Gromov–Wasserstein Problem with Cyclic Symmetry

Shoichiro Takeda Yasunori Akagi
NTT Corporation
{shoichiro.takeda, yasunori.akagi}@ntt.com

Abstract

We propose novel fast algorithms for the Gromov–Wasserstein problem (GW) with cyclic symmetry of input data. This problem naturally appears as an object-matching task, which underlies various real-world computer vision applications, e.g.*, image registration, point cloud registration, stereo matching, and 3D reconstruction. Gradient-based algorithms have been widely used to solve GW, and our main idea is to utilize the following remarkable property that emerges in GW with cyclic symmetry: By setting the initial solution to have cyclic symmetry, all intermediate solutions and matrices that appear in the gradient-based algorithms have the same cyclic symmetry until convergence. Based on this property, our gradient-based algorithms restrict the solution space to have cyclic symmetry and update only one symmetric part of solutions and matrices at each iteration, resulting in faster computation. Moreover, our algorithms solve the optimal transport problem at each iteration, which also exhibits cyclic symmetry. This problem can be solved efficiently, and as a result, our algorithms perform significantly faster. Experiments showed the effectiveness of our algorithms in synthetic and real-world data with strict and approximate cyclic symmetry.*

1. Introduction

Given a discrete probability measure and a metric matrix on each of two discrete metric-measure spaces, the discrete Gromov–Wasserstein problem (GW) finds a coupling that is as close to an isometry between the two measures as possible. The coupling distortion from isometry is quantifiable and is called Gromov–Wasserstein distance (GWD). The GWD is an effective tool that compares or matches two metric-measure spaces, and thus GW has been studied in various research areas, *e.g.*, object matching [25, 28, 42], generative modeling [10], and domain adaptation [12].

Existing approaches to solve GW include gradient-based algorithms, and the conditional gradient method (CG) was proposed recently [41]. This algorithm iterates the following three processes until convergence: (i) it computes the gradient of the objective function in GW at the current solution, (ii) solves the optimal transport problem (OT) [20] with the computed gradient as the cost to obtain the update direction, and (iii) updates the current solution in that direction. In addition, the entropy-regularized GW (EGW) has attracted much attention [28, 31, 49] because it offers a concise and fast projected gradient method (PG). This algorithm iterates the same processes as CG until convergence but solves the entropy-regularized OT (EOT) [13] instead of OT. As another line of work to solve GW faster, algorithms using special structures of input data have been proposed, *e.g.*, algorithms for GW with low-rank [29] or hierarchical [48] structures of input data. Besides, several algorithms provide closed-form solutions to GW with some special structures of input data, such as 1-dimensional input data [42] and tree metric structure [23].

In this paper, we propose novel fast algorithms for GW with a new and ubiquitous special structure, *cyclic symmetry*, of input data. Specifically, we assume K-order cyclic symmetry of input data; the input m- and n-dimensional probability measures are given as concatenations of K copies of $m'(:= m/K)$- and $n'(:= n/K)$-dimensional vectors, respectively, and the input $m \times m$ and $n \times n$ metric matrices are block circulant matrices consisting of K matrices with size $m' \times m'$ and $n' \times n'$, respectively (see Assumption 1). Such GW with cyclic symmetry naturally appears as an object-matching task, which underlies various real-world computer vision applications (see Sec. 4). To solve this new GW efficiently, we developed fast versions of CG and PG using cyclic symmetry. Our main idea is to utilize the following remarkable property that emerges in this new problem: By setting the initial solution as a $m \times n$ block circulant matrix consisting of K matrices with size $m' \times n'$, all intermediate solutions and matrices that appear in CG and PG also become $m \times n$ block circulant matrices consisting of K matrices with size $m' \times n'$ until convergence (see Theorem 1 and the later discussion). This property ensures that we can restrict the solution space to such a block circulant matrix in CG and PG and update only the K matrices with size $m' \times n'$ instead of the whole $m \times n$ matrix, which results in fast versions of CG and PG. We call this fast

CG *Cyclic CG (C-CG)* and this fast PG *Cyclic PG (C-PG)*. Moreover, OT and EOT that must be solved in C-CG and C-PG also exhibit cyclic symmetry. These problems can be solved efficiently [36], and as a result, C-CG and C-PG perform significantly faster than the original CG and PG. In this paper, we theoretically show the time complexities of C-CG and C-PG in Secs. 5.2 and 5.3, respectively.

In summary, this paper is the first to introduce the concept of symmetry to solve GW faster and proposes fast cyclic symmetry-aware algorithms C-CG and C-PG. We validated the effectiveness of our algorithms in experiments on synthetic and real-world data with strict and approximate cyclic symmetry.

2. Related Works

Mémoli [25] initially formulated the Gromov–Wasserstein problem (GW) for object matching even defined on different metric-measure spaces. The Gromov–Wasserstein distance relaxes the Gromov–Hausdorff distance and quantifies the coupling distortion between two measures from being isometry. GW is a relaxation of the quadratic assignment problem that is NP-hard. Thus, several relaxations for this problem could be also used for GW, such as SDP relaxation [19, 21] and eigenvalue relaxation [9, 24]. However, the former runs slowly due to the increased number of variables, while the latter has looser approximation bounds. Recently, the conditional gradient method (CG), which is also known as the Frank-Wolfe algorithm, was proposed to solve GW directly [41]. This algorithm at each iteration boils down to compute the gradient of the objective function in GW and solve the optimal transport problem (OT) [20] by the network simplex algorithm [1]. Moreover, the entropy-regularized GW (EGW) has attracted much attention [28, 31, 49] because it offers a concise and fast projected gradient method (PG). This algorithm is similar to CG but solves the entropy-regularized OT instead of OT by the Sinkhorn-Knopp algorithm [30]. This paper focuses on the algorithms CG for GW and PG for EGW because they are widely used in the Gromov–Wasserstein research field.

To solve GW faster, various algorithms that use special structures of input data have been proposed [23, 29, 42, 48]. Scalable algorithm [48] uses the hierarchical structure of input data and recursively solves GW for each layer. Low-rank algorithm [29] uses the low-rank structure of the input metrics and couplings. Besides, several algorithms using special structures of input data provide closed-form solutions to GW. Sliced algorithm [42] assumes input probability measures on the 1-dimensional real line and the squared Euclidean distance metrics; such GW becomes the Gromov–Monge problem that has a closed-form solution. Since the 1-dimensional line is a special case of the tree structure, the algorithm using tree metric structures of input data was also proposed [23]. Unlike the above algorithms, this paper is the first to introduce the use of a new and ubiquitous special structure, *cyclic symmetry*.

The optimal transport problem (OT) was formulated as the linear programming problem by [20], and the entropy-regularized OT (EOT) has attracted attention recently because this problem can be solved faster than OT [13]. These problems seek an optimal coupling to minimize the total cost of transporting a probability measure toward another one on the same metric-measure space. Similar to the research progress in GW, many algorithms have been proposed to solve OT faster using special structures of input data, *e.g.*, low-rankness [2, 38], translation invariant [17, 27], hierarchy [16], and submodularity [3] of input data. More recently, using cyclic symmetry of input data for OT was proposed [36]. This structure and the convexity of OT can reduce OT to a small problem with significantly fewer variables, resulting in faster computation. Similar to this research, we use cyclic symmetry of input data to solve GW faster, but this is more challenging and non-trivial because GW is non-convex while OT is convex.

