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

Abstract

Gromov-Wasserstein (GW) is a powerful tool to compare probability measures whose supports are in different metric spaces. However, GW suffers from a computational drawback since it requires to solve a complex non-convex quadratic program. In this work, we consider a specific family of cost metrics, namely, tree metrics for supports of each probability measure, to develop efficient and scalable discrepancies between the probability measures. Leveraging a tree structure, we propose to align *flows* from a root to each support instead of pair-wise tree metrics of supports, i.e., flows from a support to another support, in GW. Consequently, we propose a novel discrepancy, named *Flow-based Alignment* (FlowAlign), by matching the flows of the probability measures. FlowAlign is computationally fast and scalable for large-scale applications. Further exploring the tree structure, we propose a variant of FlowAlign, named *Depth-based Alignment* (DepthAlign), by aligning the flows hierarchically along each depth level of the tree structures. Theoretically, we prove that both FlowAlign and DepthAlign are pseudo-metrics. We also derive tree-sliced variants of the proposed discrepancies for applications without prior knowledge about tree structures for probability measures, computed by averaging FlowAlign/DepthAlign using random tree metrics, adaptively sampled from supports of probability measures. Empirically, we test our proposed approaches against other variants of GW baselines on a few benchmark tasks.

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

1 Introduction

Optimal transport (OT) theory provides a powerful set of tools to compare probability measures. OT has recently gained considerable interests in machine learning community (Cuturi, 2013; Perrot et al., 2016; Genevay et al., 2016; Muzellec and Cuturi, 2018; Luise et al., 2019; Mena and Niles-Weed, 2019; Paty and Cuturi, 2019; Togninalli et al., 2019), and played an increasingly important role in several research areas, such as computer graphics (Solomon et al., 2015; Bonneel et al., 2016; Lavenant et al., 2018; Solomon and Vaxman, 2019), domain adaptation (Courty et al., 2016, 2017; Bhushan Damodaran et al., 2018; Redko et al., 2019), and deep generative models (Arjovsky et al., 2017; Gulrajani et al., 2017; Genevay et al., 2018; Kolouri et al., 2019; Nadjahi et al., 2019; Wu et al., 2019).

When probability measures are discrete and their supports are in the same space, OT distance can be recasted as a linear programming, which can be solved by standard interior-point method algorithms. However, these algorithms are not efficient when the number of supports is large. In order to account for the scalability of the OT distance, Cuturi (2013) initiated a new research line by regularizing the OT with the entropy of the transport plans. Several efficient algorithms have been recently proposed to solve the entropic OT (Altschuler et al., 2017; Dvurechensky et al., 2018; Lin et al., 2019; Altschuler et al., 2019).

When probability measures are discrete and their supports are in different spaces, the classical OT distance is no longer valid to measure their discrepancy. In his seminal work, Mémoli (2011) introduced Gromov-Wasserstein (GW) distance to compare probability measures whose supports are in different metric spaces. Due to its flexibility, the GW distance has been used in several applications, including quantum chemistry (Peyré et al., 2016), computer graphics (Solomon et al., 2016), cross-lingual embeddings (Alvarez-Melis and Jaakkola, 2018; Grave et al., 2019), graph partitioning and matching (Xu et al., 2019a,b), and deep generative mod-

[*]: The first two authors contributed equally.

els (Bunne et al., 2019). However, the GW is a complex non-convex quadratic program and NP-hard for arbitrary inputs (Peyré and Cuturi, 2019, §10.6.3). Therefore, its computation is very costly, which hinders applications in large-scale settings where the number of supports is large.

Reposing on the entropic regularization idea from OT, Peyré et al. (2016) proposed an entropic GW (EGW) discrepancy. The EGW can be efficiently solved by the Sinkhorn algorithm under certain cases of regularization parameter and a specific family of loss functions. Nevertheless, EGW requires the regularization to be sufficiently large for a fast computation, which leads to a poor approximation of GW. Following the direction of leveraging entropic regularization, Xu et al. (2019a,b) proposed algorithmic approaches to further speed up GW for graph data. Another approach for scaling up the computation of GW is sliced GW (SGW) (Vayer et al., 2019), which relies on a one-dimensional projection of supports of the probability measures. Consequently, similar to sliced-Wasserstein, SGW albeit fast limits its capacity to capture high-dimensional structure in a distribution of supports (Le et al., 2019b; Liutkus et al., 2019). Additionally, SGW can be *only* either applied for discrete measures with the same number of supports and uniform weights, or required an artifact zero-padding for probability measures having different number of supports (Vayer et al., 2019).

Contributions. In this work, we consider a particular family of cost metrics, namely tree metrics for a space of supports of each probability measure, and aim for developing efficient and scalable discrepancies for probability measures in different spaces that can be applied as fast alternative approaches for GW, especially in large-scale applications.

Although it is well-known that one can leverage tree metrics to speed up a computation of arbitrary metrics (Bartal, 1996, 1998; Charikar et al., 1998; Indyk, 2001; Fakcharoenphol et al., 2004), our goal is rather to sample tree metrics for spaces of supports, and use them as cost metrics, similar to tree-sliced-Wasserstein (TSW) (Le et al., 2019b). However, different to TSW, one may *not* apply this idea straightforwardly by only using tree metrics as cost metrics for GW to develop scalable discrepancy for probability measures in different tree metric spaces. Therefore, by exploiting a tree structure, we propose to align *flows* from a root to each support instead of pair-wise tree metrics of supports, i.e., flows from a support to another, in GW for the probability measures. Consequently, we propose a novel discrepancy, named *Flow-based Alignment* (FlowAlign), by matching the flows of the probability measures. FlowAlign is fast for computation and scalable for large-scale applications. To further explore the tree structures, we propose to align the flows hierarchically along each depth level of the tree structures, named *Depth-based Alignment* (DepthAlign). We then prove that both FlowAlign and DepthAlign are pseudo-metrics, i.e., they are symmetric and satisfy the triangle inequality.

For applications without prior knowledge about tree structures for probability measures, we derive tree-sliced variants of FlowAlign/DepthAlign, computed by averaging FlowAlign/DepthAlign using random tree metrics, sampled by a fast adaptive method, e.g., clustering-based tree metric sampling (Le et al., 2019b, §4). We empirically illustrate that the proposed discrepancies compares favorably with state-of-the-art variants of GW baselines, e.g., EGW (Peyré et al., 2016) and SGW (Vayer et al., 2019), in applications. Especially, FlowAlign is several orders faster than EGW and at least as fast as SGW while remedies the curse of dimensionality in SGW by leveraging tree structures.

Organization. The paper is organized as follows: we review tree metrics and GW in §2. We propose two novel discrepancies: FlowAlign and DepthAlign for probability measures in different tree metric spaces in §3 and §4 respectively. In §5, we derive their tree-sliced variants for practical applications, and then give discussions and related work in §6. We evaluate the proposed discrepancies against other baselines on some benchmark tasks in §7 before concluding in §8. We have released code for our proposal[1].

Notation. We denote $[n] = \{1, 2, \ldots, n\}$, $\forall n \in \mathbb{N}_+$. For $x \in \mathbb{R}^d$, let $\|x\|_1$ be the ℓ_1-norm of x, and δ_x be the Dirac function at x. For probability measure μ, we denote $\text{supp}(\mu)$ and $|\mu|$ for the set and the number of support(s) of μ respectively.

2 Reminders on Tree Metric and GW

In this section, we first recall tree metric space and then briefly review GW between probability measures in different tree metric spaces.

2.1 Tree metric space

For a tree metric space $(\mathcal{T}, d_\mathrm{T})$, d_T is a tree metric on tree $\mathcal{T}$. The *tree metric* d_T between two nodes in $\mathcal{T}$ is equal to a length of the (unique) path between them (Semple and Steel, 2003, §7, p.145–182). Given node $x \in \mathcal{T}$, let $\Gamma(x)$ be the set of nodes in the subtree of $\mathcal{T}$ rooted at x, i.e., $\Gamma(x) = \{z \in \mathcal{T} \mid x \in \mathcal{P}(r, z)\}$ where $\mathcal{P}(r, z)$ is the (unique) path between root r and node z

[1] https://github.com/lttam/FlowBasedAlignment-GW

in $\mathcal{T}$, $\mathcal{S}(x)$ be the set of child nodes of x, and $|\mathcal{S}(\cdot)|$ is the cardinality of set $\mathcal{S}(\cdot)$. Given edge e, we write u_e and v_e for the nodes that are respectively at a shallower (i.e., closer to r) and deeper (i.e., further away from r) level of edge e, and w_e be the non-negative length of that edge. We illustrate those notions in Figure 1.

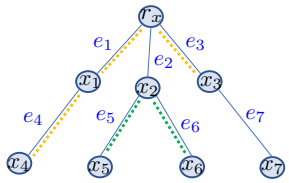


Figure 1: An illustration for a tree metric space. x_2 is at depth level 2. $\mathcal{P}(x_3, x_4)$ contains e_3, e_1, e_4 (the orange dot path), $\Gamma(x_2) = \{x_2, x_5, x_6\}$ (the green dot subtree), and $\mathcal{S}(r_x) = \{x_1, x_2, x_3\}$. For edge e_5, $v_{e_5} = x_5$ and $u_{e_5} = x_2$.

Throughout the paper, we consider two probability measures $\mu = \sum_{i\in[k]} a_i \delta_{x_i}$ (with k supports) and $\nu = \sum_{j\in[k']} b_j \delta_{z_j}$ (with k' supports) where $\text{supp}(\mu)$ and $\text{supp}(\nu)$ are in different tree metric spaces $(\mathcal{T}_X, d_{\mathcal{T}_X})$ and $(\mathcal{T}_Z, d_{\mathcal{T}_Z})$[2] respectively; $a_i, b_j \in \mathbb{R}_+, \forall i \in [k], j \in [k']$ such that $\sum_{i\in[k]} a_i = \sum_{j\in[k']} b_j = 1$. Our goal is to define discrepancies for these probability measures.

2.2 Gromov-Wasserstein with Tree Metrics

Mémoli (2011) defined Gromov-Wasserstein $\mathcal{GW}$ between μ, ν as

$$\mathcal{GW}^2(\mu,\nu) := \min_{T\in\Pi(\mu,\nu)} \sum_{i,j,i',j'} |d_{\mathcal{T}_X}(x_i, x_{i'}) - d_{\mathcal{T}_Z}(z_j, z_{j'})|^2 T_{ij} T_{i'j'}, \quad (1)$$

where $\Pi(\mu,\nu)$ is a set of the transport plans $T \in \mathbb{R}_+^{k\times k'}$ such that $\sum_{j\in[k']} T_{ij} = a_i \;|_{i\in[k]}, \sum_{i\in[k]} T_{ij} = b_j \;|_{j\in[k']}$. Intuitively, GW aligns pair-wise tree metrics of supports $d_{\mathcal{T}_X}(x_i, x_{i'}) \;|_{i,i'}$ and $d_{\mathcal{T}_Z}(z_j, z_{j'}) \;|_{j,j'}$ for μ and ν.

However, one may not scale up GW by straightforwardly using tree metrics as cost metrics as in Equation (1) like TSW (Le et al., 2019b). Therefore, we propose to leverage tree structure to align *flows* from a root to each support instead of pair-wise tree metrics of supports, i.e., flows from a support to another, in GW to develop scalable discrepancy for the probability measures. Consequently, we propose two novel discrepancies: FlowAlign and DepthAlign, detailed in §3 and §4 respectively.

3 Flow-based Alignment Discrepancy

In this section, we propose a novel, efficient and scalable discrepancy, named Flow-based Alignment (FlowAlign), for probability measures in different tree metric spaces.

[2] $d_{\mathcal{T}_X}, d_{\mathcal{T}_Z}$ are tree metrics on tree $\mathcal{T}_X, \mathcal{T}_Z$ respectively.

3.1 Definition of FlowAlign

Different from GW, FlowAlign exploits tree structures for the alignment.

