Fast 1-Wasserstein Distance Approximations Using Greedy Strategies

Guillaume Houry
ENS Paris-Saclay

Han Bao
Kyoto University

Han Zhao
UIUC

Makoto Yamada
OIST

Abstract

Among numerous linear approximation methods proposed for optimal transport (OT), tree-based methods appear to be fairly reliable, notably for language processing applications. Inspired by these tree methods, we introduce several greedy heuristics aiming to compute even faster approximations of OT. We first explicitly establish the equivalence between greedy matching and optimal transport for tree metrics, and then we show that tree greedy matching can be reduced to greedy matching on a one-dimensional line. Next, we propose two new greedy-based algorithms in one dimension: the k-Greedy and 1D-ICT algorithms. This novel approach provides Wasserstein approximations with accuracy similar to the original tree methods on text datasets while being faster in practice. Finally, these algorithms are applicable beyond tree approximations: using sliced projections of the original data still provides fairly good accuracy while eliminating the need for embedding the data in a fixed and rigid tree structure. This property makes these approaches even more versatile than the original tree OT methods.

1 INTRODUCTION

Optimal transport (OT) is an optimization problem that searches for the best matching between two distributions given a ground cost defined on their supports. It has recently gained a lot of interest from the data science community due to its ability to provide conceptual tools and efficient algorithms to handle complex data such as point clouds and probability measures. Indeed, OT is substantially related to the Wasserstein distance between probability measures (Villani, 2009). It has found many applications in various fields, including computer vision (Rabin et al., 2012, 2014), genomics (Kimmel et al., 2019; Schiebinger et al., 2019; Yang et al., 2020), language processing (Brokos et al., 2016; Kusner et al., 2015; Zhao et al., 2019), robust learning (Courty et al., 2017; Tachet des Combes et al., 2020), and fairness (Xian et al., 2023; Zhao, 2022). One key application of OT in language processing is document similarity search: document retrieval is performed by searching for the smallest Wasserstein distance between a query text and a document database. It often requires solving many instances of OT in a row, hence requiring fast algorithms for computing Wasserstein distances.

However, computing exact Wasserstein distances generally takes cubic time in the input size (Pele and Werman, 2009), which is highly demanding in practice. A popular relaxation of OT known as the Sinkhorn distance (Cuturi, 2013) and its extension (Lin et al., 2019) remain to require quadratic time. Several recent studies have tackled this computational bottleneck and reached quasi-linear complexity with additional heuristics by adding low-rank conditions to the Sinkhorn algorithm (Scetbon and Cuturi, 2022; Scetbon et al., 2021), using vector embeddings of the input distributions (Shirdhonkar and Jacobs, 2008; Tong et al., 2021; Wang et al., 2013), and formulating OT as the solution of a dynamical system that can be solved using fast numerical schemes (Li et al., 2018a,b; Liu et al., 2018).

Although very efficient for some categories of applications, these methods are not suited for the document similarity search. Indeed, the document retrieval task has two characteristics that distinguish it from most other OT applications: the ground space is high dimensional, and the support of each distribution is typically small compared to the vocabulary size. One one hand, the methods based on a dynamical formu-

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

lation of OT are designed for low dimensional inputs and rely on a grid discretization of the space, thus becoming too costly to be applied efficiently on higher dimensional settings. On the other hand, low-rank and embedding-based methods work well in high dimensions, but the computations involve the full cost matrix, meaning that they are mainly efficient when the input histograms are dense. Hence, these cannot be efficiently applied to document retrieval tasks.

Therefore, several methods were specifically designed for text applications, providing simple and efficient document retrieval implementations. Some popular algorithms are based on the relaxation of several constraints of the original OT problem, like the Relaxed Word Mover Distance (Atasu et al., 2017; Kusner et al., 2015) and the Approximate Iterative Constrained Transfers (Atasu and Mittelholzer, 2019)). More recently, another line of work used tree approximations of the ground space to take advantage of linear-time computability of tree Wasserstein distances (TWD; Le et al. (2019); Yamada et al. (2022)), providing significantly faster OT methods for document search at the cost of a lower accuracy. While the approximation performance of tree-based methods degrades, it can be drastically improved by the Flowtree algorithm (Backurs et al., 2020). However, the Flowtree improvement comes at a large amount of computational overhead. In this work, we seek a technical device to achieve a comparable approximation performance to Flowtree yet with a better computational overhead.