3. Preliminary

Notations. We denote the set of non-negative real numbers by $\mathbb{R}_{\geq 0}$. For matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d\times d}$, we denote the Frobenius inner product by $\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i,j=0}^{d-1} X_{ij} Y_{ij}$. We define the probability simplex as $\Delta^d \coloneqq \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} \mid \sum_{i=0}^{d-1} x_i = 1\}$. The d-dimensional all-ones vector is written as $\mathbf{1}_d$. We use $\mathbf{X}^{\circ 2}$ to represent element-wise squaring of $\mathbf{X}$, meaning $(\mathbf{X}^{\circ 2})_{ij} = X_{ij}^2$. The matrix norm is given by $\|\mathbf{X}\|_\infty \coloneqq \max_{i,j} |X_{ij}|$. We define the entropy of $\mathbf{X}$ as $H(\mathbf{X}) \coloneqq -\sum_{i,j=0}^{d-1} X_{ij}(\log X_{ij} - 1)$. We denote the elementwise division by $\oslash$. We denote the Kronecker product of matrices by $\otimes$. When a matrix $\mathbf{X}$ can be written as

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_0 & \mathbf{X}_1 & \cdots & \mathbf{X}_{K-1} \\ \mathbf{X}_{K-1} & \mathbf{X}_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{X}_1 \\ \mathbf{X}_1 & \cdots & \mathbf{X}_{K-1} & \mathbf{X}_0 \end{pmatrix},$$

using $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{K-1} \in \mathbb{R}^{d_1 \times d_2}$, the matrix $\mathbf{X}$ is referred to as a (d_1, d_2, K)-*block circulant matrix generated by* $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{K-1}$.

3.1. Entropic Gromov–Wasserstein Problem

We first define the original 2-Gromov–Wasserstein problem (GW) introduced by [25]. Let $\mathbb{X} = (\mathcal{X}, d_\mathcal{X}, \mathbf{a})$ be a metric-measure space where $\mathcal{X} = \{x_0, \ldots, x_{m-1}\}$ is a finite set with an associated metric $d_\mathcal{X}$ and a discrete probability measure $\mathbf{a} \in \Delta^m$. Similarly, let $\mathbb{Y} = (\mathcal{Y}, d_\mathcal{Y}, \mathbf{b})$ be another discrete metric-measure space with a discrete probability measure $\mathbf{b} \in \Delta^n$. Given similarity matrices $\mathbf{C} \in \mathbb{R}^{m\times m}_{\geq 0}$ and $\mathbf{D} \in \mathbb{R}^{n\times n}_{\geq 0}$ where $C_{ii'} \coloneqq d_\mathcal{X}(x_i, x_{i'})$ and

$D_{jj'} \coloneqq d_{\mathcal{Y}}(y_j, y_{j'})$, we define GW between $\mathbb{X}$ and $\mathbb{Y}$ as the following constrained non-convex optimization problem:

$$\min_{\mathbf{T}\in\Pi(\mathbf{a},\mathbf{b})} \sum_{i,i',j,j'} (C_{ii'} - D_{jj'})^2 T_{ij} T_{i'j'}, \tag{1}$$

where $\mathbf{T}$ is a coupling matrix between $\mathbb{X}$ and $\mathbb{Y}$, and $\Pi(\mathbf{a},\mathbf{b}) \coloneqq \{\mathbf{X} \in \mathbb{R}_{\geq 0}^{m\times n} \mid \mathbf{X}\mathbf{1}_n = \mathbf{a}, \mathbf{X}^\top \mathbf{1}_m = \mathbf{b}\}$. The optimal objective function value of (1) is called Gromov–Wasserstein distance (GWD). Through a simple calculation, the objective function in (1) can be rewritten in matrix form as $\langle \mathbf{Q} - 2\mathbf{C}\mathbf{T}\mathbf{D}^\top, \mathbf{T}\rangle$, where $\mathbf{Q} \coloneqq \mathbf{C}^{\circ 2}\mathbf{a}\mathbf{1}_n^\top + \mathbf{1}_m\mathbf{b}^\top(\mathbf{D}^{\circ 2})^\top$. Hereafter, we will use this matrix form.

In this paper, we consider the entropy-regularized GW (EGW), which is an extension of GW with the entropy regularizer introduced by [28]:

$$\min_{\mathbf{T}\in\Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{Q} - 2\mathbf{C}\mathbf{T}\mathbf{D}^\top, \mathbf{T}\rangle - \varepsilon H(\mathbf{T}), \tag{2}$$

where $\varepsilon \geq 0$. Apparently, EGW is identical to the original GW when $\varepsilon = 0$.

3.2. Gradient-based Algorithms for EGW

Several algorithms have been proposed to solve EGW. In this paper, we focus on two representative algorithms that use gradient information: the Conditional Gradient method (CG) [41] and the Projected Gradient method (PG) [28].

CG is an iterative algorithm for solving GW, *i.e.*, (2) with $\varepsilon = 0$. At each iteration, this algorithm approximates the objective function linearly around the current solution and then updates the solution in the direction obtained by minimizing the linear approximation within the feasible region.

PG is an iterative algorithm for solving EGW, *i.e.*, (2) with $\varepsilon > 0$. At each iteration, this algorithm updates the current solution in the direction of the gradient of the objective function in EGW using exponentiated gradient descent [22] and then projects the updated solution back onto the feasible region with the Kullback–Leibler metric.

Although these algorithms differ in the problems they address (GW and EGW) and their algorithm design principles (*e.g.*, how the feasible region is handled), the procedures can be described in a unified manner as below. Let $\mathbf{T}^{(\tau)}$ be the current solution and $\mathbf{G}^{(\tau)}$ denotes the gradient of the first term of (2) at $\mathbf{T}^{(\tau)}$. First, $\mathbf{G}^{(\tau)}$ is calculated as

$$\mathbf{G}^{(\tau)} = \mathbf{Q} - 2\left(\mathbf{C}^\top \mathbf{T}^{(\tau)} \mathbf{D} + \mathbf{C}\mathbf{T}^{(\tau)}\mathbf{D}^\top\right). \tag{3}$$

Then, we solve the following problem:

$$\mathbf{S}^{(\tau)} = \operatorname*{argmin}_{\mathbf{S}\in\Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{G}^{(\tau)}, \mathbf{S}\rangle - \varepsilon H(\mathbf{S}). \tag{4}$$

Finally, $\mathbf{T}^{(\tau)}$ is updated to the next solution $\mathbf{T}^{(\tau+1)}$:

$$\mathbf{T}^{(\tau+1)} = (1-\lambda)\mathbf{T}^{(\tau)} + \lambda \mathbf{S}^{(\tau)}, \tag{5}$$

where $0 \leq \lambda \leq 1$ is the step size.

In CG, the problem (4) is identical to the optimal transport problem (OT) introduced by [20] because $\varepsilon = 0$. This problem can be solved efficiently using the network simplex algorithm [1]. The step size λ can be determined by line search based on Armijo or Wolfe conditions [6, 46].

In PG, the problem (4) is identical to the entropy-regularized OT introduced by [13] because $\varepsilon > 0$, and it can be solved efficiently via the Sinkhorn-Knopp algorithm for its dual problem [30]. The step size λ is fixed to 1.