Definition 1. *The Flow-based Alignment discrepancy $\mathcal{A}_f$ between μ, ν is defined as*

$$\mathcal{A}_f^2(\mu,\nu) := \min_{r_x, r_z, T\in\Pi(\mu,\nu)} \sum_{i,j} |d_{\mathcal{T}_X}(r_x, x_i) - d_{\mathcal{T}_Z}(r_z, z_j)|^2 T_{ij}. \quad (2)$$

Intuitively, FlowAlign considers the matching for *flows* from a root to each support for probability measures based on (i) the flow lengths (i.e., tree metrics from a root to each support), and (ii) the flow masses (i.e., weights on supports corresponding to the flows). Moreover, FlowAlign also takes into account the root alignment for corresponding tree structures of tree metric spaces since the *flows* depend on which node in the tree structure has a role as the tree root. Therefore, instead of matching *pairs of supports* as in GW for probability measures in different spaces, FlowAlign exploits tree structures of the tree metric spaces to align *both tree root and supports* for the probability measures. To the best of our knowledge, our work is the first approach leveraging tree metric to align probability measures supported in different metric spaces (i.e., alternative approach for GW based on flows)[3].

One should distinguish FlowAlign from tree-(sliced)-Wasserstein which directly matches *supports* for probability measures in the *same* tree metric space. Additionally, the goal of FlowAlign is to match the probability measures μ, ν like GW, by exploiting the tree structures $(\mathcal{T}_X, d_{\mathcal{T}_X})$, $(\mathcal{T}_Z, d_{\mathcal{T}_Z})$, but not to compare trees $\mathcal{T}_X, \mathcal{T}_Z$ themselves like tree edit distance (Zhang and Shasha, 1989).

Theorem 1. *FlowAlign is a pseudo-distance. It satisfies symmetry and the triangle inequality.*

See the supplementary (§A) for the proof of Theorem 1. When $\mathcal{A}_f^2(\mu,\nu) = 0$, we can find roots r_x^* and r_z^* such that $\tilde{\mu}^* \equiv \tilde{\nu}^*$ where $\tilde{\mu}^* = \sum_i a_i \delta_{d_{\mathcal{T}_X}(r_x^*, x_i)}$ and $\tilde{\nu}^* = \sum_j b_j \delta_{d_{\mathcal{T}_Z}(r_z^*, z_j)}$. It demonstrates that μ and ν have similar 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.

While GW with tree metrics relies on *pair-wise supports alignment* to match probability measures, FlowAlign uses *flow alignment* based on geometric

[3] After our preprint, Mémoli et al. (2021) leveraged ultrametric—a special case of tree metrics—to study basic topological and geometric properties of Sturm's distance (Sturm et al., 2006) and Gromov-Wasserstein.

structures of trees. We next attempt to draw relations between them when the deepest levels of trees $\mathcal{T}_X$ and $\mathcal{T}_Z$ are equal to two.

Proposition 1. *If the deepest levels of trees $\mathcal{T}_X$ and $\mathcal{T}_Z$ are two, then $\mathcal{GW}(\mu,\nu) \leq 2\mathcal{A}_f(\mu,\nu)$.*

See the supplementary (§A) for the proof of Proposition 1.

In practical applications without prior knowledge about tree structures for probability measures, $\mathcal{T}_X$ and $\mathcal{T}_Z$ are sampled from support data points, e.g., by clustering-based tree metric sampling (Le et al., 2019b). We argue that the farthest-point clustering (Gonzalez, 1985) within the clustering-based tree metric sampling ensures that FlowAlign is invariant to rotation and translation, detailed in the supplementary (§B).

3.2 Efficient computation for FlowAlign

A naive implementation for FlowAlign $\mathcal{A}_f$ has a complexity $\mathcal{O}(N^3 \log N)$ where N is the number of nodes in tree, if one exhaustively searches the optimal pair of roots for $\mathcal{T}_X$ and $\mathcal{T}_Z$[4]. In this section, we present an efficient computation approach which reduces this complexity into nearly $\mathcal{O}(N^2)$.

Consider $\mathcal{A}_f$ between μ,ν in $\mathcal{T}_X, \mathcal{T}_Z$ rooted at r_x, r_z respectively. When one changes into the new root $\bar{r}_z$ for $\mathcal{T}_Z$, illustrated in Figure 2, there are two cases that can happen:

Case 1 : $\bar{r}_z$ is in the subtree rooted at a node in $\mathcal{S}(r_z)$, which is disjoint from $\text{supp}(\nu)$, illustrated in the left-bottom tree of Figure 2. Then, $\forall z_i \in \text{supp}(\nu)$, we have

$$d_{\mathcal{T}_Z}(\bar{r}_z, z_i) = d_{\mathcal{T}_Z}(r_z, z_i) + d_{\mathcal{T}_Z}(\bar{r}_z, r_z).$$

Therefore, the path-length order from the root is preserved.

Case 2 : $\bar{r}_z$ is in the subtree rooted at a node in $\mathcal{S}(r_z)$, containing some supports of ν, denoted as Ω_ν, illustrated in the right-bottom tree of Figure 2. Then, $\forall z_j \in \text{supp}(\nu) \setminus \Omega_\nu$, we have

$$d_{\mathcal{T}_Z}(\bar{r}_z, z_j) = d_{\mathcal{T}_Z}(r_z, z_j) + d_{\mathcal{T}_Z}(\bar{r}_z, r_z).$$

Thus, the path-length order from the root (except those for $z_i \in \Omega_\nu$) is preserved. For supports in Ω_ν (illustrated in the supplementary (§B)), there are three following sub-cases:

• **Case 2a**: For supports $z_i \in \Omega_\nu$ which $\bar{r}_z \in \mathcal{P}(r_z, z_i)$, then

$$d_{\mathcal{T}_Z}(\bar{r}_z, z_i) = d_{\mathcal{T}_Z}(r_z, z_i) - d_{\mathcal{T}_Z}(r_z, \bar{r}_z).$$

[4]More details about (aligned-root) FlowAlign complexity are given in §3.3, and in the supplementary (§C).

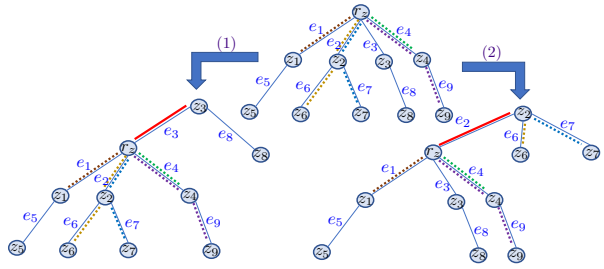


Figure 2: An illustration for an efficient computation for FlowAlign. Given $\nu = b_1\delta_{z_1} + b_2\delta_{z_6} + b_3\delta_{z_7} + b_4\delta_{z_4} + b_5\delta_{z_9}$, when the new root $\bar{r}_z = z_3$ (z_3 is in the subtree rooted at z_3 which is disjoint from $\text{supp}(\nu)$), the order of $d_{\text{T}}(r_z, z_i)\,|_{z_i \in \nu}$ is the same as that of $d_{\text{T}}(z_3, z_i)\,|_{z_i \in \nu}$, and $d_{\text{T}}(z_3, z_i) = d_{\text{T}}(r_z, z_i) + d_{\text{T}}(r_z, z_3), \forall z_i \in \text{supp}(\nu)$ (**Case 1**: the left-bottom tree). When $\bar{r}_z = z_2$ (z_2 is in the subtree rooted at z_2, and containing supports $\Omega_\nu = \{z_6, z_7\}$ of ν), the order of $d_{\text{T}}(r_z, z_i)\,|_{z_i \in \text{supp}(\nu) \setminus \Omega_\nu}$ is the same as that of $d_{\text{T}}(z_3, z_i)\,|_{z_i \in \text{supp}(\nu) \setminus \Omega_\nu}$, and $d_{\text{T}}(z_2, z_i) = d_{\text{T}}(r_z, z_i) + d_{\text{T}}(r_z, z_3), \forall z_i \in \text{supp}(\nu) \setminus \Omega_\nu$ (**Case 2**: the right-bottom tree).

So, the path-length order from the root of those supports are preserved.

• **Case 2b**: For supports $z_i \in \Omega_\nu$ which $z_i \in \mathcal{P}(r_z, \bar{r}_z)$, then

$$d_{\mathcal{T}_Z}(\bar{r}_z, z_i) = d_{\mathcal{T}_Z}(r_z, \bar{r}_z) - d_{\mathcal{T}_Z}(r_z, z_i).$$

Therefore, the path-length order from the root of those supports are reversed.

• **Case 2c**: For supports $z_i \in \Omega_\nu$ which $\bar{r}_z \notin \mathcal{P}(r_z, z_i)$ and $z_i \notin \mathcal{P}(r_z, \bar{r}_z)$, then one needs to find the corresponding closest common ancestor ζ_i of $\bar{r}_z$ and z_i, i.e., ζ_i is on both paths $\mathcal{P}(r_z, \bar{r}_z)$ and $\mathcal{P}(r_z, z_i)$, and

$$d_{\mathcal{T}_Z}(\bar{r}_z, z_i) = d_{\mathcal{T}_Z}(r_z, \bar{r}_z) + d_{\mathcal{T}_Z}(r_z, z_i) - 2d_{\mathcal{T}_Z}(r_z, \zeta_i).$$

Note that the path-length order from the root of supports having the same ζ_i is preserved.

Therefore, one only needs to merge the ordered path-lengths from the root in those above cases with the complexity nearly $\mathcal{O}(N)$ (except the degenerated instance where each above case contains only one support).

Thus, one may not need to sort for tree metrics between $\bar{r}_z$ and each support of ν by leveraging the sorted order of the tree metrics between r_z and each support. Moreover, those computational steps can be done separately for each tree. Consequently, the complexity of $\mathcal{A}_f$ reduces from $\mathcal{O}(N^3 log N)$ into nearly $\mathcal{O}(N^2)$. More details can be seen in the supplementary (§C).

3.3 Aligned-root FlowAlign

We consider a special case of FlowAlign where roots have been already aligned[5]. Therefore, we can leave out minimization step with roots in Definition 1, and name it as aligned-root FlowAlign.

Definition 2. *Assume that root r_x in $\mathcal{T}_X$ is aligned with root r_z in $\mathcal{T}_Z$. Then, the aligned-root Flow-based Alignment discrepancy $\widehat{\mathcal{A}}_f$ between μ, ν is defined as*

$$\widehat{\mathcal{A}}_f^2(\mu,\nu;r_x,r_z) := \min_{T\in\Pi(\mu,\nu)} \sum_{i,j} \left|d_{\mathtt{T}_X}(r_x,x_i) - d_{\mathtt{T}_Z}(r_z,z_j)\right|^2 T_{ij}. \tag{3}$$

The $\widehat{\mathcal{A}}_f$ in Equation (3) is equivalent to the univariate Wasserstein distance (1d-OT) between $\tilde{\mu} := \sum_i a_i \delta_{d_{\mathtt{T}_X}(r_x,x_i)}$ and $\tilde{\nu} := \sum_j b_j \delta_{d_{\mathtt{T}_Z}(r_z,z_j)}$, which is equal to the integral of the absolute difference between the generalized quantile functions of these two univariate probability distributions (Santambrogio, 2015, §2). Therefore, one only needs to *sort* flow lengths $d_{\mathtt{T}_X}(r_x,x_i)\ |_i$, and $d_{\mathtt{T}_Z}(r_z,z_j)\ |_j$ for the computation of $\widehat{\mathcal{A}}_f$, i.e., linearithmic complexity. Due to sharing the same structure as 1d-OT, $\widehat{\mathcal{A}}_f$ inherits the same properties as those of the 1d-OT. More precisely, $\widehat{\mathcal{A}}_f$ is symmetric and satisfies the triangle inequality. Additionally, $\widehat{\mathcal{A}}_f(\mu,\nu;r_x,r_z) = 0$ is equivalent to $\tilde{\mu} \equiv \tilde{\nu}$. Furthermore, one can extend the squared loss (in Equation (2) and Equation (3)) into functions which are a nonnegative convex function g applied to the difference $(x-z)$ between two tree metrics, i.e., $g(x-z)$. See the supplementary (§B) for an illustration of $\widehat{\mathcal{A}}_f$.