To better reconcile the tradeoff between accuracy and computational complexity, we propose two algorithms to approximately compute the Wasserstein distance based on greedy matching on a one-dimensional line. Historically, greedy algorithms have been used to solve optimal matching, which is closely related to optimal transport (Avis, 1983). After reviewing the connection between greedy matching and Flowtree (Section 3), we show two strategies to accelerate Flowtree by projecting tree nodes into a one-dimensional line (Section 4). Because greedy matching can be performed in nearly linear time, the proposed methods can run faster than Flowtree while enjoying the same approximation performances. Finally, in Section 5, we assess the performances of our method on text document datasets, demonstrating the relevance of our approach for language processing tasks. Overall, our contributions can be summarized as follows:

- We prove that tree OT can be reduced to a one-dimensional greedy matching problem.
- We introduce two new algorithms that approximate 1D greedy matching: k-Greedy and 1D-ICT algorithms. Our end-to-end approach provides strong approximations of 1-Wasserstein distances on text datasets while being up to twice faster than the Flowtree baseline. Moreover, k-Greedy admits Flowtree as a limit case, making it a natural alternative to the latter method.
- Additionally, we propose to use k-Greedy and 1D-ICT algorithms with sliced projections of the input data. While remaining fairly accurate, this approach is more versatile than the tree OT methods: unlike tree embeddings, sliced projections do not require fixed-support distribution, making this approach compatible with adaptive vocabulary or with more elaborate models (e.g. using attention (Sonkar et al., 2020) or dynamic (Bamler and Mandt, 2017) word embeddings).

2 BACKGROUND

OT is an optimization problem that computes the smallest transport cost required to transform one probability distribution into another. Definition 1 gives its formulation in the discrete case.

Definition 1 (Optimal transport (Villani, 2009))**.** *Let $C \in \mathbb{R}_+^{n\times m}$ be a cost matrix, $\mu \in \mathbb{R}_+^n$ and $\nu \in \mathbb{R}_+^m$ two probability vectors, satisfying $\sum_{i=1}^n \mu_i = 1$ and $\sum_{j=1}^m \nu_j = 1$, respectively. The* optimal transport *between μ and ν is the following optimization problem:*

$$L_C(\mu,\nu) = \min_{\Pi\in\mathcal{U}(\mu,\nu)} \langle C, \Pi\rangle,$$

where $\mathcal{U}(\mu,\nu) = \{\Pi \in \mathbb{R}_+^{n\times m} \mid \Pi\mathbf{1}_n = \mu,\ \Pi^T\mathbf{1}_m = \nu\}$.

The case where $n = m$ and both μ, ν are uniform corresponds to the original optimal transportation problem as formulated by Gaspard Monge (Monge, 1781). It is a special case of minimum weight matching, a classical discrete optimization problem; see, e.g., Avis (1983) for a survey on related problems.

When the cost C is a power of a distance matrix, OT satisfies several geometric properties that make it appealing to many theoretical and computational applications. Indeed, the OT solution then defines a distance on the set of probability measures, called Wasserstein distance (Definition 2). This work mainly focuses on the 1-Wasserstein case, which we will denote by $W(\cdot,\cdot)$ for conciseness.

Definition 2 (Wasserstein distances (Villani, 2009))**.** *Let $p > 0$. Let $(\mathcal{X}, d)$ be a finite metric space. Let μ, ν be two probability measures on $\mathcal{X}$, whose supports are denoted by $\mathrm{Supp}(\mu) = \{x_1, \dots, x_n\}$ and $\mathrm{Supp}(\nu) = \{y_1, \dots, y_m\}$, respectively. We define the cost matrix $C_{d,p} = [d(x_i, y_j)^p]_{i,j}$.*

The p-Wasserstein distance *between* μ *and* ν *is defined as* $W_p(\mu,\nu) = L_{C_{d,p}}(\mu,\nu)^{1/p}$ *when* $p \geq 1$, *and* $W_p(\mu,\nu) = L_{C_{d,p}}(\mu,\nu)$ *when* $p \leq 1$. *It metrizes the set of probability measures on* $\mathcal{X}$, *denoted* $\mathcal{M}_+^1(\mathcal{X})$.