4. C-EGW: EGW with Cyclic Symmetry

This section explains our assumption of cyclic symmetry for EGW (2). We assume that $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ in (2) have the following K-order cyclic symmetry.

Assumption 1. A common divisor K exists for m and n. The probability vectors $\mathbf{a}$ and $\mathbf{b}$ in (2) can be written as

$$\mathbf{a} = \mathbf{1}_K \otimes \boldsymbol{\alpha}, \qquad \mathbf{b} = \mathbf{1}_K \otimes \boldsymbol{\beta},$$

where $\boldsymbol{\alpha} \in \mathbb{R}_{\geq 0}^{m'}, \boldsymbol{\beta} \in \mathbb{R}_{\geq 0}^{n'}$ and $m' \coloneqq \frac{m}{K}, n' \coloneqq \frac{n}{K}$ are integers. Furthermore, in (2), the similarity matrix $\mathbf{C}$ is an (m', m', K)-block circulant matrix generated by $\mathbf{C}_0, \ldots, \mathbf{C}_{K-1} \in \mathbb{R}_{\geq 0}^{m'\times m'}$ and $\mathbf{D}$ is an (n', n', K)-block circulant matrix generated by $\mathbf{D}_0, \ldots, \mathbf{D}_{K-1} \in \mathbb{R}_{\geq 0}^{n'\times n'}$.

In this paper, we call EGW (2) with Assumption 1 *Cyclic EGW (C-EGW)*. This problem naturally appears as one approach to the object-matching task, which underlies various computer vision applications, as below.

Example 1 (Image Matching with Cyclic Symmetry)**.** Image matching task, which seeks a coupling of similar features between images, is essential for various visual applications such as face recognition [51], image registration [33, 40], stereo matching [4, 35], and image retrieval [18, 43]. Since images often exhibit inherent cyclic symmetry, the image matching task often uses cyclic symmetry to improve accuracy [4, 26, 32, 35, 43, 51]. We here consider an image matching task between two images with 90° rotational symmetry, *i.e.*, 4-order cyclic symmetry, as shown in Fig. 1. We define $\mathbf{a}$ and $\mathbf{b}$ as concatenations of image features (or simply intensity values) at each symmetric region in each image of Fig. 1, and $\mathbf{C}$ and $\mathbf{D}$ as the Euclidean distance matrix between the pixel positions. These $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ satisfy Assumption 1 with $K = 4$, and thus this image matching task will become C-EGW with $K = 4$.

Figure 1. Images with 4-order cyclic symmetry.

Example 2 (Point Cloud Matching with Cyclic Symmetry)**.** Point cloud matching task, which seeks a coupling of similar geometric features between point clouds, is essential for various 3D applications such as point cloud registration [5, 50] and 3D reconstruction [39, 47]. Since 3D objects, especially artificial ones, often exhibit inherent cyclic symmetry, the point clouds obtained from such 3D objects also exhibit cyclic symmetry. Thus, the point cloud matching task often uses this structure to improve accuracy [34, 39, 45]. We here consider a point cloud matching task between two point clouds with mirror symmetry, *i.e.*, 2-order cyclic symmetry, as shown in Fig. 2. We define $\mathbf{a}$ and $\mathbf{b}$ as uniform weights (or intensity values of points), each of which has a total of 1, and $\mathbf{C}$ and $\mathbf{D}$ as the Euclidean distance matrices for pairwise 3D point positions of each object in Fig. 2. These $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ satisfy Assumption 1 with $K = 2$, and thus this point cloud matching task will become C-EGW with $K = 2$.

Figure 2. Point clouds with 2-order cyclic symmetry

We can show that $\mathbf{Q}$ has a block circulant structure in C-EGW, which will be used in the next section.

Lemma 1. Under Assumption 1, the matrix $\mathbf{Q}$ in (2) is an (m', n', K)-block circulant matrix generated by $\widehat{\mathbf{Q}}, \dots, \widehat{\mathbf{Q}}$, where $\widehat{\mathbf{Q}} \coloneqq \sum_{i=0}^{K-1} \mathbf{C}_i^{\circ 2} \boldsymbol{\alpha} \mathbf{1}_{n'}^\top + \mathbf{1}_{m'} \boldsymbol{\beta}^\top (\mathbf{D}_i^{\circ 2})^\top$.

Note that all proofs are in the Appendix.

5. Fast Algorithms for C-EGW

In this section, we propose fast algorithms to solve C-EGW.

5.1. Block Circulant Intermediate Matrices

We first demonstrate that in CG and PG, introduced in Sec. 3.2, the cyclic symmetry structure of matrices generated during the algorithms, including all intermediate solutions, can be preserved under Assumption 1.

Theorem 1. Assume that $\mathbf{T}^{(\tau)}$ is an (m', n', K)-block circulant matrix. Then, under Assumption 1, $\mathbf{G}^{(\tau)}$ given by (3) is an (m', n', K)-block circulant matrix. Moreover, there exists an (m', n', K)-block circulant matrix $\mathbf{S}^{(\tau)}$ which is an optimal solution to the problem (4). Furthermore, when $\mathbf{S}^{(\tau)}$ is an (m', n', K)-block circulant matrix, $\mathbf{T}^{(\tau+1)}$ given by (5) remains an (m', n', K)-block circulant matrix.

By repeatedly applying Theorem 1, we can see that by setting the initial solution $\mathbf{T}^{(0)}$ in C-EGW as an (m', n', K)-block circulant matrix, it is possible to ensure that $\mathbf{T}^{(\tau)}$, $\mathbf{G}^{(\tau)}$, and $\mathbf{S}^{(\tau)}$ all become (m', n', K)-block circulant matrices in CG and PG until convergence (see Fig. 3).

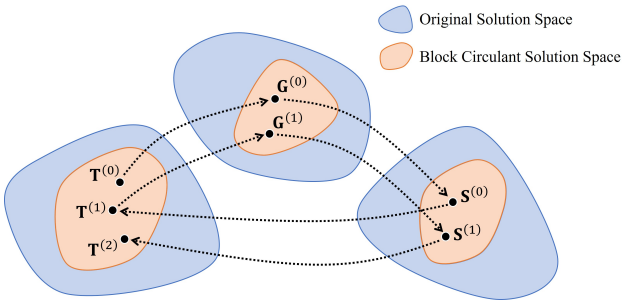

Figure 3. An illustrative flow diagram of repeatedly applying Theorem 1 for $\tau = 0, 1, 2$.

