Supplementary Material for: Flow-based Alignment Approaches for Probability Measures in Different Spaces

Tam Le[*]
RIKEN AIP

Nhat Ho[*]
The University of Texas at Austin

Makoto Yamada
Kyoto University & RIKEN AIP

This supplementary material is for the main text in [5]. We organize it as follow:

- In Appendix A, we provide proofs for theoretical results: Theorem 1, Theorem 2, and Proposition 1 in the main text.
- In Appendix B, we show more illustrations and geometric properties, and discuss about the rotational and translational invariance for Flow-based Alignment (*FlowAlign*) mentioned in the main text.
- In Appendix C, we describe further details for *FlowAlign* and Depth-based Alignment (*DepthAlign*), e.g., algorithms and complexity for *FlowAlign* and *DepthAlign* on applications with or without priori knowledge about tree structures for probability measures.
- In Appendix D, we illustrate
  - further experimental results in quantum chemistry (`qm7` dataset), document classification (`TWITTER, RECIPE, CLASSIC, AMAZON` datasets) considered in the main text;
  - time consumption for the clustering-based tree metric sampling;
  - results with different parameters for tree metric sampling;
  - and a small experimental setup for performance comparison for k-means clustering on *randomly rotated* `MNIST` dataset.
- In Appendix E, we give some brief reviews for
  - the farthest-point clustering;
  - clustering-based tree metric sampling;
  - tree metric;
  - F_β measure for clustering evaluation;
  - and more information for datasets.
- In Appendix F, we provide some further discussions.
- In Appendix G, we investigate empirical relations among the considered discrepancies (e.g., *FlowAlign*, *DepthAlign*, sliced GW (SGW), entropic GW (EGW) and the standard entropic GW where entropic regularization is used in both transportation plan optimization and objective function computation (EGW_0)) for probability measures in different spaces.

Note that we have released code for our proposal at

`https://github.com/lttam/FlowBasedAlignment-GW`

Notations. We use same notations as in the main text [5].

[*]: The first two authors contributed equally.

A Proofs

In this section, we provide the proofs for the pseudo-distances and properties of *FlowAlign* and *DepthAlign* discrepancies, i.e., Theorem 1, Theorem 2, and Proposition 1 in the main text.

A.1 Proof of Theorem 1 in the main text

From the definition of *FlowAlign* $\mathcal{A}_f$, it is symmetric, namely, $\mathcal{A}_f(\mu, \nu) = \mathcal{A}_f(\nu, \mu)$. In addition, it is clear that $\mathcal{A}_f(\mu, \mu) = 0$. Finally, we show that $\mathcal{A}_f$ also satisfies triangle inequality as in Proposition 1.

Proposition 1. *Given three probability measures μ, ν, γ in three different metric spaces $(\mathcal{T}_X, d_{\mathcal{T}_X})$, $(\mathcal{T}_Y, d_{\mathcal{T}_Y})$, and $(\mathcal{T}_Z, d_{\mathcal{T}_Z})$. Then, we have:*

$$\mathcal{A}_f(\mu, \gamma) \leq \mathcal{A}_f(\mu, \nu) + \mathcal{A}_f(\nu, \gamma).$$

Proof. It is sufficient to demonstrate that

$$\widehat{\mathcal{A}}_f(\mu, \gamma; r_x, r_y) \leq \widehat{\mathcal{A}}_f(\mu, \nu; r_x, r_z) + \widehat{\mathcal{A}}_f(\nu, \gamma; r_y, r_z), \tag{1}$$

for any roots r_x, r_y, r_z of $\mathcal{T}_X, \mathcal{T}_Y, \mathcal{T}_Z$ respectively. Our proof for the above inequality is a direct application of the gluing lemma in [14]. In particular, for any roots r_x, r_y, r_z of of $\mathcal{T}_X, \mathcal{T}_Y, \mathcal{T}_Z$, we denote $\widehat{T}^1 \in \Pi(\mu, \nu)$ and $\widehat{T}^2 \in \Pi(\nu, \gamma)$ as optim al transport plans for $\widehat{\mathcal{A}}_f(\mu, \nu; r_x, r_z)$ and $\widehat{\mathcal{A}}_f(\nu, \gamma; r_y, r_z)$ respectively. Based on the gluing lemma, there exists T with marginal of the first and the third factors as $\widehat{T}^1$ and marginal of the second and the third factors as $\widehat{T}^2$. We denote the marginal of its first and second factors as $\bar{T}$, which is a transport plan between μ and γ. Therefore, from the definition of aligned-root *FlowAlign* discrepancy, we have

$$\begin{aligned}
\widehat{\mathcal{A}}^2_f(\mu, \gamma; r_x, r_y) &\leq \textstyle\sum_{i,j} \left|d_{\mathrm{T}_X}(r_x, x_i) - d_{\mathrm{T}_Y}(r_y, y_j)\right|^2 \bar{T}_{ij} \\
&= \textstyle\sum_{i,j,k} \left|d_{\mathrm{T}_X}(r_x, x_i) - d_{\mathrm{T}_Y}(r_y, y_j)\right|^2 T_{ijk} \\
&= \textstyle\sum_{i,j,k} \left|d_{\mathrm{T}_X}(r_x, x_i) - d_{\mathrm{T}_Z}(r_z, z_k)\right|^2 T_{ijk} + \sum_{i,j,k} \left|d_{\mathrm{T}_X}(r_y, y_j) - d_{\mathrm{T}_Z}(r_z, z_k)\right|^2 T_{ijk} \\
&\quad -2 \textstyle\sum_{i,j,k} (d_{\mathrm{T}_X}(r_x, x_i) - d_{\mathrm{T}_Z}(r_z, z_k))(d_{\mathrm{T}_Y}(r_y, y_j) - d_{\mathrm{T}_Z}(r_z, z_k)) T_{ijk} \\
&\leq \widehat{\mathcal{A}}^2_f(\mu, \nu; r_x, r_z) + \widehat{\mathcal{A}}^2_f(\nu, \gamma; r_y, r_z) \\
&\quad +2 \widehat{\mathcal{A}}_f(\mu, \nu; r_x, r_z) \widehat{\mathcal{A}}_f(\nu, \gamma; r_y, r_z) \\
&= (\widehat{\mathcal{A}}_f(\mu, \nu; r_x, r_z) + \widehat{\mathcal{A}}_f(\nu, \gamma; r_y, r_z))^2,
\end{aligned} \tag{2}$$

where we used Hölder's inequality for the third term for the second inequality. As a consequence, we obtain the conclusion of the inequality in Equation (1). ■

Discussion about $\mathcal{A}^2_f(\mu, \nu) = 0$. As discussed in the main text, when $\mathcal{A}^2_f(\mu, \nu) = 0$, we can find roots r^*_x and r^*_z such that $\tilde{\mu}^* \equiv \tilde{\nu}^*$ where $\tilde{\mu}^* = \sum_i a_i \delta_{d_{\mathrm{T}_X}(r^*_x, x_i)}$ and $\tilde{\nu}^* = \sum_j b_j \delta_{d_{\mathrm{T}_Z}(r^*_z, z_j)}$. It demonstrates that μ and ν have the same weights on supports (i.e., flow masses) while the tree metrics of their supports to the corresponding root r^*_x or r^*_z (i.e., flow lengths) are identical.

A.2 Proof of Theorem 2 in the main text

In fact, from the definition of *DepthAlign* $\mathcal{A}_d$, it is clear that $\mathcal{A}_d(\mu, \nu) = \mathcal{A}_d(\nu, \mu)$ and $\mathcal{A}_d(\mu, \mu) = 0$. Furthermore, $\mathcal{A}_d$ satisfies triangle inequality as in Proposition 2.

Proposition 2. *Given three probability measures μ, ν, γ in three different metric spaces $(\mathcal{T}_X, d_{\mathcal{T}_X})$, $(\mathcal{T}_Y, d_{\mathcal{T}_Y})$, and $(\mathcal{T}_Z, d_{\mathcal{T}_Z})$. Then, we have:*

$$\mathcal{A}_d(\mu, \gamma) \leq \mathcal{A}_d(\mu, \nu) + \mathcal{A}_d(\nu, \gamma).$$

Proof. Similar to the proof of Proposition 1, it is sufficient to demonstrate that

$$\widehat{\mathcal{A}}_d(\mu, \gamma; r_x, r_y) \leq \widehat{\mathcal{A}}_d(\mu, \nu; r_x, r_z) + \widehat{\mathcal{A}}_d(\nu, \gamma; r_y, r_z), \tag{3}$$

for any roots r_x, r_y, r_z of $\mathcal{T}_X, \mathcal{T}_Y, \mathcal{T}_Z$ respectively. According to the definition of aligned-root *DepthAlign*, the above inequality is equivalent to

$$\sum_h \sum_{(x,y)\in\mathcal{M}^{1,2}_{h-1}} T^*_h(x,y)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \gamma_{\mathcal{T}^2_y}; x, y) \leq \sum_h \sum_{(x,z)\in\mathcal{M}^{1,3}_{h-1}} \bar{T}^*_h(x,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \nu_{\mathcal{T}^2_z}; x, z) + \sum_h \sum_{(y,z)\in\mathcal{M}^{2,3}_{h-1}} \tilde{T}^*_h(y,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_y}, \nu_{\mathcal{T}^2_z}; y, z), \tag{4}$$

where $\mathcal{M}^{1,2}_h, \mathcal{M}^{1,3}_h, \mathcal{M}^{2,3}_h$ are respectively sets of optimal aligned pairs at the depth level h from trees $\mathcal{T}_X$ and $\mathcal{T}_Y$, from trees $\mathcal{T}_X$ and $\mathcal{T}_Z$, and from trees $\mathcal{T}_Y$ and $\mathcal{T}_Z$; $T^*_h(x,y), \bar{T}^*_h(x,z), \tilde{T}^*_h(y,z)$ are respectively optimal matching masses for the pairs $(x,y)\in\mathcal{M}^{1,2}_{h-1}, (x,z)\in\mathcal{M}^{1,3}_{h-1}, (y,z)\in\mathcal{M}^{2,3}_{h-1}$. In order to demonstrate the above inequality, we only need to verify that

$$\sum_{(x,y)\in\mathcal{M}^{1,2}_{h}} T^*_h(x,y)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \gamma_{\mathcal{T}^2_y}; x, y) \leq \sum_{(x,z)\in\mathcal{M}^{1,3}_{h}} \bar{T}^*_h(x,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \nu_{\mathcal{T}^2_z}; x, z) + \sum_{(y,z)\in\mathcal{M}^{2,3}_{h}} \tilde{T}^*_h(y,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_y}, \nu_{\mathcal{T}^2_z}; y, z), \tag{5}$$

for any depth level $h \geq 1$. We respectively denote $\mathcal{T}^h_X = \{\bar{x}^{(h)}_1, \ldots, \bar{x}^{(h)}_{k_{1,h}}\}, \mathcal{T}^h_Y = \{\bar{y}^{(h)}_1, \ldots, \bar{y}^{(h)}_{k_{2,h}}\}, \mathcal{T}^h_Z = \{\bar{z}^{(h)}_1, \ldots, \bar{z}^{(h)}_{k_{3,h}}\}$ the set of nodes in depth level h of the trees $\mathcal{T}_X, \mathcal{T}_Y, \mathcal{T}_Z$. The inequality in Equation (5) can be rewritten as

$$\begin{aligned}\sum_{i=1}^{k_{1,h-1}} \sum_{j=1}^{k_{2,h-1}} \sum_{x\in\mathcal{S}(\bar{x}^{(h-1)}_i), y\in\mathcal{S}(\bar{y}^{(h-1)}_j)} & T^*_h(x,y)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \gamma_{\mathcal{T}^2_y}; x, y) \\ \leq & \sum_{i=1}^{k_{1,h-1}} \sum_{j=1}^{k_{3,h-1}} \sum_{x\in\mathcal{S}(\bar{x}^{(h-1)}_i), z\in\mathcal{S}(\bar{z}^{(h-1)}_j)} \bar{T}^*_h(x,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \nu_{\mathcal{T}^2_y}; x, z) \\ & + \sum_{i=1}^{k_{2,h-1}} \sum_{j=1}^{k_{3,h-1}} \sum_{y\in\mathcal{S}(\bar{y}^{(h-1)}_i), z\in\mathcal{S}(\bar{z}^{(h-1)}_j)} \tilde{T}^*_h(y,z)\widehat{\mathcal{A}}_f(\gamma_{\mathcal{T}^2_y}, \nu_{\mathcal{T}^2_y}; y, z).\end{aligned} \tag{6}$$

