Review of Wasserstein-based Graph Alignment

CALLARD Baptiste
ENS Paris-Saclay
Saclay, France
baptiste.callard@ens-paris-saclay.fr

TOCQUEC Louis
ENS Paris-Saclay
Paris, France
louis.tocquec@ens-paris-saclay.fr

ABSTRACT

Our review examines from a critical perspective the innovative approach proposed in the paper : "Wasserstein-based Graph Alignment" [9] by Hermina Petric Maretic, Mireille El Gheche et al. for the alignment of non-isomorphic graphs of different sizes. This paper presents a new way of using the Wasserstein's distance to measure the similarity between graph signals, which enables the alignment of graphs of different sizes. This study offers an algorithm which introduces a one-to-many assignment strategy of nodes, contrasting it with traditional one-to-one approaches. A new differentiable Dykstra operator is introduced to solve this assignment problem with a soft-assignment. It enables the use of the backpropagation algorithm, and therefore benefits from powerful frameworks such as PyTorch and TensorFlow. The method is flexible and produces a high-quality match, even with noisy data, and can be used for a wide range of tasks : clustering, graph alignment and community detection. It is versatile and suitable for a wide range of applications, including neuroimaging, social network analysis and bioinformatics. Nevertheless, it has some limitations that we will try to highlight. It has high computation costs and restrictive assumptions (normal distribution of the graph signal). In addition, it is not easy to assess whether a distance works better, as it cannot simply be interpreted. In fact, it is not immediately obvious that two graphs are close. Moreover, there is no available code, and thus we had to implement it ourselves using PyTorch in order to make experiments on self-made data. Our first task was to explain the mathematical tools, highlighting points that were not developed enough in the paper. Secondly, we conducted a series of experiments to complement the original paper. In particular, we studied graph alignment visualization and outlier detection methods. Our code is available on our GitHub [1].

KEYWORDS

Graph, Optimal Transport, Alignment, Classification, Community, Outlier detection

1. https://github.com/b-ptiste/Wasserstein-Graph-Alignment

Geometric Data Analysis, 2023, Paris
.

1 INTRODUCTION

Methods and applications linked to graphs play a key role in various fields such as transport networks, bioinformatics, social networks and financial networks. Although the study of graphs has been intensively explored by mathematicians, the increase in computing resources coupled with scientific advances and new tools such as Deep Learning offer new challenges and new ways of working. Representing graphs is a complex problem : in general, a graph can be considered as a discrete set of vertices and edges, but it can also be much more abstract. For example, it is sometimes necessary to consider graphs as continuous objects in order to use certain theoretical tools. It is therefore important to take into account the different representations. The comparison between graphs is not a trivial task at all because it implies local and global properties. In addition, graph alignment of different sizes is a fundamental problem with applications in chemistry for the classification and discovery of new molecules, in the study of social networks or transport networks. These tasks may require for instance generative methods that need a way of assessing the quality and distance between reference graphs and generated graphs. Additionally, to discover new drugs, we do not want to work with molecules of homogeneous sizes. Thus, to answer this type of task, it is important to have a way of computing distances between graphs of different sizes that takes into account local and global properties. It is in this context that the article "Wasserstein-based Graph Alignment" provides a truly valuable solution.

2 RELATED WORK

Graph isomorphism problem (IG), the strictest version of graph alignment that tries to determine whether two graphs are identical up to the nodes relabelling is NP-hard [23]. Historical approaches such as Ullman algorithm for sub-graph isomorphism have exponential complexity, which limits their use to very small graphs. Graph Kernel-based methods such as graphlet Kernel, the shortest path Kernel or random walk Kernel are also used for (IG) or graph alignment [20]. They offer a simple way of comparing graphs, but may not be suitable for global interactions. In the same context, Weisfeiler-Lehman test (WL) proposes an iterative algorithm to detect

isomorphism, but it has been demonstrated that some class of non-isomorphism graphs can't be distinguished in a polynomial time [6], [10]. This method can also be used to determine whether two graphs are similar by looking at the number of iterations the algorithm has performed before determining whether two graphs are not isomorphic. Some most recent papers as [8] propose a direct comparison of graph heat diffusion matrices using kernel methods but tackle only graphs of the same size.

Recent research focused on the development of efficient algorithms to address graph alignment problems. A first approach comes from optimization where graph topic has been in the heart of many works : it takes the form of a quadratic assignment problem (see [25] and [11]) where the idea is to approximate the permutation matrix as best as possible, for instance by spectral clustering with relaxation in order to transform it in a continuous problem. Spectral graph theory is the study of the graph properties linked to characteristic polynomials, eigenvalues, and eigenvectors of matrices associated with the graph, such as adjacency matrix or Laplacian matrix. Whereas an adjacency matrix depends only on vertex labelling, its specter is a graph invariant. It is important to design algorithms which are invariant to edge or vertex permutations because we want robust models.

A new way to address the problem comes from deep learning with the increase in computing power and of image/language processing works. These methods are based on classical learning algorithms that try to learn the nodes, edges or graphs representation in an end-to-end manner. Most common approach are focused on node proximity as factorization matrix methods (e.g. Laplacian eigenmaps) [19]. DeepWalk ([4]) and node2vec([1]) design algorithms to study local interactions by examining random walks in graphs. Other methods focus on learning-based structures such as struc2vec [15]. More recent works such as GCN [14] or GAT [24] use mechanisms of passing messages, readout of hidden representations to produce graph embeddings. Message passing is powerful and Xu et al. [12] showed that two distant points in embedding space give similar results with the Weisfeiler-Lehman graph isomorphism test. Thus, this active domain has developed tools to solve (IG) problem and graph alignment between 2019 and 2020 in main conferences as Graph Isomorphism Network Architecture (GIN), Higher Order Weisfeiler and Leman, k-GNN, k-hop, RelationalPooling, CLIP, k-order graph networks etc. Other methods like [18] tries to align graphs using the Sinkhorn operator to predict a soft permutation matrix.

