Journal of Machine Learning **ISSN**: 2790-2048(e), 2790-203X(p)

Fast Gradient Computation for Gromov-Wasserstein Distance

Wei Zhang [*1], Zihao Wang [†2], Jie Fan [‡1], Hao Wu [§1], and Yong Zhang [¶3]

[1]Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China
[2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong SAR, China
[3]BNRist, RIIT, Institute of Internet Industry, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract. The Gromov-Wasserstein distance is a notable extension of optimal transport. In contrast to the classic Wasserstein distance, it solves a quadratic assignment problem that minimizes the pair-wise distance distortion under the transportation of distributions and thus could apply to distributions in different spaces. These properties make Gromov-Wasserstein widely applicable to many fields, such as computer graphics and machine learning. However, the computation of the Gromov-Wasserstein distance and transport plan is expensive. The well-known Entropic Gromov-Wasserstein approach has a cubic complexity since the matrix multiplication operations need to be repeated in computing the gradient of Gromov-Wasserstein loss. This becomes a key bottleneck of the method. Currently, existing methods accelerate the computation focus on sampling and approximation, which leads to low accuracy or incomplete transport plans. In this work, we propose a novel method to accelerate accurate gradient computation by dynamic programming techniques, reducing the complexity from cubic to quadratic. In this way, the original computational bottleneck is broken and the new entropic solution can be obtained with total quadratic time, which is almost optimal complexity. Furthermore, it can be extended to some variants easily. Extensive experiments validate the efficiency and effectiveness of our method.

Keywords:
Optimal transport,
Gromov-Wasserstein distance,
Fast gradient computation algorithm,
Fast algorithm.

Article Info.:
Volume: 3
Number: 3
Pages: 282 - 299
Date: September/2024
doi.org/10.4208/jml.240416

Article History:
Received: 16/04/2024
Accepted: 25/06/2024

Communicated by:
Zhi-Qin John Xu

1 Introduction

The Gromov-Wasserstein (GW) distance [31], as an important member of optimal transport [36, 40], is a powerful tool for distribution comparison. It is related to the Gromov-Hausdorff (GH) distance [17], a fundamental distance in metric geometry that measures how far two metric spaces are from being isometric [11]. Specifically, it measures the minimal distortion of pair-wise geodesic distances under the transport plan between two probabilistic distributions, even defined on different underlying spaces. Inherited from GH distance, GW is invariant to translation, rotation, and reflection of metric space. In

[*]zhang-w20@mails.tsinghua.edu.cn
[†]Corresponding author. zwanggc@cse.ust.hk
[‡]fanj21@mails.tsinghua.edu.cn
[§]hwu@tsinghua.edu.cn
[¶]zhangyong05@tsinghua.edu.cn

this way, GW has particular advantages to applications that require preserving geometry structures including computer graphics [30,37,48], natural language processing [3], graph factorization and clustering [14,55], and machine learning [10,57]. Moreover, variants of GW distance have been proposed for wider applications. For example, unbalanced GW (UGW) extends the comparison from probabilistic distributions to positive measures [44]. Fused GW (FGW) combines the GW and Wasserstein distances by interpolating their objectives, which is shown to be particularly effective for networks [50,52] and cross-domain distributions [35].

The computation of the Gromov-Wasserstein distance boils down to solving a non-convex quadratic assignment problem that is NP-hard [22]. For this, some numerical methods built on relaxations have been developed, including convex relaxations [20, 46, 47], eigenvalue relaxations [24], etc. Nevertheless, these methods often require a large number of relaxed variables (for instance, $N^2 \times N^2$ variables in [20] where N is the number of discrete points of two spaces), resulting in high computational complexity. And they frequently provide unsatisfying solutions, especially in the presence of a symmetric metric matrix [37]. Entropic GW is a seminal and now the most popular work to compute GW distance from another perspective, which minimizes GW objective with an entropy regularization term [37,48]. In contrast to the non-entropy-based method mentioned above, it exhibits global convergence without removing constraints and offers a more concise computation. Moreover, it can adapt to solve GW variants, such as FGW [52] and UGW [44]. In each iteration of it, one first computes the GW gradient with matrix multiplications in $\mathcal{O}(N^3)$ time, which dominates the total complexity, and then solves the subproblem by the Sinkhorn algorithm [15] in $\mathcal{O}(N^2)$ time.

The computational cost of entropic GW remains unsatisfactory in large-scale scenarios. There are various methods to accelerate it (see Table 1.1 for the comparison). Scalable Gromov-Wasserstein learning method (S-GWL) [56] assumes the hierarchical structure of

Table 1.1: The comparison of different methods for the computation of GW metric and its variants. For SaGrow, the parameter s serves as a sampling parameter that dictates the quantity of specific sampled matrices. For spar-GW, the parameter s' designates the number of elements sampled from the GW gradient matrix. For LR-GW, the parameters r and d represent the presumed ranks of the distance matrices and the coupling matrices, respectively.

Method	Complexity	Exact and full-sized plan
Entropic GW and its approximations		
Entropic GW [37]	$\mathcal{O}(N^3)$	✓
S-GWL [56]	$\mathcal{O}(N^2 \log N)$	not exact
SaGroW [19]	$\mathcal{O}(N^2 s)$	not full-sized
Spar-GW [25]	$\mathcal{O}(N^2 + s'^2)$	not full-sized
LR-GW [42]	$\mathcal{O}(N(r^2 + d^2 + rd))$	not exact
AE [41]	$\mathcal{O}(N^2 \log N)$	not exact
GW on special structures		
FlowAlign [23]	$\mathcal{O}(N^2)$	tree only
FGC-GW (This work)	$\mathcal{O}(N^2)$	✓

the transportation plan and conducts the entropic GW algorithm in a multi-scale scheme. Anchor energy distance (AE) [41] approximates the quadratic assignment problem by two nested linear assignment problems. Sampled Gromov-Wasserstein (SaGroW) [19] and the importance sparsification method (Spar-GW) [25] approximate the original problem by sub-sampling original distributions. Low-rank Gromov-Wasserstein (LR-GW) [42] assumes the low-rank structures in the distance matrices and transport plan. However, these approximation or sampling methods always lead to low accurate GW distance and incomplete GW transport plans. On the other hand, closed-form solutions for GW can be found on some specific structures, for example, tree (FlowAlign) [23].

In this work, we propose a novel acceleration method to conduct the accurate entropic GW. The key of the method is to utilize the structure of the distance matrices and dynamic programming techniques to reduce the complexity of the matrix multiplication from $\mathcal{O}(N^3)$ time to $\mathcal{O}(N^2)$. It directly speeds up the gradient computation that was once the bottleneck in efficiency, which is somehow inspired by the fast Sinkhorn algorithm [27, 28] and fast algorithms for matrix multiplication [18, 21]. Therefore, we name the method fast gradient computation for GW metric (FGC-GW). In contrast to the previous methods, FGC-GW presents a rare combination of three advantages: (1) running with low time complexity, (2) computing accurate metrics, (3) inducing exact and full-sized plans. Moreover, it can be extended to some popular GW variants, including FGW and UGW. We call these extensions FGC-FGW and FGC-UGW, respectively.

This paper is organized as follows. Section 2 briefly reviews the GW distance and the entropic GW algorithm. In Section 3, we present the fast gradient computation for GW metric and generalize it to higher dimensions. Extensive experiments in Section 4 highlight the efficiency, accuracy, and effectiveness of FGC-GW and FGC-FGW. Finally, we conclude the work and discuss future work in Section 5.

