.

We describe how these graphs are generated. The first step is to generate the labels. In practice we only use 2 labels $+1$ and -1, sampled with probability $1/2$ each. But any number of labels K is possible, and label distribution is not necessarily uniform.

The structure of these graphs follows a Stochastic Blockk Model [111] (SBM). Edges are sampled at random using a matrix $M \in [0,1]^{K\times K}$: the probability of two nodes n_i and n_j of labels l_i and l_j being connected is M_{l_i,l_j}. We use $M = \begin{pmatrix} c_{+1} & c_0 \\ c_0 & c_{-1} \end{pmatrix}/N$, where c_{+1}, c_{-1} and c_0 are constants and N is the

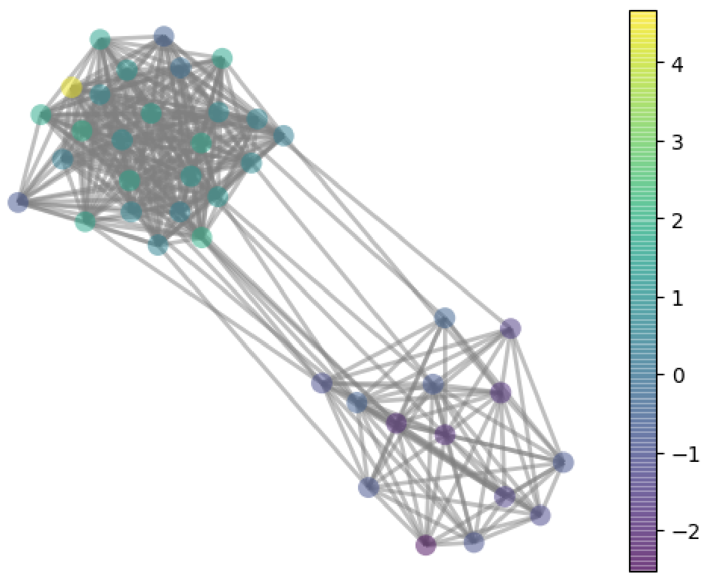


Figure 5.1: Example of a CSBM graph

graph's size. The $1/N$ factor is used to control the edge's density regardless of the graph's size.

The attributes are sampled according to a d-dimensional Gaussian model, where the Gaussian's center depends on the label l of a node. We adopt a slightly simpler approach than the original CSBM model with 1d attributes for ease of visualisation and analysis (later on in Sections 5.2.2, real data will possess multi-dimensional features). Attributes are sampled according to the law $\mathcal{N}(l, \sigma)$, where l is the node's label and σ a parameter of the experiment.

An example of a CSBM graph is plotted in Figure 5.1. The node's positions in the figure are arbitrary: they are chosen to separate the two blocks in space. The features are represented by the color of the nodes: darker nodes have a higher associated feature (we use 1-dimensional features). The labels are not represented, but the two groups are still clearly visually separable in the figure.

5.1.2 Limit of Circular Validation

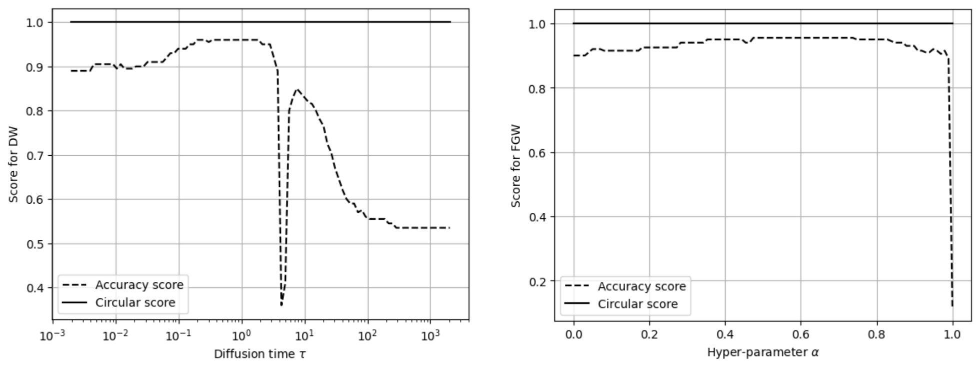
In this section, we perform a simple experiment to highlight how the circular validation criterion [98] can fail to properly assess the quality of a transport map.

We recall the definition of the circular validation score from Section 4.2.1. From the source labels l^s and a transport map γ, the pseudo-labels for the target $\tilde{l}^t_j$ are:

$$\tilde{l}^t_j = l^s_{\underset{1\leq i\leq N^s}{\operatorname{argmax}}\{\gamma_{i,j}\}}. \tag{5.1}$$

Then, the pseudo-labels for the source $\tilde{l}^s_i$ are:

$$\tilde{l}^s_i = \tilde{l}^t_{\underset{1\leq j\leq N^t}{\operatorname{argmax}}\{\gamma_{i,j}\}}. \tag{5.2}$$



(a) Score comparison for DW against hyper-parameter τ.

(b) Score comparison for FGW against hyper-parameter α.

Figure 5.2: Comparison of the (unsupervised) circular validation score and the (supervised) target graph labelling accuracy of two OT-based methods for CSBM data. Graphs are generated following the same CSBM block model; its default parameters are $N = 200$, $p_{+1,+1} = p_{-1.-1} = 0.4$, $p_{+1,-1} = p_{-1,+1} = 0.1$ and $\sigma = 2$.

Finally, the score is defined as the accuracy between the pseudo and the real source labels:

$$s(\gamma) = \frac{|\left\{i \,|\, l_i^s = \tilde{l}_i^s\right\}|}{N^s}. \tag{5.3}$$

The experiment setup is the following. Two CSBM graphs are sampled following the same distribution. For DW and FGW we compute a transport map between the two graphs with varying hyper-parameters τ and α respectively. Transport maps are then evaluated according to two criteria: the (unsupervised) circular validation score, and the (supervised) target graph labelling accuracy $\frac{|\left\{j \,|\, l_j^t = \tilde{l}_j^t\right\}|}{N^t}$.