Finally, promising results have been made using optimal transport theory. Some methods are based on Gromov-Wasserstein distance which can be efficiently computed, with the works of [7] and [22], by using the Euclidean distance between shortest path matrices. However, they still have a limited impact due to their inability to take advantage of the global structure or their use of restrictive assumptions (same graph size, hard assignments). The paper [5] moves away from traditional one-to-one assignment, introducing a flexible one-to-many node assignment strategy. Another method [2], really close to the work of "*Wasserstein-based Graph Alignment*" [9] uses graph filter and then Wasserstein distance. In "*Wasserstein-based Graph Alignment*", they follow the same framework, using Wasserstein distance to evaluate the similarity but applied to the graph signals. They handle the non-convexity of the problem with a stochastic formulation and a new differentiable Dykstra operator allowing an efficient algorithm that integrates Bayesian exploration in the optimization process. Their results demonstrate that the Wasserstein distance outperforms both Gromov-Wasserstein and Euclidean distance in graph alignment and classification tasks with structured graphs. They provide a meaningful way to efficiently align and compare graphs.

3 MATHEMATICAL BACKGROUND

3.1 Graph signal theory

We denote by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ a graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges. We denote by $|\mathcal{V}|$ the cardinal of $\mathcal{V}$, i.e the number of vertices. From this graph, we can compute is Laplacian Matrix which is defined as :

$$L = D - W$$

where D is the degree matrix which is a diagonal matrix defined such as $D_{ii} = deg(v_i)$ and A the adjacency matrix such as $W_{ij} = \mathbb{1}_{(v_i,v_j)\in E}$.

Property 3.1. *0 is an eigenvalue of L. Its multiplicity is c, where c denote the number of connected components of* $\mathcal{G}$

Proof : Suppose that the graph is one connected component, then, by denoting $\mathbb{1} \in |\mathcal{V}|$ the vector composed only of value 1, we have :

$$L\mathbb{1} = D\mathbb{1} - W\mathbb{1} = 0$$

Indeed :

$$D\mathbb{1} = \sum_{i=1}^{|\mathcal{V}|} deg(v_i)$$

and :

$$W\mathbb{1} = \sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} \mathbb{1}_{(v_i,v_j)\in E} = \sum_{i=1}^{|\mathcal{V}|} deg(v_i)$$

Thus 0 is an eigenvalue of L associated with the eigenvector $\mathbb{1}$.

Now if we suppose that $\mathcal{G}$ is composed of c connected components, we use the same reasoning as before on each connected component.

□

It means that L is singular and not of full rank. We must therefore use the correct version of the pseudo-inverse.

Now define a function $g : i \in V \to g(i) \in \mathbb{R}$ and the vector $x = (\ g(1), ..., \ g(|\mathcal{V}|)) \ \in \ \mathbb{R}^{|\mathcal{V}|}$. x is called the signal and it is equivalent to add a scalar to each node of the graph. Generally, we see the signal as a value distributed according to a probabilistic distribution. In the paper, but also in a lot of works in the literature :

$$x \sim \mathcal{N}(0, L^{\dagger})$$

where † denotes the pseudo-inverse operator.

At first this choice can be seen as an arbitrary one but actually it is quite meaningful. Indeed, in this case, strongly connected vertices have similar values, an assumption which real life applications often confirm. For instance, for a traffic map where vertices are cities, edges are roads and signal is the number of cars of each city : cities with a lot of common roads will have similar traffic. It is also the case for social networks : a group of friends is probably going to share more common interests than two people who don't know each other. Moreover, this assumption produces smooth graphs which are more convenient to work with.

3.2 Optimal Transport

Optimal Transport Theory was first introduced by G.Monge a french mathematician in 1871 in order to address the optimal resources allocation problem in a mathematical way.

Formally : let ν and μ two probabilistic distributions on two space $\mathcal{X}$ and $\mathcal{Y}$, the goal is to find the map T : $\mathcal{X} \to \mathcal{Y}$ which minimise the cost of transport, i.e :

$$\inf_{T_{\#}\nu=\mu} \int_{\mathcal{X}} c(x, T(x)) d\nu(x) \quad (*)$$

where $T_{\#}$ is the push-forward associated to T and c :$\mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is the transport cost.

In the paper, c : $(x, y) \in \mathcal{Y} \times \mathcal{Y} \to ||x - y||_2^2 \in \mathbb{R}_+$ and in this case $(*)$ is called the Wasserstein distance denoted $\mathcal{W}_2^2$.

Property 3.2. *Let $\nu \sim \mathcal{N}(0, \Sigma_1)$ and $\mu \sim \mathcal{N}(0, \Sigma_2)$ two Gaussian measures. Then, the Wasserstein distance is :*

$$\mathcal{W}_2^2(\nu, \mu) = Tr(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}}})$$

Proof : First of all, we suppose T as a linear shape i.e :

$$T : x \mapsto Ax$$

with A a symmetric positive matrix.
The proof is based on the following lemma :

Lemma 3.2.1. $T_{\#}\nu = \mu \ \Leftrightarrow A\Sigma_1 A = \Sigma_2$

(the proof is given in appendix)
Thus :

$$\begin{aligned} & A\Sigma_1 A = \Sigma_2 \\ \Leftrightarrow\ & \Sigma_1^{\frac{1}{2}} A \Sigma_1 A \Sigma_1^{\frac{1}{2}} = \Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}} \\ \Leftrightarrow\ & (\Sigma_1^{\frac{1}{2}} A \Sigma_1^{\frac{1}{2}})^2 = \Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}} \\ \Leftrightarrow\ & \Sigma_1^{\frac{1}{2}} A \Sigma_1^{\frac{1}{2}} = (\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}})^{\frac{1}{2}} \\ \Leftrightarrow\ & A = \Sigma_1^{-\frac{1}{2}} (\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}})^{\frac{1}{2}} \Sigma_1^{-\frac{1}{2}} \end{aligned}$$