In order to obtain the conclusion of inequality in Equation (6), we only need to prove that

$$\begin{aligned}\sum_{x\in\mathcal{S}(\bar{x}^{(h-1)}_i), y\in\mathcal{S}(\bar{y}^{(h-1)}_j)} & T^*_h(x,y)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \gamma_{\mathcal{T}^2_y}; x, y) \\ \leq & \sum_{x\in\mathcal{S}(\bar{x}^{(h-1)}_i), z\in\mathcal{S}(\bar{z}^{(h-1)}_l)} \bar{T}^*_h(x,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2_x}, \nu_{\mathcal{T}^2_y}; x, z) \\ & + \sum_{y\in\mathcal{S}(\bar{y}^{(h-1)}_i), z\in\mathcal{S}(\bar{z}^{(h-1)}_l)} \tilde{T}^*_h(y,z)\widehat{\mathcal{A}}_f(\gamma_{\mathcal{T}^2_y}, \nu_{\mathcal{T}^2_y}; y, z),\end{aligned}$$

for any $i \in \{1, \ldots, k_{1,h-1}\}$, $j \in \{1, \ldots, k_{2,h-1}\}$, and $l \in \{1, \ldots, k_{3,h-1}\}$. We can make use of the gluing lemma [14] to prove that inequality. In fact, there exists T with marginal of its first and third factors as $\bar{T}^*_h(x,z)$ and marginal of its second and third factors as $\bar{T}^*_h(y,z)$. We denote the marginal of its first and second factors as $\hat{T}$, which is the transport plan for probability measures with supports at $x \in \mathcal{S}(\bar{x}^{(h-1)}_i)$ and $y \in \mathcal{S}(\bar{y}^{(h-1)}_j)$. Now, we have the following inequalities

$$\begin{aligned}
&\sum_{x \in \mathcal{S}(\bar{x}_i^{(h-1)}), y \in \mathcal{S}(\bar{y}_j^{(h-1)})} T_h^*(x,y)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}_x^2}, \gamma_{\mathcal{T}_y^2}; x, y) \\
&\quad = \sum_{x \in \mathcal{S}(\bar{x}_i^{(h-1)}), y \in \mathcal{S}(\bar{y}_j^{(h-1)}), z \in \mathcal{S}(\bar{z}_l^{(h-1)})} T_{xyz}\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}_x^2}, \gamma_{\mathcal{T}_y^2}; x, y) \\
&\quad \leq \sum_{x \in \mathcal{S}(\bar{x}_i^{(h-1)}), y \in \mathcal{S}(\bar{y}_j^{(h-1)}), z \in \mathcal{S}(\bar{z}_l^{(h-1)})} T_{xyz}\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}_x^2}, \nu_{\mathcal{T}_z^2}; x, z) \\
&\qquad + \sum_{x \in \mathcal{S}(\bar{x}_i^{(h-1)}), y \in \mathcal{S}(\bar{y}_j^{(h-1)}), z \in \mathcal{S}(\bar{z}_l^{(h-1)})} T_{xyz}\widehat{\mathcal{A}}_f(\gamma_{\mathcal{T}_y^2}, \nu_{\mathcal{T}_z^2}; y, z) \\
&\quad = \sum_{x \in \mathcal{S}(\bar{x}_i^{(h-1)}), z \in \mathcal{S}(\bar{z}_l^{(h-1)})} \bar{T}_h^*(x,z)\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}_x^2}, \nu_{\mathcal{T}_y^2}; x, z) \\
&\qquad + \sum_{y \in \mathcal{S}(\bar{y}_i^{(h-1)}), z \in \mathcal{S}(\bar{z}_l^{(h-1)})} \tilde{T}_h^*(y,z)\widehat{\mathcal{A}}_f(\gamma_{\mathcal{T}_y^2}, \nu_{\mathcal{T}_y^2}; y, z).
\end{aligned}$$

As a consequence, we obtain the conclusion of the proposition. ■

Discussion about $\mathcal{A}_d(\mu, \nu) = 0$. As discussed in the main text, when $\mathcal{A}_d(\mu, \nu) = 0$, we can find roots r_x^* and r_z^* such that all the hierarchical corresponding $\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}_\cdot^2}, \nu_{\mathcal{T}_\cdot^2}; \cdot, \cdot)$ for each depth level along the trees are equal to 0. It demonstrates that μ and ν have the same weights on supports while their supports have the same depth levels. For each depth level, the tree metrics of supports in the corresponding $\mu_{\mathcal{T}_\cdot^2}, \nu_{\mathcal{T}_\cdot^2}$ to the 2-depth-level-tree roots are identical, i.e., corresponding weight edges are identical.

A.3 Proof of Proposition 1 in the main text

The proof of Proposition 1 in the main text is a direct application of Cauchy-Schwarz. In particular, since the deepest levels of trees $\mathcal{T}_X$ and $\mathcal{T}_Z$ are equal to two for some roots r_x and r_z respectively, we obtain that

$$\begin{aligned}
\mathcal{GW}^2(\mu, \nu) &= \min_{T \in \Pi(\mu,\nu)} \sum_{i,j,i',j'} \left| d_{\mathrm{T}_X}(x_i, x_{i'}) - d_{\mathrm{T}_Z}(z_j, z_{j'}) \right|^2 T_{ij} T_{i'j'} \\
&= \min_{T \in \Pi(\mu,\nu)} \sum_{i,j,i',j'} \left| d_{\mathrm{T}_X}(x_i, r_x) + d_{\mathrm{T}_X}(r_x, x_{i'}) - d_{\mathrm{T}_Z}(z_j, r_z) - d_{\mathrm{T}_Z}(r_z, z_{j'}) \right|^2 T_{ij} T_{i'j'} \\
&\leq \min_{T \in \Pi(\mu,\nu)} \sum_{i,j,i',j'} 2\left[(d_{\mathrm{T}_X}(x_i, r_x) - d_{\mathrm{T}_Z}(z_j, r_z))^2 + (d_{\mathrm{T}_X}(r_x, x_{i'}) - d_{\mathrm{T}_Z}(r_z, z_{j'}))^2 \right] T_{ij} T_{i'j'} \\
&= 4\widehat{\mathcal{A}}_f^2(\mu, \nu; r_x, r_z) \qquad (7)
\end{aligned}$$

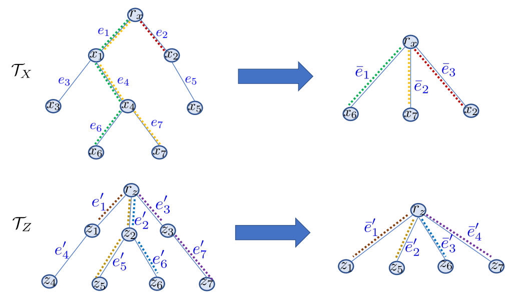
where the inequality is due to the standard Cauchy-Schwarz inequality $(a+b)^2 \leq 2(a^2+b^2)$ for any $a, b \in \mathbb{R}$. By taking the infimum over r_x, r_z, then square root on both sides of the above inequality (7), we obtain the conclusion of the proposition that

$$\mathcal{GW}(\mu, \nu) \leq 2\mathcal{A}_f(\mu, \nu).$$

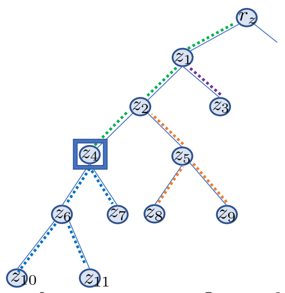
B Further illustrations, geometric properties, and discussion about rotational and translational invariance for *FlowAlign*

In this section, we provide further illustrations for *FlowAlign* mentioned in the main text.

- In Figure 1a, we illustrate the aligned-root *FlowAlign* $\widehat{\mathcal{A}}_f$.
- In Figure 1b, we illustrate for supports in Ω_ν in the **Case 2** in the efficient computation approach for *FlowAlign* $\mathcal{A}_f$.



(a) An illustration for the aligned-root *FlowAlign* $\widehat{\mathcal{A}}_f$ between $\mu = a_1\delta_{x_6} + a_2\delta_{x_7} + a_3\delta_{x_2}$ on $\mathcal{T}_X$ and $\nu = b_1\delta_{z_1} + b_2\delta_{z_5} + b_3\delta_{z_6} + b_4\delta_{z_7}$ on $\mathcal{T}_Z$.



(b) An illustration for supports in Ω_ν in the **Case 2** in the efficient computation approach for $\mathcal{A}_f$.

Figure 1: In (a), in $\widehat{\mathcal{A}}_f$, we consider the flows from the root to each support of a measure as illustrated in the corresponding right figures where the length of each edge in the right figures is equal to the length of the path from the root to that node on the corresponding tree $\mathcal{T}_.$ in the left figures respectively. In (b), assume that $\Omega_\nu = \{z_1, z_2, z_3, z_7, z_8, z_9, z_{10}, z_{11}\}$ and the new root $\bar{r}_z = z_4$ (emphasized by the square border). We have supports z_7, z_{10}, z_{11} for **Case 2a** (blue dots), supports z_1, z_2 for **Case 2b** (green dots), and supports z_3, z_8, z_9 for **Case 2c** (purple and orange dots) where the corresponding closest common ancestors $\zeta_3 = z_1$, $\zeta_8 = z_2$, and $\zeta_9 = z_2$ respectively. Note that, $\zeta_8 = \zeta_9$ (orange dots), therefore the order of supports z_8, z_9 is preserved when one changes into the new root.

Rotational and translational invariance for *FlowAlign*. In practice, we usually do not have priori knowledge about tree structures for probability measures[1]. Therefore, we need to choose or sample trees $\mathcal{T}_X$ and $\mathcal{T}_Z$ from support data points, e.g. by clustering-based tree metric sampling [6].

For the clustering-based tree metric sampling[2], the farthest-point clustering[3] within the clustering-based tree metric sampling gives the same results for rotational and/or translational support data points and for the original ones, when one uses the same corresponding point in the given finite set of support data points as its initialization). Therefore, tree metric sampled from the clustering-based tree metric sampling is rotational and translational invariance (i.e., tree structure and lengths of edges are the same, only nodes are represented for the corresponding rotational and/or translational support data points instead of the original ones.). Consequently, the *FlowAlign* has rotational and translational invariance. As showed in the main text (§6.1), the *FlowAlign* works well with the quantum chemistry (in a real dataset: `qm7`) where one needs translational and rotational invariance for the relative positions of atoms in $\mathbb{R}^3$ for each molecule.

As a trivial extension, due to rotational and translation invariance for tree metrics sampled from the clustering-based tree metric sampling, *DepthAlign* also has rotational and translational invariance.

C Further details for *FlowAlign* and *DepthAlign*

In this section, we first derive a computation for a univariate optimal transport for empirical measures. Then, we give some further details about *FlowAlign* and *DepthAlign* proposed in the main text.

C.1 Univariate optimal transport (OT) for empirical measures

Recall that the univariate OT, i.e., univariate Wasserstein, is equal to the integral of the absolute difference between the generalized quantile functions of two univariate probability distributions [11] (§2). Therefore, one only needs to sort their supports for the computation with linearithmic complexity.

In particular, given two empirical measures $\mu = \sum_{i\in[n]} \bar{a}_i\delta_{\bar{x}_i}$ and $\nu = \sum_{j\in[m]} \bar{b}_j\delta_{\bar{z}_j}$ whose supports are in one-dimensional space, i.e., $\bar{x}_i, \bar{z}_j \in \mathbb{R}, \forall i \in [n], j \in [m]$. Firstly, we sort supports of μ, ν in an increasing order, denoted as $\mu = \sum_{i\in[n]} a_i\delta_{x_i}$ and $\nu = \sum_{j\in[m]} b_j\delta_{z_j}$ (i.e., $x_1 \le x_2 \le \cdots \le x_n$, and $z_1 \le z_2 \le \cdots \le z_m$). Without loss of generality, assume that $n \ge m$, the complexity of this sorting is $\mathcal{O}(n\log n)$. We summarize the algorithm

[1]For a priori tree metric space, recall that tree metric space is finite. So, rotation/translation may not be directly well-defined in tree metric space.