The results are visible on Figure 5.2. In the first subfigure 5.2a we can see that the circular validation score is constant equal to 1, while the accuracy varies up and down with τ. Therefore, this unsupervised score can not be used to select this hyper-parameter in this scenario. The same can be seen in the second subfigure 5.2b, with FGW and its hyper-parameter α.

This behaviour may be due to the fact that the transport maps produced by both DW and FGW are sparse, and therefore the source labels are perfectly. For instance, any transport map that is a permutation matrix will always have the highest circular validation score, regardless of the original data. This experiment highlights the need of a better hyper-parameter selection procedure, such as the one we proposed in Section 4.2.

5.2 Domain Adaptation tasks

In this section we look at DA problems where source and target data are attributed, possibly labelled graphs (see Section 1.4.2 for an introduction). A concrete application would be for instance transporting an analysis made on a specific timestamp of the Wikipedia encyclopedia to other versions of it. A similar idea is to transfer models learned on one social graph to another (say from one social network to another).

We address the most complicated DA scenario where source and target domains are considered and labels are only available in the former. Data from the two domains are supposedly drawn from different but related distributions and the goal is to reduce this distribution discrepancy while benefiting from the supervised information from the source [64]. Note that when dealing with a DA task between attributed graphs, the divergence can come from three situations: (i) a shift in the feature representation of the source/target nodes; (ii) a difference in the graph structures; (iii) both of them. In this section, we study these three settings.

We perform two series of experiments dedicated to compare several optimal transport methods on graph Domain Adaptation tasks. In the first one, we use Contextual Stochastic Block Models [110] to generate synthetic data. The second one concerns the `ogbn-arxiv` graph [112] and aims at classifying papers published in a given year from articles published before.

OT-based DA methods

Under different experimental conditions, we compare in DA tasks the relevance of our diffusion distances `DW` (3.1), to state-of-the-art OT-based distances. All these OT methods from the literature have been presented in Section 2.2. We recall them here.

Let $\mathcal{G}^s$ and $\mathcal{G}^t$ be two attributed graphs of size m and n nodes, with node distributions μ and ν. Let X^s and X^t be the associated node features. Let C^s and C^t be an associated structure matrix; we use the shortest-path matrix. Let L^s and L^t be the associated graph Laplacians. Let l^s and l^t be the associated node labels.

The Wasserstein distance [51] (`W`) compares distributions via a pair-wise cost of moving samples onto another (represented by a cost matrix in the discrete case). Here it is computed by solving the following optimisation problem:

$$\mathtt{W}(\mu,\nu) = \underset{\gamma\in\Pi(a,b)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{i,j}\cdot \|X_i^s - X_j^t\|_2 \right\}. \tag{5.4}$$

The Gromov-Wasserstein distance [60] (`GW`) has been originally defined for comparing two distributions that do not necessarily lie in the same feature space. Based on intra-distribution pairwise distances/costs, `GW` provides a nice framework for computing a distance between two graphs by encoding some structure, like the shortest path between two vertices. Here it is computed by solving:

$$\mathtt{GW}(\mu,\nu) = \min_{\gamma\in\Pi(a,b)} \left\{ \sum_{i,j,k,l} |(C_{i,k}^s, C_{j,l}^t|^2 \gamma_{i,j}\gamma_{k,l}\cdot \right\}, \tag{5.5}$$

In order to consider both the features associated to the nodes and the structure of the graphs, the *Fused-Gromov-Wasserstein* (`FGW`) distance has been recently introduced in [15]. `FGW` acts as a combination of the Wasserstein distance (focusing only on the features) and the `GW` distance (considering only the structure). It is computed by solving for some $\alpha \in [0, 1]$:

$$\mathtt{FGW}_p^p(\mu, \nu) = \min_{\gamma \in \Pi(a,b)} \left\{ \sum_{i,j,k,l} \left((1-\alpha) M_{ij}^p + \alpha |C_{ik}^s - C_{jl}^t|^p \right) \gamma_{ij} \gamma_{kl} \right\}. \tag{5.6}$$

In the presence of labels (which is the case in our context), the authors of [71] introduce two possible regularisation terms, based on the Laplacian of a graph encoding the class structure. Here, we use it with the Laplacians of the graphs themselves, because this structure information is available in the context we consider. Later in the experiments, this method is named `OT_LAPLACE`. For this method, we solve:

$$\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, M(\mathbf{X}_s, \mathbf{X}_t) \rangle + \lambda_s \mathrm{Tr}(\mathbf{X}_t^T \gamma^T \mathbf{L}_s \gamma \mathbf{X}_t) + \lambda_t \mathrm{Tr}(\mathbf{X}_s^T \gamma \mathbf{L}_t \gamma^T \mathbf{X}_s) \right\} \tag{5.7}$$

$$\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, M(\mathbf{X}_s, \mathbf{X}_t + \lambda_s B_s + \lambda_t B_t) \rangle + \lambda_s \mathrm{Tr}(\mathbf{X}_t^T \gamma^T \mathbf{L}_s \gamma \mathbf{X}_t) + \lambda_t \mathrm{Tr}(\mathbf{X}_s^T \gamma \mathbf{L}_t \gamma^T \mathbf{X}_s) \right\}, \tag{5.8}$$

with $B_s = -{}^1\!/_{N_s} (\mathbf{L}_s + \mathbf{L}_s^T) \mathbf{X}_s \mathbf{X}_t^T$ and $B_t = -{}^1\!/_{N_t} \mathbf{X}_s \mathbf{X}_t^T (\mathbf{L}_t + \mathbf{L}_t^T)$.