One can remark that $A = A^T$.
A is a symmetric positive definite matrix, so A is a gradient of convex function. By Brenier's theorem (see appendix) we can conclude that T is optimal :

$$T(x) = \Sigma_1^{-\frac{1}{2}} (\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}})^{\frac{1}{2}} \Sigma_1^{-\frac{1}{2}} x$$

Now we want to compute $\mathcal{W}_2^2(\nu, \mu)$:

$$\begin{aligned} \mathcal{W}_2^2(\nu, \mu) &= \int_{\mathcal{X}} ||x - T(x)||_2^2 d\nu(x) \\ &= \mathbb{E}_\nu[x^T x] + \mathbb{E}_\nu[T(x)^T T(x)] \\ &\quad - 2\mathbb{E}_\nu[T(x)^T x] \\ &= \mathbb{E}_\nu[Tr(xx^T)] + \mathbb{E}_\nu[Tr(T(x)T(x)^T)] \\ &\quad - 2\mathbb{E}_\nu[Tr(T(x)x^T)] \\ &= Tr(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}}}) \end{aligned}$$

□

4 PROBLEM DEFINITION

4.1 One-to-many assignment

Let's define two graphs $\mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{E}_1\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \mathcal{E}_2\}$ such that their respective signals are Gaussian defined exactly the same as in section 3.1.

In the same size graph alignment problem, by using Property 3.2, we have an explicit formula for Wasserstein distance and thus the problem is well-defined. But this same size assumption, even if it gives some intuition to the general case, is a restriction. Indeed, in practice, we often want to deal with graphs of different sizes and this particular case is a

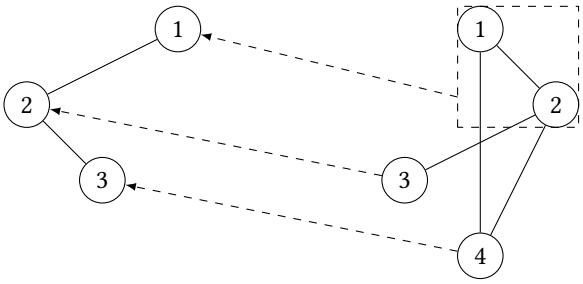

Figure 1 : one-to-many assignment

negligible part of real life problems that occur in chemistry or in networks for instance.

This paper addresses the case where the graphs have different sizes (but we still need to know which graph is smaller) and thus derives a way more general algorithm to compare graphs. We make the assumption, without losing generality, that $|\mathcal{V}_1| < |\mathcal{V}_2|$. The basic principle of the method is to use a one-to-many assignment which means that each vertex of $\mathcal{G}_2$ must be matched to a single vertices of $\mathcal{G}_1$. On the other, each node $\mathcal{G}_1$ can mapped to at most k_max nodes of $\mathcal{G}_2$. This one-to-many assignment can be efficiently described by a matrix of the set C_{hard} defined by :

$$C_{hard} = \{P \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|} \text{ such as : } \begin{cases} \forall i, \forall j,\ P_{ij} \in \{0,1\} \\ \forall i, \sum\limits_j P_{ij} \in [1, k_{max}] \\ \forall j, \sum\limits_i P_{ij} = 1 \end{cases} \}$$

where k_{max} is a number to ensure that the set C_{hard} defines a one-to-many assignment. At most k_{max} nodes in $\mathcal{G}_2$ can be matched to a node in $\mathcal{G}_1$.

Property 4.1. $1 \leq k_{max} \leq |\mathcal{V}_2| - |\mathcal{V}_1| + 1$

Proof : by contradiction, suppose that the property isn't true. The first case which is $k_{max} = 0$, is trivial so we supposed that $k_{max} > |\mathcal{V}_2| - |\mathcal{V}_1| + 1$. Then, we can consider $P \in C_{hard}$ such that P matches $|\mathcal{V}_2| - |\mathcal{V}_1| + 2$ vertices of $\mathcal{G}_2$ to one node of $\mathcal{G}_1$. But then there are only $|\mathcal{V}_1| - 2$ nodes in $\mathcal{G}_2$ to match with $|\mathcal{V}_1| - 1$ nodes of $\mathcal{G}_1$: impossible by definition of one-to-many assignment.

□

Example 4.1. *A super-simple example of a one-to-many assignment is given in the figure (1), with the associated matrix P defines as :*

$$P = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Given the true alignment matrix $P_* \in C_{hard}$, the two graphs can be aligned by transforming the Laplacian matrix L_2 in $P_* L_2 P_*^T$ such as : $\mu_{P_*}^{\mathcal{G}_2} = \mathcal{N}(0, (P_* L_2 P_*^T)^\dagger)$.
Of course P_* is unknown and thus the problem can be formulated as :

$$\min_{P \in C_{hard}} \mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu_P^{\mathcal{G}_2}) \quad (*)$$

This construction allows us to return to the simple case of two graphs of the same size, which is very convenient but it remains one huge difficulty : C_{hard} is a discrete set the problem is therefore combinatorial which made $(*)$ intractable in practice.

4.2 Optimization

First of all, the problem is relaxed by changed C_{hard} to a continuous set C_{soft} :

$$C_{soft} = \{P \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|} \text{ s.t : } \begin{cases} \forall i, \forall j,\ P_{ij} \in [0,1] \\ \forall i, \sum_j P_{ij} \in [1, k_{max}] \\ \forall j, \sum_i P_{ij} = 1 \end{cases} \}$$

Then, one can also define a operator called Dykstra which takes as input a matrix $P \in \mathbb{R}^{\mathcal{V}_1\times\mathcal{V}_2}$ and produces as output a matrix which is in C_{soft}. Formally :

$$\mathcal{A}_\tau : P' \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|} \mapsto \mathcal{A}_\tau(P') \in C_{soft}$$

where τ is a small positive constant.