[2]see Appendix E.2 for a review

[3]see Appendix E.1 for a review

Algorithm 1 Univariate optimal transport for empirical measures

```
Input: Input empirical measures with sorted supports $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$, and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$ (i.e., $x_1 \le x_2 \le \cdots \le x_n$, and $z_1 \le z_2 \le \cdots \le z_m$), and a ground distance $\ell$ (e.g., $\ell(x, z) = \ell_1(x, z) = |x - z|$).
Output: OT distance $d$ and optimal transport plan $T$.
 1: Initialize $d \leftarrow 0$, $T \leftarrow 0_{n \times m}$, $i \leftarrow 1$, $j \leftarrow 1$.
 2: while $i \le n$ and $j \le m$ do
 3:   if $a_i \le b_j$ then
 4:     $T_{ij} \leftarrow a_i$.
 5:     $d \leftarrow d + a_i \ell(x_i, z_j)$.
 6:     Update $b_j \leftarrow b_j - a_i$, $i \leftarrow i + 1$.
 7:     if $b_j == 0$ then
 8:       $j \leftarrow j + 1$.
 9:     end if
10:   else
11:     $T_{ij} \leftarrow b_j$.
12:     $d \leftarrow d + b_j \ell(x_i, z_j)$.
13:     Update $a_i \leftarrow a_i - b_j$, $j \leftarrow j + 1$.
14:     if $a_i == 0$ then
15:       $i \leftarrow i + 1$.
16:     end if
17:   end if
18: end while
```

for the univariate OT between μ and ν (whose supports are already sorted) in Algorithm 1.

The complexity of Algorithm 1 is $\mathcal{O}(n)$. Therefore, the complexity of the univariate OT for empirical measures is $\mathcal{O}(n \log n)$, or its main complexity is to sort supports of empirical measures.

C.2 *FlowAlign*

There are two types of applications: *without* or *with* priori knowledge about tree metrics for supports in probability measures.

Algorithm 2 *FlowAlign* for probability measures by sampling aligned-root tree metrics

```
Input: Input probability measures $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$, and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$.
Output: FlowAlign discrepancy $d$.
 1: Sample aligned-root tree metrics $\mathcal{T}_X, \mathcal{T}_Z$ for $\mu, \nu$ respectively, e.g., by choosing a mean of support data as its root for the clustering-based tree metric sampling [6].
 2: Construct $\tilde{\mu} \leftarrow \sum_{i \in [n]} a_i \delta_{d_{\mathcal{T}_X}(r_x, x_i)}$ and $\tilde{\nu} \leftarrow \sum_{j \in [m]} b_j \delta_{d_{\mathcal{T}_Z}(r_z, z_j)}$
 3: Sort support(s) $d_{\mathcal{T}_X}(r_x, x_i) \mid_{i \in [n]}$ and $d_{\mathcal{T}_Z}(r_z, z_j) \mid_{j \in [m]}$, then $d \leftarrow$ Algorithm 1 for $\tilde{\mu}$ and $\tilde{\nu}$.
```

C.2.1 Applications *without* priori knowledge about tree metrics for supports in probability measures

In general applications, one usually does not have priori knowledge about tree metrics for supports in probability measures. However, one can sample tree metrics for the space of supports of probability measures, e.g., using clustering-based tree metric sampling [6] (§4), for *FlowAlign*.

One can compute *FlowAlign* between $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$ and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$ as follow:

- Step 1: Sample aligned-root tree metrics $\mathcal{T}_X$ and $\mathcal{T}_Z$ for supports $x_i \mid_{i \in [n]}$, and $z_j \mid_{j \in [m]}$ of probability measures μ and ν respectively, e.g., by choosing means of support data distributions as roots when using the clustering-based tree metric sampling [6] (§4) (See Section E.2 for a review about clustering-based tree metric sampling).

- Step 2: Based on the sampled aligned-root tree metrics, *FlowAlign* (Equation (2) in the main text) is equivalent to aligned-root *FlowAlign* (Equation (4) in the main text). Consequently, *FlowAlign* between μ and ν is equivalent to the univariate OT distance between $\tilde{\mu} := \sum_{i \in [n]} a_i \delta_{d_{\mathrm{T}_X}(r_x, x_i)}$ and $\tilde{\nu} := \sum_{j \in [m]} b_j \delta_{d_{\mathrm{T}_Z}(r_z, z_j)}$ where r_x, r_z are roots of $\mathcal{T}_X, \mathcal{T}_Z$ respectively.

- Step 3: Sort supports of $\tilde{\mu}$ and $\tilde{\nu}$, and then apply Algorithm 1 to compute the univariate OT between $\tilde{\mu}$ and $\tilde{\nu}$.

We summarize this computation of *FlowAlign* in Algorithm 2. We next show a complexity analysis for *FlowAlign*:

- The complexity of Step 1 is $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa)$ where $H_{\mathcal{T}}$ is a predefined deepest level of tree $\mathcal{T}$ and κ is the number of clusters in the farthest-point clustering used in the clustering-based tree metric sampling [6]; $\bar{N}$ is the input number of supports[4]. Let N be the number of nodes in the sampled tree $\mathcal{T}$, we have $N \le \left(\kappa^{H_{\mathcal{T}}} - 1\right) / (\kappa - 1)$.

- The complexity of Step 2 is $\mathcal{O}(n H_{\mathcal{T}})$ for computing supports of $\tilde{\mu}, \tilde{\nu}$.

- The complexity of Step 3 is $\mathcal{O}(n \log n)$ as in Section C.1.

In general, one usually chooses small values for $H_{\mathcal{T}}$ and κ (e.g., $(H_{\mathcal{T}} = 6, \kappa = 4)$ are suggested parameters for the clustering-based tree metric sampling [6]); and has $n \le N$ (each support is corresponding to a node in a tree). Therefore, the overall complexity of *FlowAlign* is $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + n H_{\mathcal{T}} + n \log n)$, or approximately $\mathcal{O}(N \log N)$.

Algorithm 3 *FlowAlign* for probability measures with priori tree metrics by exhaustively searching optimal pair of roots

```
Input: Input probability measures $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$, and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$ where $x_i \mid_{i \in [n]}$ in tree $\mathcal{T}_X$, and $z_j \mid_{j \in [m]}$ in tree $\mathcal{T}_Z$.
Output: FlowAlign discrepancy $d$.
 1: Initialize $d \leftarrow \infty$.
 2: for each $x \in \mathcal{T}_X$ do
 3:     Construct $\tilde{\mu} \leftarrow \sum_{i \in [n]} a_i \delta_{d_{\mathrm{T}_X}(x, x_i)}$, then sort support(s) $d_{\mathrm{T}_X}(x, x_i) \mid_{i \in [n]}$.
 4:     for each $z \in \mathcal{T}_Z$ do
 5:         Construct $\tilde{\nu} \leftarrow \sum_{j \in [m]} b_j \delta_{d_{\mathrm{T}_Z}(z, z_j)}$, then sort support(s) $d_{\mathrm{T}_Z}(z, z_j) \mid_{j \in [m]}$.
 6:         $\tilde{d} \leftarrow$ Algorithm 1 for $\tilde{\mu}$ and $\tilde{\nu}$.
 7:         if $d > \tilde{d}$ then
 8:             $d \leftarrow \tilde{d}$.
 9:         end if
10:     end for
11: end for
```

C.2.2 Applications *with* priori knowledge about tree metrics for supports in probability measures

In some specific applications where one has a priori knowledge about tree metric for supports in each probability measure. One can compute *FlowAlign* as in Equation (2) in the main text, where one can exhaustedly search the optimal aligned roots (as summarized in Algorithm 3), or apply the efficient computation in Section 3.2 in the main text to reduce this complexity (as summarized in Algorithm 4).

Assume that one have priori knowledge about tree metrics[5] $\mathcal{T}_X, \mathcal{T}_Z$ for supports of probability measure μ, ν respectively. In general, one needs to search the optimal aligned roots r_x, r_z for tree $\mathcal{T}_X, \mathcal{T}_Z$. The complexity of exhausted search is $\mathcal{O}(N^2)$ where N is the number of nodes in trees. Additionally, the complexity of aligned-root

[4]One can use supports of the input probability measures, or a (sub)set of supports from several input probability measures, e.g., in case, supports are in non-registered, but same-dimensional spaces, to sample tree metrics having the same tree structure. Therefore, we have $\bar{N} \approx tn$, where t is the number of probability measures whose supports are used to sample tree metrics.

[5]Assume that for each tree metric, each node has at most κ child nodes, and the deepest level is $H_{\mathcal{T}}$.

FlowAlign is $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + n H_{\mathcal{T}} + n \log n)$, or approximately $\mathcal{O}(N \log N)$ (See Section C.2.1). Therefore, the overall complexity of *FlowAlign* with exhausted search for optimal aligned-roots is $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + N^2 n H_{\mathcal{T}} + N^2 n \log n + N^2)$[6], or approximately $\mathcal{O}(N^3 \log N)$.

Algorithm 4 Efficient computation for *FlowAlign* for probability measures with priori tree metrics (Following Section 3.2 in the main text)

```
Input: Input probability measures $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$, and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$ where $x_i \mid_{i \in [n]}$ in tree $\mathcal{T}_X$, and
    $z_j \mid_{j \in [m]}$ in tree $\mathcal{T}_Z$.
Output: FlowAlign discrepancy $d$.
 1: Choose $r_x$ as a root of tree $\mathcal{T}_X$, and compute $d_{\mathrm{T_X}}(r_x, x)$ and save path $\mathcal{P}(r_x, x)$ for all $x \in \mathcal{T}_X$.
 2: Construct $\tilde{\mu}_{r_x} \leftarrow \sum_{i \in [n]} a_i \delta_{d_{\mathrm{T_X}}(r_x, x_i)}$ with $r_x$ as a root of $\mathcal{T}_X$.
 3: Sort supports $d_{\mathrm{T_X}}(r_x, x_i) \mid_{i \in [n]}$.
 4: for each $x \in \mathcal{T}_X$ and $x \neq r_x$ do
 5:     % Change root from $r_x$ to $x$ for tree $\mathcal{T}_X$
 6:     Construct $\tilde{\mu}_x \leftarrow \sum_{i \in [n]} a_i \delta_{d_{\mathrm{T_X}}(x, x_i)}$ with $x$ as a root of $\mathcal{T}_X$ (note that $d_{\mathrm{T_X}}(x, x_i) = d_{\mathrm{T_X}}(r_x, x) + d_{\mathrm{T_X}}(r_x, x_i) -$
        $2 d_{\mathrm{T_X}}(r_x, \zeta_i)$ where $\zeta_i$ is the closet ancestor of $x$ and $x_i$ and $\zeta_i \in \mathcal{P}(r_x, x)$, $\zeta_i \in \mathcal{P}(r_x, x_i)$).
 7:     Sort supports $d_{\mathrm{T_X}}(x, x_i) \mid_{i \in [n]}$. (Option: using efficient computation in Section 3.2 in the main text to
        reduce the complexity from $\mathcal{O}(n \log n)$ into nearly $\mathcal{O}(n)$ by leveraging the sorted order at root $r_x$).
 8: end for
 9: Choose $r_z$ as a root of tree $\mathcal{T}_Z$, and compute $d_{\mathrm{T_Z}}(r_z, z)$ and save path $\mathcal{P}(r_z, z)$ for all $z \in \mathcal{T}_Z$.
10: Construct $\tilde{\nu}_{r_z} \leftarrow \sum_{j \in [m]} b_j \delta_{d_{\mathrm{T_Z}}(r_z, z_j)}$ with $r_z$ as a root of $\mathcal{T}_Z$.
11: Sort supports $d_{\mathrm{T_Z}}(r_z, z_j) \mid_{j \in [m]}$.
12: for each $z \in \mathcal{T}_Z$ and $z \neq r_z$ do
13:     % Change root from $r_z$ to $z$ for tree $\mathcal{T}_Z$
14:     Construct $\tilde{\nu}_z \leftarrow \sum_{j \in [m]} b_j \delta_{d_{\mathrm{T_Z}}(z, z_j)}$ with $z$ as a root of $\mathcal{T}_Z$ (note that $d_{\mathrm{T_Z}}(z, z_j) = d_{\mathrm{T_Z}}(r_z, z) + d_{\mathrm{T_Z}}(r_z, z_j) -$
        $2 d_{\mathrm{T_Z}}(r_z, \zeta_j)$ where $\zeta_j$ is the closet ancestor of $z$ and $z_j$ and $\zeta_j \in \mathcal{P}(r_z, z)$, $\zeta_j \in \mathcal{P}(r_z, z_i)$).
15:     Sort supports $d_{\mathrm{T_Z}}(z, z_i) \mid_{j \in [m]}$. (Option: using efficient computation in Section 3.2 in the main text to
        reduce the complexity from $\mathcal{O}(m \log m)$ into nearly $\mathcal{O}(m)$ by leveraging the sorted order at root $r_z$).
16: end for
17: Initialize $d \leftarrow \infty$.
18: for each $x \in \mathcal{T}_X$ do
19:     Retrieve $\tilde{\mu}_x$ with sorted support(s).
20:     for each $z \in \mathcal{T}_Z$ do
21:         Retrieve $\tilde{\nu}_z$ with sorted support(s).
22:         $\tilde{d} \leftarrow$ Algorithm 1 for $\tilde{\mu}_x$ and $\tilde{\nu}_z$.
23:         if $d > \tilde{d}$ then
24:             $d \leftarrow \tilde{d}$.
25:         end if
26:     end for
27: end for
```