The OTDA [50] algorithm is based on the Wasserstein distance, but uses a group-lasso regularisation term on the labels, to prevent points of different labels from being mapped together. Later in the experiments, this method is named `L1L2_GL`. It is computed by solving:

$$\operatorname*{argmin}_{\gamma \in \Pi(a,b)} \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{i,j} \cdot \|X_i^s - X_j^t\|_2 + \sum_j \sum_{cl} \|\gamma(\mathcal{I}_{cl}, j)\|_2 . \right\} \tag{5.9}$$

All these methods are compared with ours, the `DW` distance, for which we distinguish 5 variations:

- `DW__CV`, where τ is tuned using a circular validation criterion.
- `DW`$\mathcal{L}_1$, where τ is the minimizer of our loss function $\mathcal{L}_1$.
- `DW`$\mathcal{L}_2$, where τ is the minimizer of our loss function $\mathcal{L}_2$.
- `DW`$\mathcal{L}_{1,\varepsilon}$, which uses entropic regularisation and τ selected by minimizing $\mathcal{L}_1$.
- `DW`$\mathcal{L}_{2,\varepsilon}$, which uses entropic regularisation and τ selected by minimizing $\mathcal{L}_2$.

Tuning hyper-parameters

When no other heuristic is explicitly used, we resort to the circular validation procedure derived in [98] to select hyper-parameters (see Section 4.2.1). Although both source and target graphs are labelled, the ground truth on the vertices of $\mathcal{G}^t$ is hidden until the final evaluation. Therefore, only the ground truth labels on the vertices of $\mathcal{G}^s$ are used for the tuning procedure.

For `DW` we managed to design others unsupervised criterion that yield consistently better results (see Section 4.2). But for other methods such as `FGW`, in the absence of a better alternative, we resorted to this method for the domain adaptation tasks.

5.2.1 Domain adaptation on synthetic data

To compare all OT based methods on a DA task, CSBM are generated, and each method is used to predict the target graph's labels.

For this experiment, 50 pairs of graphs are generated. The model parameters, described in Section 5.1.1, are $N^s = N^t = 240$, $M^s = \begin{pmatrix} 0.2 & 0.05 \\ 0.05 & 0.2 \end{pmatrix}$ and $M^t = \begin{pmatrix} 0.2 & 0.05 \\ 0.05 & {}^{0.2}\!/_{3} \end{pmatrix}$, $\sigma^s = 2$ and $\sigma^t = 4$. These parameters were chosen after some trials and errors to be challenging but not impossible.

OT methods are evaluated on their capacity to predict the target labels. Using the same procedure as earlier (see Equation 4.26), pseudo-labels are computed for target nodes, and compared with the ground truth, yielding an accuracy score.

When applicable, each method has 35 attempts to tune its hyper-parameters using the circular validation procedure described in Section 4.2.1.

The results are reported in Figure 5.3. The accuracy scores of each method over the 50 graph pairs are plotted as a boxplot, displaying the median performance, the quartiles and the 10th and 90th percentiles as well as the outliers (i.e. out of the 10th and 90th percentiles). We can make the following comments. First, we can note that the Gromov Wasserstein distance is worse than random guessing. This behaviour can probably be explained by a coupling matrix that permutes the classes. Second, the approaches that take into account both the feature and the structural information (i.e. `FGW` and `DW`-based methods) outperform the competitors. Third, `DW`-based methods are better than any other method in this context, with a more stable behaviour for the regularized version of our loss function. Finally, as expected, learning τ yields a significant improvement compared to `DW_CV` based on the circular validation.

5.2.2 Domain Adaptation on real data

This second series of experiments concerns the real `ogbn-arxiv` graph [112]. Although originally designed for node classification, we cast the problem as a Domain Adaptation task and we address it using the same methods as in the previous section. Each node of the graph represents a paper published in Arxiv. A link from one node to another indicates that this later is cited by the former. The feature of a node is an embedding of the paper's title and abstract, and it lies in $\mathbb{R}^{128}$. Nodes are labelled according to their corresponding subject area among 40 possible labels. Finally, each node is associated with a publication

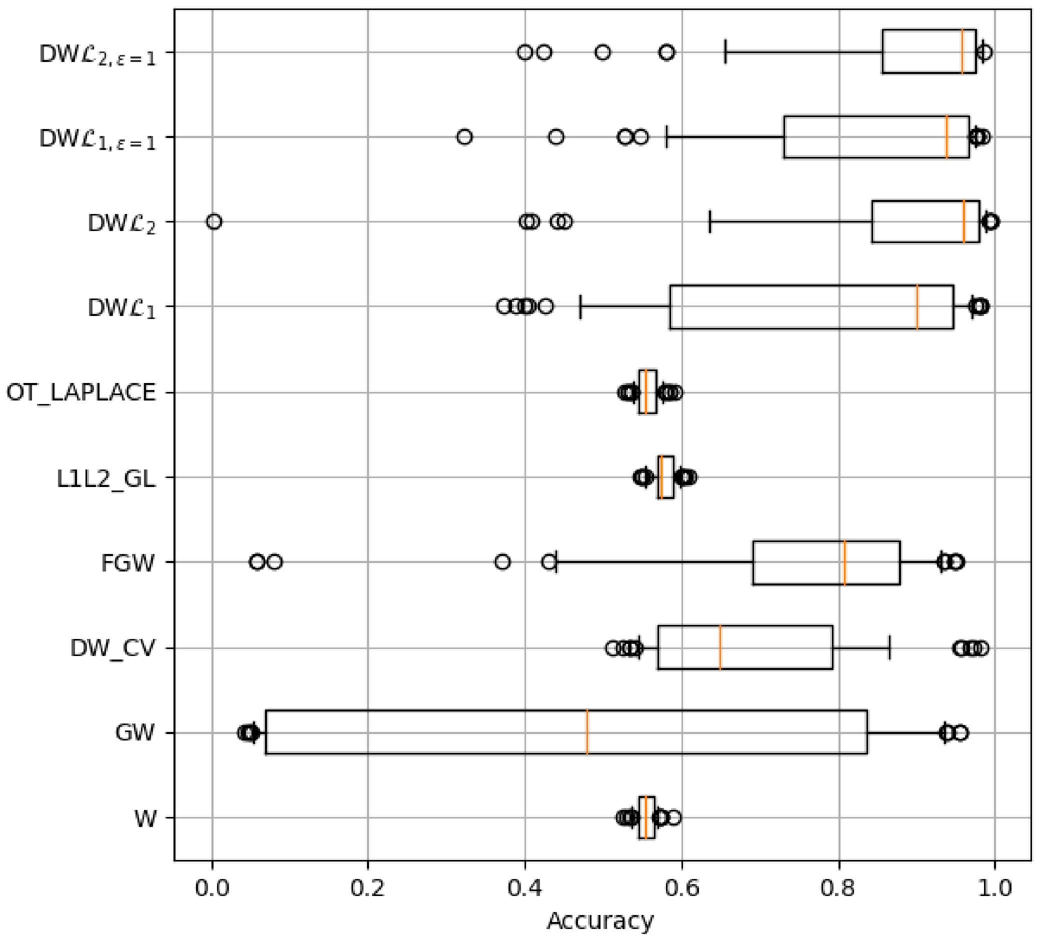
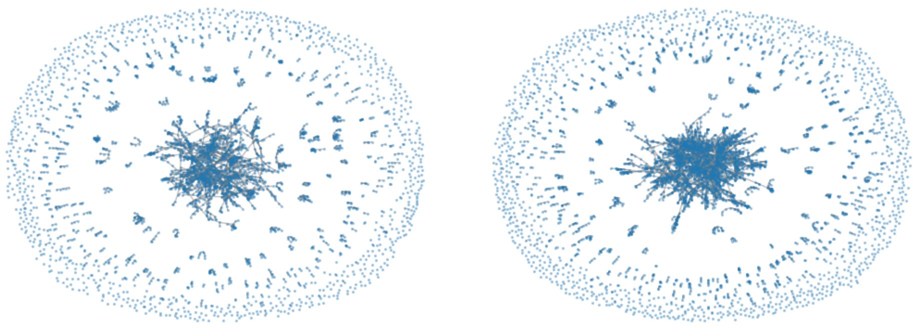