Note that in practical applications, we usually do not have prior knowledge about tree structures for probability measures. Therefore, we need to sample tree metrics for each support data space. Moreover, by leveraging geometrically spatial information, we choose means of support data distributions as roots when using the clustering-based tree metric sampling (Le et al., 2019b) as a heuristic for sampling likely suboptimal *aligned-root tree metrics*. Consequently, we can reduce the complexity of FlowAlign by using aligned-root FlowAlign.

Aligned-root FlowAlign barycenter. The aligned-root FlowAlign can be handily used for a barycenter problem, especially in large-scale applications. Given m probability measures $\mu_i\ |_{i\in[m]}$ in different tree metric spaces $(\mathcal{T}_{X_i}, d_{\mathtt{T}_{X_i}})\ |_{i\in[m]}$ with aligned-roots $r_{x_i}\ |_{i\in[m]}$ respectively, and corresponding weights $p_i\ |_{i\in[m]}$, the aligned-root FlowAlign barycenter aims to find a flow-based tree structure[6] $\Delta_{\bar{\mu}} := \{d_{\mathtt{T}_X}(r_x,x_i), a_i\}_{i\in[\mathtt{k}]}$ of an optimal probability measure $\bar{\mu}$ with at most $\mathtt{k}$ supports in $(\mathcal{T}_X, d_{\mathtt{T}_X})$ that takes the form:

$$\Delta_{\bar{\mu}} \in \arg\min_{\Delta_{\hat{\mu}}} \left(\sum_{i=1}^{m} p_i \widehat{\mathcal{A}}_f^2(\hat{\mu},\mu_i;r_x,r_{x_i}) \right), \tag{4}$$

where the roots r_{x_i} in $\mathcal{T}_{X_i}\ |_{i\in[m]}$ are aligned with root r_x in $\mathcal{T}_X$. The barycenter problem in Equ. (4) is equivalent to the free-support 1d-OT barycenter efficiently solved, e.g., by using Alg. 2 in (Cuturi and Doucet, 2014).

4 Depth-based Alignment Discrepancy

FlowAlign only focuses on flows from a root to each support and tree root alignment, but ignores the depth level of supports in trees. In this section, we take into account the depth level of supports, and propose Depth-based Alignment (DepthAlign) discrepancy $\mathcal{A}_d$ by considering the alignment for flows hierarchically for each depth level along the tree structures. We first introduce some necessary definitions to define $\mathcal{A}_d$. Recall that, $\mathcal{S}(x)$ is a set of child nodes of x in $\mathcal{T}$.

Definition 3. *Given node x in $\mathcal{T}$, a 2-depth-level tree $\mathcal{T}_x^2$ is an induced subtree of $\mathcal{T}$, rooted as x and spanned by x and its children $\mathcal{S}(x)$.*

Let $V(\mathcal{T}_x^2)$ be the set of vertices of $\mathcal{T}_x^2$. Following Definition 3, $V(\mathcal{T}_x^2)$ contains x and all $\bar{x} \in \mathcal{S}(x)$. Moreover, given μ in $\mathcal{T}$, we have a corresponding $\mu_{\mathcal{T}_x^2}$ in $\mathcal{T}_x^2$, defined as $\mu_{\mathcal{T}_x^2} = \sum_i \bar{a}_i \delta_{\bar{x}_i}$ where $\bar{x}_i \in V(\mathcal{T}_x^2)$, and if $\bar{x}_i \neq x$ then

$$\bar{a}_i = \mu(\Gamma(\bar{x}_i))/\mu(\Gamma(x)),$$

otherwise

$$\bar{a}_i = 1 - \sum_{\bar{x}_j \in \mathcal{S}(x)} \mu(\Gamma(\bar{x}_j))/\mu(\Gamma(x)).$$

In order to define DepthAlign, we start with its special case when roots are aligned. Let $\mathcal{M}_h$ and $T_h^*(x,z)$ be a set of optimal aligned pairs and the optimal matching mass for the pair (x,z) at the depth level h respectively.

Definition 4. *Assume that root r_x in $\mathcal{T}_X$ is aligned with root r_z in $\mathcal{T}_Z$. Then, the aligned-root Depth-based Alignment $\widehat{\mathcal{A}}_d$ between μ, ν is defined as*

$$\widehat{\mathcal{A}}_d(\mu,\nu;r_x,r_z) := \sum_h \sum_{(x,z)\in\mathcal{M}_{h-1}} T_{h-1}^*(x,z) \widehat{\mathcal{A}}_f(\mu_{\mathcal{T}_x^2}, \nu_{\mathcal{T}_z^2}; x, z), \tag{5}$$

where h is the considered depth level, starting from 2 to the deepest level of the lower tree between $\mathcal{T}_X$ and $\mathcal{T}_Z$;

[5] By root alignment, we mean the optimal pair of roots in (r_x^*, r_z^*, T^*) in Equation (2).

[6] Recall that $\widehat{\mathcal{A}}_f$ is equivalent to 1d-OT, Δ_μ is a representation (e.g., supports and weights) in 1d-OT problem of a probability measure μ in $\widehat{\mathcal{A}}_f$ problem.

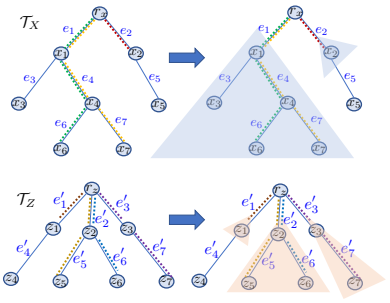


Figure 3: An illustration for aligned-root DepthAlign $\widehat{\mathcal{A}}_d$ between $\mu = a_1\delta_{x_6} + a_2\delta_{x_7} + a_3\delta_{x_2}$ on $\mathcal{T}_X$ and $\nu = b_1\delta_{z_1} + b_2\delta_{z_5} + b_3\delta_{z_6} + b_4\delta_{z_7}$ on $\mathcal{T}_Z$. In $\widehat{\mathcal{A}}_d$, we consider the optimal alignment at each depth level h. At $h = 1$, root r_x is trivially aligned for root r_z. Since r_x, r_z have their child nodes, the alignment (r_x, r_z) is recursive into $h = 2$. For μ in $\mathcal{T}_X$, r_x has 2 subtrees rooted at x_1, x_2. Thus, $V(\mathcal{T}^2_{r_x}) = \{r_x, x_1, x_2\}$, and $\mu_{\mathcal{T}^2_{r_x}} = (a_1 + a_2)\delta_{x_1} + a_3\delta_{x_2}$. Similarly, for ν in $\mathcal{T}_Z$, $\nu_{\mathcal{T}^2_{r_z}} = b_1\delta_{z_1} + (b_2 + b_3)\delta_{z_2} + b_4\delta_{z_3}$. The recursive procedure is repeated until the deepest level of the lower tree where only simple cases exist.

$\mathcal{M}_1 = \{(r_x, r_z)\}$; $T^*_1(r_x, r_z) = 1$. *Solving the optimal transport maps* $T^*_h(\cdot,\cdot)$ *of subproblems* $\widehat{\mathcal{A}}_f$ *at depth level* h *will form a set of optimal aligned pairs* $\mathcal{M}_h$.

Intuitively, at each depth level h, we consider the alignment for the corresponding 2-depth-level trees. Note that the 2-depth-level tree structures are at the same depth level h for both $\mathcal{T}_X$ and $\mathcal{T}_Z$, and one can consider $\widehat{\mathcal{A}}_f$ for such alignment. Moreover, at $h = 1$, $\widehat{\mathcal{A}}_d$ trivially matches r_x to r_z with optimal matching mass 1. Solving the optimal transport maps $T^*_h(\cdot,\cdot)$ of subproblems $\widehat{\mathcal{A}}_f$ at depth level h will form a set of optimal aligned pairs $\mathcal{M}_h$. Then, $T^*_h(\cdot,\cdot)$ and $\mathcal{M}_h$ are used in the next depth level $(h+1)$. Thus, the matching procedure is recursive along all depth levels in trees, illustrated in Figure 3. The simple case of the recursive procedure is that either at least one node of considered pair does not have child nodes, or sum of weights of child nodes in the corresponding 2-depth-level tree is equal to 0.

When roots of trees $\mathcal{T}_X$ and $\mathcal{T}_Z$ are not aligned yet, we optimize the root alignment as follow:

$$\mathcal{A}_d(\mu,\nu) := \min_{r_x, r_z} \widehat{\mathcal{A}}_d(\mu,\nu; r_x, r_z), \tag{6}$$

which is referred to as DepthAlign. Similar to FlowAlign, we have a following theorem:

Theorem 2. *DepthAlign is a pseudo-distance. It satisfies symmetry and the triangle inequality.*

See the supplementary (§A) for the proof of Theorem 2. When $\mathcal{A}_d(\mu,\nu) = 0$, we can find roots r^*_x and r^*_z such that all the hierarchical corresponding $\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2}, \nu_{\mathcal{T}^2}; \cdot, \cdot)$ for each depth level along the trees are equal to 0. It demonstrates that μ and ν have similar weights on supports while their supports have similar depth levels, and for each depth level, the tree metrics of supports in the corresponding $\mu_{\mathcal{T}^2}, \nu_{\mathcal{T}^2}$ to the 2-depth-level-tree roots are identical, i.e., corresponding weight edges are identical.

DepthAlign computes a set of $\widehat{\mathcal{A}}_d$ subproblems hierarchically for each tree depth level. The number of $\widehat{\mathcal{A}}_d$ subproblems depends on the tree deepest level, and the number of assignments at each depth level. DepthAlign takes into account the depth level of each support which is not addressed in FlowAlign, but DepthAlign is much slower than FlowAlign.

5 Tree-sliced Variants by Sampling Tree Metrics

In practical applications, one usually do not have prior knowledge about tree structure for each probability measure. Computing FlowAlign/DepthAlign requires to choose or sample tree metrics for each space of supports. One way is to optimize tree metrics for probability measures from input data for a given task. However, this approach may be expensive and gives an extra cost for the computation of FlowAlign/DepthAlign.

Following the approach of TSW (Le et al., 2019b), we propose the tree-sliced variants for our proposed discrepancies, computed by averaging the corresponding FlowAlign/DepthAlign using randomly sampled tree metrics. We use fast adaptive methods, e.g., clustering-based tree metric sampling (Le et al., 2019b), to sample tree metrics from support data points of a given task.

Definition 5. *Given* μ, ν *supported on a set in which tree metric spaces* $\{(\mathcal{T}_{X_i}, d_{\mathcal{T}_{X_i}}) \mid i \in [n]\}$ *and* $\{(\mathcal{T}_{Z_i}, d_{\mathcal{T}_{Z_i}}) \mid i \in [n]\}$ *can be defined respectively, the tree-sliced variants of (aligned-root) FlowAlign/DepthAlign is defined as an average of corresponding (aligned-root) FlowAlign/DepthAlign for* μ, ν *on* $(\mathcal{T}_{X_i}, d_{\mathcal{T}_{X_i}})$, *and* $(\mathcal{T}_{Z_i}, d_{\mathcal{T}_{Z_i}})$ *respectively.*

As discussed in (Le et al., 2019b), the average over different random tree metrics can reduce quantization effects or clustering sensitivity problems in which data points may be clustered to adjacent but different clusters respectively in tree metric sampling. Moreover, the complexity of tree metric sampling is negligible. Indeed, for clustering-based tree metric sampling[7], its complexity is $\mathcal{O}(H_{\mathcal{T}} m \log \kappa)$ when one fixes the same number

[7] In principle, one can control the number of nodes in sampled trees by hyperparameters $(H_{\mathcal{T}}, \kappa)$ of the clustering-based tree metric sampling. We empirically follow suggestions in (Le et al., 2019b) to sample each tree containing about 4000 nodes.

of clusters κ for the farthest-point clustering (Gonzalez, 1985) and sets $H_{\mathcal{T}}$ for the predefined deepest level of tree $\mathcal{T}$, and m is the number of input data points.