$$\begin{aligned} \mathcal{A}_\tau(P') &= argmax_{P\in C_{soft}}(< P, P' > -\tau \sum_{ij} P_{ij} \log_\odot(P_{ij})) \\ &= argmin_{P\in C_{soft}} \sum_{ij} \left(P_{ij} \log_\odot(P_{ij}) - P_{ij} \frac{P'_{ij}}{\tau} \right) \\ &= argmin_{P\in C_{soft}} \sum_{ij} \left[P_{ij} \log_\odot(P_{ij}) - P_{ij} \log_\odot(\exp_\odot \frac{P'_{ij}}{\tau}) \right] \\ &= argmin_{P\in C_0\cap C_1} KL(P \,||\, \exp_\odot(P'/\tau)) \end{aligned}$$

Where $\odot$ is the entry-wise operator ($exp_\odot$ and $log_\odot$ are the analogous notations) and C_0 and C_1 are defined as :

$$C_0 = \{P \in \mathbb{R}_+^{|\mathcal{V}_1|\times|\mathcal{V}_2|} \mid \sum_{j=0}^{|\mathcal{V}_2|} P_{ij} \in [1, k_{max}]\}$$

$$C_1 = \{P \in \mathbb{R}_+^{|\mathcal{V}_1|\times|\mathcal{V}_2|} \mid \sum_{i=0}^{|\mathcal{V}_1|} P_{ij} = 1\}$$

We can leverage easily that C_0 and C_1 are two convex sets. Thus, following [3], we can define their associated projectors according to the Kullback-Leibler divergence defined by :

$$\mathcal{P}_{C_i}^{KL}(P) = argmin_{P'\in C_i} KL(P'||P)$$

Definition 4.1. *The Kullback-Leibler projectors of the matrix* $P \in \mathbb{R}_+^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$ *on the sets* C_0 *and* C_1 *are given with the following formula :*

$$\mathcal{P}^{KL}_{C_0}(P) = \boldsymbol{diag}\left(\left[\frac{\max(1, \min(\sum_j P_{ij}, k_{max}))}{\sum_j P_{ij}}\right]_i\right) \times P$$

$$\mathcal{P}^{KL}_{C_1}(P) = P \times \boldsymbol{diag}\left(\left[\frac{1}{\sum_i P_{ij}}\right]_j\right)$$

Given a matrix P, the goal of $\mathcal{P}^{KL}_{C_0}(P)$ is to normalize the rows of P in order to have the condition described in C_0 and $\mathcal{P}^{KL}_{C_1}(P)$ normalizes the columns of P.

Example 4.2. *For* $k_{max} = 2$ *and the matrix* $P = \begin{pmatrix} 3 & 1 & 5 \\ 1 & 0 & 0 \end{pmatrix}$

$$\mathcal{P}^{KL}_{C_0}(P) = \begin{pmatrix} 2/9 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 3 & 1 & 5 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 6/9 & 2/9 & 10/9 \\ 1 & 0 & 0 \end{pmatrix} \in C_0$$

$$\mathcal{P}^{KL}_{C_1}(P) = \begin{pmatrix} 3 & 1 & 5 \\ 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1/4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/5 \end{pmatrix} = \begin{pmatrix} 3/4 & 1 & 1 \\ 1/4 & 0 & 0 \end{pmatrix} \in C_1$$

In order to compute $\mathcal{A}_\tau(P)$ we use the Dykstra Algorithm which uses the Kullback-Leibler projects previously defined. The output of the algorithm, when $\tau \to 0$, is a one-to-many assignment matrix.

Definition 4.2. *Dykstra is an algorithm defined by the following steps :*

Algorithm 1 Dykstra Algorithm

Input : $P \in \mathbb{R}_+^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$
t=0
$P_0 = \exp_\odot(P\backslash\tau)$
$Q_{-1} = Q_0 = \mathbb{1}^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$
for *t = 0, 1,...* ***do***
  $P_{t+1} \leftarrow \mathcal{P}^{KL}_{C_{t \bmod 2}}(P_t \odot Q_{t-1})$
  $Q_{t+1} = \frac{Q_{t-1}\odot P_t}{P_{t+1}}$
end for
Output : $\mathcal{A}_\tau(P_\infty)$

We use the Dykstra operator to leverage a new problem :

$$\min_{P\in\mathbb{R}^{\mathcal{V}_1\times\mathcal{V}_2}} \mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(P)}) \quad (\diamond)$$

It can be seen as a relaxed version of $(*)$ with a random initialization (the only condition is on the shape of the matrix). By recalling that the cost function $\mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(P)})$ can be expressed as :

$$Tr(L_1^\dagger + \mathcal{A}_\tau(P)^t L_2^\dagger \mathcal{A}_\tau(P) - 2\sqrt{L_1^{\dagger\frac{1}{2}}\mathcal{A}_\tau(P)^t L_2^\dagger \mathcal{A}_\tau(P) L_1^{\dagger\frac{1}{2}}})$$

and because the Dykstra operator is differentiable [16], we can apply the Gradient Descent framework to solve $(\diamond)$. Nevertheless, the problem is still non-convex and may cause the optimization task to converge towards a local minima. Thus we need to develop a stochastic optimization process in order to solve it. We suppose here that $P \sim q_\theta$ a multivariate Gaussian distribution with parameters $\theta = (\mu, \sigma)$. We can easily derive the following upper bound :

$$\min_{P\in\mathbb{R}^{\mathcal{V}_1\times\mathcal{V}_2}} \mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(P)}) \le \mathbb{E}_{P\sim q_\theta}[\mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(P)})]$$