As described in Section 3.2 in the main text, those computational steps, e.g., tree metrics between a root to each support, and sorting for those tree metrics between a root to each support or its efficient computation, can be done separately for each tree before one applies Algorithm 1 for those sorted tree metrics between a root and each support, then compares those N^2 values to find the optimal pair of roots, one can reduce the complexity of *FlowAlign*

- into $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + N n H_{\mathcal{T}} + n \log n + N^2)$, or $\mathcal{O}(N^2)$ for Case 1 in the main text, since one needs to compute tree metrics from a root to each support with complexity $\mathcal{O}(n H_{\mathcal{T}})$ for N times due to changing a root in a tree; sort tree metrics between a root to each support for only 1 time; and compare aligned-root *FlowAlign* results for N^2 cases of pairs of roots.
- or nearly into $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + N n H_{\mathcal{T}} + n \log n + N n + N^2)$, or nearly $\mathcal{O}(N^2)$ for Case 2 in the main text,

[6]Naively computing N^2 aligned-roots *FlowAlign*, and comparing those N^2 values to obtain the optimal.

since one needs to compute tree metrics from a root to each support with complexity $\mathcal{O}(nH_{\mathcal{T}})$ for N times due to changing a root in a tree; sort tree metrics between a root to each support for only 1 time; merge some ordered arrays with complexity nearly $\mathcal{O}(n)$ for N times due to changing a root in a tree; and compare aligned-root *FlowAlign* results for N^2 cases of pairs of roots.

When the degenerated case happens, one needs to merge n ordered arrays, where each array only has 1 node. Therefore, the complexity is $\mathcal{O}(n \log n)$, or one simply needs to resort for those n tree metrics from a root to each support when changing a root of a tree. Hence, the overall complexity for the degenerated case is $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + NnH_{\mathcal{T}} + Nn \log n + N^2)$, or approximately $\mathcal{O}(N^2 \log N)$.

Thus, for *FlowAlign*, one can reduce its complexity $\mathcal{O}(N^3 \log N)$ for a naive implementation (Algorithm 3) into nearly $\mathcal{O}(N^2)$ (or into $\mathcal{O}(N^2 \log N)$ for the degenerate case) with the proposed efficient computation in Section 3.2 in the main text (Algorithm 4).

Note that when one can not screen out any aligned-root *FlowAlign*s, the computation of *FlowAlign* requires at least N^2 comparisons among aligned-root *FlowAlign* (choosing the optimal pair of roots from the values of N^2 aligned-root *FlowAlign*). Therefore, for this case, $\mathcal{O}(N^2)$ is also the optimal complexity for *FlowAlign*.

C.3 *DepthAlign*

Similar to *FlowAlign* in Section C.2, there are two types of applications: *without* or *with* priori knowledge about tree metrics for supports in probability measures. For general applications where one usually does not have priori knowledge about tree metrics for probability measures, one can apply clustering-based tree metric sampling [6] to sample tree metrics for supports of probability measures. For some specific applications where one knows tree metric for each probability measure, one needs to search the optimal aligned roots, e.g., by exhausted search.

C.3.1 Applications *without* priori knowledge about tree metrics for supports in probability measures

For those general applications without priori knowledge about tree metrics for supports in probability measures, one can use clustering-based tree metric approach to sample tree metrics for supports of the probability measures.

One can compute *DepthAlign* between $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$ and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$ as follow:

- Step 1: Sample aligned-root tree metrics $\mathcal{T}_X$ and $\mathcal{T}_Z$ for supports $x_i \mid_{i \in [n]}$ and $z_j \mid_{j \in [m]}$ in probability measures μ and ν respectively (similar to Step 1 for *FlowAlign*).

- Step 2: Based on the sampled aligned-root tree metrics, *DepthAlign* (Equation (7) in the main text) is equivalent to aligned-root *DepthAlign* (Equation (6) in the main text). For each probability measure, we construct 2-depth-level tree for all nodes from the tree root to each support of the probability measure as in Algorithm 5[7].

Algorithm 5 Construct 2-depth-level tree

```
Input: Input empirical measure $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$, tree metric $\mathcal{T}_X$.
Output: Set of 2-depth-level trees for $\mu$.
1: Construct a set of paths $S_{\tilde{p}}$ for supports $x_i \mid_{i \in [n]}$ where each element is a path from a root to each support.
2: From the set of paths $S_{\tilde{n}}$, construct a set of nodes where each node belongs to at least one path of the set of paths $S_{\tilde{p}}$.
3: For each node in $S_{\tilde{n}}$, construct a 2-depth-level tree for that node in tree $\mathcal{T}$ for $\mu$, as in Section 4 in the main text.
4: Gather all those 2-depth-level trees to form the set of 2-depth-level trees for $\mu$.
```

[7] Constructing 2-depth-level tree for all nodes from the tree root to each support of probability measures as in Algorithm 5 can be considered as a preprocessing step since those 2-depth-level trees are needed during the hierarchical alignment along each deep level in trees for a computation of *DepthAlign*.

Algorithm 6 *DepthAlign* for probability measures by sampling aligned-root tree metrics

```
Input: Input probability measures $\mu = \sum_{i \in [n]} a_i \delta_{x_i}$, and $\nu = \sum_{j \in [m]} b_j \delta_{z_j}$.
Output: DepthAlign discrepancy $d$.
 1: Sample aligned-root tree metrics $\mathcal{T}_X, \mathcal{T}_Z$ for $\mu, \nu$ respectively, e.g., by choosing a mean of support data as its
    root for the clustering-based tree metric sampling [6].
 2: Construct $S_{\mathcal{T}_X}, S_{\mathcal{T}_Z}$: sets of 2-depth-level trees for $\mu, \nu$ in tree $\mathcal{T}_X, \mathcal{T}_Z$ respectively by using Algorithm 5.
 3: Initialization with a pair of roots $(r_x, r_z)$ of tree $\mathcal{T}_X, \mathcal{T}_Z$ respectively, and the optimal matching mass
    $T^*(r_x, r_z) = 1$, and $d \leftarrow 0$.
 4: Push $\{(r_x, r_z), T^*(r_x, r_z)\}$ into a queue Q.
 5: while Q is not empty do
 6:     Pull $(x, z), T^*(x, z)$ from queue Q.
 7:     Get corresponding 2-depth-level trees: $\mathcal{T}_x^2, \mathcal{T}_z^2$ for $\mu, \nu$ from $S_{\mathcal{T}_X}, S_{\mathcal{T}_Z}$ respectively.
 8:     % For two simple 2-depth-level trees, the discrepancy between $\mu_{\mathcal{T}_x^2}, \nu_{\mathcal{T}_z^2}$ is equal to 0.
 9:     if One of two 2-depth-level trees $\mathcal{T}_x^2, \mathcal{T}_z^2$ is simple, but not both of them then
10:         if $\mathcal{T}_x^2$ is simple then
11:             Get all paths from $x$ to each support of $\mu$ in the subtree of $\mathcal{T}_X$ rooted at $x$.
12:             Normalize for weights of those supports of corresponding paths.
13:             Compute $\tilde{d}$ as a sum of weighted lengths for those paths.
14:             $d \leftarrow d + T^*(x, z)\tilde{d}$.
15:         else
16:             Get all paths from $z$ to each support of $\nu$ in the subtree of $\mathcal{T}_Z$ rooted at $z$.
17:             Normalize for weights of those supports of corresponding paths.
18:             Compute $\tilde{d}$ as a sum of weighted lengths for those paths.
19:             $d \leftarrow d + T^*(x, z)\tilde{d}$.
20:         end if
21:     else if Two 2-depth-level trees $\mathcal{T}_x^2, \mathcal{T}_z^2$ are not simple then
22:         Sort the distances from a root to each node in tree $\mathcal{T}_x^2, \mathcal{T}_z^2$.
23:         Compute univariate OT $\tilde{d}$ and the optimal transportation plan $\tilde{T}^*$ for empirical measures with sorted
            supports $\mu_{\mathcal{T}_x^2}, \nu_{\mathcal{T}_z^2}$ by using Algorithm 1.
24:         $d \leftarrow d + T^*(x, z)\tilde{d}$.
25:         Compute weighted optimal transport plan $\tilde{T}^* \leftarrow T^*(x, z)\tilde{T}^*$.
26:         For all child nodes $u, v$ of $\mathcal{T}_x^2, \mathcal{T}_z^2$ respectively, if their optimal matching mass $\tilde{T}^*(u, v) > 0$, push
            $\{(u, v), \tilde{T}^*(u, v)\}$ into queue Q.
27:     end if
28: end while
```

- Step 3: Compute the aligned-root *DepthAlign* (Equation (6) in the main text). It starts from a comparison between 2-depth-level tree constructed from each root of $\mathcal{T}_X$, and $\mathcal{T}_Z$ for μ and ν respectively, with optimal matching mass 1.
    - If it is *not* a simple case[8], then we compute the disrepancy between 2-depth-level tree as aligned-root *FlowAlign* by simply sorting supports and using Algorithm 1. Then, we push all the matching pairs between child nodes and their optimal matching mass into the queue.
    - If it is a simple case where both two nodes of the considered pair do not have child nodes, or sum of their child-node weights is equal to 0, then their discrepancy is equal to 0.
    - If it is a simple case where one of two considered nodes, but not both of them, does not have child nodes, or sum of its child-node weights is equal to 0, then their discrepancy is equal to sum of normalized-weighted lengths of paths from that node to supports of a corresponding measure which are in the subtree rooted at that node.

  We stop the computation when the queue is empty. The aligned-root *DepthAlign* is equal to sum of all weighted discrepancies between 2-depth-level trees.

[8]A simple case for a pair of considered nodes is defined as: at least one node of the considered pair does not have child nodes, or sum of its child-node weights is equal to 0.

We summarize the computation for *DepthAlign* by sampling aligned-root tree metrics in Algorithm 6.

We next give a complexity analysis for *DepthAlign*:

- Recall that the complexity of sampling tree metric $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa)$ where $H_{\mathcal{T}}$ is a predefined deepest level of tree $\mathcal{T}$ and κ is the number of clusters in the farthest-point clustering for the clustering-based tree metric sampling [6]; $\bar{N}$ is the input number of supports.
- The complexity of constructing 2-depth-level trees is $\mathcal{O}(n H_{\mathcal{T}} \kappa)$ (we have n supports, each path from a root to a support has less than $H_{\mathcal{T}}$ nodes, and each 2-depth-level tree has less than or equal $(\kappa + 1)$ nodes).
- The complexity to compute the univariate OT between 2-depth-level trees is $\mathcal{O}(\kappa \log \kappa)$.
- At deep level $(h+1)$, the number of nodes is not more than κ^h. So, the number of pairs of nodes at deep level $(h+1)$ is not more than κ^{2h}. Let Δ be the number of comparisons Δ for 2-depth-level trees, we have $\Delta \leq \left(\kappa^{2H_{\mathcal{T}}} - 1\right) / \left(\kappa^2 - 1\right)$.