Much like one-dimensional projections do not have interesting properties in a distortion viewpoint, but remain useful for SGW (or for sliced Wasserstein (Rabin et al., 2011)). We believe trees with high distortion can be useful for FlowAlign/DepthAlign (similar to TSW (Le et al., 2019b, §4)). Moreover, excessive efforts to optimize the problem in Equation (2) for each randomly sampled tree metric would be self-defeating since it would lead to overfitting within the computation of FlowAlign itself (see (Peyré and Cuturi, 2019, §8.4) and (Le et al., 2019b, §4)). Therefore, we apply the clustering-based tree metric sampling where we choose population means as tree roots as a heuristic based on geometrically spatial information to sample *aligned-root* tree metrics which are likely suboptimal in applications. Consequently, we can reduce the computational complexity of tree-sliced variants of FlowAlign/DepthAlign by using their aligned-root formulations with those randomly sampled *aligned-root* tree metrics.

6 Discussion and Related Work

For specific applications with prior knowledge about tree metrics for probability measures, one can apply FlowAlign, or consider DepthAlign if the known tree structure is important for the applications. Moreover, if roots of those known tree metrics are already aligned, one can use the corresponding aligned-root formulations to reduce the complexity. For general applications without prior knowledge about tree metrics for probability measures, one can use a heuristic to sample aligned-root tree metrics, e.g., by choosing a mean of support data as its root for the clustering-based tree metric sampling (Le et al., 2019b), and use the aligned-root formulations for an efficient computation.

Ultrametric (a.k.a, non-Archimedean or isosceles metric (Shkarin, 2004)) is a special case of tree metrics. Mémoli et al. (2019) employed ultrametric to study geometric and computational properties of Gromov-Hausdoff, and later basic topological and geometric properties of Sturm's distance (Sturm et al., 2006) and Gromov-Wasserstein in (Mémoli et al., 2021). Additionally, tree metrics are also used for other OT problems, e.g., tree-Wasserstein barycenter (Le et al., 2019a) and entropy partial transport (Le and Nguyen, 2021).

7 Experiments

We evaluate our proposed discrepancies for quantum chemistry and document classification with *random linear transformation* word embeddings[8]. We also carry out the large-scale FlowAlign barycenter problem within k-means clustering for point clouds of handwritten digits in `MNIST` dataset *rotated arbitrarily* in the plane as in (Peyré et al., 2016).

Setup. We consider two baselines: (i) SGW (Vayer et al., 2019) and (ii) EGW (Peyré et al., 2016). In all of our experiments, we do not have prior knowledge about tree metrics for probability measures. Therefore, we apply the clustering-based tree metric sampling (Le et al., 2019b) where means of support data points are chosen as tree roots, as a heuristic to sample *aligned-root* tree metrics from support data points. Thus, we can leverage the aligned-root formulations for both FlowAlign (FA) and DepthAlign (DA) to reduce their complexity. For SGW, we follow Vayer et al. (2019) to add artifact zero-padding for discrete measures having different numbers of supports, and use the binomial expansion to reduce its complexity. For EGW, we use the entropic regularization to optimize transport plan, but exclude it when computing GW, which gives comparative or better performances than those of standard EGW. We also apply the log-stabilized Sinkhorn (Schmitzer, 2019). We observe that the quality of EGW is better when entropic regularization becomes smaller, but the computation is considerably slower. In our experiments, the computation for EGW is either usually blown up, or too slow for evaluation when entropic regularization is less than or equal 1. We run experiments with Intel Xeon CPU E7-8891v3 (2.80GHz), and 256GB RAM. Reported time consumption for all methods has already included their corresponding preprocessing, e.g., tree metric sampling for FlowAlign and DepthAlign, or one-dimensional projection for SGW.

7.1 Applications

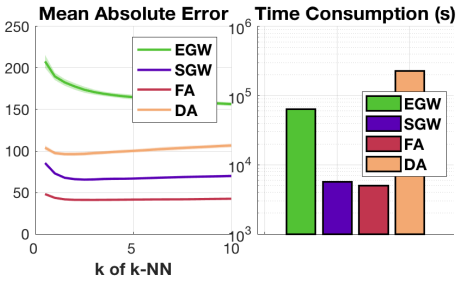


Figure 4: MAE and time consumption of k-NN regression on `qm7` for EGW (eps=5), SGW (10 slices), FA (10 tree-slices), and DA (1 tree-slice).

Quantum chemistry. We consider a regression problem on molecules for `qm7` dataset as in (Peyré et al., 2016). The task is to predict atomization energies for

[8] We apply different random linear transformation for word embedding in each document.

molecules based on similar labeled molecules instead of estimating them through expensive numerical simulations (Rupp et al., 2012; Peyré et al., 2016).

For simplicity, we only used the relative locations in $\mathbb{R}^3$ of atoms in molecules, *without information about atomic nuclear charges* as the experiments in (Rupp et al., 2012; Peyré et al., 2016). Therefore, a molecule with t atoms is represented as a probability measure $\mu = \frac{1}{t}\sum_{i=1}^{t}\delta_{x_i}$ where x_i is the relative location of atom i in $\mathbb{R}^3$. We randomly split 80%/20% for training and test sets, and repeat 20 times. As in (Peyré et al., 2016), we use k-nearest neighbor (k-NN) regression.

Document classification with non-registered word embeddings. We also evaluate our proposed discrepancies for document classification with non-registered word embeddings in `TWITTER`, `RECIPE`, `CLASSIC`, and `AMAZON` datasets. For each document in these datasets, we apply different *random linear transformation* for *word2vec* word embedding (Mikolov et al., 2013), pre-trained on Google News[9], containing about 3 million words/phrases. *word2vec* maps those words/phrases into $\mathbb{R}^{300}$. Following Kusner et al. (2015); Le et al. (2019b), we remove SMART stop words (Salton and Buckley, 1988), and drop words in documents if they are not in the pre-trained *word2vec*. Therefore, each document can be considered as an empirical measure where each word and its frequency are regarded as a support and its weight respectively. We randomly split 80%/20% for training and test sets, and repeat 20 times.

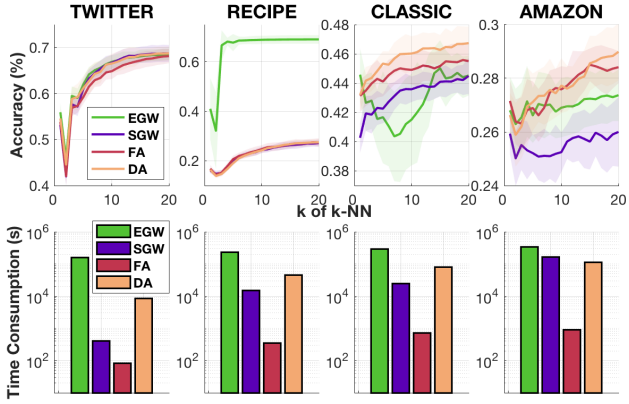


Figure 5: Accuracy and time consumption of k-NN classification on document datasets for EGW (eps=5 in `TWITTER`, and eps=10 in others), SGW (10 slices), FA (10 tree-slices), and DA (1 tree-slice).

Performance results, time consumption and discussions. The results of averaged mean absolute value (MAE) for different k in k-NN regression, and time consumption of quantum chemistry in `qm7` dataset are illustrated in Figure 4, while the results of averaged accuracy for different k in k-NN, and time consumption of document classification with non-registered word embeddings in `TWITTER`, `RECIPE`, `CLASSIC`, and `AMAZON` datasets are shown in Figure 5.

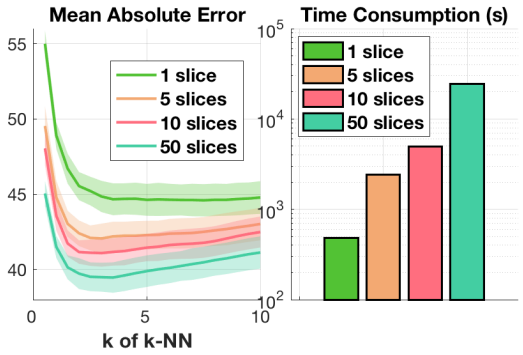


Figure 6: Trade-off between performances and time consumption for FlowAlign w.r.t. tree slices in `qm7`.

The computational time of FlowAlign is at least comparative to that of SGW, and several-order faster than that of EGW. Especially, in `CLASSIC`, it took 12 minutes for FlowAlign (10 slices), while 6.8 hours for SGW (10 slices), and 3.3 days for EGW (entropic regularization eps=10). Moreover, the performances of FlowAlign compare favorably with other baselines, except EGW in RECIPE dataset. FlowAlign performs better when the number of tree slices is increased, but its time consumption is also increased linearly. We show this trade-off on `qm7` in Figure 6. For DepthAlign, its performances are comparative with other baselines. However, DepthAlign is slow in practice due to solving a large number of sub-problems, i.e., aligned-root FlowAlign between corresponding 2-depth-level trees. For EGW, its performances are improved when the entropic regularization small enough for the problems, but its computational time is considerably increased which makes EGW unsuitable for large-scale applications. The value of entropic regularization is important for performances of EGW, e.g., EGW performs well in `RECIPE`, and is comparative with other approaches on other datasets. For SGW, its computational time is slow down when document lengths are large, e.g., in `AMAZON` dataset, since it requires to use extra artificial zeros padding and uniform weights for probability measures with different number of supports (i.e., documents with different lengths), while other approaches work with an original number of supports (i.e., unique words in documents), and general weights (i.e., frequencies of unique words) for supports in probability measures.

Similar to tree metric sampling for TSW (Le et al., 2019b), we observe that the clustering-based tree metric sampling for FlowAlign and DepthAlign is fast and its time consumption is negligible compared to that of either FlowAlign or DepthAlign. For examples, for each

[9] https://code.google.com/p/word2vec

tree metric sampling with the suggested parameters (e.g., the predefined deepest level $H_{\mathcal{T}} = 6$, and the number of clusters $\kappa = 4$ for the farthest-point clustering), it only took about $0.4, 1.5, 11.0, 17.5, 20.5$ seconds for `qm7`, `TWITTER`, `RECIPE`, `CLASSIC`, and `AMAZON` datasets respectively.

We further randomly sample 1 million pairs of distributions in `qm7` dataset and compare FlowAlign $\mathcal{A}_f$ using the optimization in Equation (2) (i.e., optimize the roots r_x, r_z and the transport plan T), and *aligned-root* FlowAlign $\widehat{\mathcal{A}}_f$ in Equation (3) with the heuristic of sampling *aligned-root* tree metrics by using clustering-based tree metric sampling where we choose population means as tree roots (i.e., only optimize the transport plan T with the sampled suboptimal *aligned-root* tree metrics). There are 68.12% of those randomly sampled suboptimal aligned-root tree metrics using the heuristic for $\widehat{\mathcal{A}}_f$ that are actually *optimal* when we optimize $\mathcal{A}_f$ with those sampled trees (the optimization problem for FlowAlign in Equation (2)), and the average relative difference $(\widehat{\mathcal{A}}_f - \mathcal{A}_f)/\mathcal{A}_f$ is 0.0244. Therefore, the sampled *aligned-root* tree metrics are likely suboptimal that not only help to reduce the complexity of the proposed discrepancies by using their aligned-root formulations, but also play as an efficient regularization for the computation of FlowAlign/DepthAlign in applications. Moreover, there is no need for paying too much efforts to optimize the proposed discrepancies with respect to each randomly sampled tree metric since it would lead to overfitting within the computation of FlowAlign/DepthAlign (Peyré and Cuturi, 2019, §8.4), (Le et al., 2019b, §4).

Further experimental results about performances and time consumptions of the discrepancies with different parameters (e.g., entropic regularization in EGW, and number of (tree) slices in SGW, FlowAlign and DepthAlign), and clustering-based tree metric sampling with different parameters (i.e., $H_{\mathcal{T}}, \kappa$ for tree depths and branches respectively) can be seen in the supplementary (§D).

We emphasize that the proposed discrepancies are novel for probability measures in different tree metric spaces, and we do not try to mimic or approximate either GW (with tree metrics) or EGW/SGW. Besides their relations in Proposition 1, we also thoroughly investigate their empirical relations in the supplementary (§G).