We also use the "re-parameterization trick" described in [13] to obtain :

$$P = \mu + \sigma \odot \epsilon$$

where ϵ is such that $\epsilon_{ij} \sim \mathcal{N}(0, 1)$.
The stochastic version of the problem $(\diamond)$ consists to optimize the previous upper bound and can be expressed as :

$$\min_{\eta,\sigma} \mathbb{E}_{\epsilon\sim q_{unit}}[\mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(\eta+\sigma\odot\epsilon)})] \quad (\bullet)$$

where $q_{unit} = \Pi_{ij}\mathcal{N}(0, 1)$.
The aim goal is to design q_θ such that it puts all of its mass on a minimizer of the cost function.
We introduce the following notation :

$$\mathcal{I}(\eta, \sigma) = \mathbb{E}_{\epsilon\sim q_{unit}}[\mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(\eta+\sigma\odot\epsilon)})]$$

which is differentiable [21].
We can define :

$$\tilde{\mathcal{I}}(\mu, \sigma) = \frac{1}{S}\sum_{s=1}^{S} \mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(\eta+\sigma\odot\epsilon_s)})$$

where $S \in \mathbb{N}$ and $\epsilon_s \sim q_{unit}$, the empirical mean of $\mathcal{I}(\eta, \sigma)$, such that the gradient can be approximated by :

$$\nabla\mathcal{I}(\eta, \sigma) \approx \frac{1}{S}\sum_{s=1}^{S} \nabla\mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(\eta+\sigma\odot\epsilon_s)})$$

The algorithm proposed in [9] to solve $(\bullet)$ is the following :

Algorithm 2 Wasserstein-based graph alignment algorithm

Input : $\mathcal{G}_1$ and $\mathcal{G}_2$
Input : S$\in \mathbb{N}$, step size $\gamma > 0$, $\tau > 0$
Input : μ_0, σ_0
$\mu \leftarrow \mu_0$
$\sigma \leftarrow \sigma_0$
while no convergence **do**
    Draw sample ϵ_1,...,ϵ_S from q_{unit}
    $\tilde{I}(\mu, \sigma) \leftarrow \frac{1}{S} \sum_{s=1}^{S} \mathcal{W}_2^2(\nu^{\mathcal{G}_1}, \mu^{\mathcal{G}_2}_{\mathcal{A}_\tau(\eta+\sigma\odot\epsilon_s)})$
    $\mu \leftarrow \mu - \gamma \nabla_\mu \tilde{I}(\mu, \sigma)$
    $\sigma \leftarrow \sigma - \gamma \nabla_\sigma \tilde{I}(\mu, \sigma)$
end while
Output : $P = \mathcal{A}_\tau(\mu^*)$

5 PERSONAL EXPERIMENTS

We have re-implemented Algorithm 2 from scratch using PyTorch since the code is not available. We made our experiments reproducible on our GitHub repository in a notebook. We used S=10, τ=2, $\gamma \in (0.5, 1.5)$, nb_iter_dykstra=20, this was the best configuration in the original paper using grid search. The authors studied their method on graph alignment, community detection and graph classification. We chose to study graph alignment by working on visualisation. Then, we proposed an original approach, even though it is close to clustering, which is the detection of outliers in a family of graphs that we created.

5.1 Graph Alignment : visualisation for hard-assignment and soft-assignment

This experiment allows us to visualise matching in the context of a hard-assignment and then soft-assignment. We created graphs $\mathcal{G}_2$ from $\mathcal{G}_1$ by adding nodes and edges randomly so that the graphs share a similar underlying structure. Next, we studied the matrix P after convergence.

The figure (2) is a snapshot of the alignment at the end of the training. We provide a comprehensive video on our GitHub (for hard and soft-assignment) that shows the evolution of alignment over iteration. In Graph $\mathcal{G}_1$, we can consider that three communities can be defined [2] : {red, orange}, {yellow, light green, light blue}, {dark blue, purple}. In the soft-assignment, the nodes with a high centrality : yellow and light blue have a hard assignment (very high probability). In the video, yellow and light blue are assigned first in $\mathcal{G}_2$. On the other hand, the other nodes are assigned to several nodes of $\mathcal{G}_1$ which have a close signal (i.e. a similar degree and neighbourhood).

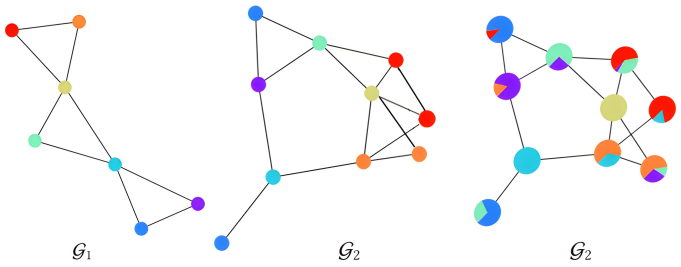

Figure 2 : Hard and soft-assignment from $\mathcal{G}_2$ to $\mathcal{G}_1$

We also noticed that the algorithm converged better when the symmetry was broken (see. figure(3)). It is well known that breaking symmetry in optimization is a good thing, because otherwise there are many equivalent optimal solutions, which creates instability. Here, we can see that the nodes close in the graph $\mathcal{G}_1$ are also close in $\mathcal{G}_2$ and have the same neighbourhood. It is interesting to see that because of the yellow node, the central node in light blue connected by an edge is divided into light green and light blue. Indeed, there is an edge between light green and yellow in $\mathcal{G}_1$. Although the alignment is good, it is not perfect.

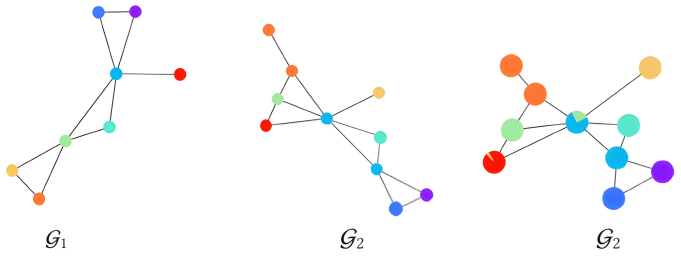

Figure 3 : Hard and soft-assignment from $\mathcal{G}_2$ to $\mathcal{G}_1$

5.2 Alignment and Outlier detection

We created a family of graphs. We call it $GF(n, p)$ which is parameterised by n the number of cycles and p the number of nodes per cycle. This allows us to generate graphs with similar underlying structures and properties. For example, in figure (4), $\mathcal{G}_0$ is $GF(3, 3)$, $\mathcal{G}_1$ is $GF(3, 4)$ and $\mathcal{G}_2$ is $GF(3, 5)$. We also constructed $\mathcal{G}_3$ as a random Erdős-Rényi graph.
We then calculated the distances between the different graphs over 5 iterations. The results in figure (7) shows that the distances between the $GF(n, p)$ graphs are respectively smaller than their distances from the Erdős-Rényi random graph. The method is also more robust on similar graphs, as shown by the standard deviation. This means we can detect that $\mathcal{G}_3$