The worst-case complexity of exact OT solvers is cubic in the input size, although simplex methods behave quadratically in practice (Bonneel et al., 2011). The entropic (or Sinkhorn) regularizations of OT take quadratic time, with the advantages of being implementable on GPU and differentiable with respect to the input points (Cuturi, 2013), and hence are more suited to large-scale and deep-learning-oriented OT applications. Quadratic complexity yet remains too slow in the most challenging settings.

2.1 Tree Wasserstein Distances

Several linear-time heuristics have been proposed to further scale up Wasserstein computations. Among them, tree methods approximate Wasserstein distances by replacing the true distance d with a tree metric $d_{\mathcal{T},w}$ for some well-chosen tree, in the form given in Definition 3.

Definition 3 (Tree metric)**.** *Let* $\mathcal{T} = (\mathcal{X}, \mathcal{V}, \mathcal{E})$ *be a tree, where* $\mathcal{V}$ *denotes the nodes of the tree,* $\mathcal{X} \subset \mathcal{V}$ *the set of leaves, and* $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ *its edges. Let* $w : \mathcal{E} \to \mathbb{R}_+$ *be a weight function on the edges of* $\mathcal{T}$. *The* tree metric $d_{\mathcal{T},w}$ *is the shortest path distance on* $\mathcal{T}$*:*

$$\forall x, y \in \mathcal{X}, d_{\mathcal{T},w}(x,y) = \min_{u_1 \dots u_k \in \mathcal{C}(x,y)} \sum_{i=1}^{k-1} w_{(u_i,u_{i+1})}$$

with $\mathcal{C}(x,y)$ *the sets of paths* $(u_1,\dots,u_k)$ *from* x *to* y.

The quality of the tree Wasserstein distance (TWD) approximation mainly relies on the choice of the approximation tree $\mathcal{T}$ and edge weights w. Popular tree embeddings used in the literature are Quadtrees (Samet, 1984) and Clustertrees (Gonzalez, 1985). The latter choice allows us to specify the height of the embedding tree, leading to better control over the tree construction. By contrast, Clustertrees do not provide straightforward weightings on the tree edges and require additional constructions to apply TWD.

A closed-form formula of 1-Wasserstein distances on tree metric spaces enables us to compute TWD particularly fast. The resulting tree Wasserstein approximations provide fairly accurate estimates of the true Wasserstein distances and can be further improved by taking the average Wasserstein estimate over several different trees, as investigated in Le et al. (2019) and Takezawa et al. (2021).

2.2 Flowtree

The Flowtree algorithm, proposed in Backurs et al. (2020), provides a strong improvement over the initial TWD by introducing a small variation in the tree methods. Instead of computing the Wasserstein distances directly in the tree embedding space, Flowtree computes the OT matching $\tilde{\Pi}_{\mathcal{T},w}$ in the tree space but computes the Wasserstein distance with the true ground distance. By following the notations of Definition 1, the Flowtree approximation can be written as follows:

$$\tilde{W}(\mu,\nu) = \langle C_d, \tilde{\Pi} \rangle$$
$$\text{where} \quad \tilde{\Pi} \in \operatorname*{arg\,min}_{\Pi \in \mathcal{U}(\mu,\nu)} \langle C_{d_{\mathcal{T},w}}, \Pi \rangle,$$

with C_d denoting the cost matrix for the true distance d and $C_{d_{\mathcal{T},w}}$ the tree approximation cost matrix.

Although drastically more accurate than TWD, this approach is significantly slower. Moreover, the running time of Flowtree is depends on the tree height, which makes it less suited to deep tree embeddings. Hence, there is still potential for enhancing the computational aspect of Flowtree.