7.2 Large-scale FlowAlign barycenter within k-means clustering

We applied FlowAlign barycenter (§3.3), using Algorithm 2 in (Cuturi and Doucet, 2014) where we set `k` = 100 for the maximum number of supports in barycenters, into a larger machine learning pipeline such as k-means clustering on `MNIST` dataset where point clouds of handwritten digits are *rotated arbitrarily* in the plane as in (Peyré et al., 2016). For each handwritten digit, we randomly extracted 6000 point clouds. We evaluated k-means with FlowAlign for 60K, 120K, 240K, 480K, and 960K handwritten-digit point clouds where each handwritten digit is *randomly rotated* $1, 2, 4, 8$, and 16 times respectively. Furthermore, we grouped the handwritten digit 6 and digit 9 together due to applying random rotation. We used k-means++ initialization technique (Arthur and Vassilvitskii, 2007), set 20 for the maximum iterations of k-means, and repeated 10 times with different random seeds for k-means++ initialization. In Figure 7, we show the averaged time consumption and F_β measure (Manning et al., 2008) where β is chosen as in (Le and Cuturi, 2015) for the results of k-means clustering with FlowAlign. Note that, in these settings, the barycenter problem from EGW has extremely slow running time. A small experimental setup for performance comparison can be found in the supplementary (§D).

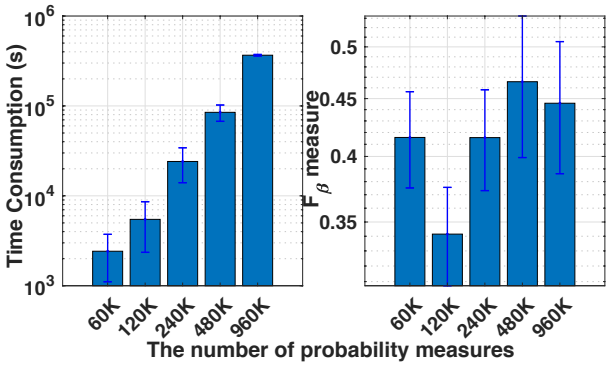


Figure 7: Time consumption and F_β measure for k-means clustering with FlowAlign for *randomly rotated* `MNIST`.

8 Conclusion

We proposed two discrepancies FlowAlign and DepthAlign for probability measures whose supports are in different tree metric spaces. The FlowAlign is not only several order faster than EGW and at least as fast as SGW while remedies its curse of dimensionality, but its performances also compare favorably with those of variants of GW baselines. Moreover, FlowAlign can be applied for large-scale applications (e.g., a million probability measures) which are usually prohibited for (entropic) GW. The questions about sampling efficiently tree metrics from support data points for the proposed discrepancies, or using them for more involved parametric inference are left for the future work.

Acknowledgements

We thank Marco Cuturi for fruitful discussions and anonymous reviewers for their comments. TL acknowledges the support of JSPS KAKENHI Grant number 20K19873. MY was supported by the JSPS KAKENHI Grant Number 20H04243.

References

Altschuler, J., Bach, F., Rudi, A., and Niles-Weed, J. (2019). Massively scalable Sinkhorn distances via the Nyström method. In *Advances in Neural Information Processing Systems*, pages 4429–4439.

Altschuler, J., Weed, J., and Rigollet, P. (2017). Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in neural information processing systems*, pages 1964–1974.

Alvarez-Melis, D. and Jaakkola, T. (2018). Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1881–1890.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035.

Bartal, Y. (1996). Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193.

Bartal, Y. (1998). On approximating arbitrary metrices by tree metrics. In *ACM Symposium on Theory of Computing (STOC)*, volume 98, pages 161–168.

Bhushan Damodaran, B., Kellenberger, B., Flamary, R., Tuia, D., and Courty, N. (2018). Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 447–463.

Bonneel, N., Peyré, G., and Cuturi, M. (2016). Wasserstein barycentric coordinates: histogram regression using optimal transport. *ACM Trans. Graph.*, 35(4):71–1.

Bunne, C., Alvarez-Melis, D., Krause, A., and Jegelka, S. (2019). Learning generative models across incomparable spaces. In *International Conference on Machine Learning*.

Charikar, M., Chekuri, C., Goel, A., Guha, S., and Plotkin, S. (1998). Approximating a finite metric by a small number of tree metrics. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 379–388.

Courty, N., Flamary, R., Habrard, A., and Rakotomamonjy, A. (2017). Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2016). Optimal transport for domain adaptation. *Pattern analysis and machine intelligence (PAMI)*, 39(9):1853–1865.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300.

Cuturi, M. and Doucet, A. (2014). Fast computation of Wasserstein barycenters. In *International conference on machine learning*, pages 685–693.

Dvurechensky, P., Gasnikov, A., and Kroshnin, A. (2018). Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376.

Fakcharoenphol, J., Rao, S., and Talwar, K. (2004). A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497.

Genevay, A., Cuturi, M., Peyré, G., and Bach, F. (2016). Stochastic optimization for large-scale optimal transport. In *Advances in neural information processing systems*, pages 3440–3448.

Genevay, A., Peyre, G., and Cuturi, M. (2018). Learning generative models with sinkhorn divergences. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1608–1617.

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

Grave, E., Joulin, A., and Berthet, Q. (2019). Unsupervised alignment of embeddings with Wasserstein procrustes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1880–1890.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Indyk, P. (2001). Algorithmic applications of low-distortion geometric embeddings. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 10–33.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. (2019). Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. In *International conference on machine learning*, pages 957–966.

Lavenant, H., Claici, S., Chien, E., and Solomon, J. (2018). Dynamical optimal transport on discrete surfaces. In *SIGGRAPH Asia 2018 Technical Papers*, page 250. ACM.

Le, T. and Cuturi, M. (2015). Unsupervised Riemannian metric learning for histograms using Aitchison transformations. In *International Conference on Machine Learning*, pages 2002–2011.

Le, T., Huynh, V., Ho, N., Phung, D., and Yamada, M. (2019a). On scalable variant of wasserstein barycenter. *arXiv preprint arXiv:1910.04483*.

Le, T. and Nguyen, T. (2021). Entropy partial transport with tree metrics: Theory and practice. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Le, T., Yamada, M., Fukumizu, K., and Cuturi, M. (2019b). Tree-sliced variants of Wasserstein distances. In *Advances in neural information processing systems*, pages 12283–12294.

Lin, T., Ho, N., and Jordan, M. (2019). On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3982–3991.

Liutkus, A., Simsekli, U., Majewski, S., Durmus, A., and Stöter, F.-R. (2019). Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4104–4113.

Luise, G., Salzo, S., Pontil, M., and Ciliberto, C. (2019). Sinkhorn barycenters with free support via Frank-Wolfe algorithm. In *Advances in Neural Information Processing Systems*, pages 9318–9329.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.

Mémoli, F. (2011). Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11(4):417–487.

Mémoli, F., Munk, A., Wan, Z., and Weitkamp, C. (2021). The ultrametric gromov-wasserstein distance. *arXiv preprint arXiv:2101.05756*.

Mémoli, F., Smith, Z., and Wan, Z. (2019). Gromov-hausdorff distances on p-metric spaces and ultrametric spaces. *arXiv preprint arXiv:1912.00564*.

Mena, G. and Niles-Weed, J. (2019). Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. In *Advances in Neural Information Processing Systems*, pages 4543–4553.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Muzellec, B. and Cuturi, M. (2018). Generalizing point embeddings using the wasserstein space of elliptical distributions. In *Advances in Neural Information Processing Systems*, pages 10237–10248.

Nadjahi, K., Durmus, A., Simsekli, U., and Badeau, R. (2019). Asymptotic guarantees for learning generative models with the sliced-Wasserstein distance. In *Advances in Neural Information Processing Systems*, pages 250–260.

Paty, F.-P. and Cuturi, M. (2019). Subspace robust Wasserstein distances. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5072–5081.

Perrot, M., Courty, N., Flamary, R., and Habrard, A. (2016). Mapping estimation for discrete optimal transport. In *Advances in Neural Information Processing Systems*, pages 4197–4205.

Peyré, G. and Cuturi, M. (2019). Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Peyré, G., Cuturi, M., and Solomon, J. (2016). Gromov-Wasserstein averaging of kernel and distance matrices. In *Proceedings of the International Conference on Machine Learning*.

Peyré, G., Cuturi, M., and Solomon, J. (2016). Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. (2011). Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 435–446.

Redko, I., Courty, N., Flamary, R., and Tuia, D. (2019). Optimal transport for multi-source domain adaptation under target shift. In *International Conference on Artificial Intelligence and Statistics*, pages 849–858.

Rupp, M., Tkatchenko, A., Müller, K.-R., and Von Lilienfeld, O. A. (2012). Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Santambrogio, F. (2015). *Optimal transport for applied mathematicians*. Birkäuser.

Schmitzer, B. (2019). Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481.

Semple, C. and Steel, M. (2003). Phylogenetics. *Oxford Lecture Series in Mathematics and its Applications*.

Shkarin, S. (2004). Isometric embedding of finite ultrametric spaces in banach spaces. *Topology and its Applications*, 142(1-3):13–17.

Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. (2015). Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66.

Solomon, J., Peyré, G., Kim, V. G., and Sra, S. (2016). Entropic metric alignment for correspondence problems. *ACM Transactions on Graphics (TOG)*, 35(4):72.

Solomon, J. and Vaxman, A. (2019). Optimal transport-based polar interpolation of directional fields. *ACM Transactions on Graphics (TOG)*, 38(4):1–13.

Sturm, K.-T. et al. (2006). On the geometry of metric measure spaces. *Acta mathematica*, 196(1):65–131.

Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., and Borgwardt, K. (2019). Wasserstein Weisfeiler-Lehman graph kernels. In *Advances in Neural Information Processing Systems*, pages 6436–6446.

Vayer, T., Flamary, R., Tavenard, R., Chapel, L., and Courty, N. (2019). Sliced Gromov-Wasserstein. *Advances in Neural Information Processing Systems*.

Wu, J., Huang, Z., Acharya, D., Li, W., Thoma, J., Paudel, D. P., and Gool, L. V. (2019). Sliced Wasserstein generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3713–3722.

Xu, H., Luo, D., and Carin, L. (2019a). Scalable Gromov-Wasserstein learning for graph partitioning and matching. In *Advances in neural information processing systems*.

Xu, H., Luo, D., Zha, H., and Carin, L. (2019b). Gromov-Wasserstein learning for graph matching and node embedding. In *International conference on machine learning*.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

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

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

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 $\mathcal{A}_f^2(\mu,\nu) = 0$. As discussed in the main text, when $\mathcal{A}_f^2(\mu,\nu) = 0$, 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

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

where $\mathcal{M}_h^{1,2}, \mathcal{M}_h^{1,3}, \mathcal{M}_h^{2,3}$ 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 $(x,y)\in\mathcal{M}_{h-1}^{1,2}, (x,z)\in\mathcal{M}_{h-1}^{1,3}, (y,z)\in\mathcal{M}_{h-1}^{2,3}$. In order to demonstrate the above inequality, we only need to verify that

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

for any depth level $h \geq 1$. We respectively denote $\mathcal{T}_X^h = \{\bar{x}_1^{(h)},\ldots,\bar{x}_{k_{1,h}}^{(h)}\}, \mathcal{T}_Y^h = \{\bar{y}_1^{(h)},\ldots,\bar{y}_{k_{2,h}}^{(h)}\}, \mathcal{T}_Z^h = \{\bar{z}_1^{(h)},\ldots,\bar{z}_{k_{3,h}}^{(h)}\}$ 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

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

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

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

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 $x\in\mathcal{S}(\bar{x}_i^{(h-1)})$ and $y\in\mathcal{S}(\bar{y}_j^{(h-1)})$. Now, we have the

following inequalities

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

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 $\widehat{\mathcal{A}}_f(\mu_{\mathcal{T}^2},\nu_{\mathcal{T}^2};\cdot,\cdot)$ 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 $\mu_{\mathcal{T}^2},\nu_{\mathcal{T}^2}$ 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

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

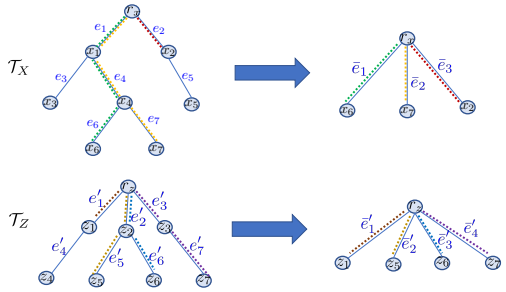
where the inequality is due to the standard Cauchy-Schwarz inequality $(a+b)^2 \le 2(a^2+b^2)$ for any $a,b\in\mathbb{R}$. By taking the infimum over r_x, r_z, then square root on both sides of the above inequality (7), we obtain the conclusion of the proposition that

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

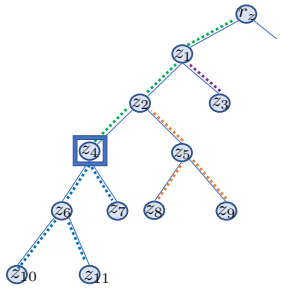
B Further illustrations, geometric properties, and discussion about rotational and translational invariance for *FlowAlign*

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., $x_1 \leq x_2 \leq \cdots \leq x_n$, and $z_1 \leq z_2 \leq \cdots \leq z_m$), and a ground distance ℓ (e.g., $\ell(x,z) = \ell_1(x,z) = |x-z|$).
Output: OT distance d and optimal transport plan T.