Therefore, one can implement the computation of *DepthAlign* with a complexity $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + n H_{\mathcal{T}} \kappa + \Delta \kappa \log \kappa)$.

C.3.2 Applications *with* priori knowledge about tree metrics for supports in probability measures

For some specific applications where one has a priori knowledge about tree metric for supports in each probability measures, one can easily tailor Algorithm 6 with existing tree metrics to compute aligned-root *DepthAlign*. Thus, for the *DepthAlign*, one needs to search the optimal pair of roots for the given tree metrics[9], where one uses the aligned-root *DepthAlign* for each pair of roots. Overall, the complexity of *DepthAlign* with exhausted search for the optimal pair of roots is approximately $\mathcal{O}(N^2 n H_{\mathcal{T}} \kappa + N^2 \Delta \kappa \log \kappa)$.

D Further experimental results

We denote EGW_0 for the standard entropic Gromov-Wasserstein where we use entropic regularization for both optimizing the transport plan and computing entropic GW.

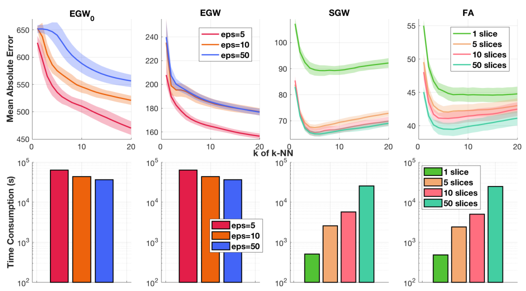
D.1 Further experimental results on quantum chemistry and document classification

We illustrate the trade-off between performances and time consumption for the discrepancies for probability measures in different spaces when their parameters are changed, e.g., entropic regularization in EGW and EGW_0, and the number of (tree) slices in SGW, *FlowAlign* (FA), and *DepthAlign* (DA)

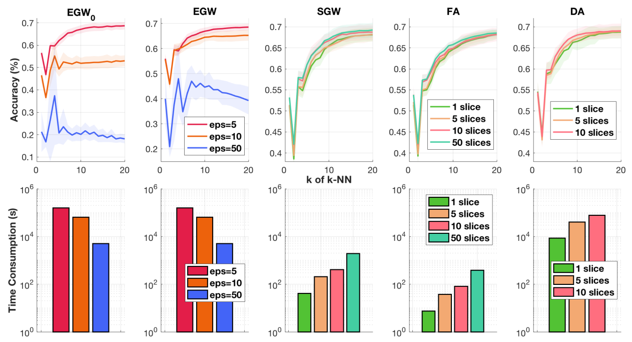
- for quantum chemistry (`qm7` dataset) in Figure 2a,
- for document classification
  - in `TWITTER` dataset in Figure 2b,
  - in `RECIPE` dataset in Figure 2c,
  - in `CLASSIC` dataset in Figure 2d,
  - in `AMAZON` dataset in Figure 2e.

The entropic term in standard entropic GW (EGW_0) may harm its performances (comparing with EGW) (e.g., in `qm7, RECIPE, CLASSIC` datasets illustrated in Figure 2a, Figure 2c, Figure 2d respectively). Performances of EGW and the standard EGW_0 are improved when entropic regularization (eps) is smaller, but their computational time is considerably increased.

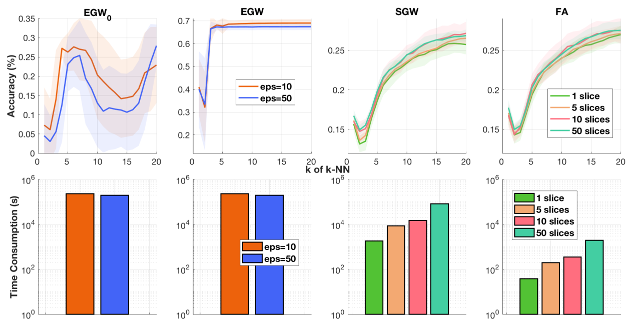
[9] Assume that for each tree metric, each node has at most κ child nodes, the tree deep is $H_{\mathcal{T}}$, and the number of nodes in tree is about N.



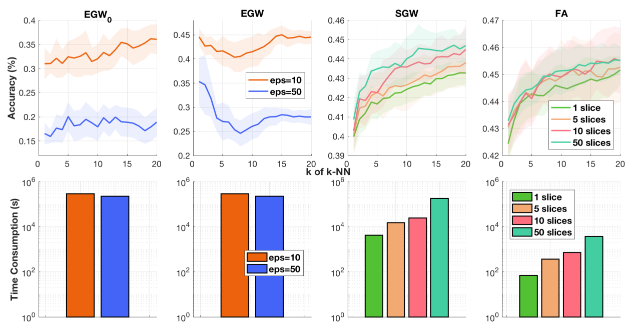
(a) In `qm7` dataset.



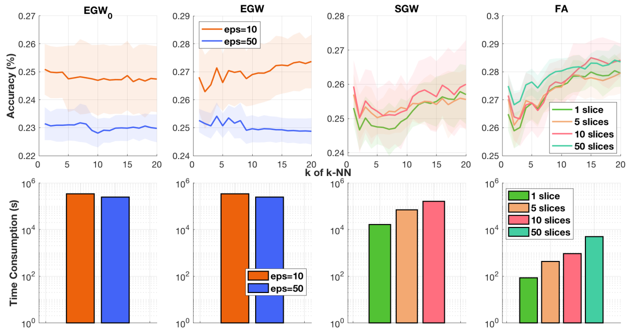
(b) In `TWITTER` dataset.



(c) In `RECIPE` dataset.



(d) In `CLASSIC` dataset.



(e) In `AMAZON` dataset.

Figure 2: In (a), results of MAE and time consumption for the discrepancies with different parameters (e.g. entropic regularization in EGW_0/EGW, and number of (tree) slices in SGW/FA) in k-NN regression in `qm7` dataset. For clustering-based tree metric approach, we used its suggested parameters ($\kappa = 4$, $H_{\mathcal{T}} = 6$). In (b, c, d, e), results of averaged accuracy and time consumption for the discrepancies with different parameters, e.g., entropic regularization in EGW_0/EGW, and the number of (tree) slices in SGW/FA in k-NN in `TWITTER, RECIPE, CLASSIC, AMAZON` datasets respectively. For clustering-based tree metric approach, we used its suggested parameters ($\kappa = 4$, $H_{\mathcal{T}} = 6$).

D.2 Time consumption for the clustering-based tree metric sampling

Time consumption for tree metric sampling by the clustering-based tree metric method [6] is negligible in computation for both *FlowAlign* and *DepthAlign*. Indeed, we illustrate time consumption for tree metric sampling with different parameters, e.g., the predefined deepest level $H_{\mathcal{T}}$, the number of clusters κ, for the clustering-based tree metric sampling [6] in `qm7, TWITTER, RECIPE, CLASSIC, AMAZON` datasets in Figure 3. For examples, for each tree metric sampling with the suggested parameters ($H_{\mathcal{T}} = 6, \kappa = 4$), it only took about 0.4 seconds for `qm7` dataset, 1.5 seconds for `TWITTER` dataset, 11.0 seconds for `RECIPE` dataset, 17.5 seconds for `CLASSIC` dataset, and 20.5 seconds for `AMAZON` dataset. Furthermore, we give a brief review for the clustering-based tree metric sampling in Section E.2.

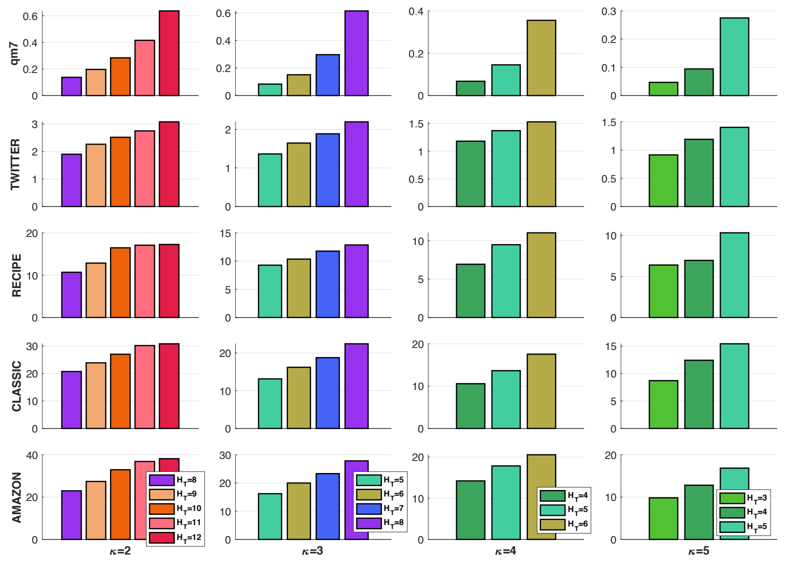


Figure 3: Time consumption (seconds) for a tree metric sampling in *FlowAlign* and *DepthAlign* by the clustering-based tree metric method [6] with different parameters (e.g. the predefined deepest level $H_{\mathcal{T}}$, the number of clusters κ) in quantum chemistry (`qm7` dataset), and document classification (`TWITTER, RECIPE, CLASSIC, AMAZON` datasets).

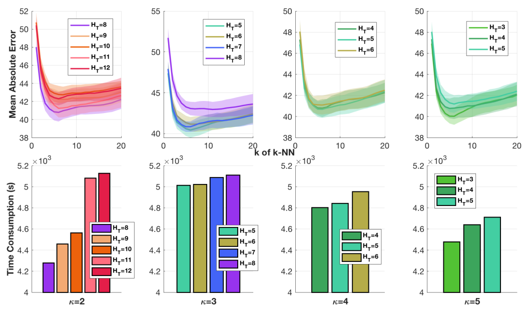
D.3 Experiment results with different parameters for tree metric sampling

We illustrate results of mean absolute error (MAE) and time consumption for *FlowAlign* (10 tree slices) with different parameters, e.g., the predefined deepest level $H_{\mathcal{T}}$, the number of clusters κ, in the clustering-based tree metric sampling:

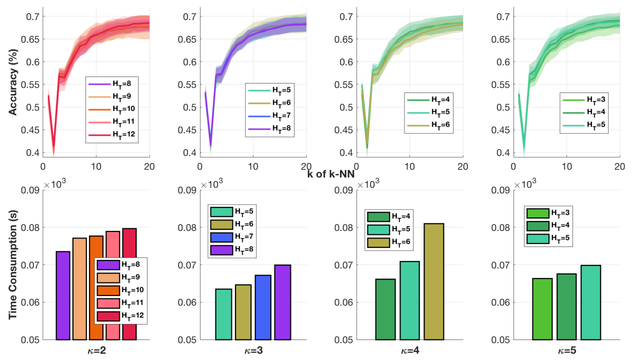
- in `qm7` dataset in Figure 4a,
- in `TWITTER` dataset in Figure 4b,
- in `RECIPE` dataset in Figure 4c,
- in `CLASSIC` dataset in Figure 4d,
- in `AMAZON` dataset in Figure 4e.

D.4 Further experimental results: k-means clustering on a small experimental setup for performance comparison on random rotated `MNIST` dataset

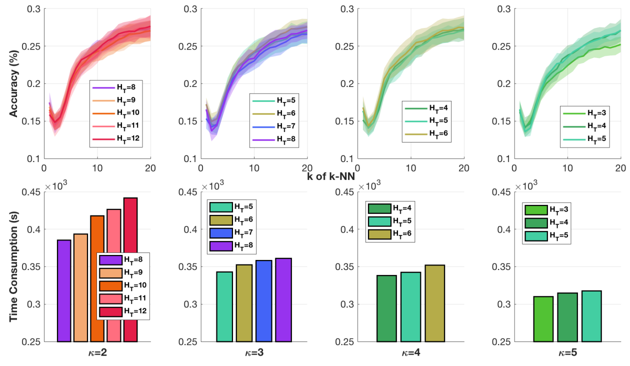
We follow the small experimental setup for performance comparison on *random rotated* `MNIST` dataset as in [9]. We randomly select 50 point clouds from each digits 0 to 4, apply k-means clustering with $k = 5$, and k-means++



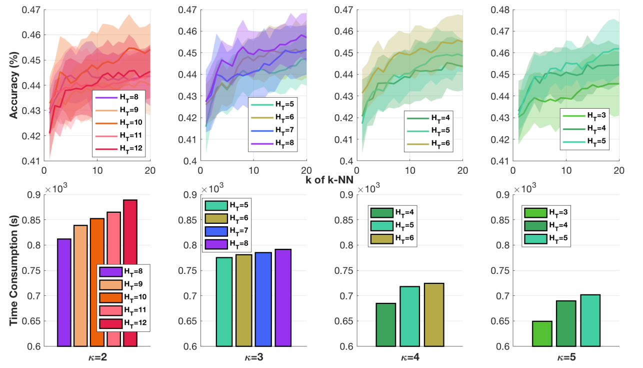
(a) In `qm7` dataset.