3 RELEVANCE OF GREEDY MATCHING AS A PROXY FOR OPTIMAL TRANSPORT

OT is closely related to matching problems, for which greedy strategies are very classical. The principle, described in Algorithm 1, is to match unassigned points iteratively in ascending order of pairwise distance.

Algorithm 1: Greedy Optimal Transport.

input : Cost matrix C of size $n \times m$, normalized weight vectors $\mu \in \mathbb{R}_+^n$, $\nu \in \mathbb{R}_+^m$.
output: $\tilde{\Pi}$ matching matrix of size $n \times m$.

$\tilde{\Pi}_{i,j} = 0$ for $(i,j) \in \{1,\dots,n\} \times \{1,\dots,m\}$
for $(i,j) \in \{1,\dots,n\} \times \{1,\dots,m\}$ *sorted by increasing value of* $C_{i,j}$ **do**
  $u = \min(\mu_i, \nu_j)$
  $\tilde{\Pi}_{i,j} = u$
  $\mu_i = \mu_i - u$
  $\nu_j = \nu_j - u$
return $\tilde{\Pi}$

For maximum matching problems, the greedy matching provides a constant multiplicative bound on the optimal cost (Avis, 1983). Minimum matching does not enjoy such guarantees, with a $\Omega(n^{\log_2(3/2)})$ worst-case approximation ratio, where n denotes the number of points (Reingold and Tarjan, 1981). Since that

Table 1: Relative Error and Correlation with true 1-Wasserstein distance of the greedy OT on some text datasets (introduced in Section 5.1).

DATASET	REL. ERR.	CORR.
TWITTER	$0.4 \cdot 10^{-2}$	0.999
BBC	$3.7 \cdot 10^{-2}$	0.997
AMAZON	$2.2 \cdot 10^{-2}$	0.995
20NEWS	$3.1 \cdot 10^{-2}$	0.994

OT is closely related to minimum matching, we cannot expect strong theoretical guarantees about greedy matching for OT. However, the practical performances of the greedy algorithm are highly promising on both real and synthetic data. The strong empirical results of greedy matchings have been observed in Ottolini and Steinerberger (2023), and our pilot study showed the same conclusions in Table 1 and supplementary material. Note that the worst greedy matching case given in Reingold and Tarjan (1981, Figure 1) corresponds to a very specific arrangement of points separated by exponentially increasing distances, which is too pessimistic in practice. These impressive empirical performances drive us to focus on greedy heuristics for 1-Wasserstein distances.

Greedy matchings arise naturally when working with trees because an OT solution on a tree is always a greedy assignment. Indeed, each tree metric admits an ultrametric sharing the same tree structure (Proposition 1). Moreover, we find that tree metrics with the same tree structure share the same set of OT solutions (Proposition 2) and that the greedy algorithm solves the ultrametric OT exactly (Proposition 3). These results are due to the cyclical monotony of optimal matching, which makes tree OT completely determined by the order of lowest common ancestors in the tree. The proofs are provided in the supplementary material.

Definition 4 (Ultrametric (Leclerc, 1981))**.** *An* ultrametric *is a distance function d such that:*

$$\forall x, y, z \in \mathcal{X}, \quad d(x, z) \leq \max\left(d(x, y), d(y, z)\right).$$

Proposition 1 (Leclerc (1981))**.** *Let $(\mathcal{X}, d)$ be an ultrametric space. Then, there exists a tree $\mathcal{T}$ and a weight function $w : \mathcal{E} \to \mathbb{R}_+$ such that $d = d_{\mathcal{T},w}$. Reciprocally, if $\mathcal{T}$ is a tree, then there is a weight function $w : \mathcal{E} \to \mathbb{R}_+$ such that $d_{\mathcal{T},w}$ is an ultrametric on $\mathcal{X}$.*

Proposition 2. *For each tree $\mathcal{T}$, the set of optimal matchings associated to the metric $d_{\mathcal{T},w}$ is identical for any weight function w on its edges: for any $w, w' : \mathcal{E} \to \mathbb{R}_+$ and $\mu, \nu \in \mathcal{M}_+^1(\mathcal{X})$, we have:*

$$\operatorname*{arg\,min}_{\Pi \in \mathcal{U}(\mu,\nu)} \langle d_{\mathcal{T},w}, \Pi \rangle = \operatorname*{arg\,min}_{\Pi \in \mathcal{U}(\mu,\nu)} \langle d_{\mathcal{T},w'}, \Pi \rangle.$$