Initialize $d \leftarrow 0$, $T \leftarrow 0_{n\times m}$, $i \leftarrow 1$, $j \leftarrow 1$.
while $i \leq n$ and $j \leq m$ **do**
  if $a_i \leq b_j$ **then**
    $T_{ij} \leftarrow a_i$.
    $d \leftarrow d + a_i\ell(x_i, z_j)$.
    Update $b_j \leftarrow b_j - a_i$, $i \leftarrow i+1$.
    if $b_j == 0$ **then**
      $j \leftarrow j+1$.
    end if
  else
    $T_{ij} \leftarrow b_j$.
    $d \leftarrow d + b_j\ell(x_i, z_j)$.
    Update $a_i \leftarrow a_i - b_j$, $j \leftarrow j+1$.
    if $a_i == 0$ **then**
      $i \leftarrow i+1$.
    end if
  end if
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 μ, ν respectively, e.g., by choosing a mean of support data as its root for the clustering-based tree metric sampling [6].
2: Construct $\tilde{\mu} \leftarrow \sum_{i\in[n]} a_i\delta_{d_{\mathrm{T}_X}(r_x,x_i)}$ and $\tilde{\nu} \leftarrow \sum_{j\in[m]} b_j\delta_{d_{\mathrm{T}_Z}(r_z,z_j)}$
3: Sort support(s) $d_{\mathrm{T}_X}(r_x, x_i)\,|_{i\in[n]}$ and $d_{\mathrm{T}_Z}(r_z, z_j)\,|_{j\in[m]}$, 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 $x_i\,|_{i\in[n]}$, and $z_j\,|_{j\in[m]}$ 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 $N \leq \left(\kappa^{H_{\mathcal{T}}} - 1\right) / (\kappa - 1)$.
- 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 $n \leq N$ (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.

```
Initialize d ← ∞.
for each x ∈ T_X do
    Construct μ̃ ← Σ_{i∈[n]} a_i δ_{d_{T_X}(x, x_i)}, then sort support(s) d_{T_X}(x, x_i) |_{i∈[n]}.
    for each z ∈ T_Z do
        Construct ν̃ ← Σ_{j∈[m]} b_j δ_{d_{T_Z}(z, z_j)}, then sort support(s) d_{T_Z}(z, z_j) |_{j∈[m]}.
        d̃ ← Algorithm 1 for μ̃ and ν̃.
        if d > d̃ then
            d ← d̃.
        end if
    end for
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 $\mathcal{O}(\bar{N}H_{\mathcal{T}}\log\kappa + nH_{\mathcal{T}} + n\log n)$, 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 $\mathcal{O}(\bar{N}H_{\mathcal{T}}\log\kappa + N^2nH_{\mathcal{T}} + N^2n\log n + N^2)$[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 $x_i\ |_{i\in[n]}$ in tree $\mathcal{T}_X$, and $z_j\ |_{j\in[m]}$ in tree $\mathcal{T}_Z$.
Output: *FlowAlign* discrepancy d.

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 $x \in \mathcal{T}_X$.
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 $\mathcal{T}_X$.
Sort supports $d_{\mathrm{T}_X}(r_x, x_i)\ |_{i\in[n]}$.
for each $x \in \mathcal{T}_X$ and $x \neq r_x$ **do**
  % *Change root from* r_x *to* x *for tree* $\mathcal{T}_X$
  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 $d_{\mathrm{T}_X}(x, x_i) = d_{\mathrm{T}_X}(r_x, x) + d_{\mathrm{T}_X}(r_x, x_i) - 2d_{\mathrm{T}_X}(r_x, \zeta_i)$ where ζ_i is the closet ancestor of x and x_i and $\zeta_i \in \mathcal{P}(r_x, x)$, $\zeta_i \in \mathcal{P}(r_x, x_i)$).
  Sort supports $d_{\mathrm{T}_X}(x, x_i)\ |_{i\in[n]}$. (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).
end for
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 $z \in \mathcal{T}_Z$.
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 $\mathcal{T}_Z$.
Sort supports $d_{\mathrm{T}_Z}(r_z, z_j)\ |_{j\in[m]}$.
for each $z \in \mathcal{T}_Z$ and $z \neq r_z$ **do**
  % *Change root from* r_z *to* z *for tree* $\mathcal{T}_Z$
  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 $d_{\mathrm{T}_Z}(z, z_j) = d_{\mathrm{T}_Z}(r_z, z) + d_{\mathrm{T}_Z}(r_z, z_j) - 2d_{\mathrm{T}_Z}(r_z, \zeta_j)$ where ζ_j is the closet ancestor of z and z_j and $\zeta_j \in \mathcal{P}(r_z, z)$, $\zeta_j \in \mathcal{P}(r_z, z_i)$).
  Sort supports $d_{\mathrm{T}_Z}(z, z_i)\ |_{j\in[m]}$. (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).
end for
Initialize $d \leftarrow \infty$.
for each $x \in \mathcal{T}_X$ **do**
  Retrieve $\tilde{\mu}_x$ with sorted support(s).
  for each $z \in \mathcal{T}_Z$ **do**
    Retrieve $\tilde{\nu}_z$ with sorted support(s).
    $\tilde{d} \leftarrow$ Algorithm 1 for $\tilde{\mu}_x$ and $\tilde{\nu}_z$.
    if $d > \tilde{d}$ **then**
      $d \leftarrow \tilde{d}$.
    end if
  end for
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 $\mathcal{O}(\bar{N}H_{\mathcal{T}}\log\kappa + NnH_{\mathcal{T}} + n\log n + N^2)$, 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 $\mathcal{O}(nH_{\mathcal{T}})$ for N times due to changing a root in a tree; sort tree metrics between a root to each support for only 1 time; and compare aligned-root *FlowAlign* results for N^2 cases of pairs of roots.
- or nearly into $\mathcal{O}(\bar{N}H_{\mathcal{T}}\log\kappa + NnH_{\mathcal{T}} + n\log n + Nn + N^2)$, 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 $\mathcal{O}(\tilde{N} H_{\mathcal{T}} \log \kappa + NnH_{\mathcal{T}} + Nn \log n + N^2)$, 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 $x_i \,|_{i \in [n]}$ and $z_j \,|_{j \in [m]}$ 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 μ.