Figure 5.3: Median, quartile and decile accuracy of various OT methods on the task of transferring the labels of $\mathcal{G}^s$ to $\mathcal{G}^t$, over 50 sampling of source/target graph pairs.



(a) Year 2006 (b) Year 2007

Figure 5.4: Structure of the `ogbn-arxiv` graph, restricted to two different time periods. Node positions are for readability of the structure, and do not relate to the node labels or features.

year. In Figure 5.4 two sub-graphs are represented, restricted to nodes of year ≤ 2006 (Figure 5.4a) and year ≤ 2007 (Figure 5.4b).

In our setting, the source graph corresponds to the papers published before 2006. Its size is $m = 3678$. The target graph contains the articles published before 2007 ($n = 4980$ nodes). This makes the source graph a sub-graph of the target one and therefore, the DA accuracy is measured only on nodes of year 2007:

$$\mathrm{acc}(\gamma) = \frac{1}{1302} \sum_{j=3679}^{4980} \delta_{\hat{l}^t_j = l^t_j}. \tag{5.10}$$

For `GW`, the source and target cost matrices are built from the shortest-distances in the graph. Because the graph is not connected and the solver cannot handle infinite costs between two nodes, infinite values are replaced by twice the longest length path. For `FGW`, the same cost matrices are used, along with the pairwise Euclidean distance between the features. For the hyper-parameter α, 10 values are sampled uniformly in $[0, 1]$ and the best one is selected using circular validation. For `DW`, the hyper-parameter τ is either determined by circular validation among 10 logarithmically spaced values in $[10^{-3}, 10^{1}]$, or chosen by minimizing $\mathcal{L}_1$ and $\mathcal{L}_2$. Finally, the size of the impostor graph $\mathcal{G}^0$ is set to 500 nodes.

The test accuracies are reported in Table 5.1 as well as the computation times. Hyper-parameter tuning is included is the computation time, but cost matrix computation is excluded for `FGW` and `GW` as they are constant and assumed pre-computed. We can note that `DW` outperforms the competitors in terms of accuracy and remains much cheaper than `FGW` from a computational angle. The results also confirm that learning τ by minimizing the triplet loss $\mathcal{L}$ yields much better results than the circular validation.

All experiments are written in Python and use the libraries `POT` [113] for optimal transport methods, `PyGSP` [114] for graph generation, `OGB` [112] for corresponding graph dataset and `NumPy` and `SciPy` for other computations. The code is available at the following address: `https://gitlab.aliens-lyon.fr/`

Table 5.1: Computation time and test accuracy of various OT-based DA methods on the `ogbn-arxiv` graph restricted to years ≤ 2006 (for the source) and ≤ 2007 (for the target) with 3678 and 4980 nodes respectively.

Method	Computation time	Test accuracy
$\texttt{DW}\mathcal{L}_{1,\varepsilon=0.1}$	105s	54%
$\texttt{DW}\mathcal{L}_{2,\varepsilon=0.1}$	117s	54%
$\texttt{DW}\mathcal{L}_{1}$	92s	41%
$\texttt{DW}\mathcal{L}_{2}$	124s	41%
L2L1_GL	1502s	34%
`DW_CV`	69s	35%
`FGW`	1489s	35%
OT_LAPLACE	4472s	25%
`W`	4s	30%
`GW`	393s	17%

`dbarbe/manuscript-experiments`.

Conclusion

The contribution of this manuscript takes the form of the definition and the analysis, both theoretical and algorithmic, of a new family of *Diffusion-Wasserstein distances* (`DW`). `DW` lies at the intersection of Graph Signal Processing [7] and Optimal Transport [42]. It generalizes the Wasserstein distance by offering a way to compare attributed graphs, and behaves very efficiently when used for addressing Domain Adaptation tasks.

In Chapter 1, we laid down the theoretical notions and tools required for our contributions. We defined the kind of data that we manipulate: attributed graphs [23]. These objects combine two modalities: (i) a graph, a set of nodes and link between them, that represents the structure of the object, and (ii) attributes, a fixed-sized real-valued vector for each node, that represent a description of the individual nodes. They are very generic data models that encompass a lot of objects. They are difficult to study, because of their intrinsic non-linearity and the fact that they combine multiple modalities. Then, we recalled some notions of Graph Signal Processing, a field dedicated to the study of signals defined on graphs. In particular, we studied a specific graph filter: the heat diffusion in graphs [32]. It is a dynamical process on graphs that mimics the physical process of heat diffusing in a material. This tool is one of the building blocks of `DW`; we use it to merge the structure information and the feature information. Finally, we gave an introduction to the field of Optimal Transport. We gave the definition of the Wasserstein distance [51], a tool that both defines a distance between distributions, and a joint distribution between them that allows mapping sampled from one to the other.