2 Gromov-Wasserstein distance

Given two probability density functions $u_{\mathcal{X}}(x)$ and $v_{\mathcal{Y}}(y)$ defined in metric spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively, the Gromov-Wasserstein distance [31] is defined as

$$\begin{aligned} &GW^2\big(u_{\mathcal{X}}(x), d_{\mathcal{X}}, v_{\mathcal{Y}}(y), d_{\mathcal{Y}}\big) \\ =& \inf_{\gamma(x,y)\in\mathcal{S}} \int_{\mathcal{X}^2\times\mathcal{Y}^2} |d_{\mathcal{X}}(x,x') - d_{\mathcal{Y}}(y,y')|^2 \gamma(x,y)\gamma(x',y') \,\mathrm{d}x\,\mathrm{d}x'\,\mathrm{d}y\,\mathrm{d}y', \\ &\mathcal{S} = \left\{ \gamma(x,y) \;\middle|\; \int_{\mathcal{Y}} \gamma(x,y)\,\mathrm{d}y = u_{\mathcal{X}}(x), \int_{\mathcal{X}} \gamma(x,y)\,\mathrm{d}x = v_{\mathcal{Y}}(y) \right\}. \end{aligned} \tag{2.1}$$

Here $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ are arbitrary metrics on $\mathcal{X}$ and $\mathcal{Y}$. Usually, the k-th power of distances is preferable [36]. For example, in 1D space,

$$d_{\mathcal{X}}(x,x') = |x-x'|^{k_{\mathcal{X}}}, \quad d_{\mathcal{X}}(y,y') = |y-y'|^{k_{\mathcal{Y}}},$$

where $k_{\mathcal{X}}, k_{\mathcal{Y}} \in \mathbb{N}^+$. For numerical computation, we discretize two probability distributions on two uniform grids with spaces of $h_{\mathcal{X}}$ and $h_{\mathcal{Y}}$. Then, the discrete distributions $\boldsymbol{u}_{\mathcal{X}}$ and $\boldsymbol{u}_{\mathcal{Y}}$ can be represented by vectors

$$\boldsymbol{u}_{\mathcal{X}} = (u_1, u_2, \cdots, u_M), \quad \boldsymbol{v}_{\mathcal{X}} = (v_1, v_2, \cdots, v_N).$$

In this paper, our discussion is general for any $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$. For the sake of simplicity, we assume $k_{\mathcal{X}} = k_{\mathcal{Y}} = k$. Naturally, there are two symmetric distance matrices

$$\begin{aligned} D_{\mathcal{X}} &= \left[d_{ij}^{\mathcal{X}}\right]_{M\times M}, \quad d_{ij}^{\mathcal{X}} = h_{\mathcal{X}}^k |i-j|^k, \\ D_{\mathcal{Y}} &= \left[d_{pq}^{\mathcal{Y}}\right]_{N\times N}, \quad d_{pq}^{\mathcal{Y}} = h_{\mathcal{Y}}^k |p-q|^k. \end{aligned} \tag{2.2}$$

Therefore, the Gromov-Wasserstein distance is discretized as the following quadratic assignment problem:

$$\begin{aligned} &GW^2(\boldsymbol{u}_{\mathcal{X}}, D_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}}, D_{\mathcal{Y}}) = \min_{\Gamma\in S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}})} \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma), \\ &\mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) = \sum_{i,j}^{M}\sum_{p,q}^{N} \left(d_{ij}^{\mathcal{X}} - d_{pq}^{\mathcal{Y}}\right)^2 \gamma_{ip}\gamma_{jq}, \\ &S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}}) = \left\{ \Gamma = [\gamma_{ip}]_{M\times N} \,\middle|\, \sum_{i=1}^{M} \gamma_{ip} = v_p, \sum_{p=1}^{N} \gamma_{ip} = u_i, \gamma_{ip} \geq 0, \ \forall\, i, p \right\}. \end{aligned}$$

It is non-convex and intractable to solve [48].

Entropic GW is a seminal and now the most popular model to approximate GW distance, proposed originally in [37]. It tries to minimize the GW objective with an entropic regularization term

$$\begin{aligned} &GW_\varepsilon^2(\boldsymbol{u}_{\mathcal{X}}, D_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}}, D_{\mathcal{Y}}) = \min_{\Gamma\in S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}})} \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) + \varepsilon H(\Gamma), \\ &H(\Gamma) := \sum_{i=1}^{M}\sum_{p=1}^{N} \gamma_{ip}(\ln \gamma_{ip} - 1), \end{aligned} \tag{2.3}$$

where $\varepsilon > 0$ is the regularization parameter.

2.1 Mirror descent method

We here introduce mirror decent [7, 53], which is the best-known method to solve problem (2.3). Given a differentiable function $h(\Gamma)$ on $\mathbb{R}^{M\times N}$ that is λ-strongly convex with a specific norm $\|\cdot\|$, a mirror map can be defined as

$$\nabla h(\Gamma) : \mathbb{R}^{M\times N} \ \rightarrow \ \mathbb{R}^{M\times N}. \tag{2.4}$$

The inverse map exists due to the differentiability and strong convexity. The l-th iteration step of the mirror descent method for solving a general constrained optimization problem $\min_{\Gamma\in\Phi\subset\mathbb{R}^{M\times N}} f(\Gamma)$ is as follows:

i. Map to the dual space: $\eta^{(l)} = \nabla h(\Gamma^{(l)})$.

ii. Perform gradient descent in the dual space with step τ: $\eta^{(l+1)} = \eta^{(l)} - \tau\nabla f(\Gamma^{(l)})$.

iii. Map back to the primal space: $\Gamma^{(l+1/2)} = \nabla^{-1} h(\eta^{(l+1)})$.

iv. Project $\Gamma^{(l+1/2)}$ back into the feasible region: $\Gamma^{(l+1)} = \min_{\Gamma \in \Phi} D_h(\Gamma \| \Gamma^{(l+1/2)})$, where

$$D_h(\Gamma \| \Gamma') := h(\Gamma) - h(\Gamma') - \langle \nabla h(\Gamma'), \Gamma - \Gamma' \rangle$$

is the Bregman divergence [9].

When h takes the negative entropy function (1-strongly convex concerning $\|\cdot\|_1$), i.e. $h{=}H$, the associated mirror map, its inverse, and the Bregman divergence are formulated as

$$\nabla H(\Gamma) = \ln(\Gamma), \quad \nabla^{-1} H(\eta) = e^{\eta}, \quad D_H(\Gamma \| \Gamma') = \sum_{i}^{M} \sum_{p}^{N} \left(\gamma_{ip} \ln \left(\frac{\gamma_{ip}}{\gamma'_{ip}} \right) - \gamma_{ip} + \gamma'_{ip} \right).$$

In this context, the mirror descent step of entropic GW is ultimately represented as

$$\begin{aligned} \Gamma^{(l+1)} &= \min_{\Gamma \in S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}})} D_H\Big(\Gamma \| \big(\Gamma^{(l)} \odot e^{-\tau \nabla f(\Gamma^{(l)})}\big)\Big), \\ \nabla f(\Gamma^{(l)}) &= \nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma^{(l)}) + \varepsilon \ln(\Gamma^{(l)}), \end{aligned} \tag{2.5}$$

where $\odot$ is the Hadamard product. It is shown in [8] that the above minimization problem is equivalent to the following regularized transport problem:

$$\Gamma^{(l+1)} = \operatorname*{arg\,min}_{\Gamma \in S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}})} \langle \Pi, \Gamma \rangle + \epsilon H(\Gamma) \tag{2.6}$$

with

$$\Pi = -\varepsilon \ln \left(\Gamma^{(l)} \odot e^{-\tau \nabla f(\Gamma^{(l)})} \right) = \tau\varepsilon \nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma^{(l)}) + (\tau\varepsilon^2 - \varepsilon) \ln(\Gamma^{(l)})$$

being cost matrix, which can be solved with the Sinkhorn algorithm [15].