1: Construct a set of paths $S_{\tilde{p}}$ for supports $x_i \,|_{i \in [n]}$ 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 μ, 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 μ.

[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.

Sample aligned-root tree metrics $\mathcal{T}_X, \mathcal{T}_Z$ for μ, ν respectively, e.g., by choosing a mean of support data as its root for the clustering-based tree metric sampling [6].
Construct $S_{\mathcal{T}_X}, S_{\mathcal{T}_Z}$: sets of 2-depth-level trees for μ, ν in tree $\mathcal{T}_X, \mathcal{T}_Z$ respectively by using Algorithm 5.
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 $d \leftarrow 0$.
Push $\{(r_x, r_z), T^*(r_x, r_z)\}$ into a queue `Q`.
while `Q` is not empty **do**
  Pull $(x, z), T^*(x, z)$ from queue `Q`.
  Get corresponding 2-depth-level trees: $\mathcal{T}_x^2, \mathcal{T}_z^2$ for μ, ν from $S_{\mathcal{T}_X}, S_{\mathcal{T}_Z}$ respectively.
  % For two simple 2-depth-level trees, the discrepancy between $\mu_{\mathcal{T}_x^2}, \nu_{\mathcal{T}_z^2}$ is equal to 0.
  if One of two 2-depth-level trees $\mathcal{T}_x^2, \mathcal{T}_z^2$ is simple, but not both of them **then**
    if $\mathcal{T}_x^2$ is simple **then**
      Get all paths from x to each support of μ in the subtree of $\mathcal{T}_X$ rooted at x.
      Normalize for weights of those supports of corresponding paths.
      Compute $\tilde{d}$ as a sum of weighted lengths for those paths.
      $d \leftarrow d + T^*(x, z)\tilde{d}$.
    else
      Get all paths from z to each support of ν in the subtree of $\mathcal{T}_Z$ rooted at z.
      Normalize for weights of those supports of corresponding paths.
      Compute $\tilde{d}$ as a sum of weighted lengths for those paths.
      $d \leftarrow d + T^*(x, z)\tilde{d}$.
    end if
  else if Two 2-depth-level trees $\mathcal{T}_x^2, \mathcal{T}_z^2$ are not simple **then**
    Sort the distances from a root to each node in tree $\mathcal{T}_x^2, \mathcal{T}_z^2$.
    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.
    $d \leftarrow d + T^*(x, z)\tilde{d}$.
    Compute weighted optimal transport plan $\tilde{T}^* \leftarrow T^*(x, z)\tilde{T}^*$.
    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`.
  end if
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 $\mathcal{O}(nH_{\mathcal{T}}\kappa)$ (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 $\mathcal{O}(\bar{N} H_{\mathcal{T}} \log \kappa + nH_{\mathcal{T}}\kappa + \Delta\kappa \log \kappa)$.

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 $\mathcal{O}(N^2 nH_{\mathcal{T}}\kappa + N^2\Delta\kappa \log \kappa)$.

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.

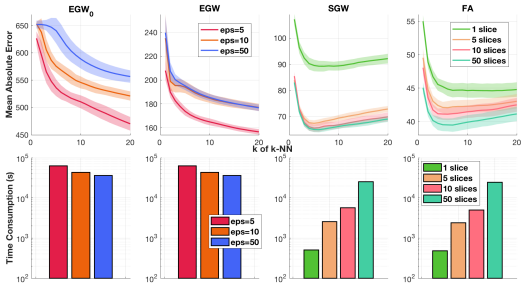
D.1 Further experimental results on quantum chemistry and document classification

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)

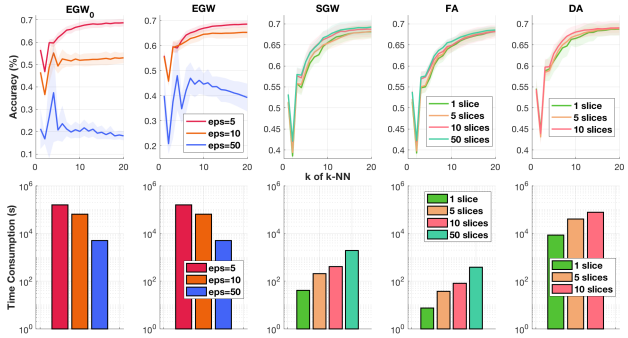
- for quantum chemistry (`qm7` dataset) in Figure 2a,
- for document classification
  - in `TWITTER` dataset in Figure 2b,
  - in `RECIPE` dataset in Figure 2c,
  - in `CLASSIC` dataset in Figure 2d,
  - in `AMAZON` dataset in Figure 2e.

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.

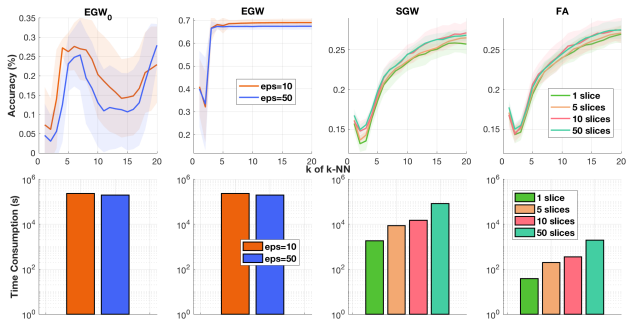
[9] Assume that for each tree metric, each node has at most κ child nodes, the tree deep is $H_{\mathcal{T}}$, and the number of nodes in tree is about N.



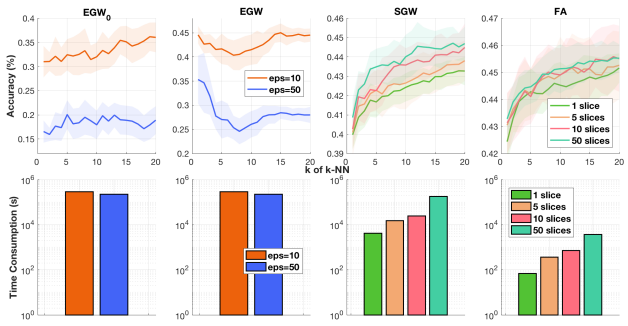
(a) In `qm7` dataset.



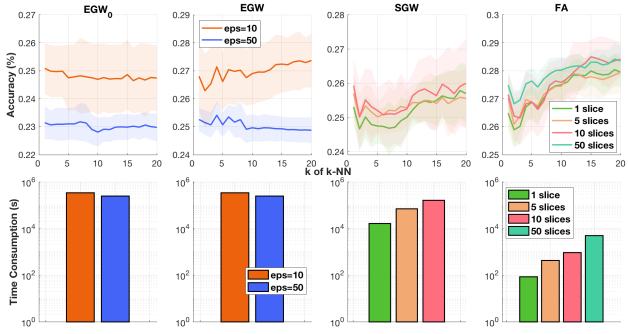
(b) In `TWITTER` dataset.



(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.

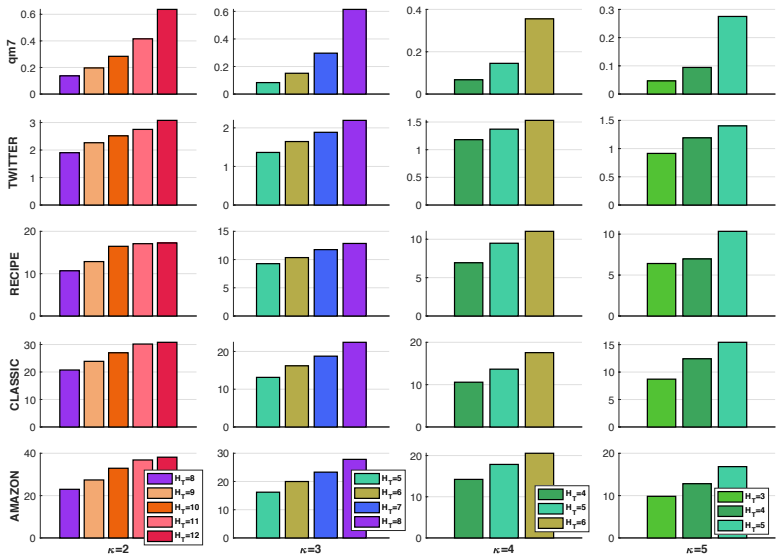


Figure 3: Time consumption (seconds) for a tree metric sampling in *FlowAlign* and *DepthAlign* by the clustering-based tree metric method [6] with different parameters (e.g. the predefined deepest level $H_{\mathcal{T}}$, the number of clusters κ) in quantum chemistry (`qm7` dataset), and document classification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

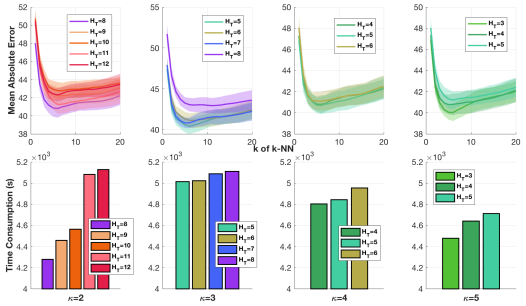
D.3 Experiment results with different parameters for tree metric sampling

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:

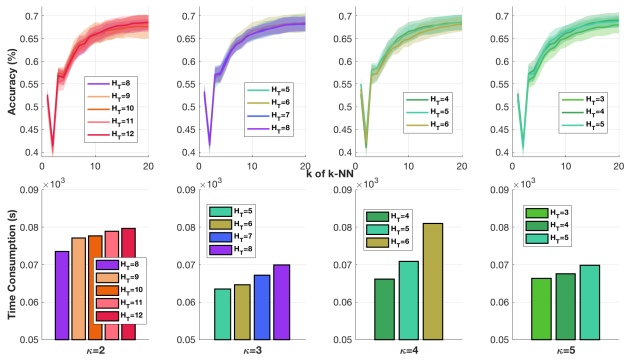
- in `qm7` dataset in Figure 4a,
- in `TWITTER` dataset in Figure 4b,
- in `RECIPE` dataset in Figure 4c,
- in `CLASSIC` dataset in Figure 4d,
- in `AMAZON` dataset in Figure 4e.

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

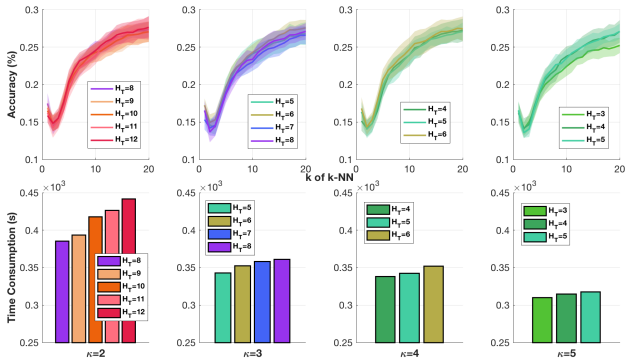