2. Note : Other interesting clusters of communities may be proposed.

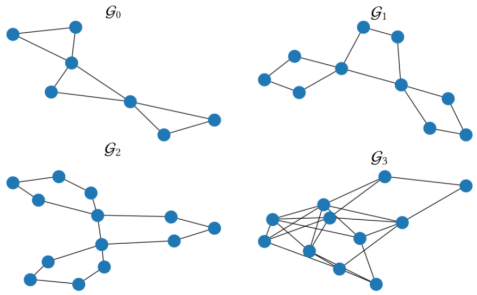

Figure 4 : Three graphs of $GF(n, p)$ and a random Erdős-Rényi graph

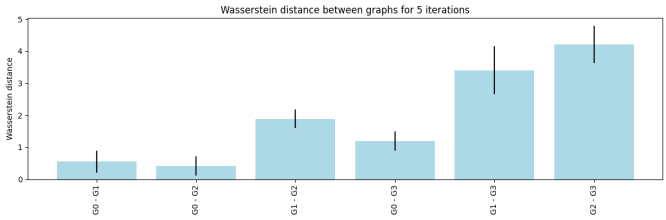

Figure 5 : Mean and std of the median of the last 100 steps over 10 iterations

is an outlier to the others based on the distance gap.

In figure (6), $\mathcal{G}_0$ is $GF(3, 5)$, $\mathcal{G}_1$ is $GF(4, 5)$ and $\mathcal{G}_2$ is $GF(5, 5)$. We also constructed $\mathcal{G}_3$ as a random Erdős-Rényi graph.

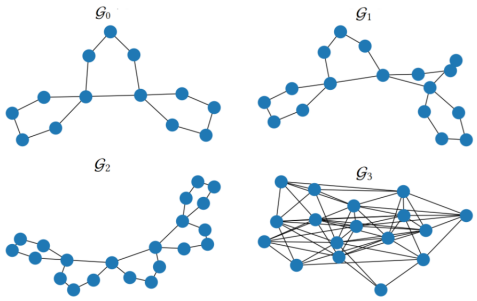

Figure 6 : Three graphs of $GF(n, p)$ and a random Erdős-Rényi graph

We then computed the distances between the different graphs over 5 iterations. The results in figure (7) highlights that respectively the distances between graphs in class $GF(n, p)$ are smaller than their distance from the Erdős-Rényi random graph. We obtain results similar to those obtained previously and manage to detect the outlier.
We also did the same experiments, taking a reference graph and randomly adding nodes and edges, then comparing the distances. We also added a random Erdős-Rényi graph (see.

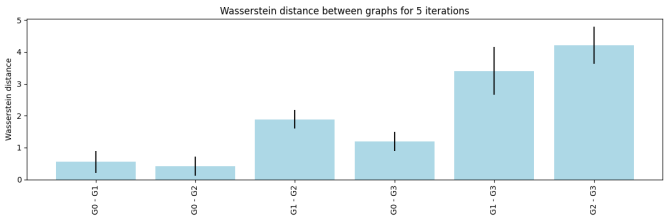

Figure 7 : Mean and std of the median of the last 100 steps over 10 iterations

figures (8) and (9)). We wanted to show that the method was able to adapt to a 60-node graph with at least 10 extra nodes and 5 extra edges.

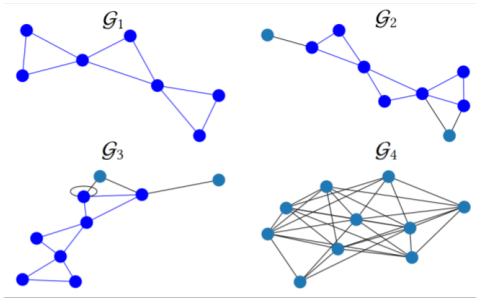

Figure 8 : $\mathcal{G}_1$ has 7 nodes, $\mathcal{G}_2$ and $\mathcal{G}_3$ have 2 more nodes and at least 2 more edges. $\mathcal{G}_4$ is the Erdős-Rényi graph with 10 nodes.

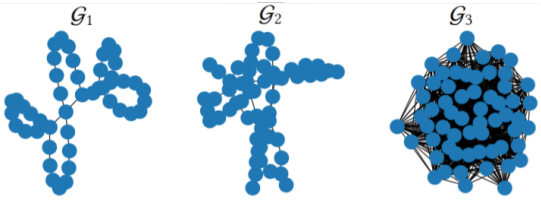

Figure 9 : $\mathcal{G}_1$ has 50 nodes, $\mathcal{G}_2$ and $\mathcal{G}_3$ have 10 more nodes and at least 5 more edges. $\mathcal{G}_4$ is the Erdős-Rényi graph with 60 nodes.

The first two graphs are indeed closer to $\mathcal{G}_1$ according to the Wasserstein distance than graph 4 (see figure (10)). In figure (11), the distance is fairly large for both graphs, but smaller for the graph $\mathcal{G}_2$ constructed from $\mathcal{G}_1$, as one might expect. On the other hand, it is complicated to interpret the result completely because the distance between two graphs is by definition abstract.