As we stated before, the evaluation of Π is time-consuming. More concretely, evaluating the component $\nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma)$ requires $\mathcal{O}(M^2 N^2)$ time for the reason that its (i, p)-th entry reads

$$\left[\nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) \right]_{ip} = 2 \sum_{j=1}^{M} \sum_{q=1}^{N} \left(d_{ij}^{\mathcal{X}} - d_{pq}^{\mathcal{Y}} \right)^2 \gamma_{jq}. \tag{2.7}$$

That is far more than the $\mathcal{O}(MN)$ time of the Sinkhorn algorithm. Fortunately, it is observed in [37] that $\nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma)$ can be decomposed into a constant term $\mathcal{C}_1$ and a linear term of T. Specifically,

$$\begin{aligned} &\nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) = \mathcal{C}_1 - 4 D_{\mathcal{X}} \Gamma D_{\mathcal{Y}}, \\ &\mathcal{C}_1 = 2\big((D_{\mathcal{X}} \odot D_{\mathcal{X}}) \boldsymbol{u}_{\mathcal{X}} \mathbf{1}_N^{\top} + (D_{\mathcal{Y}} \odot D_{\mathcal{Y}}) \boldsymbol{u}_{\mathcal{Y}} \mathbf{1}_M^{\top} \big). \end{aligned}$$

Computing $\mathcal{C}_1$ costs $\mathcal{O}(M^2 + N^2 + MN)$ time and would be only performed once. Therefore, with the decomposition, the overall complexity of the solution of Entropic GW is reduced to $\mathcal{O}(MN^2 + M^2 N)$, dominated by the computation of $D_{\mathcal{X}} \Gamma D_{\mathcal{Y}}$. This complexity is still unacceptable in practice and worth our further exploration.

Remark 2.1. In the existing literature, $\tau = 1/\varepsilon$ is suggested [37]. Then Eq. (2.6) turns to

$$\Gamma^{(l+1)} \leftarrow \underset{\Gamma \in S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}})}{\arg\min} \left\langle \nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma^{(l)}), \Gamma \right\rangle + \varepsilon H(\Gamma).$$

We follow this in the subsequent discussion. Since it takes only $\mathcal{O}(MN)$ time to calculate $\ln(\Gamma^{(l)})$, whether τ equals $1/\varepsilon$ makes no difference in our statement about complexity.

Remark 2.2 (Entropic Algorithm for FGW)**.** Let $C = [c_{ip}]_{M\times N}$ be an additional cost matrix between $\boldsymbol{u}_{\mathcal{X}}$ and $\boldsymbol{v}_{\mathcal{Y}}$. Fused GW (FGW) minimizes the objective

$$\bar{\mathcal{E}}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) = (1-\theta) \cdot \sum_{i=1}^{M} \sum_{p=1}^{N} c_{ip}^2 \gamma_{ip} + \theta \cdot \sum_{i,j}^{M} \sum_{p,q}^{N} \left(d_{ij}^{\mathcal{X}} - d_{pq}^{\mathcal{Y}}\right)^2 \gamma_{ip}\gamma_{jq}$$

in the region $S(\boldsymbol{u}_{\mathcal{X}}, \boldsymbol{v}_{\mathcal{Y}})$, where $\theta \in [0,1]$ balances the effect of the linear and quadratic assignment [52]. The iteration formula (2.6) applies to FGW as well. Analogous to GW, its gradient has the decomposition

$$\begin{aligned}
&\nabla \bar{\mathcal{E}}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) = \mathcal{C}_2 - 4\theta \cdot D_{\mathcal{X}} \Gamma D_{\mathcal{Y}}, \\
&\mathcal{C}_2 = (1-\theta) \cdot C \odot C + 2\theta \cdot \left((D_{\mathcal{X}} \odot D_{\mathcal{X}})\boldsymbol{u}_{\mathcal{X}} \mathbf{1}_N^\top + (D_{\mathcal{Y}} \odot D_{\mathcal{Y}})\boldsymbol{u}_{\mathcal{Y}} \mathbf{1}_M^\top\right).
\end{aligned}$$

C_2 can also be computed in $\mathcal{O}(M^2 + N^2 + MN)$ time, so $D_{\mathcal{X}}\Gamma D_{\mathcal{Y}}$ still dominates the overall complexity.

Remark 2.3 (Entropic Algorithm for UGW)**.** Given $\rho > 0, \hat{S} = \{\Gamma | \gamma_{ip} \geq 0, \forall i, p\}$, the unbalanced Gromov-Wasserstein divergence is defined as

$$\min_{\Gamma \in \hat{S}} \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma) + \rho KL\left((\Gamma \mathbf{1}) \otimes (\Gamma \mathbf{1}) | \boldsymbol{u}_{\mathcal{X}} \otimes \boldsymbol{u}_{\mathcal{X}}\right) + \rho KL\left((\Gamma^\top \mathbf{1}) \otimes (\Gamma^\top \mathbf{1}) | \boldsymbol{v}_{\mathcal{Y}} \otimes \boldsymbol{u}_{\mathcal{Y}}\right).$$

The key to its entropic algorithms is to solve

$$\begin{aligned}
\min_{\Gamma \in \hat{S}} &\left\langle \frac{1}{2} \nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma^{(l)}) + g(\Gamma^{(l)}), \Gamma \right\rangle \\
&+ \mathbf{1}^\top \Gamma^{(l)} \mathbf{1} \left(\rho \mathrm{KL}(\Gamma \mathbf{1} | \boldsymbol{u}_{\mathcal{X}}) + \rho \mathrm{KL}\left(\Gamma^\top \mathbf{1} | \boldsymbol{u}_{\mathcal{Y}}\right) + \varepsilon \mathrm{KL}(\Gamma | \boldsymbol{u}_{\mathcal{X}} \otimes \boldsymbol{u}_{\mathcal{Y}})\right)
\end{aligned}$$

at the l-th iteration [44]. Still the computation of $\nabla \mathcal{E}_{D_{\mathcal{X}}, D_{\mathcal{Y}}}(\Gamma^{(l)})$ or rather $D_{\mathcal{X}}\Gamma^{(l)} D_{\mathcal{Y}}$ is to blame for $\mathcal{O}(M^2N + MN^2)$ complexity, while the other parts take no more than $\mathcal{O}(M^2 + N^2 + MN)$ time. The method proposed in this paper will also apply here.

3 Fast gradient computation

In this section, we present an efficient method that computes $D_{\mathcal{X}}\Gamma D_{\mathcal{Y}}$ in $\mathcal{O}(MN)$ time. For the sake of simplicity, we assume $M = N$. [1]

[1]The method can easily handle the case where M is not equal to N. This assumption is without loss of generality.

First, we note that the distance matrices $D_{\mathcal{X}}, D_{\mathcal{Y}}$ satisfy

$$D_{\mathcal{X}} = h_{\mathcal{X}}^k \tilde{D}, \quad D_{\mathcal{Y}} = h_{\mathcal{Y}}^k \tilde{D},$$

where

$$\tilde{D} = L + L^\top, \quad L = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 2 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \\ (N-1) & (N-2) & \cdots & 1 & 0 \end{pmatrix}^{\odot k}. \tag{3.1}$$

The operator $\odot k$ refers to raising each element of the matrix to the power of k. Therefore, $D_{\mathcal{X}} \Gamma D_{\mathcal{Y}}$ can be expanded as

$$\begin{aligned} D_{\mathcal{X}} \Gamma D_{\mathcal{Y}} &= h_{\mathcal{X}}^k h_{\mathcal{Y}}^k (\tilde{D} \Gamma \tilde{D}) = h_{\mathcal{X}}^k h_{\mathcal{Y}}^k \big(L\Gamma L + L\Gamma L^\top + L^\top \Gamma L + L^\top \Gamma L^\top\big) \\ &= h_{\mathcal{X}}^k h_{\mathcal{Y}}^k \big(L(L^\top \Gamma^\top)^\top + L(L\Gamma^\top)^\top + L^\top (L^\top \Gamma^\top)^\top + L^\top (L\Gamma^\top)^\top\big). \end{aligned} \tag{3.2}$$