In Chapter 2, we recalled various distances and OT methods between attributed graphs and similar objects. All of them were selected because of their relevance to `DW`: they either served as inspiration, or can be used to perform the same task. We recalled their formal definitions, mathematical complexity, and gave the pros and cons of each method. The Graph Diffusion Distance [67] simply measures a similarity between graphs of the same size; its use of the diffusion operator served as inspiration. The Gromov-Wasserstein distance [60] is a variant of the Wasserstein distance used to compare distributions supported on different spaces. The Fused-Gromov-Wasserstein distance [15] was introduced to define a distance between distributions on attributed graphs; it is the tool closest to our method. Laplacian regularization [71] is a term that can be added to the existing Wasserstein distance to take into account labels or a graph structure. Finally, the ℓ_1-ℓ_2 group-lasso regularizer [73] can be added to the existing Wasserstein distance too to take into account labels.

In Chapter 3, we gave the formal definition of our new distance, the Diffusion-Wasserstein distance. We proved several theoretical properties, namely that (i) it generalizes the Wasserstein distance, (ii) its limit when the diffusion time goes to infinity is the difference of the means of the features (iii) bounds on its expected value and (iv) conditions for it to define a metric. We also introduced variants: adding regularization, modifying other methods with diffusion and using other filters than diffusion.

In Chapter 4, we discussed specific points about the implementation and the use of `DW`. We first discussed a fast implementation of the diffusion process based on Chebyshev polynomials [79]. We detail the approximation process, and devise new bounds on the approximation error that improve upon the literature, allowing to choose a smaller truncation order to get below a given error. The complete algorithm is detailed, and experiments support this approach. Then, we discussed the choice of the hyper-parameter τ, the diffusion time, in a Domain Adaptation context. We devised a method to select this hyper-parameter tailored to `DW`, which involves the minimization of two possible loss functions, inspired by the existing notion of triplet loss.

Finally, in Chapter 5 we performed various experiments in Domain Adaptation to validate our design of `DW`. Our new method outperforms others OT-based methods both on synthetic and real data. This experiment also validates the use of our new loss functions to choose the hyper-parameter τ. On top of getting the best performances in the contexts studied, these experiments also show that our method runs faster than almost all methods we studied.

Perspectives The research work presented in this manuscript opens the following research directions:

- Our approach to optimising the hyper-parameter τ by minimising a triplet-based (or quadruplet) loss function proved to be successful compared to the use of the circular validation criterion. We think this approach has potential for other families of distances indexed by one or more hyper-parameters. Applying this method in other contexts requires to design a suitable notion of impostor, and the corresponding loss, and find a suitable loss optimisation algorithm (as not all distances are $\mathcal{C}^1$.

- In Section 3.3.1 we hinted at the possibility of going beyond diffusion, by using graph filters other than the diffusion filter studied in this manuscript. We believe the study of different graph filters to merge the feature and structure information is promising, as suggested by our results here with the diffusion. These methods would be able to leverage the existing body of work on Graph Signal Processing, and retain the computational advantage and all the variants of Optimal Transport.

- Our Diffusion-Wasserstein Distance was studied in the context of Domain Adaptation. As computing `DW` defines both a distance and a transport map, future research directions include trying to use it for other Machine Learning tasks on attributed graphs, like as a kernel for classification. The main difficulty will come from the definition of impostors in these new contexts. In the case of classification for instance they could be picked from the dataset rather than synthesized, and the loss could be averaged over multiple pairs.

- The losses we defined to select `DW`'s hyper-parameters assume that τ^s and τ^t are equal. Designing an improved method that do not rely on this assumption is necessary, as there will be cases where having them different is necessary for optimal performances. We hinted at a few possibilities in Section 4.2.4, but did not find a definitive answer yet, so this possibility remains open for future research.

- Privacy is a huge concerns for some attributed graphs (such as social graphs). All the distance computation methods presented here assume a perfect knowledge of the data, which may not be desirable in some cases. Therefore, another possible research direction is the design of privacy-preserving distances between attributed graphs. This topic that have been addressed on some distances (see [115] for instance) but not on this type of data yet.

Bibliography

[1] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[2] W. Zhang *et al.*, "Shift-invariant pattern recognition neural network and its optical architecture," in *Proceedings of annual conference of the Japan Society of Applied Physics*, 1988.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[5] B. Wu, W. Chen, Y. Fan, Y. Zhang, J. Hou, J. Liu, and T. Zhang, "Tencent ml-images: A large-scale multi-label image database for visual representation learning," *IEEE Access*, vol. 7, pp. 172683–172693, 2019.

[6] R. J. Trudeau, *Introduction to graph theory.* Courier Corporation, 2013.

[7] A. Ortega, P. Frossard, J. Kovačević, J. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[8] A. Sanfeliu and K. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Trans. Syst. Man Cybern.*, vol. 13, no. 3, pp. 353–362, 1983.

[9] D. Hammond, Y. Gur, and C. Johnson, "Graph diffusion distance: A difference measure for weighted graphs based on the graph laplacian exponential kernel," 12 2013.

[10] I. Jovanović and Z. Stanić, "Spectral distances of graphs," *Linear Algebra and its Applications*, vol. 436, no. 5, pp. 1425–1435, 2012.

[11] N. M. Kriege, F. D. Johansson, and C. Morris, “A survey on graph kernels,” *Applied Network Science*, vol. 5, no. 1, p. 6, 2020.

[12] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-lehman graph kernels,” *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, 2011.

[13] C. Morris, K. Kersting, and P. Mutzel, “Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs,” in *ICDM*, pp. 327–336, IEEE Computer Society, 2017.

[14] M. Togninalli, M. E. Ghisu, F. Llinares-López, B. Rieck, and K. M. Borgwardt, “Wasserstein weisfeiler-lehman graph kernels,” in *NeurIPS*, pp. 6436–6446, 2019.