Proposition 3. *If $(\mathcal{X}, d)$ is an ultrametric space, then the matching output by Algorithm 1 is optimal.*

Although not explicit in Backurs et al. (2020), the relation between greedy matching and tree OT is central in the Flowtree implementation. The strong results of greedy OT imply that the Flowtree performance is mainly due to the greedy nature of the tree matching rather than the tree metric approximation itself. To improve Flowtree computations, two strategies hence emerge: (i) speeding up the greedy matching itself; (ii) performing greedy matching on other types of embeddings enjoying better properties than trees.

4 GREEDY-BASED HEURISTICS FOR OPTIMAL TRANSPORT

We exploit the link between greedy matching and tree OT to design faster variants of the Flowtree algorithm. Flowtree relies on a tree greedy matching that runs in $O((n + m)h)$ time, where h denotes the tree height. Although satisfactory when the tree is shallow (as is usually the case with Quadtrees), the dependence in h can significantly impact the running time when the vocabulary size is large or when using deeper tree embeddings. For the first time, we propose efficient data structures that eliminate the dependency in h, fastening the greedy matching in deep tree settings.

4.1 Method 1: 1D k-Greedy Matching

Our initial observation is that the nodes of a tree can be encoded in the real line while preserving the pairwise similarity order (Proposition 4). The construction, illustrated in Figure 1, is detailed in the supplementary material (Section B.1). We can therefore reduce the tree greedy matching problem to a 1D greedy assignment.

Proposition 4. *Let $\mathcal{T}$ be a rooted tree and $\mathcal{X}$ its set of leaves. Then, there exists a mapping $f : \mathcal{X} \to \mathbb{R}$ that defines a distance $d_f(x, y) = |f(x) - f(y)|$ such that for each ultrametric d on $\mathcal{T}$ and for each $x, y, z \in \mathcal{X}$,*

$$d(x, y) < d(x, z) \implies d_f(x, y) < d_f(x, z).$$

In particular, the greedy matching on d_f provides an optimal matching for any tree distance on $\mathcal{T}$.

Computing Nearest Neighbors (NN) in one dimension is particularly efficient: as stated in Proposition 5, the NN mapping between two sets of 1D points is always non-decreasing. Exploiting this fact, we can compute the list of all NNs in one single iteration over the data.

Proposition 5. *Let $x_1 \leq \cdots \leq x_n \in \mathbb{R}$ and $y_1 \leq \cdots \leq y_m \in \mathbb{R}$. Let $\tau : \{1, \ldots, n\} \mapsto \{1, \ldots, m\}$ be the*

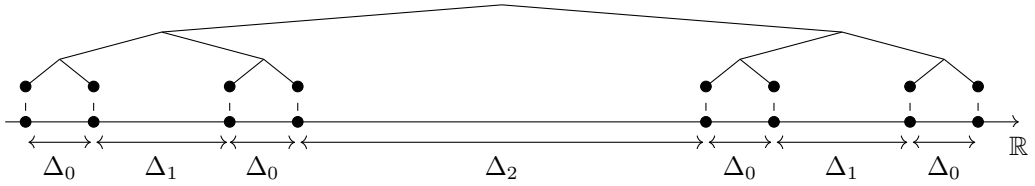

Figure 1: The 1D-embedding of the nodes of a tree. By adjusting the gaps $\Delta_0, \Delta_1, \Delta_2$, we can ensure that the 1D distance between two points increases with the height of their lowest common ancestor.

nearest neighbor map between the two sets of points, defined as:

$$\forall i \in \{1, \dots, n\}, \quad \tau(i) = \underset{j \in \{1,\dots,m\}}{\arg\min} |x_i - y_j|.$$

Then, τ is non-decreasing.