Next, we show that for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)^\top \in \mathbb{R}^N$, $\boldsymbol{y} = L\boldsymbol{x}$ can be implemented with $\mathcal{O}(N)$ element-wise operations. $L^\top \boldsymbol{x}$ can be computed in a similar way. As a result, the total time cost of computing equation (3.2) will be no more than $\mathcal{O}(N^2)$. To be concrete,

$$\boldsymbol{y} = \left(0, \sum_{j=1}^{1} (2-j)^k x_j, \cdots, \sum_{j=1}^{N-1} (N-j)^k x_j\right)^\top. \tag{3.3}$$

Defining $N \times (k+1)$ elements

$$a_{ir} = \sum_{j=1}^{i-1} (i-j)^{r-1} x_j, \quad i \in \{1, \ldots, N\}, \quad r \in \{1, \ldots, k+1\},$$

we obtain that the i-th entry of $\boldsymbol{y}$

$$y_i = a_{i,k+1}, \quad i \in \{1, \ldots, N\}.$$

A significant observation is that

$$\begin{aligned} a_{i+1,r} &= x_i + \sum_{j=1}^{i-1} (i-j+1)^{r-1} x_j = x_i + \sum_{j=1}^{i-1} \sum_{s=1}^{r} \binom{r-1}{s-1} (i-j)^{s-1} x_j \\ &= x_i + \sum_{s=1}^{r} \binom{r-1}{s-1} \sum_{j=1}^{i-1} (i-j)^{s-1} x_j = x_i + \sum_{s=1}^{r} \binom{r-1}{s-1} a_{is} \\ &= x_i + a_{i1} + \binom{r-1}{1} a_{i2} + \cdots + \binom{r-1}{r-1} a_{ir}. \end{aligned} \tag{3.4}$$

In other word, we can calculate $a_{i+1,r}$ by making linear combination of $\{a_{is}\}_{s=1}^{r}$ recursively. With $\{a_{is}\}_{s=1}^{r}$ and all the binomial coefficients being known[2], it needs only $r-1$ multiplications and r additions to get $a_{i+1,r}$.

In view of the fact that $a_{1r}=0$ for any r, we conclude that it takes

$$(N-1)\sum_{r=1}^{k+1}(r-1)=(N-1)\frac{k(k+1)}{2}$$

multiplications and

$$(N-1)\sum_{r=1}^{k+1}r=(N-1)\frac{(k+2)(k+1)}{2}$$

additions to evaluate all a_{ir} in the order of

$$a_{11},\cdots,a_{N,1},\cdots,a_{1,k+1},\cdots,a_{N,k+1}.$$

Therefore, the computation of $\boldsymbol{y}=L\boldsymbol{x}$ is finished in $\mathcal{O}(k^2N)$ time. In practice, k refers to the power of the distance and typically takes 1 or 2. So the total cost is $\mathcal{O}(N)$ for short.

3.1 Extension to high dimension

In this part, we illustrate the generalization of FGC-GW to 2D space. And there is no essential difference to generalizing the algorithm to higher dimensional space.

Consider two probabilistic distributions

$$\boldsymbol{u}_{\mathcal{X}}=\begin{pmatrix}u_{11}&u_{12}&\cdots&u_{1n}\\u_{21}&u_{22}&\cdots&u_{2n}\\\vdots&\vdots&\ddots&\vdots\\u_{n1}&u_{n2}&\cdots&u_{nn}\end{pmatrix},\quad \boldsymbol{v}_{\mathcal{X}}=\begin{pmatrix}v_{11}&v_{12}&\cdots&v_{1n}\\v_{21}&v_{22}&\cdots&v_{2n}\\\vdots&\vdots&\ddots&\vdots\\v_{n1}&v_{n2}&\cdots&v_{nn}\end{pmatrix}$$

on two uniform 2D grids with both sizes $n\times n$, which contributes to the total number of grid points $N=n^2$. For simplicity, we set both the horizontal and vertical spacing of space $\mathcal{X}$ to $h_{\mathcal{X}}$ and those of space $\mathcal{Y}$ to $h_{\mathcal{Y}}$. Ordering the grid points of $\boldsymbol{u}_{\mathcal{X}}$ and $\boldsymbol{v}_{\mathcal{Y}}$ in column-major order, by analogy with (2.2), we get two distance matrices on 2D where

$$D_{\mathcal{X}}=h_{\mathcal{X}}^{k}\hat{D},\quad D_{\mathcal{Y}}=h_{\mathcal{Y}}^{k}\hat{D}. \tag{3.5}$$

Here

$$\hat{D}=\begin{pmatrix}D_1&D_1+J&D_1+2J&\cdots&D_1+(n-1)J\\D_1+J&D_1&D_1+J&\cdots&D_1+(n-2)J\\D_1+2J&D_1+J&D_1&\cdots&D_1+(n-3)J\\\vdots&\vdots&\vdots&\ddots&\vdots\\D_1+(n-1)J&D_1+(n-2)J&D_1+(n-3)J&\cdots&D_1\end{pmatrix}^{\odot k}_{n^2\times n^2} \tag{3.6}$$

[2]In fact, all the binomial coefficients can be computed in $\mathcal{O}(k^2)$ time [39].

with

$$D_1 = \begin{pmatrix} 0 & 1 & 2 & \cdots & n-1 \\ 1 & 0 & 1 & \cdots & n-2 \\ 2 & 1 & 0 & \cdots & n-3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n-1 & n-2 & n-2 & \cdots & 0 \end{pmatrix}_{n\times n}, \quad J = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}_{n\times n}.$$

Since the distance matrices in 2D and 1D take different forms, we here use the notation $\hat{D}$ for the 2D case to distinguish it from $\tilde{D}$ in Eq. (3.1). The (i,j)-th block in $\hat{D}$ represents the k-th power of the pairwise distances between the points in the i-th and j-th columns. At this point,

$$D_{\mathcal{X}}\Gamma D_{\mathcal{Y}} = h_{\mathcal{X}}^k h_{\mathcal{Y}}^k(\hat{D}\Gamma\hat{D}) = h_{\mathcal{X}}^k h_{\mathcal{Y}}^k\big(\hat{D}(\hat{D}\Gamma^\top)^\top\big). \tag{3.7}$$

Notice that any $\Gamma \in \mathbb{R}^{n^2\times n^2}$ can be partitioned into n^2 vectors in $\mathbb{R}^{n^2}$. Whereafter, we show that for any $x \in \mathbb{R}^{n^2}$, $\hat{D}x$ can be performed in $\mathcal{O}(n^2)$ time. In this way, (3.7) can be finished in total $\mathcal{O}(N^2) = \mathcal{O}(n^4)$ time.

One can expand $\hat{D}$ as

$$\hat{D} = \sum_{r=0}^{k} \binom{k}{r} D_1^{\odot r} \otimes D_1^{\odot k-r}$$

with the notion $\otimes$ representing the Kronecker product. Therefore,

$$\hat{D}x = \sum_{r=0}^{k} \binom{k}{r} \big(D_1^{\odot r} \otimes D_1^{\odot k-r}\big)x = \sum_{r=0}^{k} \binom{k}{r} \mathrm{vec}\big(D_1^{\odot k-r}\mathrm{mat}(x)D_1^{\odot r}\big), \tag{3.8}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix, that is, for $Q = [q_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$,

$$\mathrm{vec}(Q) = (q_{11}, \cdots, q_{1n}, q_{21}, \cdots, q_{2n}, \cdots, q_{n1}, \cdots, q_{nn})^\top \in \mathbb{R}^{n^2}.$$

And $\mathrm{mat}(\cdot)$ denotes the matrixization of a vector, i.e. the inverse transformation of vectorization. Note the similarity between $D_1^{\odot k-r}\mathrm{mat}(x)D_1^{\odot r}$ and $\tilde{D}\Gamma\tilde{D}$ in Eq. (3.2). It is obvious that $D_1^{\odot k-r}\mathrm{mat}(x)D_1^{\odot r}$ can be computed in $\mathcal{O}(k^2n^2)$ time, by leveraging the approach in 1D case. Totally, the cost of computing $\hat{D}x$ in Eq. (3.8) is $\mathcal{O}(k^4n^2)$, for short, $\mathcal{O}(n^2)$.