[15] T. Vayer, L. Chapel, R. Flamary, R. Tavenard, and N. Courty, “Fused gromov-wasserstein distance for structured objects,” *Algorithms*, vol. 13, no. 9, p. 212, 2020.

[16] Y. Ollivier, H. Pajot, and C. Villani, eds., *Optimal Transport - Theory and Applications*, vol. 413 of *London Mathematical Society lecture note series.* Cambridge University Press, 2014.

[17] A. Barbe, M. Sebban, P. Gonçalves, P. Borgnat, and R. Gribonval, “Transport optimal entre graphes exploitant la diffusion de la chaleur,” in *CAP 2020-Conférence sur l’Apprentissage Automatique*, 2020.

[18] A. Barbe, M. Sebban, P. Gonçalves, P. Borgnat, and R. Gribonval, “Graph diffusion wasserstein distances,” in *ECML/PKDD (2)*, vol. 12458 of *Lecture Notes in Computer Science*, pp. 577–592, Springer, 2020.

[19] S. Marcotte, A. Barbe, R. Gribonval, T. Vayer, M. Sebban, P. Borgnat, and P. Gonçalves, “Fast multiscale diffusion on graphs,” *CoRR*, vol. abs/2104.14652, 2021.

[20] A. Barbe, P. Gonçalves, M. Sebban, P. Borgnat, R. Gribonval, and T. Vayer, “Optimization of the diffusion time in graph diffused-wasserstein distances: Application to domain adaptation,” in *IEEE International Conference on Tools with Artificial Intelligence*, 2021.

[21] D. Barbe, P. Borgnat, P. Gonçalves, and M. Sebban, “Transport optimal sous contrainte de régularité pour l’adaptation de domaines entre graphes avec attributs,” in *GRETSI 2019-XXVIIème Colloque francophonede traitement du signal et des images*, pp. 1–4, 2019.

[22] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[23] C. Bothorel, J. D. Cruz, M. Magnani, and B. Micenková, “Clustering attributed graphs: Models, measures and methods,” *Netw. Sci.*, vol. 3, no. 3, pp. 408–444, 2015.

[24] F. R. Chung and F. C. Graham, *Spectral graph theory.* No. 92, American Mathematical Soc., 1997.

[25] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *ICLR (Poster)*, OpenReview.net, 2017.

[26] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *ICLR (Poster)*, OpenReview.net, 2018.

[27] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NIPS*, pp. 1024–1034, 2017.

[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” in *ICLR*, OpenReview.net, 2019.

[29] S. Kolouri, N. Naderializadeh, G. K. Rohde, and H. Hoffmann, “Wasserstein embedding for graph learning,” in *ICLR*, OpenReview.net, 2021.

[30] T. Cai, S. Luo, K. Xu, D. He, T. Liu, and L. Wang, “Graphnorm: A principled approach to accelerating graph neural network training,” in *ICML*, vol. 139 of *Proceedings of Machine Learning Research*, pp. 1204–1215, PMLR, 2021.

[31] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, “An end-to-end deep learning architecture for graph classification,” in *AAAI*, pp. 4438–4445, AAAI Press, 2018.

[32] *Partial Differential Equations.* Springer-Verlag New York, 1982.

[33] H. S. Carslaw and J. C. Jaeger, “Conduction of heat in solids,” tech. rep., Clarendon Press,, 1959.

[34] P. Wilmott, S. Howson, S. Howison, J. Dewynne, *et al.*, *The mathematics of financial derivatives: a student introduction.* Cambridge university press, 1995.

[35] D. J. Griffiths and D. F. Schroeter, *Introduction to quantum mechanics.* Cambridge University Press, 2018.

[36] S. T. Acton, “Chapter 20 - diffusion partial differential equations for edge detection,” in *The Essential Guide to Image Processing* (A. Bovik, ed.), pp. 525–552, Boston: Academic Press, 2009.

[37] J. Sporring, L. Florack, M. Nielsen, and P. Johansen, *Gaussian Scale-Space Theory.* USA: Kluwer Academic Publishers, 1997.

[38] J. Klicpera, S. Weißenberger, and S. Günnemann, “Diffusion improves graph learning,” in *NeurIPS*, pp. 13333–13345, 2019.

[39] D. Thanou, X. Dong, D. Kressner, and P. Frossard, “Learning heat diffusion graphs,” *IEEE Trans. Signal Inf. Process. over Networks*, vol. 3, no. 3, pp. 484–499, 2017.

[40] R. A. Rossi and N. K. Ahmed, “The network data repository with interactive graph analytics and visualization,” in *AAAI*, pp. 4292–4293, AAAI Press, 2015.

[41] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” in *Conference NeurIPS, December 5-8, Nevada (US)* (C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 2292–2300, 2013.

[42] G. Peyré and M. Cuturi, “Computational Optimal Transport,” *arXiv*, Mar. 2018.

[43] G. Monge, *Mémoire sur la théorie des déblais et des remblais.* De l’Imprimerie Royale, 1781.

[44] A. Tolstoi, “Methods of finding the minimal total kilometrage in cargo transportation planning in space,” *TransPress of the National Commissariat of Transportation*, vol. 1, pp. 23–55, 1930.

[45] A. Fawzi, O. Fawzi, and P. Frossard, “Analysis of classifiers’ robustness to adversarial perturbations,” *Mach. Learn.*, vol. 107, no. 3, pp. 481–508, 2018.

[46] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, pp. 2672–2680, 2014.

[47] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *ICML*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 214–223, PMLR, 2017.

[48] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, *Advances in domain adaptation theory.* Elsevier, 2019.

[49] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, “A survey on domain adaptation theory,” *CoRR*, vol. abs/2004.11829, 2020.

[50] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, “Optimal transport for domain adaptation,” *Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, 2017.

[51] L. V. Kantorovich, “On the translocation of masses,” *Journal of mathematical sciences*, vol. 133, no. 4, pp. 1381–1382, 2006.

[52] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *ICML*, vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 957–966, JMLR.org, 2015.

[53] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, 2000.

[54] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," in *NeurIPS*, pp. 1186–1195, 2018.