Then, we can compute 1D greedy matchings by the following procedure: while some points remain unmatched, we identify the unmatched point whose distance to its NN is minimal and match this point with its NN. Although computing the NN list is very fast, the global procedure would still have a quadratic running time because the number of loop iterations is too large to offer a better complexity. This can be solved by restricting the number of iterations: after k updates of the NN list, we assign the unmatched points in arbitrary order without updating the NN list anymore. The pseudo-code of this k-Greedy method is given in Algorithm 2, and the detailed implementation is described in the supplementary material (Section B.3).

While $\tilde{k} < k$, k-Greedy matches points by increasing distance. Therefore, when $k \geq n + m$, k-Greedy is actually equivalent to the Flowtree method. For smaller values of k, the furthest pairs of points are matched arbitrarily. Using adequate data structures, the full algorithm can be implemented in $O((k+1)(n+m)\log(n))$ time, offering acceleration compared to the original Flowtree whenever $(k+1)\log(n) \ll h$.

4.2 Method 2: Iterative Constrained Transfer Algorithm

We propose a faster but less precise algorithm for 1D matching, the 1D-ICT algorithm. The Iterative Constrained Transfer (ICT) algorithm, proposed in Atasu and Mittelholzer (2019), solves a relaxed variant of OT where the feasible set $\mathcal{U}(\mu, \nu)$ is replaced with

$$\mathcal{U}_{ICT}(\mu, \nu) = \{\Pi \in \mathbb{R}_+^{n \times m} \mid \Pi \mathbf{1}_n = \mu,\ \Pi_{i,j} \leq \nu_j\ \forall i, j\}.$$

ICT is generally expensive to solve; however, the computations can be made fast in one dimension. Our 1D-ICT algorithm (Algorithm 3) runs in $O(n + m)$ time on uniform input distributions. It is also very data-efficient; besides the input data, it only requires $O(1)$ additional space.

Algorithm 2: 1D k-Greedy algorithm.

input : Points $x_1, \dots, x_n \in \mathcal{X}$ and $y_1, \dots, y_m \in \mathcal{X}$ sorted by values of f, weight vectors $\mu \in \mathbb{R}^n$, $\nu \in \mathbb{R}^m$, mapping function $f : \mathcal{X} \to \mathbb{R}$, the true distance d, $k \geq 0$.
output: The 1-Wasserstein estimate $\tilde{W}$.

```
W̃ = 0
E_x = {1, ..., n}
E_y = {1, ..., m}
τ = nearest neighbor mapping between x_1, ..., x_n and y_1, ..., y_m for the function f.
for k̃ = 0 to k do
    Sort E_x by increasing value of |f(x_i) − f(y_τ(i))|
    for i ∈ E_x do
        label 0
        u = min(μ_i, ν_τ(i))
        W̃ = W̃ + u · d(x_i, y_τ(i))
        μ_i = μ_i − u
        ν_τ(i) = ν_τ(i) − u
        if ν_τ(i) = 0 then E_y = E_y \ {τ(i)}
        if μ_i = 0 then
            E_x = E_x \ {i}
        else
            τ(i) = arg min_{j ∈ E_y} {|f(x_i) − f(y_j)|}
            if k̃ = k then goto 0
    if E_x = ∅ then break
return W̃
```

4.3 Sliced Extensions

Algorithms 2 and 3 are not specific to trees and can be used with any 1D embedding. Taking inspiration from Sliced Wasserstein distances (Rabin et al., 2012), we can use orthogonal projections of the data to apply our algorithms without relying on a tree embedding of the vocabulary. Since random projections contain less structure than tree embeddings, the Sliced greedy approach should be less accurate than the tree version. However, Sliced projections are more polyvalent and flexible than tree embeddings because they do not rely on a static embedding structure; therefore, this

Algorithm 3: One-dimensional ICT.

input : Points $x_1, \dots, x_n \in \mathcal{X}$ and $y_1, \dots, y_m \in \mathcal{X}$ sorted by values of f, weight vectors $\mu \in \mathbb{R}^n$, $\nu \in \mathbb{R}^m$, mapping function $f : \mathcal{X} \to \mathbb{R}$, the true distance d.

output: The 1-Wasserstein estimate $\tilde{W}$.