We follow the small experimental setup for performance comparison on *random rotated* `MNIST` dataset as in [9]. We randomly select 50 point clouds from each digits 0 to 4, apply k-means clustering with $k = 5$, and k-means++



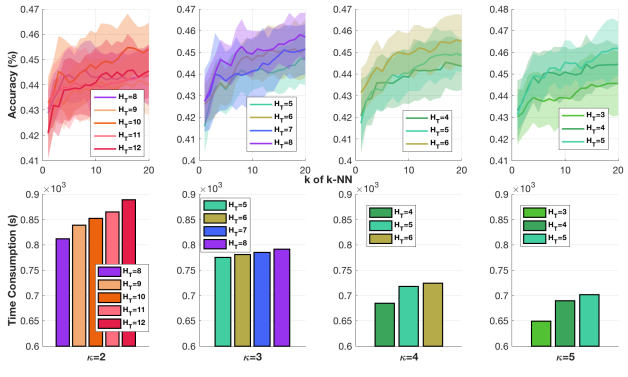
(a) In `qm7` dataset.



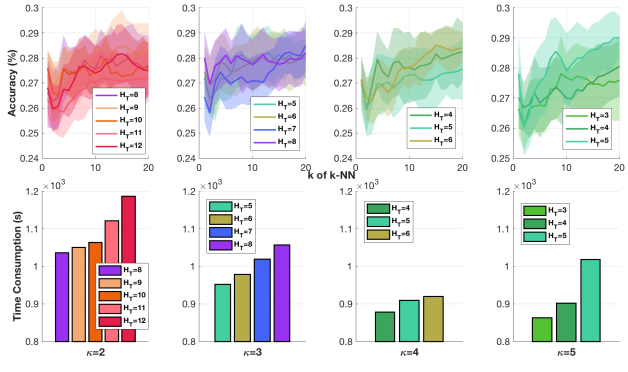
(b) In `TWITTER` dataset.



(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.

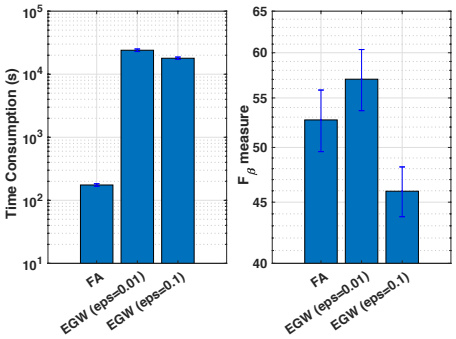


Figure 5: Results of k-means clustering on a small experiment setup for performance comparison on *random rotated* `MNIST` dataset.

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: $X = \{x_i \mid_{i\in[n]}\}$ is a set of n input data points, and κ is the predefined number of clusters for the farthest-point clustering.
Output: A set of clustering centers $C = \{c_i \mid_{i\in[\kappa]}\}$, and cluster indices for $x_i \mid_{i\in[n]}$.

Initialize $C \leftarrow \varnothing$.
$c_1 \leftarrow$ a random data point $x \in X$.
$C \leftarrow c_1$.
$i \leftarrow 1$.
while $i < \kappa$ and $n - i > 0$ **do**
  $i \leftarrow i + 1$.
  $c_i \leftarrow \max_{x\in X} \min_{c\in C} \|x - c\|$. % (find the farthest point $x \in X$ to C).
  $C \leftarrow C \cup c_i$. % (add the new cluster center into C).
end while
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 $\mathcal{T}$, κ is the predefined number of clusters for the farthest-point clustering.
Output: tree metric $\mathcal{T}$

if $m > 0$ **then**
  if $h > 0$ **then**
    Node $\tilde{x}_c \leftarrow$ a center of X, e.g., the mean data point of X.
    Length of edge $(\tilde{x}_p, \tilde{x}_c) \leftarrow$ distance $(\tilde{x}_p, \tilde{x}_c)$.
  else
    Node $\tilde{x}_c \leftarrow \tilde{x}_p$.
  end if
  if $m > 1$ and $h < H_{\mathcal{T}}$ **then**
    Run the farthest-point clustering for X into κ clusters $X_i \mid_{i \in [\kappa]}$.
    for each cluster $X_i \mid_{i \in [\kappa]}$ **do**
      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 κ for the farthest-point clustering).
    end for
  end if
end if

E.3 Tree metric

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

Definition 1. *A metric $d : \Omega \times \Omega \to \mathbb{R}_+$ 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

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

and recall

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

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:

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

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; $\mathtt{D}$, $\mathtt{S}$ 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.

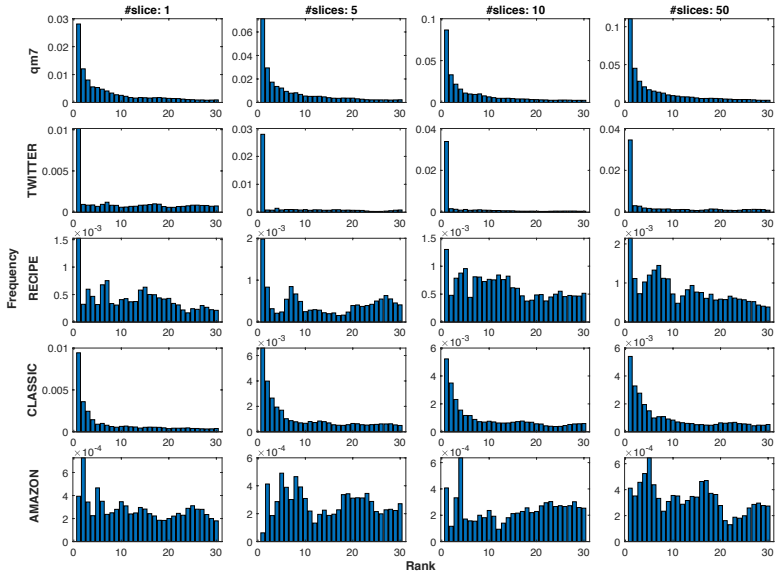


Figure 6: Empirical relation: $d_\alpha :=$ FA and $d_\beta :=$ DA. We used DA with 1 tree slice for `qm7`, 10 tree slices for `TWITTER`, 5 tree slices for `RECIPE`, 1 tree slice for `CLASSIC`, and 1 tree slice for `AMAZON`.

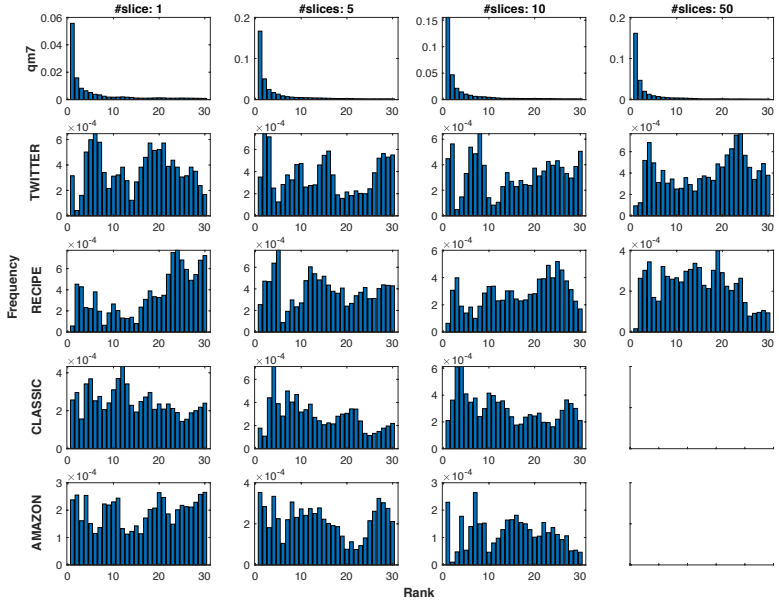


Figure 7: Empirical relation: $d_\alpha :=$ SGW and $d_\beta :=$ FA (10 tree slices). For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger number of slices.

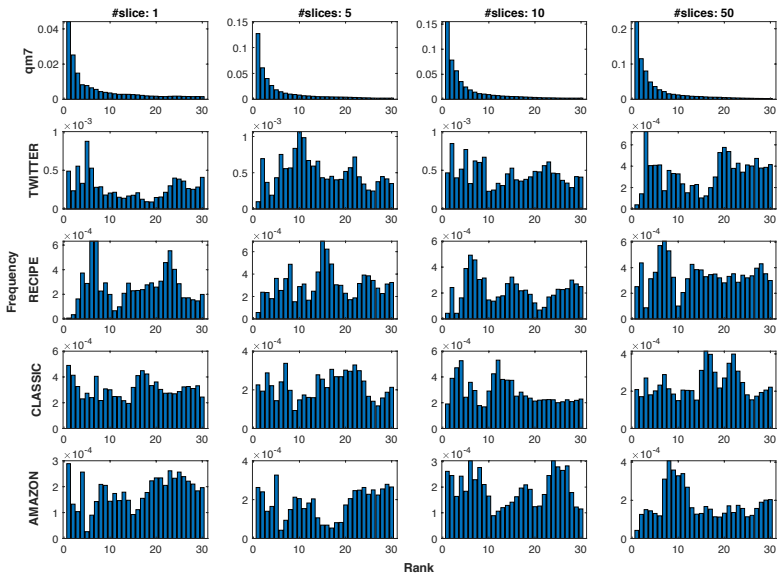


Figure 8: Empirical relation: $d_\alpha :=$ FA and $d_\beta :=$ SGW (10 slices).

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.

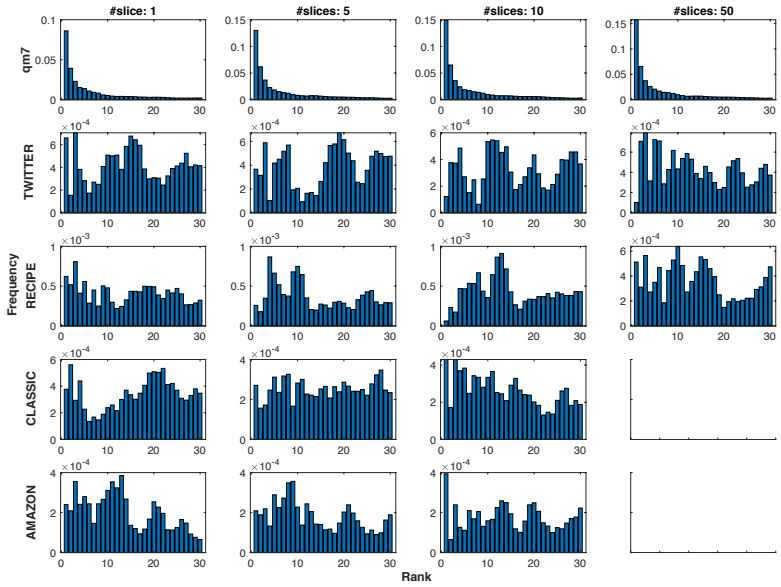


Figure 9: Empirical relation: $d_\alpha :=$ SGW and $d_\beta :=$ DA.We used DA with 1 tree slice for `qm7`, 10 tree slices for `TWITTER`, 5 tree slices for `RECIPE`, 1 tree slice for `CLASSIC`, and 1 tree slice for `AMAZON`. For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due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

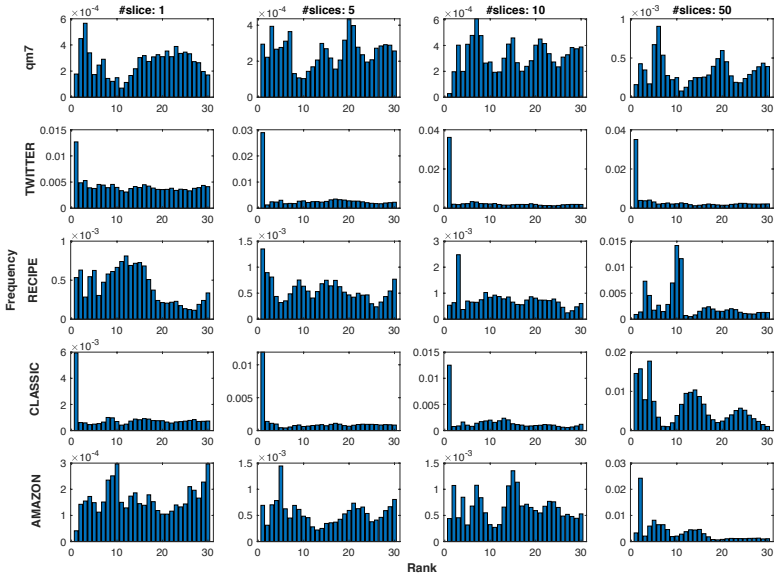


Figure 10: Empirical relation: $d_\alpha :=$ FA and $d_\beta :=$ EGW$_0$. The entropic regularization for EGW$_0$ is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets.

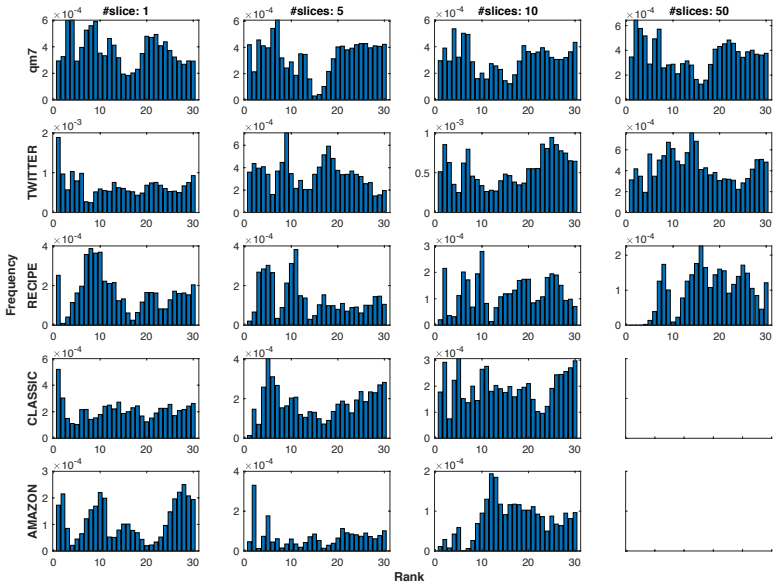


Figure 11: Empirical relation: $d_\alpha :=$ SGW and $d_\beta :=$ EGW$_0$. The entropic regularization for EGW$_0$ is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets. For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with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

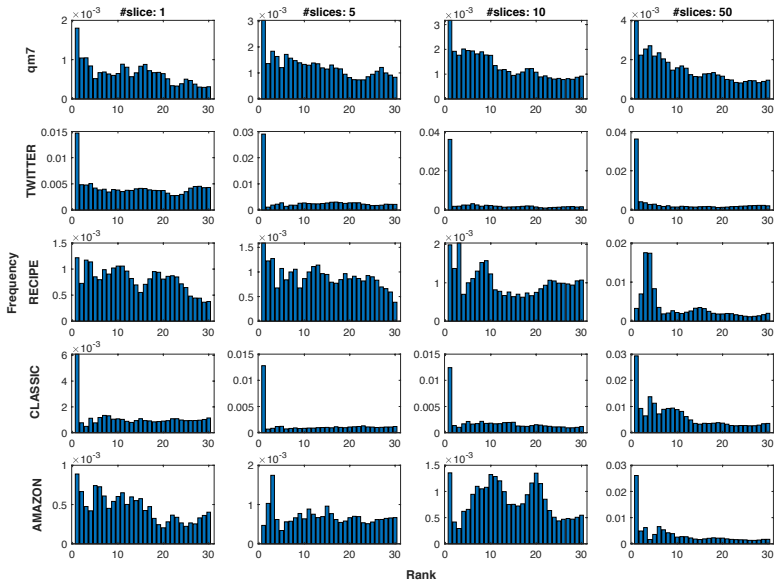


Figure 12: Empirical relation: $d_\alpha :=$ FA and $d_\beta :=$ EGW. The entropic regularization for EGW is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets.

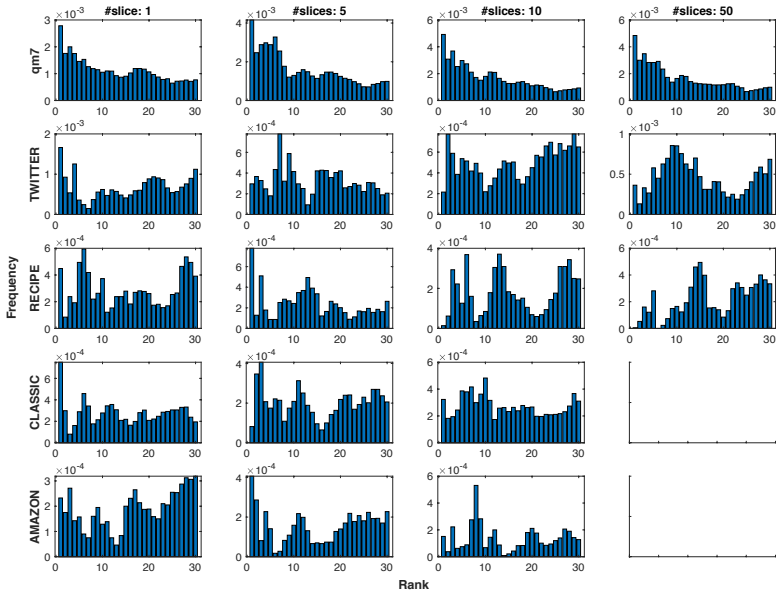


Figure 13: Empirical relation: $d_\alpha :=$ SGW and $d_\beta :=$ EGW. The entropic regularization for EGW is set 5 for `qm7` and `TWITTER` datasets, and 10 for `RECIPE, CLASSIC, AMAZON` datasets. For SGW in `CLASSIC, AMAZON` datasets, we only evaluate it until 10 slices due to its slowness with a larger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