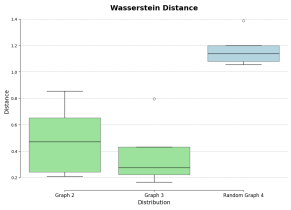

Figure 10 : Distance for configuration on figure (8)

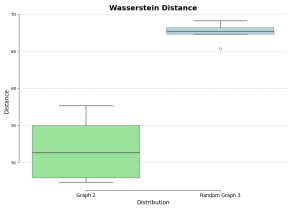

Figure 11 : Distance for configuration on figure (9)

6 DISCUSSIONS

Despite an interesting framework to solve the hard task of graph alignment where the graphs are not of the same size, the algorithm is expensive in terms of computations with a complexity $\Theta(n^3)$. Thus, its application to huge graphs is limited. For instance, comparison of protein or social networks analysis may require huge time, at least with our current implementation, which is perhaps naive. This complexity is explained in part by the use of eigenpairs. In addition, the method can run into difficulties when there is a disparity between the sizes of the graphs or when the graphs have symmetries.

Furthermore, the main assumption of the soft graph signal is not always satisfied and can be very easily broken (social networks, financial networks or biological networks). We might want to modify our normal distribution assumption, but all "close forms" formulas vanish. While smooth graph signals effectively capture global properties of graphs, they may lack of descriptive power for other potentially interesting properties. The author of the paper [17] addresses this limitation. Their new transport-based optimal distance is able to prioritise different spectral information in observed graphs.

Regarding to the paper experiments, the algorithm is only compared to Optimal transport methods, however, these methods may not be the best way to solve the problem. Methods should be compared with optimization, deep learning and approaches from other communities on various benchmarks. Moreover, it can be rough to interpret results and performances in graph alignment when it comes to huge random graphs because no a priori alignment can be leveraged. Nevertheless, the robustness can be assessed in part by create two graphs with common structures and a random graph : intuitively, the distance should be less between the first two. The fact that the code is not available is also regrettable. Although the pseudo code appears to be simple to implement, we encountered difficulties because many of the subtleties for making the method converge are not addressed in the paper. The method is robust to initialization but there can be cases where matrices become ill-conditioned and cause the method to diverge.

On our side, there are still avenues to explore for future work. We would like to speed up implementation and facilitate convergence by studying schedulers or initialization that allow the model not to fall into local minima (a highly non-convex problem). We have started some experimentation (see. Appendix). If we were able speed it up, it could be packaged and used with the PyTorch model.

7 CONCLUSION

Graph topic is a really interesting field which has many applications in real life issues. It has, at the same time, strong mathematical and algorithmic components. We both found the paper relevant by the way it uses previous works to leverage a new algorithm to address a difficult problem as the graph alignment. We tried to make a summary of the mathematical tools necessary to have an understanding of this algorithm and to provide some properties and details not developed in the paper. Moreover we also did an implementation of the procedure using PyTorch to give more examples of application of the method. We commented on the results obtained and gave interpretations. Finally, it seems that the algorithm offers an efficient way to compute the distance between two graphs of different sizes under some good assumptions (see limits described in section 6.).

RÉFÉRENCES

[1] Jure Leskoveci Aditya Grover. 2016. node2vec : Scalable Feature Learning for Networks. (2016).
[2] Amélie Barbe, Marc Sebban, Paulo Gonçalves, Pierre Borgnat, and Rémi Gribonval. 2020. Graph diffusion wasserstein distances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 577–592.
[3] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. 2015. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing* 37, 2 (2015), A1111–A1138.
[4] Steven Skiena Bryan Perozzi, Rami Al-Rfou. 2014. DeepWalk : Online Learning of Social Representations. (2014).
[5] Yihe Dong and Will Sawin. 2020. Copt : Coordinated optimal transport on graphs. *Advances in Neural Information Processing Systems* 33 (2020), 19327–19338.
[6] S. Evdokimov and I. N. Ponomarenko. 1999. On highly closed cellular algebras and highly closed isomorphisms. (1999).
[7] Justin Solomon Gabriel Peyré, Marco Cuturi. 2016. Gromov-Wasserstein Averaging of Kernel and Distance Matrices. (2016).
[8] David K Hammond, Yaniv Gur, and Chris R Johnson. 2013. Graph diffusion distance : A difference measure for weighted graphs based on the graph Laplacian exponential kernel. In *2013 IEEE global conference on signal and information processing*. IEEE, 419–422.

[9] Matthias Minder Giovanni Chierchia Pascal Frossard Hermina Petric Maretic, Mireille El Gheche. [n. d.]. Wasserstein-based Graph Alignment. ([n. d.]).

[10] M. Furer J. Cai and N. Immerman. 1992. An optimal lower bound on the number of variables for graph identification. *Combinatorica* (1992).

[11] Bo Jiang, Jin Tang, Chris Ding, Yihong Gong, and Bin Luo. 2017. Graph matching via multiplicative update algorithm. *Advances in neural information processing systems* 30 (2017).

[12] Jure Leskovec Stefanie Jegelka Keyulu Xu, Weihua Hu. 2019. How Powerful are Graph Neural Networks? *AAAI 19* (2019).

[13] Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML]

[14] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv :1609.02907* (2016).

[15] Daniel R. Figueiredo Leonardo F.R. Ribeiro, Pedro H.P. Saverese. 2017. struc2vec : Learning Node Representations from Structural Identity. *KDD 17* (2017).

[16] Giulia Luise, Alessandro Rudi, Massimiliano Pontil, and Carlo Ciliberto. 2018. Differential properties of sinkhorn approximation for learning with wasserstein distance. *Advances in Neural Information Processing Systems* 31 (2018).

[17] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. 2022. FGOT : Graph distances based on filters and optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7710–7718.

[18] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv :1802.08665* (2018).

[19] Partha Niyogi Mikhail Belkin. 2002. Using manifold structure for partially labelled classification. *NIPS'02* (2002).

[20] Fredrik D. Johansson Christopher Morris Nils M. Kriege. 2020. A survey on graph kernels. *Applied Network Science* (2020).