$\tilde{W} = 0$
$j = 1$
for $i = 1$ *to* n **do**
  while $j < m$ *and* $|f(y_{j+1}) - f(x_i)| < |f(y_j) - f(x_i)|$ **do**
    $j = j + 1$
  $j_\mathrm{l} = j$
  $j_\mathrm{r} = j$
  $c = \mu_i$
  while $c > 0$ **do**
    $j_\mathrm{next} = \arg\min_{j_\mathrm{next} \in \{j_\mathrm{l}, j_\mathrm{r}\}} \{|f(y_{j_\mathrm{next}}) - f(x_i)|\}$
    $u = \min(c, \nu_{j_\mathrm{next}})$
    $\tilde{W} = \tilde{W} + u \cdot d(x_i, x_{j_\mathrm{next}})$
    $c = c - u$
    if $j_\mathrm{next} = j_\mathrm{l}$ **then** $j_\mathrm{l} = j_\mathrm{l} - 1$
    if $j_\mathrm{next} = j_\mathrm{r}$ **then** $j_\mathrm{r} = j_\mathrm{r} + 1$
return $\tilde{W}$

approach provide an interesting generalization of the tree methods when the vocabulary is not fixed. Moreover, Sliced projections can be computed on-the-fly without any data preprocessing. Finally, k-Greedy always provides an upper bound of the true Wasserstein distance because it always outputs a feasible matching for OT. Using the idea of Mahey et al. (2023), we can take the minimum greedy Wasserstein estimate over several random 1D projections to further improve the approximation quality of the k-Greedy algorithm and obtain more stable estimates of Wasserstein distances.

Note that sliced greedy matching is really close to Projected distances (Rowland et al., 2019), a variant of Sliced Wasserstein that aims to better estimate Wasserstein distances by computing the OT matching matrix on 1D projections of the inputs and evaluate it back in the original space. The difference with our method is that we compute a (truncated) greedy 1D matching rather than the 1D optimal transport matching. Experiments will show that this slight change actually yields strong improvements over the original Projected Wasserstein on tested datasets.

5 EXPERIMENTS

We now evaluate the approximation quality of our methods on Nearest Neighbour (NN) search tasks, on various real and synthetic datasets. In all the following experiments, the Euclidean distance was used as the ground cost for computing the 1-Wasserstein distances.

Table 2: Number of elements, vocabulary size, and average support size (word count) of the datasets.

DATASET	ELEMS.	VOC.	SUPP.
TWITTER	1754	4489	10.4
BBC	1225	11382	121.8
AMAZON	9198	27479	41.0
20NEWS	11314	400000	115.9

5.1 Setup

We test our algorithms on various text datasets: Twitter (Huang et al., 2016),[1] BBC (Greene and Cunningham, 2006),[2] Amazon Movies and TV (Ni et al., 2019),[3] and 20News (Backurs et al., 2020).[4] The text documents were preprocessed following the methodology of Kusner et al. (2015): after stopwords removal, each document was embedded as a Bag-of-Vectors using deep neural embeddings. The Twitter dataset was embedded using word2vec (Church, 2017) with an embedding dimension of 300; BBC, Amazon, and 20News using Glove (Pennington et al., 2014) with dim. 50. The statistics of each dataset are given in Table 2.

We also conduct experiments on random datasets: the random vocabularies $\mathcal{X}$ contain 10^5 points sampled from the standard Gaussian distribution in $\mathbb{R}^d$ with varying values of d, and each data element contains $n = 100$ points uniformly sampled from $\mathcal{X}$.