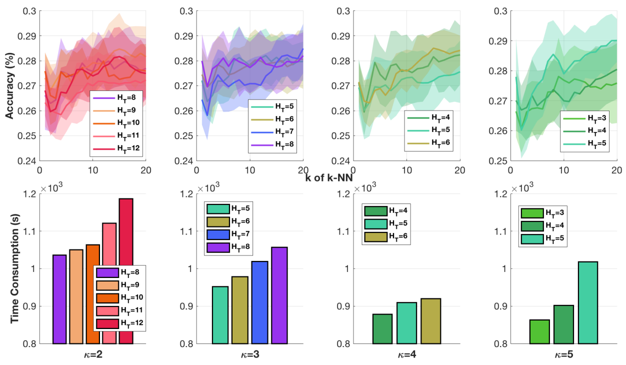
(b) In `TWITTER` dataset.



(c) In `RECIPE` dataset.



(d) In `CLASSIC` dataset.



(e) In `AMAZON` dataset.

Figure 4: Results of mean absolute error and time consumption for *FlowAlign* (10 tree slices) with different parameters (e.g. the predefined deepest level $H_{\mathcal{T}}$, the number of clusters κ) in the clustering-based tree metric sampling.

initialization. We show the performance comparison for k-means clustering in Figure 5[10]. The performances of *FlowAlign* are comparative with EGW. Moreover, *FlowAlign* is several order faster than EGW. The performances of EGW are better when entropic regularization (eps) is smaller, but the time consumption is also higher.

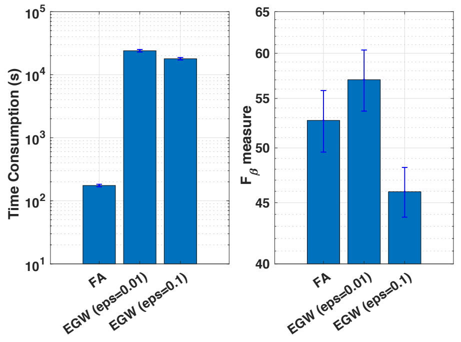


Figure 5: Results of k-means clustering on a small experiment setup for performance comparison on *random rotated* `MNIST` dataset.

E Some brief reviews

We give brief reviews for the farthest-point clustering [3] (a more detail summarization and discussion can be seen in [6]), the clustering-based tree metric sampling [6], tree metric in [12], and F_β measure for clustering evaluation [8] where β is chosen as in [4].

E.1 The farthest-point clustering

The farthest-point clustering [3] is a simple fast greedy approach for a κ-center problem. The κ-center problem is defined as finding a partition of n points into κ clusters to minimize the maximum radius of clusters. The complexity of a direct implementation, e.g., Algorithm 7, is $\mathcal{O}(n\kappa)$. Moreover, by using the algorithm in [2][11], the complexity for the farthest-point clustering can be reduced into $\mathcal{O}(n \log \kappa)$.

Algorithm 7 The farthest point clustering

```
Input: $X = \left\{x_i \mid_{i \in [n]}\right\}$ is a set of $n$ input data points, and $\kappa$ is the predefined number of clusters for the
    farthest-point clustering.
Output: A set of clustering centers $C = \left\{c_i \mid_{i \in [\kappa]}\right\}$, and cluster indices for $x_i \mid_{i \in [n]}$.
 1: Initialize $C \leftarrow \varnothing$.
 2: $c_1 \leftarrow$ a random data point $x \in X$.
 3: $C \leftarrow c_1$.
 4: $i \leftarrow 1$.
 5: while $i < \kappa$ and $n - i > 0$ do
 6:     $i \leftarrow i + 1$.
 7:     $c_i \leftarrow \max_{x \in X} \min_{c \in C} \|x - c\|$.          % (find the farthest point $x \in X$ to $C$).
 8:     $C \leftarrow C \cup c_i$.                                  % (add the new cluster center into $C$).
 9: end while
10: Each data point $x \in X$ is assigned to its nearest cluster center $c \in C$.
```

E.2 Clustering-based tree metric sampling

The clustering-based tree metric sampling [6] is a practical fast approach to sample tree metric from input data points. Its main idea is to use a (fast) clustering method, e.g., the farthest-point clustering, to cluster input data points hierarchically to build a tree structure, as summarized in Algorithm 8[12]. As discussed in [6], one can use any clustering method for the clustering-based tree metric sampling. The farthest-point clustering is suggested

[10]The barycenter for SGW is not published yet.
[11]Code is available at https://github.com/vmorariu/figtree/blob/master/matlab/figtreeKCenterClustering.m
[12]Code is available at https://github.com/lttam/TreeWasserstein/blob/master/BuildTreeMetric_HighDim_V2.m

due to its fast computation (see Section E.1). The complexity of the clustering-based tree metric sampling for n input data points where one uses the same number of clusters κ for the farthest-point clustering and $H_{\mathcal{T}}$ for the predefined deepest level of tree $\mathcal{T}$, is $\mathcal{O}(nH_{\mathcal{T}} \log \kappa)$. Therefore, the clustering-based tree metric sampling is very fast for applications.

A cluster sensitivity problem. As discussed in [6], for data points near a border of adjacent, but different clusters, they are close to each other but in different clusters. The fact that whether those data points are clustered in the same cluster or not, depends on an initialization of the farthest-point clustering. Therefore, by leveraging various clustering results, obtained with different initializations for the farthest-point clustering, e.g. as in our proposed flow-based alignment approaches: *FlowAlign* and *DepthAlign*, one can reduce an affect of the cluster sensitivity problem.

Algorithm 8 Clustering-based tree metric (with the farthest-point clustering)

```
Input: $X$ is a set of $m$ input data points, $\tilde{x}_p$ is a parent node for those input data points in $X$, $h$ is a current
    depth level, $H_{\mathcal{T}}$ is the predefined deepest level of tree $\mathcal{T}$, $\kappa$ is the predefined number of clusters for the
    farthest-point clustering.
Output: tree metric $\mathcal{T}$
 1: if $m > 0$ then
 2:   if $h > 0$ then
 3:     Node $\tilde{x}_c \leftarrow$ a center of $X$, e.g., the mean data point of $X$.
 4:     Length of edge $(\tilde{x}_p, \tilde{x}_c) \leftarrow$ distance $(\tilde{x}_p, \tilde{x}_c)$.
 5:   else
 6:     Node $\tilde{x}_c \leftarrow \tilde{x}_p$.
 7:   end if
 8:   if $m > 1$ and $h < H_{\mathcal{T}}$ then
 9:     Run the farthest-point clustering for $X$ into $\kappa$ clusters $X_i \mid_{i \in [\kappa]}$.
10:     for each cluster $X_i \mid_{i \in [\kappa]}$ do
11:       Recursive the clustering-based tree metric for input data points in set $X_i$, a parent node $\tilde{x}_c$, a current
            depth level $(h + 1)$ with the predefined deepest level $H_{\mathcal{T}}$ for tree $\mathcal{T}$, and the predefined number of
            clusters $\kappa$ for the farthest-point clustering).
12:     end for
13:   end if
14: end if
```

E.3 Tree metric

We recall the definition of tree metric in [12] (§7, p.145–182).

Definition 1. *A metric $d : \Omega \times \Omega \rightarrow \mathbb{R}_+$ is a tree metric on a finite set Ω if there exists a tree $\mathcal{T}$ with non-negative edge lengths such that all elements of Ω are nodes in $\mathcal{T}$, and for $x, z \in \Omega$, $d(x, z)$ equals to the length of the (unique) path in $\mathcal{T}$ between x and z.*

E.4 F_β measure for clustering evaluation

We summarize the F_β measure for clustering evaluation as in [8] where β is chosen as in [4]. The main idea is that a pair of data points is assigned to the same cluster if and only if they are in the same class and otherwise. We have some following quantities:

- TP: the number of a true positive decisions which assign a pair of data points in the same class to the same cluster.
- TN: the number of a true negative decisions which assign a pair of data points in the different classes to the different clusters.
- FP: the number of a false positive decisions which assign a pair of data points of different classes to the same cluster.

- FN: the number of a false negative decisions which assign a pair of data points of the same class to different clusters.

Consequently, we have the precision

$$\mathbf{P} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{8}$$

and recall

$$\mathbf{R} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{9}$$

Note that we usually have many more pairs of data points in different classes than in the same class in clustering. Therefore, we need to penalize false negative error more strongly than false positive error. $\mathbf{F}_\beta$ measure can take into account of this idea by using a scalar $\beta > 1$, defined as follow:

$$\mathbf{F}_\beta = \frac{\left(\beta^2 + 1\right) \mathbf{P}\mathbf{R}}{\beta^2 \mathbf{P} + \mathbf{R}}. \tag{10}$$

Following [4], we plug Equation (8), and Equation (9) into Equation (10), and observe that $\mathbf{F}_\beta$ penalizes false negative error β^2 times more than false positive error. Then, we can set

$$\beta = \sqrt{\frac{|\mathtt{D}|}{|\mathtt{S}|}}, \tag{11}$$

where $|\cdot|$ denotes a cardinality of a set let; D, S are sets of pairs of data points in different and same classes respectively.

E.5 More information about datasets

Quantum chemistry. One can download qm7 dataset from: http://quantum-machine.org/datasets/. We emphasize that for simplicity, we only use the Cartesian coordinate of each atom ($\mathbb{R}^3$) in the molecules. We do *not* use the atomic nuclear charge for each molecule for the atomization energy prediction task as used in experiments of [9, 10].

There are 7165 molecules and each molecule has no more than 23 atoms in qm7 dataset.

Document classification with non-registered word embeddings. One can download document datasets, e.g., TWITTER, RECIPE, CLASSIC, AMAZON datasets from: https://github.com/mkusner/wmd.

After preprocessing, there are

- 3108 documents in 3 classes where each document length is not more than 29 in TWITTER dataset,
- 4370 documents in 15 classes where each document length is not more than 628 in RECIPE dataset,
- 7093 documents in 4 classes where each document length is not more than 348 in CLASSIC dataset,
- 8000 documents in 4 classes where each document length is not more than 4592 in AMAZON dataset.

F Some further discussions

Further discussions about results on RECIPE dataset. The proposed methods and SGW do not work well as compared to EGW in RECIPE dataset in Figure 5 in the main text. A possible reason is that the number of classes in RECIPE is larger than other documents datasets (TWITTER, CLASSIC, AMAZON). While RECIPE has 15 classes, other document datasets have only 3 or 4 classes. Additionally, another potential reason is that RECIPE is very imbalanced among its classes. Some classes have many samples (e.g., class ID9 has 1299 samples, class ID15 has 972 samples) while some other classes has much less samples (e.g., the number of samples of class ID1, class ID8, class ID10 or class ID12 is less than 40). Moreover, note that as illustrated in Figure 2c, performances of standard entropic GW (denoted EGW_0) are only comparative with other approaches in the `RECIPE` dataset; we also observed that the entropic regularization term (eps) in EGW may be relatively small enough for `RECIPE` since reducing eps (from 50 to 10) just slightly changes performances of EGW. It is totally different to the cases for EGW in other document datasets. Furthermore, as discussed in §6, the quality of EGW is better when the entropic regularization term (eps) becomes smaller, but the computation of EGW is considerably slower.