This remarkable property ensures that we can restrict the solution space to (m', n', K)-block circulant matrices in CG and PG. Therefore, we hereafter restrict $\mathbf{T}^{(\tau)}, \mathbf{G}^{(\tau)}$, and $\mathbf{S}^{(\tau)}$ to (m', n', K)-block circulant matrices generated by $\mathbf{T}_0^{(\tau)}, \dots, \mathbf{T}_{K-1}^{(\tau)} \in \mathbb{R}_{\geq 0}^{m' \times n'}$, $\mathbf{G}_0^{(\tau)}, \dots, \mathbf{G}_{K-1}^{(\tau)} \in \mathbb{R}_{\geq 0}^{m' \times n'}$, and $\mathbf{S}_0^{(\tau)}, \dots, \mathbf{S}_{K-1}^{(\tau)} \in \mathbb{R}_{\geq 0}^{m' \times n'}$, respectively. Using this restriction and Lemma 1, we can rewrite (3) as

$$\mathbf{G}_k^{(\tau)} = \widehat{\mathbf{Q}} - 2 \sum_{i,j=0}^{K-1} \left(\mathbf{C}_i^\top \mathbf{T}_{j-i}^{(\tau)} \mathbf{D}_{j-k} + \mathbf{C}_i \mathbf{T}_{j-i}^{(\tau)} \mathbf{D}_{j-k}^\top \right), \quad (6)$$

where $\mathbf{T}_{j-i}^{(\tau)}$ and $\mathbf{D}_{j-k}$ are simplifications of $\mathbf{T}_{j-i \mod K}^{(\tau)}$ and $\mathbf{D}_{j-k \mod K}$, respectively. We can rewrite (4) as

$$\begin{aligned} &\mathbf{S}_0^{(\tau)}, \dots, \mathbf{S}_{K-1}^{(\tau)} \\ &= \operatorname*{argmin}_{\mathbf{S}_0, \dots, \mathbf{S}_{K-1} \in \Gamma(\boldsymbol{\alpha}, \boldsymbol{\beta})} \sum_{k=0}^{K-1} \langle \mathbf{G}_k^{(\tau)}, \mathbf{S}_k \rangle - \varepsilon H(\mathbf{S}_k), \end{aligned} \quad (7)$$

where $\Gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) \coloneqq \{\mathbf{X}_0, \dots, \mathbf{X}_{K-1} \in \mathbb{R}_{\geq 0}^{m' \times n'} \mid \sum_{k=0}^{K-1} \mathbf{X}_k \mathbf{1}_{n'} = \boldsymbol{\alpha}, \sum_{k=0}^{K-1} \mathbf{X}_k^\top \mathbf{1}_{m'} = \boldsymbol{\beta}\}$. We can rewrite (5) as

$$\mathbf{T}_k^{(\tau+1)} = (1 - \lambda) \mathbf{T}_k^{(\tau)} + \lambda \mathbf{S}_k^{(\tau)}. \quad (8)$$

5.2. Fast Conditional Gradient Method for C-GW

We here propose a fast CG for C-GW, *i.e.*, (2) with $\varepsilon = 0$ and Assumption 1. The processes (6)–(8) can be computed faster than the original ones (3)–(5) because they update only K matrices with the size $m' \times n'$. Moreover, because (7) with $\varepsilon = 0$ is equivalent to OT with cyclic symmetry in [36], we can show the following proposition, which extends Theorem 1 in [36] to the case where $m \neq n$. The proof is almost the same as the $m = n$ case.

Proposition 1. We consider the following OT

$$\mathbf{Z}^{(\tau)} = \operatorname*{argmin}_{\mathbf{Z} \in \Pi'(\boldsymbol{\alpha}, \boldsymbol{\beta})} \left\langle \mathbf{W}^{(\tau)}, \mathbf{Z} \right\rangle, \quad (9)$$

where $\Pi'(\boldsymbol{\alpha}, \boldsymbol{\beta}) \coloneqq \{\mathbf{X} \in \mathbb{R}_{\geq 0}^{m' \times n'} \mid \mathbf{X} \mathbf{1}_{n'} = \boldsymbol{\alpha}, \mathbf{X}^\top \mathbf{1}_{m'} = \boldsymbol{\beta}\}$ and

$$W_{ij}^{(\tau)} \coloneqq \min_{0 \leq k \leq K-1} G_{ijk}^{(\tau)}. \quad (10)$$

Algorithm 1 C-CG for C-GW

Input: $\mathbf{a} \in \Delta^m, \mathbf{b} \in \Delta^n, \mathbf{C} \in \mathbb{R}_{\geq 0}^{m \times m}, \mathbf{D} \in \mathbb{R}_{\geq 0}^{n \times n}$ which satisfy Assumption 1 and $\varepsilon = 0$

Initialize $\mathbf{T}^{(0)}$ as an (m', n', K)-block circulant matrix generated by $\mathbf{T}_0^{(0)}, \ldots, \mathbf{T}_{K-1}^{(0)} \in \mathbb{R}_{\geq 0}^{m' \times n'}$.
for $\tau = 0, 1, \ldots$ **do**
  Calculate $\mathbf{G}_0^{(\tau)}, \ldots, \mathbf{G}_{K-1}^{(\tau)}$ by (6)
  Calculate $\mathbf{W}^{(\tau)}$ whose entry is given by (10)
  Solve (9) to obtain $\mathbf{Z}^{(\tau)}$
  Calculate $S_{ijk}^{(\tau)}$ for i, j, k by the relationship (11)
  Calculate $\mathbf{T}_0^{(\tau+1)}, \ldots, \mathbf{T}_{K-1}^{(\tau+1)}$ by (8)
end for

Then, we will obtain the optimal solution to (7) with $\varepsilon = 0$, $\mathbf{S}_0^*, \ldots, \mathbf{S}_{K-1}^*$, as follows.

$$S_{ijk}^* = \begin{cases} Z_{ij}^{(\tau)} & \text{if } k = \min\left(\underset{0 \leq k \leq K-1}{\operatorname{argmin}} \, G_{ijk}^{(\tau)}\right), \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Proposition 1 indicates that we will obtain the optimal solution to (7) with $\varepsilon = 0$ by solving the small OT (9) instead, which has significantly few only $m'n'$ variables. Therefore, the original processes (3), (4), and (5) in CG can be reduced to the small ones (6), (9), and (8), respectively. We call the proposed algorithm *Cyclic Conditional Gradient method* (C-CG), summarized in Algorithm 1.

We evaluate the time complexity of C-CG (Algorithm 1). It mainly depends on the matrix-vector product in (6) and solving the small OT (9). The former requires $O(mn(m' + n'))$, which is improved from $O(mn(m + n))$ in CG that uses (3). From the report of [37], the latter requires $O((m' + n')m'n' \log(m' + n') \log((m' + n')\|\mathbf{W}^{(\tau)}\|_\infty))$ when using the network simplex algorithm to solve (9), which is improved from $O((m+n)mn \log(m+n) \log((m+n)\|\mathbf{G}^{(\tau)}\|_\infty))$ in CG that solves (4) in terms of the dimensional reduction and $\|\mathbf{W}^{(\tau)}\|_\infty \leq \|\mathbf{G}^{(\tau)}\|_\infty$ by (10). Therefore, for C-GW, C-CG will achieve a better time complexity than CG. Note that, in C-CG, the time complexity of line search to determine λ in (8) will also be improved because the objective function value of (2) can be assessed by using only the block matrices $\{\mathbf{C}_k^{(\tau)}, \mathbf{D}_k^{(\tau)}, \mathbf{T}_k^{(\tau)}\}_{k=0}^{K-1}$.

5.3. Fast Projected Gradient Method for C-EGW

We here propose a fast PG for C-EGW, *i.e.*, (2) with $\varepsilon > 0$ and Assumption 1. Similar to the discussion in Sec. 5.2, because (7) with $\varepsilon > 0$ is equivalent to EOT with cyclic symmetry in [36], we can show the following proposition, which extends Theorem 2 in [36] to the case where $m \neq n$. The proof is almost the same as the $m = n$ case.