[21] Joe Staines and David Barber. 2012. Variational Optimization. arXiv:1212.4507 [stat.ML]

[22] Rémi Flamary Romain Tavenard Nicolas Courty Titouan Vayer, Laetitia Chapel. 2018. Optimal Transport for structured data with application on graphs. (2018).

[23] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *Electr. J. Comb.* (1976).

[24] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

[25] Junchi Yan, Xu-Cheng Yin, Weiyao Lin, Cheng Deng, Hongyuan Zha, and Xiaokang Yang. 2016. A short survey of recent advances in graph matching. In *Proceedings of the 2016 ACM on international conference on multimedia retrieval*. 167–174.

APPENDIX

Complement on Optimal Transport

THEOREM 7.1. *(Brenier's theorem) - In the case $X = Y = \mathbb{R}^d$ and $c(x, y) = ||x - y||_2$, if ν has a density with respect to the Lebesgue measure, then there exists a unique optimal Monge map T. This map is characterized by being the unique gradient of a convex function i.e $T = \nabla\phi$ with ϕ convex and such that $(\nabla\phi)_{\#}\nu = \mu$.*

Proof - (Lemma 3.2.1)
By denoting f_μ (resp. f_ν the density of $\mu \sim \mathcal{N}(0, \Sigma_2)$ (resp. $\nu \sim \mathcal{N}(0, \Sigma_1)$, we have :

$$\begin{aligned} f_\mu(T(x)) &= \det(2\pi\Sigma_2)^{-\frac{1}{2}} \exp(T(x)^t \Sigma_2^{-1} T(x)) \\ &= \det(2\pi\Sigma_2)^{-\frac{1}{2}} \exp(x^t A^t \Sigma_2^{-1} A x) \\ &= \det(2\pi\Sigma_2)^{-\frac{1}{2}} \exp(x^t \Sigma_1^{-1} x) \end{aligned}$$

Moreover since T is a linear map, we have :

$$|detT'(x)| = detA = \left(\frac{\det(\Sigma_2)}{\det(\Sigma_1)}\right)^{\frac{1}{2}}$$

Therefore, we have $f_\nu = |detT'(x)| f_\mu$ meaning that $T_{\#}\nu = \mu$. □

(Proof taken from Optimal Transport course of G.Peyré)

Additional experiences

We also noticed that the convergence of the method could be unstable at initialization. Indeed, the inversion of Laplacian can be complex with singular matrices (even with the pseudo-inverse and conditioning). So we wanted to study initialization and convergence. We compared an initialization with a reduced centred normal distribution and a Xavier distribution. We also tried adding a scheduler to reduce the learning rate after a certain number of iterations.We tested several examples :

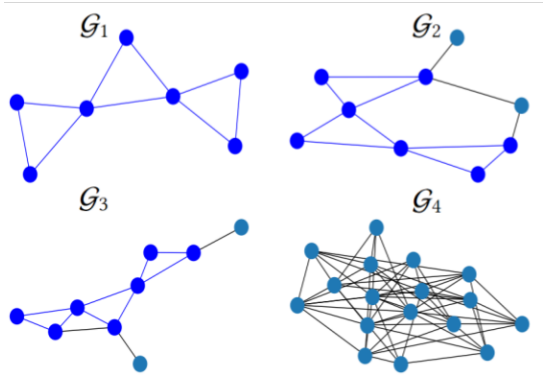

FIGURE 12 : Random Graphs

We can see that the distance is impacted by the initialisation but that the order of magnitude of the different distances is similar.

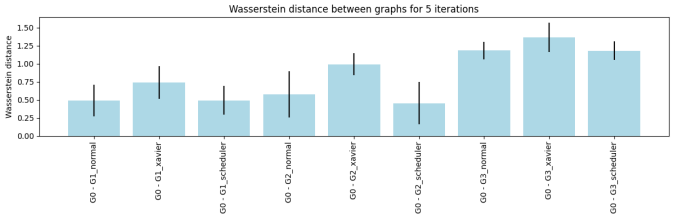

Figure 13 : Mean and std of the median of the last 100 steps over 10 iterations

We also noticed that using initialisation with Xavier enabled convergence with less instability and converge faster on this small example. In addition, the scheduler doesn't add much value.

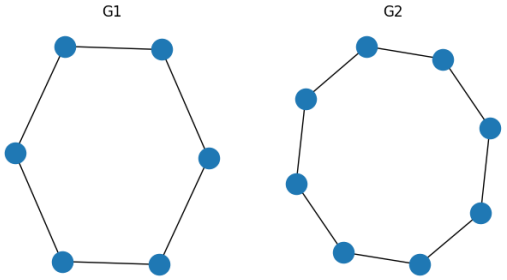

Figure 14 : Graphs use for comparison of convergence.

We can see on a very simple case that on the first iterations the method initialized with normal weights is more unstable. However, this does notcompletely solve the instability problems, even after convergence, on larger instances (due to the inversion of Laplacian).

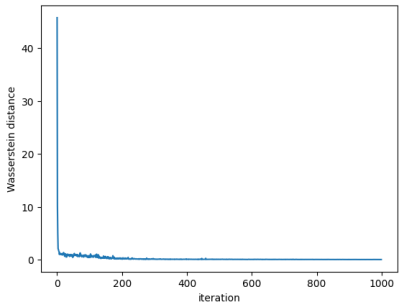

Figure 15 : Convergence with normal $\mathcal{N}(0,1)$ between $\mathcal{G}_1$ and $\mathcal{G}_2$

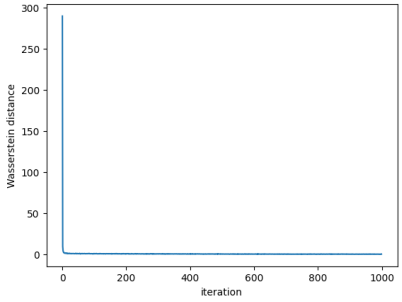

Figure 16 : Convergence with Xavier between $\mathcal{G}_1$ and $\mathcal{G}_2$