Network flow. In combinatorial optimization, network flow is a class of computational problems in which the input is a graph with capacities on its edges [1]. The minimum-cost flow problem is one of popular classes of network flow problems. Especially, optimal transport (OT) for probability measures whose supports are in the same space, can be regarded as one of instances of the minimum-cost flow problems, and one can use the network simplex algorithm to solve it. However, for GW, the supports of input probability measures are in different spaces. Therefore, one may not use algorithms for minimum-cost flow problems, e.g., network simplex, to optimize the alignment in GW problem with tree metrics (Equation (1) in the main text) where supports of input probability measures are in different tree metric spaces.

Recall that our proposed flow-based alignment approaches (i.e., *FlowAlign* and *DepthAlign*) for probability measures in different tree metric spaces is based on matching both *flows* from a root to each support in the probability measure, and root alignment for the corresponding tree structures. Thus, one should distinguish between our proposed flow-based alignment approaches in *FlowAlign* and *DepthAlign* for probability measures in different tree metric spaces, and algorithms for minimum-cost flow problems. Note that the flows of our flow-based representation shares the same spirit with the flows modeled in the proof for the closed-form computation of tree-Wasserstein distance [6] (§3).

Tree metric sampling. Our goal is ***not*** to approximate the GW distance between probability measures whose supports are in the Euclidean space (i.e., the ground metric is Euclidean metric), but rather to sample tree metrics for each space of supports, and then use those random sampled tree metrics as ground metrics for supports of input probability measures in GW, similar to tree-sliced-Wasserstein [6].

Similar to the case of one-dimensional projections for sliced Wasserstein, or sliced GW, which do not give good properties from a distortion point of view, but remain useful for sliced Wasserstein or sliced GW in applications, we believe that tree metrics with a large distortion can be *useful*, similar to the case of tree-sliced-Wasserstein in practical applications.

A correction of the binomial expansion trick in [13]. There is a typo in the binomial expansion trick in [13]. We correct it as follows:

$$\begin{aligned}\sum_{i,j}\left((x_i - x_j)^2 - (y_{\sigma_i} - y_{\sigma_j})^2\right)^2 = {} & 2n\left(\sum_i x_i^4\right) - 8\left(\sum_i x_i^3\right)\left(\sum_i x_i\right) + 6\left(\sum_i x_i^2\right)^2 \\ & + 2n\left(\sum_i y_i^4\right) - 8\left(\sum_i y_i^3\right)\left(\sum_i y_i\right) + 6\left(\sum_i y_i^2\right)^2 \\ & - 4\left(\sum_i x_i^2\right)\left(\sum_i y_i^2\right) - 4n\left(\sum_i x_i^2 y_{\sigma_i}^2\right) \\ & + 8\left(\sum_i x_i\right)\left(\sum_i x_i y_{\sigma_i}^2\right) + 8\left(\sum_i y_i\right)\left(\sum_i x_i^2 y_{\sigma_i}\right) \\ & - 8\left(\sum_i x_i y_{\sigma_i}\right)^2. \end{aligned} \tag{12}$$

The σ in Equation (12) is a permutation. Note that the binomial expansion trick can be applied for GW when one uses the squared ℓ_2 loss and input probability measures have the same number of supports with uniform weights as considered in sliced GW [13].

G Empirical relation for discrepancies for probability measures in different spaces

We emphasize that the proposed *FlowAlign* and *DepthAlign* are two ***novel*** discrepancies for probability measures in different tree metric spaces, and we do not try to mimic or approximate either the entropic GW or sliced GW.

In this section, we investigate an empirical relation between a pair of discrepancies, e.g., let denote those considered discrepancies as d_α and d_β. We carried out following experiments[13]:

For a query point q, we denote q_{NN} as the nearest neighbor of q with respect to d_α. Then, we investigate the frequency of rank order of q_{NN} among nearest neighbor of q with respect to d_β. For those experiments, we randomly split 90%/10% for training and test. Reported results are averaged over 1000 runs.

We recall some following notations: FA for *FlowAlign*, DA for *DepthAlign*, SGW for sliced GW, EGW for entropic GW (only use entropic regularization for transport plan optimization, but exclude for computing entropic GW objective) and EGW_0 for standard entropic GW (use entropic regularization for both transport plan optimization and objective computation). We also recall that supports for probability measures are in low-dimensional spaces (dim=3) in `qm7` dataset; and in high-dimensional spaces (dim=300) in `TWITTER, RECIPE, CLASSIC, AMAZON` datasets. The number of supports for probability measures are small in `qm7` (#supports ≤ 23), and `TWITTER` (#supports ≤ 29) datasets; and are large in `RECIPE` (#supports ≤ 628), `CLASSIC` (#supports ≤ 348), and `AMAZON` (#supports ≤ 4592) datasets.

G.1 Empirical relation between FA and DA

We set $d_\alpha :=$ FA and $d_\beta :=$ DA. Figure 6 illustrates an empirical relation between FA and DA in `qm7, TWITTER, RECIPE, CLASSIC, AMAZON` datasets. We used DA with 1 tree slice for `qm7`, 10 tree slices for `TWITTER`, 5 slices for `RECIPE`, 1 tree slice for `CLASSIC`, and 1 tree slice for `AMAZON`.

The empirical results show that FA agrees with some aspects of DA, especially when the number of support for probability measures is small (information about relative deep levels of supports is small), and the degree of agreement may increase when supports for probability measures are low-dimensional space (tree structure becomes simpler) as in `qm7`. From Figure 6, we also observe that the degree of agreement decreases when the number of supports in datasets increases.

Recall that DA is a generalized version of FA which takes into account deep levels of supports in tree structures of tree metric spaces. When the number of supports for probability measures is large, the information about relative deep levels of supports is increased. Therefore, DA operates differently to FA, e.g., in `RECIPE` and `AMAZON` datasets. In addition, tree structure for high-dimensional spaces of supports (e.g., in `TWITTER` and `CLASSIC` datasets) is usually more complex than that of low-dimensional space of supports (e.g., in `qm7` dataset). Thus, DA behaves more similar to FA in `qm7` dataset than in `TWITTER` and `CLASSIC` datasets.

G.2 Empirical relation between FA and SGW

We first set $d_\alpha :=$ SGW and $d_\beta :=$ FA (10 tree slices). For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices. We illustrate an empirical relation between FA and SGW in `qm7, TWITTER, RECIPE, CLASSIC, AMAZON` datasets in Figure 7.

Secondly, we set $d_\alpha :=$ FA and $d_\beta :=$ SGW (10 slices). We illustrate another empirical relation between FA and SGW in `qm7, TWITTER, RECIPE, CLASSIC, AMAZON` datasets in Figure 8.

The empirical results show that SGW and FA may agree with each other some aspects when supports are in low-dimensional spaces, e.g., in `qm7` dataset, but they become more different when supports are in high-dimensional spaces, e.g., in document datasets: `TWITTER, RECIPE, CLASSIC, AMAZON` datasets. Note that a one-dimensional space is a special case of tree metric (a tree metric is a chain). For supports in low-dimensional spaces, both projecting those supports in one-dimensional spaces and using tree metric sampling seem to be able to capture the structure of a distribution of supports at a certain level. However, for supports in high-dimensional spaces, projecting the supports in one-dimensional space limits its capacity to capture the structure of a distribution of supports [7] while sampling tree metric can remedy this problem [6].

[13]The experimental setup is similar to that of [6] for investigating an empirical relation between tree-sliced-Wasserstein and optimal transport with Euclidean ground metric.

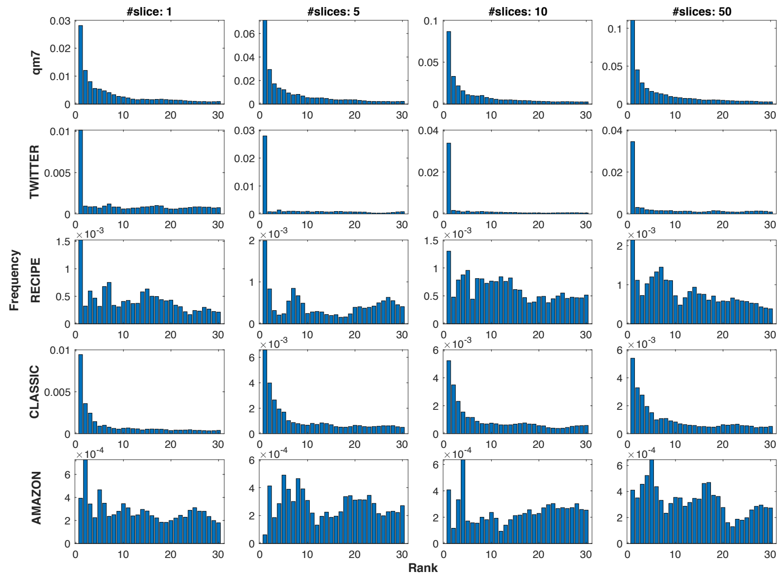


Figure 6: Empirical relation: $d_\alpha :=$ FA and $d_\beta :=$ DA. We used DA with 1 tree slice for `qm7`, 10 tree slices for `TWITTER`, 5 tree slices for `RECIPE`, 1 tree slice for `CLASSIC`, and 1 tree slice for `AMAZON`.

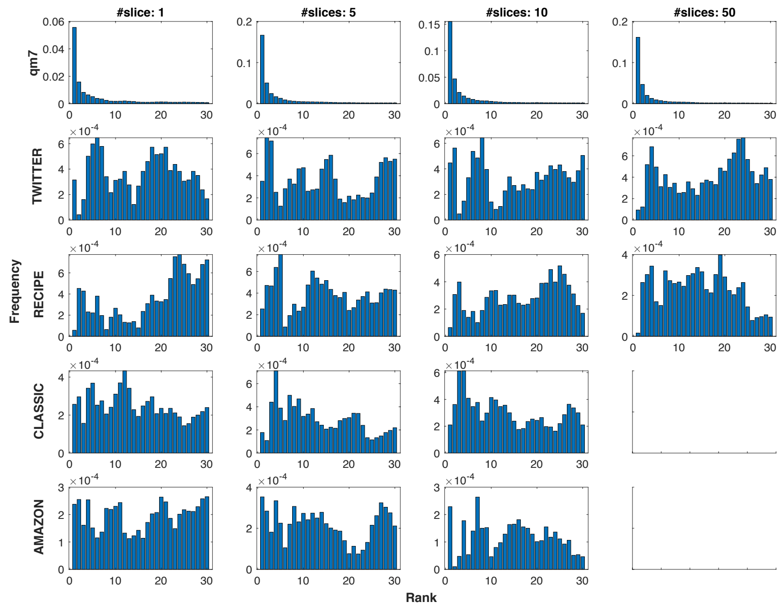


Figure 7: Empirical relation: $d_\alpha :=$ SGW and $d_\beta :=$ FA (10 tree slices). For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

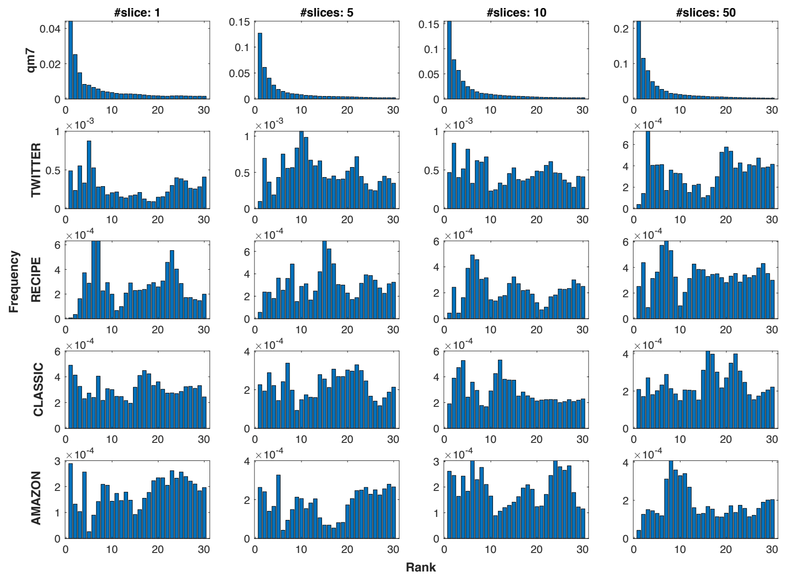


Figure 8: Empirical relation: $d_\alpha :=$ FA and $d_\beta :=$ SGW (10 slices).