[55] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pp. 460–467, IEEE Computer Society, 2009.

[56] S. Reich, "A nonparametric ensemble transform method for bayesian inference," *SIAM J. Sci. Comput.*, vol. 35, no. 4, 2013.

[57] S. Ferradans, N. Papadakis, G. Peyré, and J. Aujol, "Regularized discrete optimal transport," *SIAM J. Imaging Sci.*, vol. 7, no. 3, pp. 1853–1882, 2014.

[58] M. Cuturi and A. Doucet, "Fast computation of wasserstein barycenters," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 685–693, JMLR.org, 2014.

[59] G. D. Cañas and L. Rosasco, "Learning probability measures with respect to optimal transport metrics," in *NIPS*, pp. 2501–2509, 2012.

[60] G. Peyré, M. Cuturi, and J. Solomon, "Gromov-wasserstein averaging of kernel and distance matrices," in *Int. Conf. on Machine Learning*, vol. 48, pp. 2664–2672, 2016.

[61] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Summer school on machine learning*, pp. 169–207, Springer, 2003.

[62] V. Vapnik, *The nature of statistical learning theory.* Springer science & business media, 1999.

[63] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1-2, pp. 151–175, 2010.

[64] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, *Advances in Domain Adaptation Theory.* Elsevier, ISBN 9781785482366, p. 187, Aug. 2019.

[65] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *NIPS*, pp. 3730–3739, 2017.

[66] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," in *AAAI*, pp. 4058–4065, AAAI Press, 2018.

[67] D. K. Hammond, Y. Gur, and C. R. Johnson, "Graph diffusion distance: A difference measure for weighted graphs based on the graph laplacian exponential kernel," in *GlobalSIP*, pp. 419–422, IEEE, 2013.

[68] R. P. Brent, "Algorithms for minimization without derivatives, chap. 4," 1973.

[69] F. Mémoli, "Gromov-wasserstein distances and the metric approach to object matching," *Found. Comput. Math.*, vol. 11, no. 4, pp. 417–487, 2011.

[70] E. M. Loiola, N. M. M. de Abreu, P. O. B. Netto, P. Hahn, and T. M. Querido, "A survey for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 176, no. 2, pp. 657–690, 2007.

[71] R. Flamary, N. Courty, A. Rakotomamonjy, and D. Tuia, "Optimal transport with Laplacian regularization," in *NIPS 2014, Workshop on Optimal Transport and Machine Learning*, (Montréal, Canada), Dec. 2014.

[72] M. Frank, P. Wolfe, *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[73] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, 2017.

[74] K. Bredies, D. A. Lorenz, and P. Maass, "A generalized conditional gradient method and its connection to an iterative shrinkage method," *Comput. Optim. Appl.*, vol. 42, no. 2, pp. 173–193, 2009.

[75] C. R. Givens and R. M. Shortt, "A class of Wasserstein metrics for probability distributions.," *Michigan Mathematical Journal*, vol. 31, no. 2, pp. 231 – 240, 1984.

[76] J. van Oostrum, "Bures-wasserstein geometry," 2020.

[77] A. H. Al-Mohy and N. J. Higham, "Computing the action of the matrix exponential, with an application to exponential integrators," *SIAM J. Scientific Computing*, vol. 33, no. 2, pp. 488–511, 2011.

[78] L. Bergamaschi and M. Vianello, “Efficient computation of the exponential operator for large, sparse, symmetric matrices,” *Numer. Linear Algebra Appl.*, vol. 7, no. 1, pp. 27–45, 2000.

[79] D. I. Shuman, P. Vandergheynst, and P. Frossard, “Chebyshev polynomial approximation for distributed signal processing,” in *Distributed Computing in Sensor Systems, 7th IEEE International Conference and Workshops, DCOSS 2011, Barcelona, Spain, 27-29 June, 2011, Proceedings*, pp. 1–8, IEEE Computer Society, 2011.

[80] N. J. Higham and A. H. Al-Mohy, “Computing matrix functions,” *Acta Numer.*, vol. 19, pp. 159–208, 2010.

[81] M. Popolizio and V. Simoncini, “Acceleration techniques for approximating the matrix exponential operator,” *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 657–683, 2008.

[82] M. A. Botchev and L. A. Knizhnerman, “ART: adaptive residual-time restarting for krylov subspace matrix exponential evaluations,” *J. Comput. Appl. Math.*, vol. 364, 2020.

[83] V. Druskin and L. Knizhnerman, “Two polynomial methods of calculating functions of symmetric matrices,” *USSR Computational Mathematics and Mathematical Physics*, vol. 29, no. 6, pp. 112–121, 1989.

[84] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials.* CRC press, 2002.

[85] R. M. Mattheij, S. W. Rienstra, and J. T. T. Boonkkamp, *Partial differential equations: modeling, analysis, computation.* SIAM, 2005.

[86] O. De la Cruz Cabrera, M. Matar, and L. Reichel, “Analysis of directed networks via the matrix exponential,” *Journal of Computational and Applied Mathematics*, vol. 355, pp. 182–192, 2019.

[87] H. Zhuang, S.-H. Weng, and C.-K. Cheng, “Power grid simulation using matrix exponential method with rational krylov subspaces,” in *2013 IEEE 10th International Conference on ASIC*, pp. 1–4, IEEE, 2013.

[88] M. Pusa and J. Leppänen, “Computing the matrix exponential in burnup calculations,” *Nuclear science and engineering*, vol. 164, no. 2, pp. 140–150, 2010.

[89] L. Li and D. L. Sussman, “Graph matching via multi-scale heat diffusion,” in *IEEE BigData*, pp. 1157–1162, IEEE, 2019.

[90] M. Mehra, A. Shukla, and G. Leugering, “An adaptive spectral graph wavelet method for pdes on networks,” *Adv. Comput. Math.*, vol. 47, no. 1, p. 12, 2021.

[91] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "The spectral graph wavelet transform: Fundamental theory and fast computation," in *Vertex-Frequency Analysis of Graph Signals*, pp. 141–175, Springer, 2019.

[92] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *International ACM Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 24, 2018.