Algorithm 2 C-PG for C-EGW

Input: $\mathbf{a} \in \Delta^m, \mathbf{b} \in \Delta^n, \mathbf{C} \in \mathbb{R}_{\geq 0}^{m \times m}, \mathbf{D} \in \mathbb{R}_{\geq 0}^{n \times n}$ which satisfy Assumption 1 and $\varepsilon > 0$

Initialize $\mathbf{T}^{(0)}$ as an (m', n', K)-block circulant matrix generated by $\mathbf{T}_0^{(0)}, \ldots, \mathbf{T}_{K-1}^{(0)} \in \mathbb{R}_{\geq 0}^{m' \times n'}$.
for $\tau = 0, 1, \ldots$ **do**
  Calculate $\mathbf{G}_0^{(\tau)}, \ldots, \mathbf{G}_{K-1}^{(\tau)}$ by (6)
  Calculate $\mathbf{L}^{(\tau)}$ whose entry is given by (13)
  Solve (12) via Remark 1 to obtain $\mathbf{p}$ and $\mathbf{q}$.
  Calculate $S_{ijk}^{(\tau)}$ for i, j, k by the relationship (14)
  Calculate $\mathbf{T}_0^{(\tau+1)}, \ldots, \mathbf{T}_{K-1}^{(\tau+1)}$ by (8)
end for

Proposition 2. The Fenchel dual of (7) with $\varepsilon > 0$ is

$$\underset{\mathbf{u} \in \mathbb{R}^{m'}, \mathbf{v} \in \mathbb{R}^{n'}}{\operatorname{argmax}} \langle \mathbf{u}, \boldsymbol{\alpha} \rangle + \langle \mathbf{v}, \boldsymbol{\beta} \rangle - \varepsilon \sum_{i=0}^{m'-1} \sum_{j=0}^{n'-1} p_i L_{ij}^{(\tau)} q_j, \tag{12}$$

where $p_i = \exp(u_i/\varepsilon), q_j = \exp(v_j/\varepsilon)$, and

$$L_{ij}^{(\tau)} := \sum_{k=0}^{K-1} \exp\left(-\frac{G_{ijk}^{(\tau)}}{\varepsilon}\right). \tag{13}$$

Let u_i^* and v_j^* be the optimal solutions to (12), we can describe $p_i^* = \exp(u_i^*/\varepsilon)$ and $q_j^* = \exp(v_j^*/\varepsilon)$, respectively. Then, we will obtain the optimal solutions to (7), $\mathbf{S}_0^*, \ldots, \mathbf{S}_{K-1}^*$ as follows.

$$S_{ijk}^* = p_i^* q_j^* \exp\left(-\frac{G_{ijk}^{(\tau)}}{\varepsilon}\right). \tag{14}$$

Proposition 2 indicates that we will obtain the optimal solution to (7) with $\varepsilon > 0$ by solving the small dual problem (12) instead, which has significantly few only $m' + n'$ variables. Therefore, the original processes (3), (4), and (5) in PG can be reduced to the small ones (6), (12), and (8), respectively. We call the proposed algorithm *Cyclic Projected Gradient method* (C-PG), summarized in Algorithm 2.

Remark 1. The problem (12) can be solved efficiently using the Sinkhorn-Knopp algorithm [13, 30]; instead of optimizing $\mathbf{u}$ and $\mathbf{v}$ directly, we iteratively optimize $\mathbf{p}$ and $\mathbf{q}$ as $\mathbf{p} \leftarrow \boldsymbol{\alpha} \oslash (\mathbf{L}^{(\tau)}\mathbf{q})$ and $\mathbf{q} \leftarrow \boldsymbol{\beta} \oslash ((\mathbf{L}^{(\tau)})^\top \mathbf{p})$.

We evaluate the time complexity of C-PG (Algorithm 2). It mainly depends on the matrix-vector product in (6) and solving the small dual problem of EOT (12). The former is the same as in Algorithm 1. The latter using the Sinkhorn-Knopp algorithm [13, 30] requires $O(m'n')$ at each iteration, which is improved from $O(mn)$ in PG that solves (4) using the same algorithm. Therefore, for C-EGW, C-PG will achieve a better time complexity than PG.

6. Experiments

To validate the effectiveness of our algorithms, C-CG and C-PG, we conducted experiments on synthetic and real-world data with *strict* and *approximate* cyclic symmetry, respectively. Specifically, the synthetic data provide $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ that satisfy Assumption 1 *strictly* but the real-world data provide them that satisfy Assumption 1 *approximately* due to slight distortion and displacement. We cannot apply C-CG and C-PG to such real-world cases directly because they rely on Assumption 1 strictly. However, we can adapt them via a simple preprocess called *symmetric alignment* (details will appear in Sec. 6.3). We here compared four algorithms: CG using the network simplex algorithm [41], C-CG using the network simplex algorithm (Algorithm 1), PG [28], and C-PG (Algorithm 2). As an initial solution for CG or PG, we created a uniform random matrix in $[0, 1)^{m \times n}$ and scaled it to lie within $\Pi(\mathbf{a}, \mathbf{b})$ using the Sinkhorn-Knopp algorithm [30]. Also, as an initial solution for C-CG or C-PG, we created K uniform random matrices in $[0, 1)^{m' \times n'}$, scaled each of them to lie within $\Pi'(\frac{1}{K}\boldsymbol{\alpha}, \frac{1}{K}\boldsymbol{\beta})$ using the Sinkhorn-Knopp algorithm, and then constructed an (m', n', K)-block circulant matrix generated by the scaled K matrices. For each algorithm, we checked GWD, defined by the objective function value of (1), and the computation time, measured from data input to solution output. Each algorithm ran until the update of the GWD value was below 1.0×10^{-6}. All experiments were performed in Python on an Ubuntu 20.04 desktop with AMD Ryzen Threadripper 3970X 32-core CPU and 256 GB memory. In CG and C-CG, we determined λ using the Armijo line-search in SciPy library [44] and used the network simplex algorithm in LEMON [15].

6.1. Performance Analysis with Synthetic Data

We here evaluated whether our C-CG and C-PG show GWDs compatible with those of CG and PG but faster computations, respectively, in *different* synthetic data.

We created 100 random pairs of synthetic 2D point sets with 10-order rotational symmetry on a 2D plane (for details, see Appendix C). The number of 2D points in each point set was 2000. Thus, $m = n = 2000$. For the input data to each algorithm, we set $\mathbf{a} = \frac{1}{m}\mathbf{1}_m$ and $\mathbf{b} = \frac{1}{n}\mathbf{1}_n$, and $\mathbf{C}$ and $\mathbf{D}$ as the Euclidean distance matrices for pairwise 2D point positions of each point set. The input data $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ satisfy Assumption 1 with $K = 10$ strictly. Because this input data also satisfy Assumption 1 for all K that are divisors of 10 greater than 1, *i.e.*, $K \in \{2, 5, 10\}$, we conducted experiments for using each K in C-CG and C-PG; using a larger K makes C-CG and C-PG faster. In PG and C-PG, we set $\varepsilon = 0.004$ as low as possible while avoiding overflow, which is often caused by the Sinkhorn iterations [13].

Table 1 lists the results. C-CG showed GWDs compatible with those of CG but faster computation times when a larger K value was used. This trend was also observed between PG and C-PG. These results validated the effectiveness of C-CG and C-PG for C-EGW; they achieve faster computations than CG and PG when using the higher cyclic symmetry structure hidden in the input data (*i.e.*, larger K value) without degrading GWDs.