4 Numerical experiments

In this section, we perform several experiments to validate the effectiveness and efficiency of the proposed method. We conduct experiments over the 1D and 2D random distributions, time series [52], and images [16, 59]. For the experiments on random distributions, we evaluate both Gromov-Wasserstein and Fused Gromov-Wasserstein (FGW) metrics. We compare the FGC with the original computation by Entropic (Fused) Gromov-Wasserstein. For time series [52] and image data [16,59], we consider computing the FGW

metric to quantify their similarity. We emphasize that FGW allows for the inclusion of feature information in addition to structure information. This characteristic leads to FGW being viewed as more suitable for the task in comparison to GW [52].

To ensure the fairness of evaluation, we consider sequential implementation in this paper. The original entropic (F)GW and FGC implementations are realized using the C++ language, leveraging the vector inner-product functionality offered by the Eigen library. All experiments are executed with a single-thread program over a platform with 128 G RAM and one Intel(R) Xeon(R) Gold 5117 CPU @2.00 GHz.

4.1 1D random distributions

We consider two 1D random distributions $u_{\mathcal{X}} = (u_1, \cdots, u_N)$ and $v_{\mathcal{Y}} = (v_1, \cdots, v_N)$ on uniform grid points

$$x_i = y_i = \frac{i-1}{N-1}, \quad i = 1, 2, \ldots, N.$$

All u_i and v_i are sampled from a uniform distribution over $[0, 1]$ and then normalized so that $u_{\mathcal{X}}$ and $v_{\mathcal{Y}}$ are distributions. Our objective is to compare the performance and computational cost of our FGC algorithms and the original entropic algorithms on computing the corresponding GW metric and FGW metric ($\theta = 0.5$). We take $k = 1$ for $D_{\mathcal{X}}, D_{\mathcal{Y}}$ in (2.2) and $c_{ip} = |i - p|$ for $C = [c_{ip}]$ in Remark 2.2. The number of iterations of mirror descent (2.6) is set to 10. We test 100 random experiments for each N.

In Table 4.1, we show the averaged running time of the algorithms and the difference in the transport plans. We can see that FGC has an overwhelming advantage in computational speed. Moreover, it produces almost identical transport plans as the original entropic algorithms.

To study the efficiency advantage of FGC further, we visualize the relationship between time cost and the number of grid points in Fig. 4.1. Note that the two axes are log scales. By data fitting, we find empirical $\mathcal{O}(N^{2.22})$ and $\mathcal{O}(N^{2.20})$ complexities of FGC on GW and FGW while those of the original algorithms are $\mathcal{O}(N^{3.04})$and $\mathcal{O}(N^{3.02})$.

Table 4.1: 1D random distribution. Comparison between algorithms with FGC and the original ones with the different number of grid points N. For GW and FGW, the regularization parameter $\varepsilon = 0.002$. Column for $\|P_{Fa} - P\|_F$ validates the correctness of the results by FGC.

Metric	N	Computational time (s)		Speed-up ratio	$\|P_{Fa} - P\|_F$
		FGC	Original		
GW	500	4.97×10^{-1}	4.40×10^{0}	8.85	5.12×10^{-15}
	1000	2.13×10^{0}	3.46×10^{1}	16.2	4.33×10^{-15}
	2000	1.01×10^{1}	2.80×10^{2}	27.7	2.87×10^{-15}
	4000	5.01×10^{1}	2.44×10^{3}	48.7	2.04×10^{-15}
FGW	500	6.00×10^{-1}	4.58×10^{0}	7.63	1.08×10^{-15}
	1000	2.54×10^{0}	3.53×10^{1}	13.9	8.27×10^{-16}
	2000	1.20×10^{1}	2.83×10^{2}	23.6	5.78×10^{-16}
	4000	5.73×10^{1}	2.47×10^{3}	43.1	4.11×10^{-16}

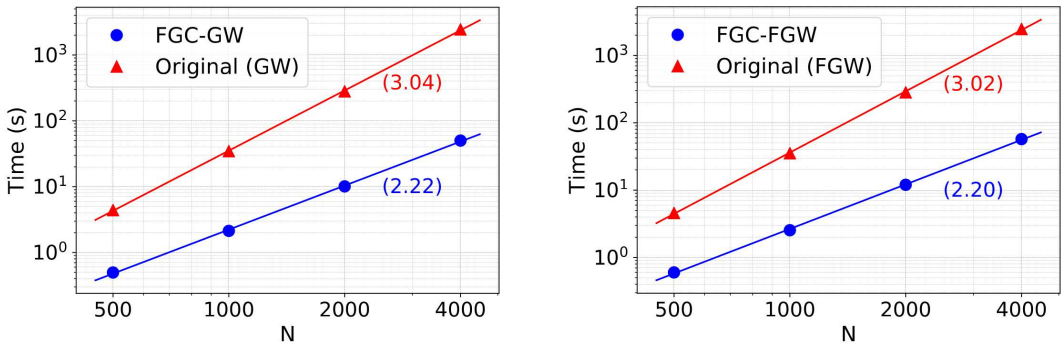

Figure 4.1: 1D random distributions. The computation time for GW (left) and FGW (right) under different N. The numbers are the fitted slopes, representing the empirical computational complexities.

4.2 2D random distributions

Next, we investigate the performance of FGC on 2D random distributions. The supports of distributions are $N = n \times n$ grid points uniform scattered on $[0,1] \times [0,1]$.

Table 4.2 and Fig. 4.2 show the results. It can be seen that the fast algorithms keep the computational advantage and make large-scale tasks possible. We also estimate their empirical complexities, $\mathcal{O}(N^{2.29})$ for GW and $\mathcal{O}(N^{2.30})$ for FGW while those of the original versions are $\mathcal{O}(N^{3.02})$ and $\mathcal{O}(N^{3.02})$, respectively.

4.3 Time series

Time series data is widely generated on a daily basis in various application domains, including bioinformatics [5, 6], finance [4, 29], engineering [32, 34], and others. Modelling different types of time series data has emerged as a significant focus in machine learning research in recent years [1, 13]. As a pre-work, it is highly important to find a good simi-

Table 4.2: 2D random distributions. Comparison between algorithms with FGC and the original ones. For GW and FGW, the regularization parameter $\varepsilon = 0.004$. Column for $\|P_{Fa} - P\|_F$ validates the correctness of the results by FGC. A dash means the running time exceeds 10 hours.