[93] G. Phillips, *Interpolation and Approximation by Polynomials.* CMS Books in Mathematics, Springer, 2003.

[94] M. Abramowitz, I. A. Stegun, and R. H. Romer, "Handbook of mathematical functions with formulas, graphs, and mathematical tables," 1988.

[95] R. Riener and M. Harders, *Virtual reality in medicine.* Springer Science & Business Media, 2012.

[96] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," 2021.

[97] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2018.

[98] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 770–787, 2010.

[99] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *NIPS*, pp. 41–48, MIT Press, 2003.

[100] J. Altschuler, J. Weed, and P. Rigollet, "Near-linear time approximation algorithms for optimal transport via sinkhorn iteration," in *NIPS*, pp. 1964–1974, 2017.

[101] S. Dhouib, I. Redko, and C. Lartizien, "On learning a large margin classifier for domain adaptation based on similarity functions," in *21 eme Conférence sur l'Apprentissage Automatique (CAp)*, 2019.

[102] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky, "Domain-adversarial training of neural networks," in *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition, pp. 189–209, Springer, 2017.

[103] I. Redko, N. Courty, R. Flamary, and D. Tuia, "Optimal transport for multi-source domain adaptation under target shift," in *AISTATS*, vol. 89 of *Proceedings of Machine Learning Research*, pp. 849–858, PMLR, 2019.

[104] A. Bellet, A. Habrard, and M. Sebban, “A survey on metric learning for feature vectors and structured data,” *CoRR*, vol. abs/1306.6709, 2013.

[105] A. Bellet, A. Habrard, and M. Sebban, *Metric Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2015.

[106] E. N. Gilbert, “Random Graphs,” *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141 – 1144, 1959.

[107] J. M. Altschuler and E. Boix-Adserà, “Wasserstein barycenters can be computed in polynomial time in fixed dimension,” *J. Mach. Learn. Res.*, vol. 22, pp. 44:1–44:19, 2021.

[108] G. Luise, S. Salzo, M. Pontil, and C. Ciliberto, “Sinkhorn barycenters with free support via frank-wolfe algorithm,” in *NeurIPS*, pp. 9318–9329, 2019.

[109] M. Sommerfeld, J. Schrieber, Y. Zemel, and A. Munk, “Optimal transport: Fast probabilistic approximation with exact solvers,” *J. Mach. Learn. Res.*, vol. 20, pp. 105:1–105:23, 2019.

[110] Y. Deshpande, S. Sen, A. Montanari, and E. Mossel, “Contextual stochastic block models,” in *NeurIPS 2018, Montréal, Canada.*, pp. 8590–8602, 2018.

[111] P. W. Holland, K. B. Laskey, and S. Leinhardt, “Stochastic blockmodels: First steps,” *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[112] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, “Open graph benchmark: Datasets for machine learning on graphs,” *CoRR*, vol. abs/2005.00687, 2020.

[113] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer, “Pot: Python optimal transport,” *Journal of Machine Learning Research*, vol. 22, no. 78, pp. 1–8, 2021.

[114] M. Defferrard, L. Martin, R. Pena, and N. Perraudin, “Pygsp: Graph signal processing in python,” Oct. 2017.

[115] J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, and M. Zohner, “Gshade: Faster privacy-preserving distance computation and biometric identification,” in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pp. 187–198, 2014.

Abstract:

This thesis is about the definition and study of the Diffusion-Wasserstein distances between attributed graphs.

An attributed graph is a collection of points with individual descriptions (features) and links between them (structure), like molecules or a social network. The Diffusion-Wasserstein distance is a generalization of the Wasserstein distance. It defines a metric for attributed graphs, and allow the computation of a transport map between them. It exploits the graph diffusion process to define new features for the nodes, and compare them. The diffusion time τ acts as a hyper-parameter, weighting the relative importance of the features and the structure. Compared to other transport distances that take into account features and a graph structure, the Diffusion-Wasserstein distance is faster to compute and yields better results in Domain Adaptation tasks.

The computational side of the diffusion process received special attention. We used polynomial approximation using a Chebychev basis to accelerate these computations. We proved a new bound for the approximation error of the exponential operator, and showed how parts of the computation can be re-used for new diffusion times τ.

A specific heuristic was developed to choose the diffusion time τ. It works by minimizing a function akin to a triplet-loss. I uses impostors, synthetic graphs built to be dissimilar in specific ways to the original attributed graphs whose distance is required. This heuristic yields better results for our method than the circular validation criterion used in Domain Adaptation.

Résumé :

Ces travaux portent sur la définition et l'étude de la distance de Diffusion-Wasserstein entre graphes attribués.

Les graphes attribués sont des collections de points avec une description individuelle (attributs) et des liens entre eux (structure de graphe), comme une molécule ou un réseau social. La distance de Diffusion-Wasserstein est une généralisation de la distance de Wasserstein ; elle permet de définir une distance entre des graphes attribués, et de calculer un plan de transport entre eux. Son fonctionnement exploite la diffusion dans le graphe pour définir de nouveaux attributs et les comparer. Le temps de diffusion τ joue le rôle d'hyper-paramètre, contrôlant l'importance donnée aux attributs et à la structure. Comparée à d'autres distances de transport qui prennent en compte attributs et structure de graphe, la distance de Diffusion-Wasserstein est plus rapide à calculer, et donne les meilleures performances dans plusieurs tâches d'apprentissage.

Le calcul du processus de diffusion a reçu une attention particulière. Ces travaux ont porté sur une approximation à base de polynômes de Tchebychev. Cette approximation permet d'accélérer le calcul de la diffusion. Une nouvelle borne de l'erreur d'approximation qui améliore l'état de l'art a été prouvée, et une méthode pour réutiliser une partie des calculs pour de nouveaux temps de diffusion τ a été montrée.

Un heuristique spéciale pour le choix du temps de diffusion τ a été définie. Celle-ci se base sur la minimisation d'une fonction, semblable à une triplet-loss,

qui utilise des imposteurs, des graphes attribués construits pour être dissimilaires aux données initiales. Cette heuristique obtient de meilleurs résultats que le critère de validation circulaire utilisé jusqu'ici.