Algo.	K	GWD ($\times 10^{-3}$)	# of iterations	Time (s)
CG [41]	–	3.75 ± 2.13	10.08 ± 5.71	16.82 ± 10.49
C-CG	2	3.76 ± 2.14	9.60 ± 5.84	3.52 ± 2.17
	5	3.74 ± 2.14	8.87 ± 5.45	1.01 ± 0.62
	10	3.75 ± 2.13	8.97 ± 4.27	0.64 ± 0.3
PG [28]	–	6.88 ± 2.13	19.56 ± 20.0	6.73 ± 6.78
C-PG	2	6.88 ± 2.15	20.60 ± 18.63	3.56 ± 3.20
	5	6.89 ± 2.14	17.97 ± 16.21	1.47 ± 1.33
	10	6.89 ± 2.13	17.41 ± 17.13	0.94 ± 0.93

Table 1. Performance analysis with synthetic data in Sec. 6.1. "Algo." means "Algorithm." "#" means "the number." "s" is the symbol for seconds. Mean $\pm$ SD is shown in each result.

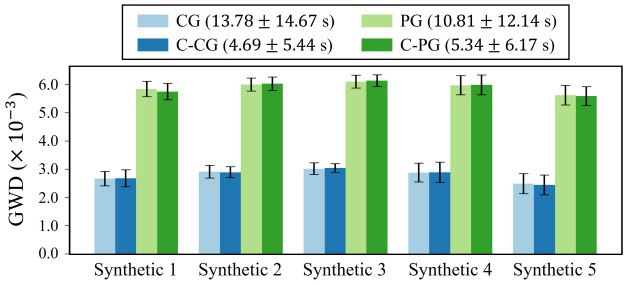

Figure 4. Sensitivity to different initial solutions in Sec. 6.2. Each of Synthetic 1 to 5 indicates the different synthetic pair data. The time listed to the right of each algorithm name represents Mean $\pm$ SD of the computation time to obtain all GWDs. Note that PG and C-PG showed larger GWDs than those of CG and C-CG because they solved C-EGW having the entropy regularizer.

6.2. Sensitivity to Different Initial Solutions

Since GW is non-convex, the result depends on an initial solution. In C-CG and C-PG, the initial solution is restricted to an (m', n', K)-block circulant matrix. This raises a concern that C-CG and C-PG may reach a worse solution than CG and PG, which allows for more flexible initial solutions. Therefore, we here evaluated the sensitivity of each algorithm to different initial solutions in the *same* synthetic data.

As in Sec. 6.1, we created 5 random pairs of synthetic 2D point sets with 2-order rotational symmetry and set the input data $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ to each algorithm. We randomly set 50 different initial solutions for each algorithm in each synthetic pair data. Other settings were the same in Sec. 6.1.

Figure 4 shows the results. In the same synthetic pair

data, C-CG and C-PG showed GWDs compatible with those of CG and PG but faster computation times, respectively. These results indicate that the constraint in C-CG and C-PG, which restricts the initial solution to an (m', n', K)-block circulant matrix, does not always degrade GWDs.

6.3. 2D Shape Analysis with Real-World Data

Similar to the existing Gromov–Wasserstein studies [8, 31], we evaluated whether GWDs obtained by our algorithms are useful to distinguish different 2D objects even if the objects have approximate cyclic symmetry.

We used the publicly available 2D structure dataset [7] and selected ten objects from each of the four classes, bone, glass, butterfly, and bell, with approximate mirror symmetry ($K = 2$). Each object consists of 100 to 200 vertices and triangles, and we used the vertices as a representative feature for the object as in [8, 31]. For the input data to CG and PG, we selected a pair of objects with m and n vertices, respectively, and set $\mathbf{a} = \frac{1}{m}\mathbf{1}_m$ and $\mathbf{b} = \frac{1}{n}\mathbf{1}_n$. Then, we defined $\mathbf{C}$ and $\mathbf{D}$ as the Euclidean distance matrices for pairwise vertex positions of each object. Because these real-world 2D objects satisfy Assumption 1 approximately due to slight displacement and distortion of shape, we cannot apply C-CG and C-PG to these data directly. Therefore, for the input data to C-CG and C-PG, we here first aligned the vertices of each object to exhibit *strict* mirror symmetry (for details, see Appendix D) and then set $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ in the same manner of CG and PG. This alignment preprocess is called *symmetric alignment*, mentioned at the beginning of Sec. 6. Figure 5 left panels show the original vertices and aligned ones of each object; we can see that the shape differences were not significant. Note that we also tested CG and PG with the symmetric alignment in this experiment for a fair comparison. For PG and C-PG, we set $\varepsilon = 0.002$.

For the evaluation, we computed the pairwise GWD between all objects using each algorithm. Also, to evaluate the discriminative power of the pairwise GWD obtained by each algorithm, we conducted a simple classification task as in [8, 25]; one object is randomly selected from each class and performs 1-nearest neighbor classification of the remaining objects using the pairwise GWD. This classification was repeated 10000 times, and we obtained a confusion matrix for each algorithm. That is, (i, j)-th entry of the confusion matrix indicates the probability that the classifier, which is based on the pairwise GWD obtained by each algorithm, will assign class j to an object with the true class i.

Figure 6 shows the results. CG produced almost the same pairwise GWDs and confusion matrices when using and not using the symmetric alignment. Also, because the symmetric alignment preprocess was so much faster than CG itself, the difference in total computation time to obtain the pairwise GWD between CG and CG with the symmetric alignment was negligible. In contrast, C-CG showed better

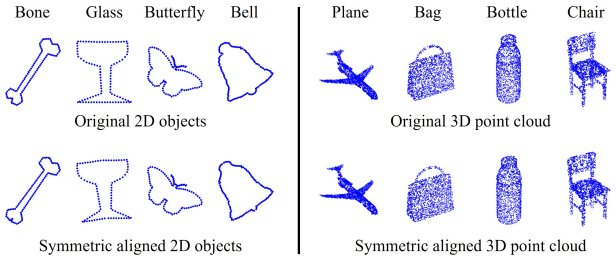

Figure 5. Input data examples in Sec. 6.3 (left) and Sec. 6.4 (right).

results than CG: (i) its pairwise GWD showed larger values between different classes, (ii) its confusion matrix was closer to the identity matrix, which indicates the classifications were done more successfully, and (iii) its total computation time was the fastest. The reason for (i) and (ii) is that CG, with or without the symmetric alignment, often reached worse local solutions due to its large solution space, while C-CG often reached better local solutions thanks to its smaller solution space that considers cyclic symmetry of input data. This implies that C-CG can properly use the inherent cyclic symmetry of input data emerging clearly by the symmetric alignment (for details, see Appendix E). This trend was also observed among PG, PG with the symmetric alignment, and C-PG. These results indicate that combining C-CG and C-PG with the symmetric alignment can be more effective than CG and PG in distinguishing different real-world 2D objects with approximate cyclic symmetry.

6.4. 3D Shape Analysis with Real-World Data

Following the existing studies [8, 25, 31], we also conducted experiments on a 3D object version of Sec. 6.3, which have more complex shapes and bigger data size.