Metric	$N = n \times n$	Computational time (s)		Speed-up ratio	$\|P_{Fa} - P\|_F$
		FGC	Original		
GW	30×30	1.73×10^0	2.46×10^1	14.2	3.03×10^{-14}
	60×60	5.53×10^1	1.66×10^3	30.0	7.94×10^{-15}
	90×90	3.01×10^2	1.85×10^4	61.5	6.75×10^{-15}
	120×120	9.65×10^2	–	–	–
FGW	30×30	1.84×10^0	2.50×10^1	13.6	2.56×10^{-14}
	60×60	5.90×10^1	1.71×10^3	29.0	1.48×10^{-15}
	90×90	3.22×10^2	1.89×10^4	58.7	1.00×10^{-15}
	120×120	1.08×10^3	–	–	–

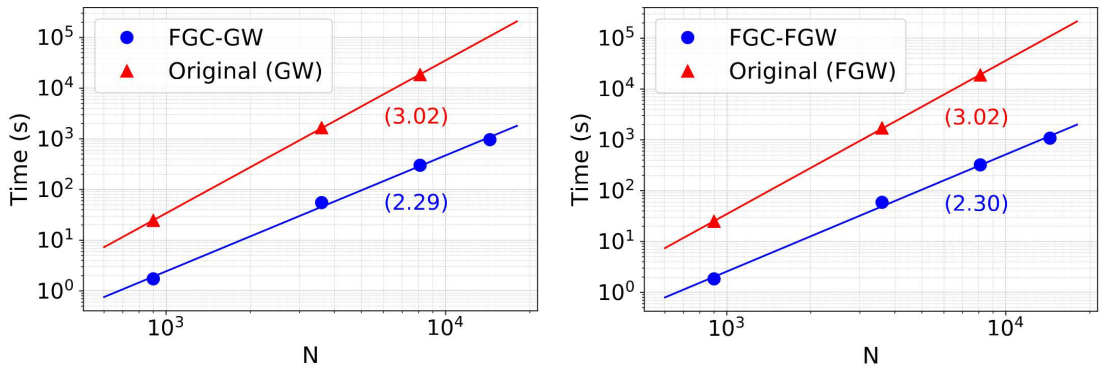

Figure 4.2: 2D random distributions. The computation time for GW (left) and FGW (right) under different N. The numbers are the fitted slopes, representing the empirical computational complexities.

larity measure for time series data [1,45]. Superior to the GW metric, FGW can effectively incorporate both signal strength (linear term) and time information (quadratic term), enabling comparatively accurate alignment of time series waveforms. Consequently, it is considered more appropriate for defining time series similarity [52].

We here investigate the acceleration effect of FGC on the FGW metric in time series. Consider a series in [0,1] that consists of two humps with heights of 0.5 and 0.8. We construct the other series by moving the humps around. Now we would like to align them using the transport plan of FGW by setting $k = 1$ for $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ in Eq. (2.2) and C as the signal strength difference. After uniform dividing the time axis, we get N sampling points for each series and then compute FGW ($\theta = 0.5$) with two algorithms. Likewise, we repeat the experiment 100 times and record the time and plans.

Table 4.3 reports the average results and Fig. 4.3 presents the $\mathcal{O}(N^{2.19})$ empirical complexity of FGC on the left. As expected, the fast algorithm demonstrates a clear computational speed advantage. Moreover, we give the plan at $N = 200$ explicitly on the right of Fig. 4.3. The good alignments illustrate that FGC can enhance the application of FGW for measuring time series similarity and thus underpins machine learning downstream tasks including time series generation and classification [1,13].

Table 4.3: Time series tasks with FGW metric. Comparison between the fasts algorithms with FGC and the original ones with the different number of grid points N. Column for $\|P_{Fa} - P\|_F$ validates the correctness of the results by FGC.

N	Computational time (s)		Speed-up ratio	$\|P_{Fa} - P\|_F$
	FGC-FGW	original		
400	3.78×10^{-1}	2.43×10^{0}	6.43	2.02×10^{-15}
800	1.59×10^{0}	1.91×10^{1}	12.0	1.45×10^{-15}
1600	7.02×10^{0}	1.54×10^{2}	21.9	1.08×10^{-15}
3200	3.59×10^{1}	1.24×10^{3}	34.5	7.17×10^{-16}

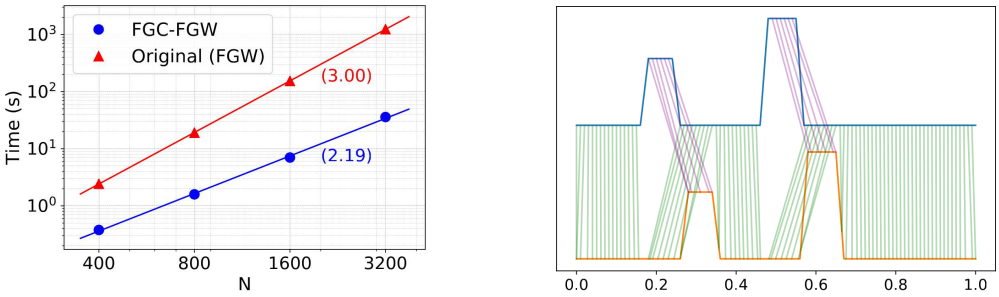

Figure 4.3: Time series alignment with FGW metric. Left: The time comparison on computing FGW metric for different numbers of sample points with entropic GW with or without FGC. Right: The visualized alignment between source (Blue) and target (Orange) time series at $N = 1600$. The lines across two time series represent the transport plan.

4.4 Image

Measuring the similarity between images is another meaningful application of FGW [52]. Intuitively, good alignment between images implies a quite accurate depiction of similarity [52]. Likewise, FGW does a better job than GW in this task, because it can take advantage of not only pixel coordinates (quadratic term) but also pixel values (linear term). Nevertheless, suffering from high computational complexity, FGW is prevented from applying to high-resolution images. By reducing the computational time significantly, FGC makes it possible.

4.4.1 Three invariances on handwritten digits image data

The advanced capability of FGW in capturing image similarity can be demonstrated by its invariant alignment under types of transformations, such as translation, rotation, and reflection. We will show that FGC acceleration preserves the exact same invariance. We first selected one 28×28 image representing digits 3 from MNIST dataset [16], then we made the new ones by translation, mirroring, and rotation of the original image. The objective is to align the original image with the others. We use the Manhattan distance on the pixel coordinate gird for $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$, i.e. take $k = 1, h_{\mathcal{X}} = h_{\mathcal{Y}} = 1$ in Eq. (3.5). C is constructed by calculating the difference in the pixel gray levels between source pixels and target pixels. Referring to [52], we take $\theta = 0.1$.

The average results of 100 runs are shown in Table 4.4. It is observed that our fast algorithm defeats the original one again, which costs about 3s and achieves about 10 times the speedup. Fig. 4.4 shows the actual transport plans. For a clear view, plans are marked with two colors. Graphically, the FGW metric makes each pixel of the digit aligned well.

4.4.2 Complex deformation on horse image data

In this part, to further emphasize the practical value of FGC on images, we conduct large-scale experiments on two 450×300 images of a running horse captured from a video [59]. The shapes of the horse in the images reveal complex deformation during running.

Table 4.4: Handwritten digits task with FGW metric. Comparison between FGC-FGW and the original algorithm on aligning three pairs of images under different types of transformations.

Invariance	Computational time (s)		Speed-up ratio	$\|P_{Fa} - P\|_F$
	FGC-FGW	Original		
Translation	2.86×10^0	2.86×10^1	10.0	6.96×10^{-14}
Rotation	2.34×10^0	2.26×10^1	9.66	1.51×10^{-14}
Reflection	2.34×10^0	2.27×10^1	9.70	1.39×10^{-14}

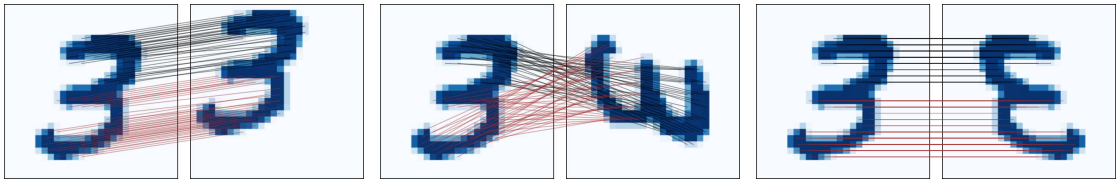

Figure 4.4: Handwritten digits task with FGW metric. The original image is matched to the new ones made by translation (left), rotation (mid), and reflection (right). The black lines and red lines represent transport plans.