G.3 Some other empirical relations

G.3.1 Empirical relation between SGW and DA

We set $d_\alpha :=$ SGW and $d_\beta :=$ DA. Figure 9 illustrates an empirical relation between SGW and DA in `qm7`, `TWITTER`, `RECIPE`, `CLASSIC`, `AMAZON` datasets. We used DA with 1 tree slice for `qm7`, 10 tree slices for `TWITTER`, 5 tree slices for `RECIPE`, 1 tree slice for `CLASSIC`, and 1 tree slice for `AMAZON`. For SGW in `CLASSIC`, `AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

The empirical results show that SGW agrees with some aspects of DA when supports for probability measures are in a low-dimensional space (tree structure becomes simpler), as in `qm7` dataset. When supports for probability measure are in a low-dimensional space (e.g., in `qm7` dataset), both SGW and DA agree with some aspect of FA (see more discussions in Section G.1, and Section G.2), or SGW and DA agrees with each other some aspects. However, when supports for probability measure are in high-dimensional spaces (e.g., in document datasets: `TWITTER`, `RECIPE`, `CLASSIC`, `AMAZON` datasets), they become different (similar to the empirical relation between SGW and FA as discussed in Section G.2).

G.3.2 Empirical relation between FA/SGW and EGW$_0$/EGW

For those experiments, the entropic regularization for EGW$_0$/EGW is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE`, `CLASSIC`, `AMAZON` datasets. For SGW in `CLASSIC`, `AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

Empirical relation between FA/SGW and EGW$_0$. We first set $d_\alpha :=$ FA and $d_\beta :=$ EGW$_0$. We illustrate an empirical relation between FA and EGW$_0$ in `qm7`, `TWITTER`, `RECIPE`, `CLASSIC`, `AMAZON` datasets in Figure 10.

Secondly, we set $d_\alpha :=$ SGW and $d_\beta :=$ EGW$_0$. We illustrate an empirical relation between SGW and EGW$_0$ in `qm7`, `TWITTER`, `RECIPE`, `CLASSIC`, `AMAZON` datasets in Figure 11.

Empirical relation between FA/SGW and EGW. We first set $d_\alpha :=$ FA and $d_\beta :=$ EGW. We illustrate an empirical relation between FA and EGW in `qm7`, `TWITTER`, `RECIPE`, `CLASSIC`, `AMAZON` datasets in Figure 12.

Secondly, we set $d_\alpha :=$ SGW and $d_\beta :=$ EGW. We illustrate an empirical relation between SGW and EGW in `qm7`, `TWITTER`, `RECIPE`, `CLASSIC`, `AMAZON` datasets in Figure 13.

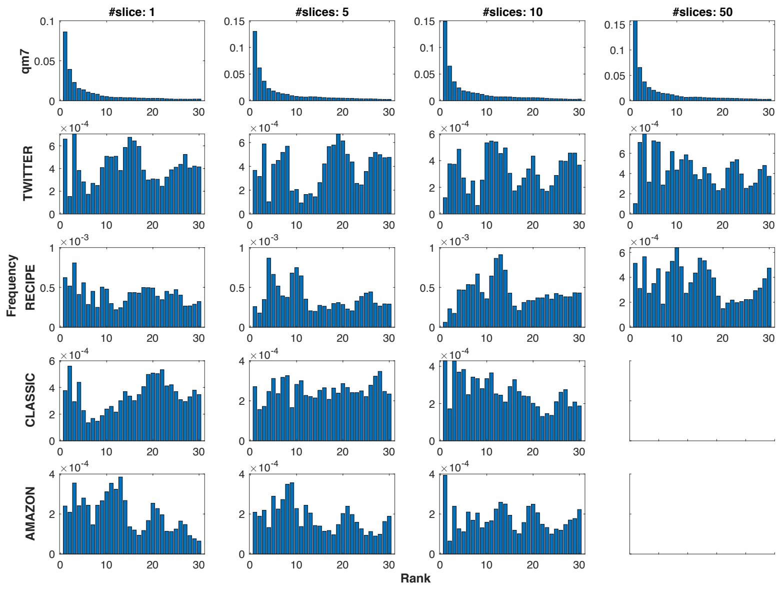


Figure 9: Empirical relation: d_α := SGW and d_β := DA.We used DA with 1 tree slice for `qm7`, 10 tree slices for `TWITTER`, 5 tree slices for `RECIPE`, 1 tree slice for `CLASSIC`, and 1 tree slice for `AMAZON`. For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

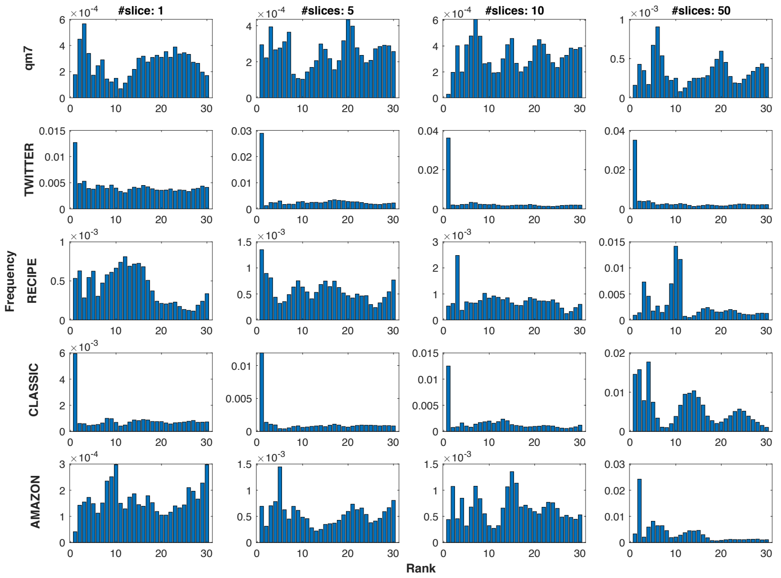


Figure 10: Empirical relation: d_α := FA and d_β := EGW_0. The entropic regularization for EGW_0 is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets.

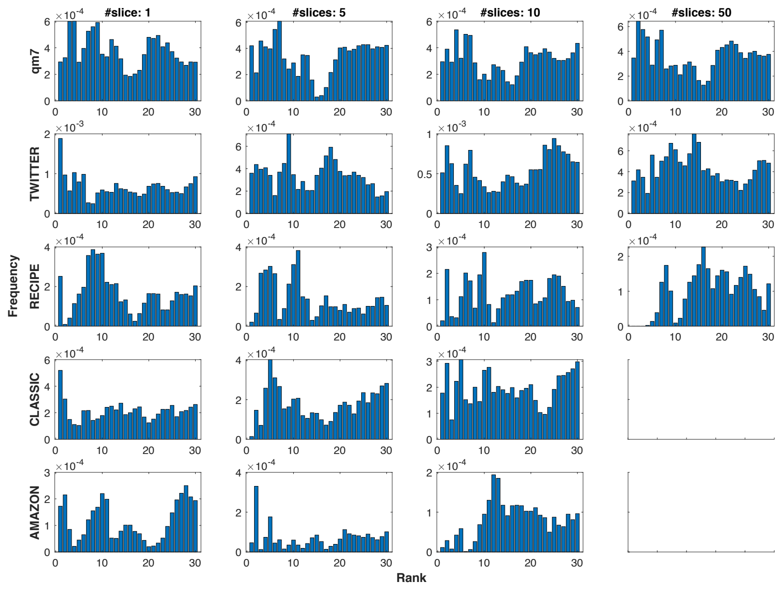


Figure 11: Empirical relation: d_α := SGW and d_β := EGW$_0$. The entropic regularization for EGW$_0$ is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets. For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

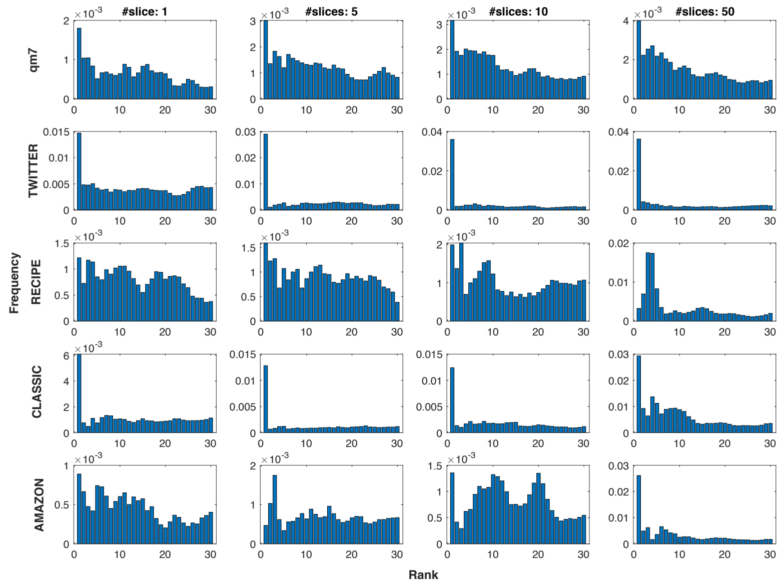


Figure 12: Empirical relation: d_α := FA and d_β := EGW. The entropic regularization for EGW is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets.

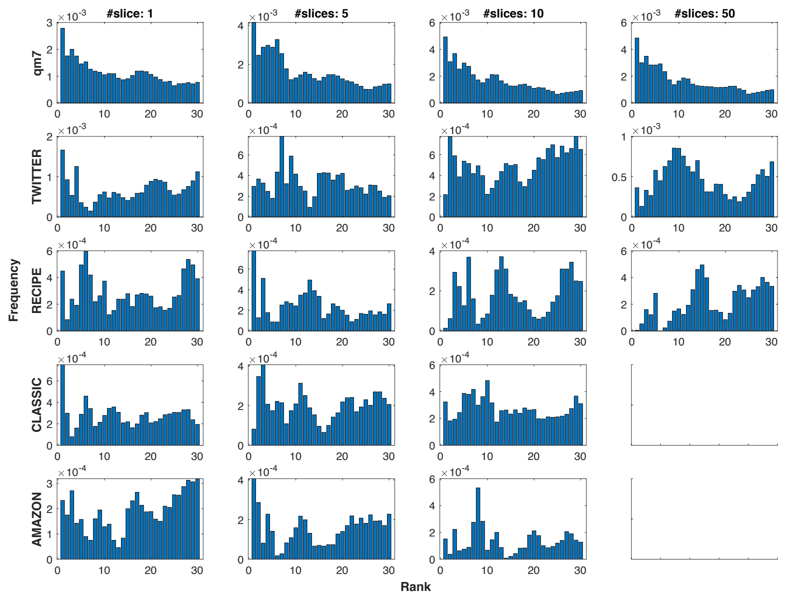


Figure 13: Empirical relation: $d_\alpha :=$ SGW and $d_\beta :=$ EGW. The entropic regularization for EGW is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets. For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

Discussions. It seems that there is no much empirical relation between FA/SGW and EGW$_0$/EGW on `qm7, TWITTER, RECIPE, CLASSIC, AMAZON` datasets.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows. 1988.

[2] Tomas Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444. ACM, 1988.

[3] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[4] Tam Le and Marco Cuturi. Unsupervised Riemannian metric learning for histograms using Aitchison transformations. In *International Conference on Machine Learning*, pages 2002–2011, 2015.

[5] Tam Le, Nhat Ho, and Makoto Yamada. Flow-based alignment approaches for probability measures in different spaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021.

[6] Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced variants of Wasserstein distances. In *Advances in neural information processing systems*, pages 12283–12294, 2019.

[7] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4104–4113, 2019.

[8] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[9] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672, 2016.

[10] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[11] Filippo Santambrogio. *Optimal transport for applied mathematicians*. Birkäuser, 2015.

[12] Charles Semple and Mike Steel. Phylogenetics. *Oxford Lecture Series in Mathematics and its Applications*, 2003.

[13] Titouan Vayer, Rémi Flamary, Romain Tavenard, Laetitia Chapel, and Nicolas Courty. Sliced Gromov-Wasserstein. *Advances in Neural Information Processing Systems*, 2019.

[14] Cédric Villani. *Topics in Optimal Transportation.* American Mathematical Society, 2003.