We used the publicly available 3D structure dataset [11] and selected ten objects from each of the four classes, plane, bag, bottle, and chair, respectively, with approximate mirror symmetry ($K = 2$) along the YZ-axes plane. Each object is provided as 3D mesh data, and we approximated the object as a point cloud with 2000 points using trimesh library [14]. In the same manner of Sec. 6.3, we set the input data $\mathbf{a}, \mathbf{b}, \mathbf{C}$ and $\mathbf{D}$ to CG and PG by using each pair of the original point clouds, and set them to C-CG and C-PG by using each pair of the symmetrically aligned point clouds exhibiting strict mirror symmetry. For details of the symmetric alignment for these point clouds, see Appendix F. Figure 5 right panels show the original point cloud and the aligned ones of each 3D object. Other settings were the same in Sec. 6.3.

Figure 7 shows the results. Similar to Sec. 6.3, C-CG and C-PG showed better pairwise GWDs, confusion matrices, and faster computations than those of CG and PG, respectively. These results indicate that combining C-CG and C-PG with the symmetric alignment can be more effective than CG and PG in distinguishing different real-world 3D objects with approximate cyclic symmetry.

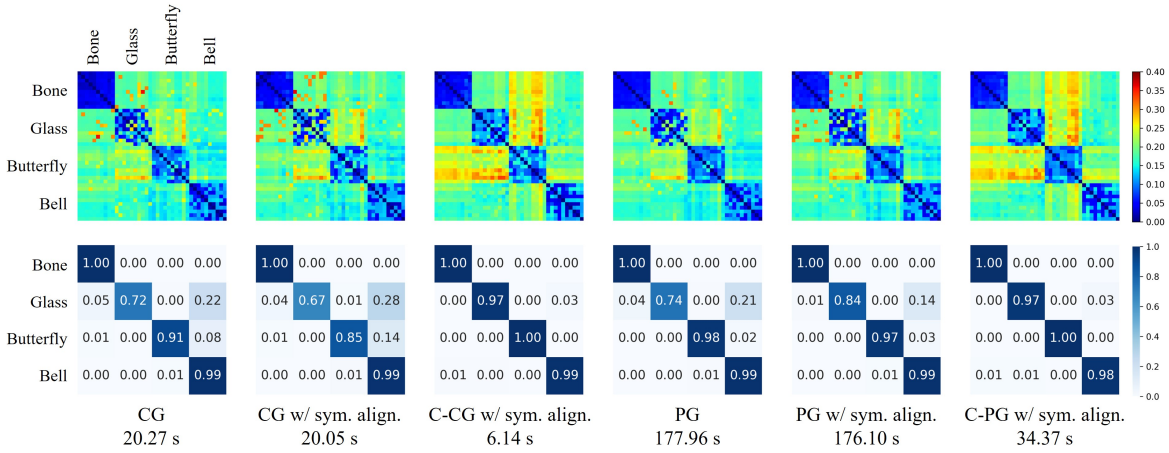

Figure 6. Experimental results in Sec. 6.3. Top panels show the pairwise GWD between all objects using each algorithm. Bottom panels show the confusion matrices obtained by each pairwise GWD. The time listed under each algorithm name represents the total computation time to obtain each pairwise GWD, which includes the symmetric alignment time if used.

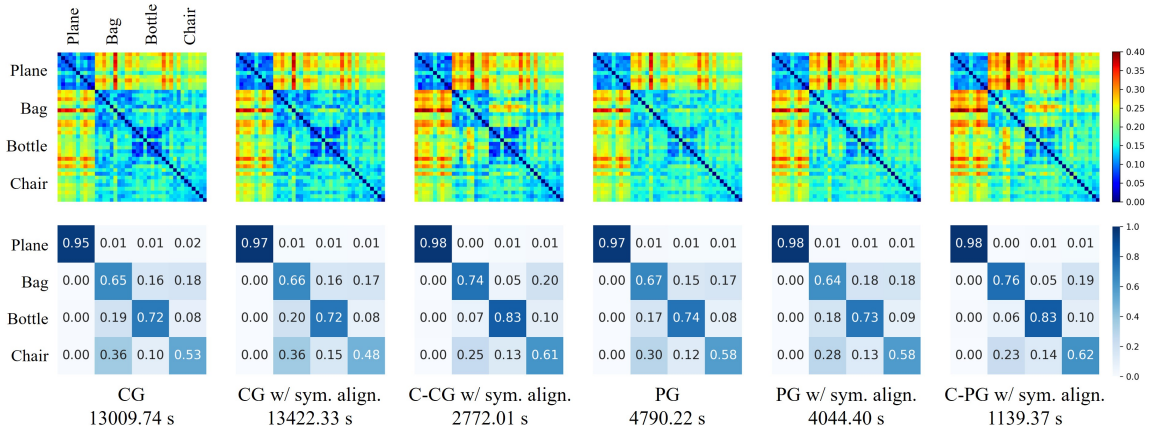

Figure 7. Experimental results in Sec. 6.4. The details are the same in Fig. 6.

7. Discussions and Limitations

We here list the future issues: (I) Our algorithms rely on Assumption 1 and need the symmetric alignment for objects with approximate cyclic symmetry. We experimentally confirmed that this process is not as computationally heavy as the main process of solving C-EGW in Secs. 6.3 and 6.4. However, in more complex real-world settings, it may lead to slower runtime or worse solutions. Thus, a theoretically guaranteed fast algorithm with symmetric alignment is desirable. (II) Our algorithms are limited to cyclic symmetry and must know its order K in advance. Generalizing for other symmetries without prior knowledge of their order is left for future work. (III) The output of our algorithms is restricted to the (m', n', K)-block circulant matrix. However, there exists an instance of C-EGW whose globally optimal solution is never such a matrix (for details, see Appendix G). A fast algorithm that guarantees reaching a globally optimal solution by using cyclic symmetry is desirable. (IV) As a first step, we incorporated cyclic symmetry into the two most widely used GW algorithms, namely CG and PG. Extending this approach to other GW algorithms is a promising direction for future research. (V) Combining other special structures, such as low-rank and hierarchical structures, with symmetry has excellent potential but is non-trivial because our approach may break such structures. We will explore this potential in the future.

8. Conclusion

We introduced a new problem, EGW with cyclic symmetry (C-EGW), and proposed novel fast algorithms for C-EGW. Our algorithms restrict the solution space to have cyclic symmetry with theoretical guarantees, thereby dramatically reducing the number of variables in the gradient-based algorithms and achieving fast computation. Diverse experiments showed the effectiveness of our algorithms in synthetic and real-world data with cyclic symmetry. This paper was the first to explore the possibility of symmetry in solving GW faster, paving the way for future followers.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993. 2, 3

[2] Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. Massively Scalable Sinkhorn Distances via the Nyström Method. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019. 2

[3] David Alvarez-Melis, Tommi Jaakkola, and Stefanie Jegelka. Structured Optimal Transport. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2018. 2

[4] Michel Antunes and João P. Barreto. SymStereo: Stereo Matching using Induced Symmetry. *International Journal of Computer Vision*, 109(3):187–208, 2014. 3

[5] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 4

[6] Larry Armijo. Minimization of Functions having Lipschitz Continuous First Partial Derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966. 3

[7] Axel Carlier, Kathryn Leonard, Stefanie Hahmann, Geraldine Morin, and Misha Collins. The 2D shape structure dataset: A user annotated open access database. *Computers & Graphics*, 58:23–30, 2016. 7