Before alignment, they are subsampled to $N = n \times n$ first and then converted to grayscale images. We set $D_{\mathcal{X}}, D_{\mathcal{Y}}$ and C the same as the handwritten digits task, except for taking $h_{\mathcal{X}} = h_{\mathcal{Y}} = 100/n$ to make $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ comparable with C in magnitude. We computed FGW with various $\theta = 0.4, 0.6, 0.8$ to show that FGC captures the original invariance with complexity advantage.

Table 4.5 reports the average results of 100 runs and Fig. 4.5 presents the $\mathcal{O}(N^{2.32})$ empirical complexity of FGC at $\theta = 0.8$ on the left. FGC enables the processing of all images in less than 10 minutes regardless of N while the original algorithm would struggle to deal with those of size 100×100. When $N = 80$, FGC brings a 40x acceleration roughly. And this would be promoted to more than 60x at $N = 100$. Fig. 4.5 also shows the plans obtained at $\theta = 0.8, N = 100$. We can see that the horse's head, tail, and legs are aligned well.

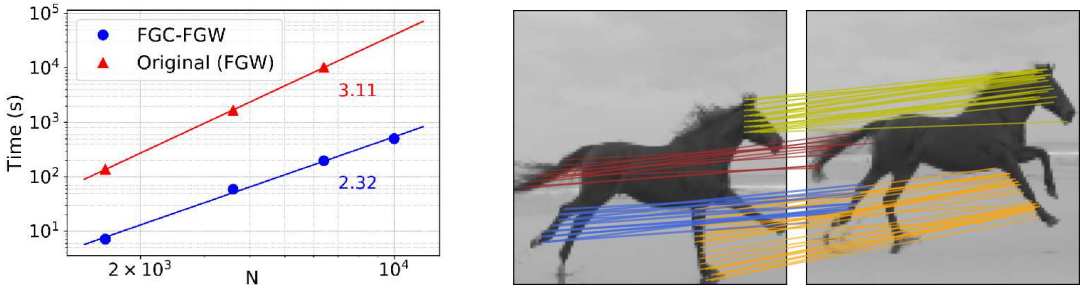

Figure 4.5: Horse images task with FGW metric. Left: the time comparison between FGC-FGW and the original algorithm with $\theta = 0.8$. The numbers attached are fitted slops, representing empirical complexities. Right: the transport plans with $\theta = 0.8$ on 100×100 images.

Table 4.5: Horse images task with FGW metric. Comparison between FGC-FGW and the original algorithm with different N and θ. A dash means the running time exceeds 10 hours.

θ	$N = n \times n$	Computational time (s)		Speed-up ratio	$\|P_{Fa} - P\|_F$
		FGC-FGW	Original		
0.4	40×40	7.08×10^0	1.39×10^2	19.6	4.25×10^{-16}
	60×60	5.92×10^1	1.68×10^3	28.4	3.54×10^{-16}
	80×80	1.99×10^2	9.59×10^3	48.2	2.87×10^{-16}
	100×100	5.02×10^2	–	–	–
0.6	40×40	7.20×10^0	1.39×10^2	19.3	7.30×10^{-16}
	60×60	5.91×10^1	1.66×10^3	28.1	5.89×10^{-16}
	80×80	1.99×10^2	9.79×10^3	49.2	4.65×10^{-16}
	100×100	5.07×10^2	–	–	–
0.8	40×40	7.18×10^0	1.38×10^2	19.2	1.01×10^{-15}
	60×60	5.92×10^1	1.67×10^3	28.2	8.09×10^{-16}
	80×80	1.98×10^2	1.03×10^4	52.0	6.88×10^{-16}
	100×100	5.04×10^2	–	–	–

5 Conclusion and future work

In this paper, we demonstrate a special recursive relation over uniform grids and develop the Fast GW Gradient Computation. Thus, the computation of GW can be conducted in $\mathcal{O}(N^2)$ time, which is almost optimal. Compared to the approximation or sampling-based methods, the fast algorithms produce full-sized and exact solutions as original entropic ones. Moreover, compared to other closed-form GW solutions over other special structures such as trees, our method can be used to accelerate the computation of a wide scope of GW variants as long as the GW gradient is required, including unbalanced GW [44], co-optimal transport [51], and fixed support GW barycenter [37]. Empirical evaluations show that our FGC has a clear advantage in terms of time complexity and accelerates the cases from tens to hundreds of times. It is also validated that full-sized transport plans produced by our algorithm are exact under various settings.

A more interesting question is whether FGC extends to other discrete cases and regularization methods. In fact, FGC's effectiveness is attributed to the examination of the structure of the distance matrix. Under non-uniform grids, the lower and upper triangular parts of the distance matrix maintain special structures similar to the lower (upper) collinear triangular matrix (L-CoLT/U-CoLT matrix) proposed in [28]. This property enables an analogous recursive relation, resulting in the same $\mathcal{O}(N^2)$ time complexity. For general point clouds, the matrices also exhibit certain structures. Fully exploiting them is meaningful but needs further investigation. For graphs, if high-quality node embeddings are available, it is promising to be transformed into a point cloud problem. On the other hand, for other regularization methods, such as Xu's approach [56, 58], the FGC method remains applicable because computational bottlenecks still exist in the calculation of the GW gradient.

Finally, we demonstrate that FGC effectively enhances the application of the FGW metric in quantifying data similarity, which serves as an indispensable component in various machine learning tasks or stages, such as data preprocessing [2, 49], unsupervised learning [33, 54], recommendation systems [26, 38], and anomaly detection [12, 43]. The alignment achieved by the transport plan highlights the quality of the computed similarity. Future work is expected to apply FGC directly to specific machine learning downstream tasks, where some challenges, such as high dimensional data, may need to be overcome.

6 Acknowledgements

The authors would also thank the anonymous reviewers for their great efforts and valuable comments in improving the quality of the manuscript.

This work was supported by the National Natural Science Foundation of China (Grant No. 12271289).

References

[1] A. Abanda, U. Mori, and J. A. Lozano, A review on distance based time series classification, *Data Min. Knowl. Discov.*, 33(2):378–412, 2019.

[2] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*, CRC Press, 2014.

[3] D. Alvarez-Melis and T. Jaakkola, Gromov-Wasserstein alignment of word embedding spaces, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 1881–1890, 2018.

[4] J. Arroyo, R. Espínola, and C. Maté, Different approaches to forecast interval time series: A comparison in finance, *Comput. Econ.*, 37(2):169–191, 2011.

[5] Z. Bar-Joseph, Analyzing time series gene expression data, *Bioinformatics*, 20(16):2493–2503, 2004.

[6] Z. Bar-Joseph, A. Gitter, and I. Simon, Studying and modelling dynamic biological processes using time-series gene expression data, *Nat. Rev. Genet.*, 13(8):552–564, 2012.

[7] A. Beck and M. Teboulle, Mirror descent and nonlinear projected subgradient methods for convex optimization, *Oper. Res. Lett.*, 31(3):167–175, 2003.

[8] J. D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré, Iterative Bregman projections for regularized transportation problems, *J. Sci. Comput.*, 37(2):A1111–A1138, 2015.

[9] L. M. Bregman, The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming, *USSR Comput. Math. Math. Phys.*, 7(3):200–217, 1967.

[10] C. Bunne, D. Alvarez-Melis, A. Krause, and S. Jegelka, Learning generative models across incomparable spaces, in: *Proceedings of the 36th International Conference on Machine Learning*, PMLR, 97:851–861, 2019.

[11] D. Burago, Y. Burago, and S. Ivanov, *A Course in Metric Geometry. Graduate Studies in Mathematics*, in: *Graduate Studies in Mathematics*, AMS, Vol. 33, 2001.