[8] Florian Beier, Robert Beinert, and Gabriele Steidl. On a Linear Gromov–Wasserstein Distance. *IEEE Transactions on Image Processing*, 31:7292–7305, 2022. 7

[9] Florian Bernard, Christian Theobalt, and Michael Moeller. DS: Tighter Lifting-Free Convex Relaxations for Quadratic Matching Problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[10] Charlotte Bunne, David Alvarez-Melis, Andreas Krause, and Stefanie Jegelka. Learning Generative Models across Incomparable Spaces. In *Proceedings of the International Conference on Machine Learning*, 2019. 1

[11] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 7

[12] Laetitia Chapel, Mokhtar Z. Alaya, and Gilles Gasso. Partial Optimal Transport with applications on Positive-Unlabeled Learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2020. 1

[13] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Proceedings of the Advances in Neural Information Processing Systems*, 2013. 1, 2, 3, 5, 6

[14] Dawson-Haggerty et al. trimesh. `https://trimesh.org/`, 2019. 7

[15] Balázs Dezső, Alpár Jüttner, and Péter Kovács. LEMON – an Open Source C++ Graph Template Library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45, 2011. Second Workshop on Generative Technologies. 6

[16] Mourad El Hamri, Younès Bennani, and Issam Falih. Hierarchical Optimal Transport for Unsupervised Domain Adaptation. *Machine Learning*, 111(11):4159–4182, 2022. 2

[17] Pascal Getreuer. A Survey of Gaussian Convolution Algorithms. *Image Processing On Line*, 3:286–310, 2013. 2

[18] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proceedings of the European Conference on Computer Vision*, 2008. 3

[19] Junyu Chen and Binh Nguyen and Yong Sheng Soh. Semidefinite Relaxations of the Gromov-Wasserstein Distance. In *NeurIPS 2023 Workshop Optimal Transport and Machine Learning*, 2023. 2

[20] Leonid Kantorovich. On the Translocation of Masses (in Russian). In *Doklady Akademii Nauk*, page 227, 1942. 1, 2, 3

[21] Itay Kezurer, Shahar Z. Kovalsky, Ronen Basri, and Yaron Lipman. Tight Relaxation of Quadratic Matching. In *Proceedings of the Eurographics Symposium on Geometry Processing*, 2015. 2

[22] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation*, 132(1):1–63, 1997. 3

[23] Tam Le, Nhat Ho, and Makoto Yamada. Flow-based Alignment Approaches for Probability Measures in Different Spaces . In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2021. 1, 2

[24] Marius Leordeanu and Martial Hebert. A Spectral Technique for Correspondence Problems Using Pairwise Constraints. In *Proceedings of the International Conference on Computer Vision*, 2005. 2

[25] Facundo Mémoli. Gromov-Wasserstein Distances and the Metric Approach to Object Matching. *Foundations of Computational Mathematics*, 11(4):417–487, 2011. 1, 2, 7

[26] Shuchao Pang, Anan Du, Mehmet A. Orgun, Yan Wang, Quan Z. Sheng, Shoujin Wang, Xiaoshui Huang, and Zhenmei Yu. Beyond CNNs: Exploiting Further Inherent Symmetries in Medical Image Segmentation. *IEEE Transactions on Cybernetics*, pages 1–12, 2022. 3

[27] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport: With Applications to Data Science*. Now Publishers, 2019. 2

[28] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-Wasserstein Averaging of Kernel and Distance Matrices. In *Proceedings of the International Conference on Machine Learning*, 2016. 1, 2, 3, 6

[29] Meyer Scetbon, Gabriel Peyré, and Marco Cuturi. Linear-Time Gromov Wasserstein Distances using Low Rank Couplings and Costs. In *Proceedings of the International Conference on Machine Learning*, 2022. 1, 2

[30] Richard Sinkhorn. Diagonal Equivalence to Matrices with Prescribed Row and Column Sums. In *The American Mathematical Monthly*, page 402–405, 1967. 2, 3, 5, 6

[31] Justin Solomon, Gabriel Peyré, Vladimir G. Kim, and Suvrit Sra. Entropic Metric Alignment for Correspondence Prob-

lems. *ACM Transactions on Graphics*, 35(4):1–13, 2016. 1, 2, 7

[32] Taeyong Song, Sunok Kim, and Kwanghoon Sohn. Unsupervised Deep Asymmetric Stereo Matching with Spatially-Adaptive Self-Similarity . In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 3

[33] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. Deformable Medical Image Registration: A Survey. *IEEE Transactions on Medical Imaging*, 32(7):1153–1190, 2013. 3

[34] Pablo Speciale, Martin R. Oswald, Andrea Cohen, and Marc Pollefeys. A Symmetry Prior for Convex Variational 3D Reconstruction. In *Proceedings of the European Conference on Computer Vision*, 2016. 4

[35] Richard Szeliski and Daniel Scharstein. Symmetric Sub-pixel Stereo Matching. In *Proceedings of the European Conference on Computer Vision*, 2002. 3

[36] Shoichiro Takeda, Yasunori Akagi, Naoki Marumo, and Kenta Niwa. Optimal Transport with Cyclic Symmetry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. 2, 4, 5

[37] Robert E Tarjan. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Mathematical Programming*, 78(2):169–177, 1997. 5

[38] Evgeny Tenetov, Gershon Wolansky, and Ron Kimmel. Fast Entropic Regularized Optimal Transport Using Semidiscrete Cost Approximation. *SIAM Journal on Scientific Computing*, 40(5):A3400–A3422, 2018. 2

[39] Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Proceedings of the International Conference on Computer Vision*, 2005. 4

[40] Alexis Thual, Huy Tran, Tatiana Zemskova, Nicolas Courty, Rémi Flamary, Stanislas Dehaene, and Bertrand Thirion. Aligning individual brains with Fused Unbalanced Gromov-Wasserstein. In *Proceedings of the Advances in Neural Information Processing Systems*, 2022. 3

[41] Vayer Titouan, Nicolas Courty, Romain Tavenard, Chapel Laetitia, and Rémi Flamary. Optimal Transport for structured data with application on graphs. In *Proceedings of the International Conference on Machine Learning*, 2019. 1, 2, 3, 6

[42] Vayer Titouan, Rémi Flamary, Nicolas Courty, Romain Tavenard, and Laetitia Chapel. Sliced Gromov-Wasserstein. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019. 1, 2

[43] Giorgos Tolias, Yannis Kalantidis, and Yannis Avrithis. SymCity: Feature Selection by Symmetry for Large Scale Image Retrieval. In *Proceedings of the ACM International Conference on Multimedia*, 2012. 3

[44] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 6

[45] Wang, Weiyue and Ceylan, Duygu and Mech, Radomir and Neumann, Ulrich. 3DN: 3D Deformation Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 4

[46] Philip Wolfe. Convergence Conditions for Ascent Methods. *SIAM review*, 11(2):226–235, 1969. 3

[47] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised Learning of Probably Symmetric Deformable 3D Objects From Images in the Wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 4

[48] Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching. In *Proceedings of the Advances in Neural Information Processing Systems*, 2019. 1, 2

[49] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-Wasserstein Learning for Graph Matching and Node Embedding. In *Proceedings of the International Conference on Machine Learning*, 2019. 1, 2

[50] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2021. 4

[51] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face Recognition: A Literature Survey. *ACM Computing Surveys*, 35(4):399–458, 2003. 3