[12] V. Chandola, A. Banerjee, and V. Kumar, Anomaly detection: A survey, *ACM Comput. Surv. (CSUR)*, 41(3):15, 2009.

[13] K. Cheng, S. Aeron, M. C. Hughes, and E. L. Miller, Dynamical Wasserstein barycenters for time-series modeling, *Adv. Neural Inf. Process. Syst.*, 34:27991–28003, 2021.

[14] S. Chowdhury and T. Needham, Generalized spectral clustering via Gromov-Wasserstein learning, in: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, PMLR, 130:712–720, 2021.

[15] M. Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport, *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., Vol. 26, 2013.

[16] L. Deng, The MNIST database of handwritten digit images for machine learning research [best of the web], IEEE Signal Process. Mag., 29(6):141–142, 2012.
[17] M. Gromov, *Metric Structures for Riemannian and Non-Riemannian Spaces*, Birkhäuser, 2007.
[18] P. Indyk and S. Silwal, Faster linear algebra for distance matrices, in: *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 35:35576–35589, 2022.
[19] T. Kerdoncuff, R. Emonet, and M. Sebban, Sampled Gromov Wasserstein, *Mach. Learn.*, 110(8):2151–2186, 2021.
[20] I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman, Tight relaxation of quadratic matching, *Comput. Graph. Forum*, 34(5):115–128, 2015.
[21] P. Koev, Matrices with displacement structure – a survey, 1999. `https://math.mit.edu/~plamen/files/mds.pdf`
[22] T. C. Koopmans and M. Beckmann, Assignment problems and the location of economic activities, *Econometrica*, 25(1):53–76, 1957.
[23] T. Le, N. Ho, and M. Yamada, Flow-based alignment approaches for probability measures in different spaces, in: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, PMLR, 130:3934–3942, 2021.
[24] M. Leordeanu and M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, IEEE, 2:1482–1489, 2005.
[25] M. Li, J. Yu, H. Xu, and C. Meng, Efficient approximation of Gromov-Wasserstein distance using importance sparsification, *J. Comput. Graph. Statist.*, 32(4):1512–1523, 2023.
[26] X. Li, Z. Qiu, X. Zhao, Z. Wang, Y. Zhang, C. Xing, and X. Wu, Gromov-Wasserstein guided representation learning for cross-domain recommendation, in: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, ACM, 1199–1208, 2022.
[27] Q. Liao, J. Chen, Z. Wang, B. Bai, S. Jin, and H. Wu, Fast Sinkhorn I: An O(N) algorithm for the Wasserstein-1 metric, *Commun. Math. Sci.*, 20(7):2053–2067, 2022.
[28] Q. Liao, Z. Wang, J. Chen, B. Bai, S. Jin, and H. Wu, Fast Sinkhorn II: Collinear triangular matrix and linear time accurate computation of optimal transport, *J. Sci. Comput.*, 98(1):1, 2024.
[29] S. Majumdar and A. K. Laha, Clustering and classification of time series using topological data analysis with applications to finance, *Expert Syst. Appl.*, 162:113868, 2020.
[30] F. Memoli, Spectral Gromov-Wasserstein distances for shape matching, in: *2009 IEEE 12th International Conference on Computer Vision Workshops*, IEEE, 256–263, 2009.
[31] F. Mémoli, Gromov-Wasserstein distances and the metric approach to object matching, *Found. Comput. Math.*, 11(4):417–487, 2011.
[32] M. Mudelsee, Trend analysis of climate time series: A review of methods, *Earth-Sci. Rev.*, 190:310–322, 2019.
[33] K.P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.
[34] D. Pastén, Z. Czechowski, and B. Toledo, Time series analysis in earthquake complex networks, *Chaos*, 28(8):083128, 2018.
[35] H. P. Maretic, M. El Gheche, G. Chierchia, and P. Frossard, GOT: An optimal transport framework for graph comparison, *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., Vol. 32, 2019.
[36] G. Peyre and M. Cuturi, Computational optimal transport, *Found. Trends Mach. Learn.*, 11(5-6):355–607, 2019.
[37] G. Peyré, M. Cuturi, and J. Solomon, Gromov-Wasserstein averaging of kernel and distance matrices, in: *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, 48:2664–2672, 2016.
[38] F. Ricci, Recommender systems in tourism, in: *Handbook of e-Tourism*, Springer, 457–474, 2022.
[39] T. Rolfe, Binomial coefficient recursion: The good, and the bad and ugly, *ACM SIGCSE Bulletin*, 33(2):35–36, 2001.
[40] F. Santambrogio, *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*, in: *Progress in Nonlinear Differential Equations and Their Applications*, Springer, Vol. 87, 2015.
[41] R. Sato, M.Cuturi, M. Yamada, and H. Kashima, Fast and robust comparison of probability measures in heterogeneous spaces, *arXiv:2002.01615*, 2020.
[42] M. Scetbon, G. Peyré, and M. Cuturi, Linear-time Gromov Wasserstein distances using low rank cou-

plings and costs, in: *Proceedings of the 39th international conference on machine learning*, PMLR, 162:19347–19365, 2022.
[43] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks, *Med. Image Anal.*, 54:30–44, 2019.
[44] T. Sejourne, F. X. Vialard, and G. Peyré, The unbalanced Gromov Wasserstein distance: Conic formulation and relaxation, *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 34:8766–8779, 2021.
[45] J. Serra, and J. L. Arcos, An empirical evaluation of similarity measures for time series classification, *Knowl.-Based Syst.*, 67:305–314, 2014.
[46] J. Solomon, L. Guibas, and A. Butscher, Dirichlet energy for analysis and synthesis of soft maps, *Comput. Graph. Forum*, 32(5):197–206, 2013.
[47] J. Solomon, A. Nguyen, A. Butscher, M. Ben-Chen, and L. Guibas, Soft maps between surfaces, *Comput. Graph. Forum*, 31(5):1617–1626, 2012.
[48] J. Solomon, G. Peyré, V. G. Kim, and S. Sra, Entropic metric alignment for correspondence problems, *ACM Trans. Graph.*, 35(4):1–13, 2016.
[49] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Education India, 2018.
[50] V. Titouan, N. Courty, R. Tavenard, C. Laetitia, and Rémi Flamary, Optimal transport for structured data with application on graphs, in: *Proceedings of the 36th International Conference on Machine Learning*, PMLR, 6275–6284, 2019.
[51] V. Titouan, I. Redko, R. Flamary, and N. Courty, Co-optimal transport, *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 33:17559–17570, 2020.
[52] T. Vayer, L. Chapel, R. Flamary, R. Tavenard, and N. Courty, Fused Gromov-Wasserstein distance for structured objects, *Algorithms*, 13(9):212, 2020.
[53] N. K. Vishnoi, *Algorithms for Convex Optimization*, Cambridge University Press, 2021.
[54] D. Xu, and Y. Tian, A comprehensive survey of clustering algorithms, *Ann. Data Sci.*, 2:165–193, 2015.
[55] H. Xu, Gromov-Wasserstein factorization models for graph clustering, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(4):6478–6485, 2020.
[56] H. Xu, D. Luo, and L. Carin, Scalable Gromov-Wasserstein learning for graph partitioning and matching, in: *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 3052–3062, 2019.
[57] H. Xu, D. Luo, R. Henao, S. Shah, and L. Carin, Learning autoencoders with relational regularization, in: *Proceedings of the 37th International Conference on Machine Learning*, PMLR, 119:10576–10586, 2020.
[58] H. Xu, D. Luo, H. Zha, and L. C. Duke, Gromov-Wasserstein learning for graph matching and node embedding, in: *International Conference on Machine Learning*, PMLR, 97:6932–6941, 2019.
[59] Running horse, `https://www.bilibili.com/video/BV1u54y1v7JR.4`, 4.4.2, 2020.