Each dataset is evaluated on a query set containing 1000 additional documents; for each query element, the goal of the NN search is to find the element of the database that minimizes the 1-Wasserstein distance with the query. The quality of the search is assessed using the NN recall metric: given $K \in \mathbb{N}$, the recall@K is the proportion of queries for which the true NN is among the K NNs retrieved by the approximation method. The correlation between the true and approximated 1-Wasserstein distances is additionally monitored: given a dataset with N elements and $M = 1000$ the number of query elements, we compute the $N \cdot M$ true and approximate 1-Wasserstein distances between each pair of query and dataset element, and measure the Pearson correlation between

[1] https://github.com/gaohuang/S-WMD
[2] http://mlg.ucd.ie/datasets/bbc.html
[3] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/
[4] https://github.com/ilyaraz/ot_estimators

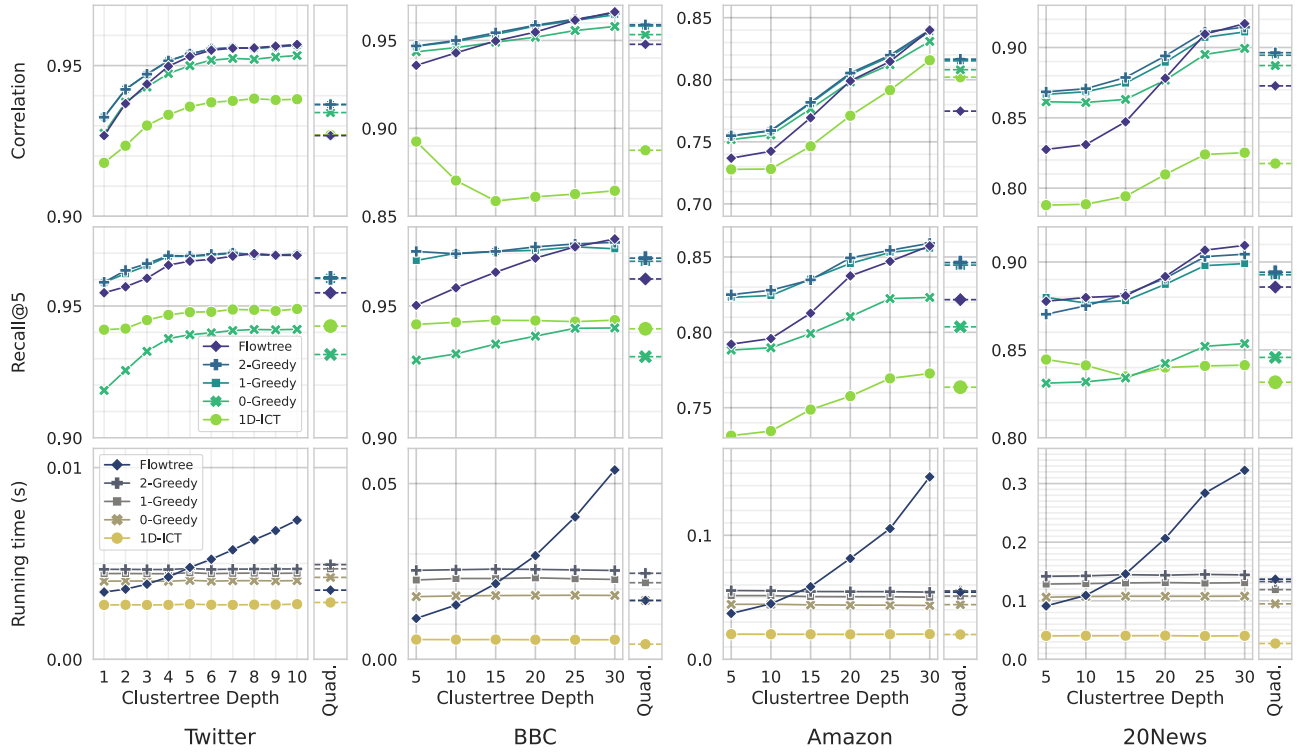

Figure 2: Accuracy and running time of the tree approximation algorithms for Quadtrees and Clustertrees embeddings (average over 20 random initializations). The running time is the average time required to compute the Wasserstein distances between one query and the full dataset.

the list of the $N \cdot M$ true Wasserstein distances and the $N \cdot M$ approximate values.

We evaluate 1D-ICT and k-Greedy algorithms both on tree embeddings (using Quadtree and Clustertrees) and sliced embeddings. Since our main focus is tree approximation methods, we mainly compare our methods to Flowtree and tree Wasserstein distance (TWD). We also provide a comparison with Projected Wasserstein distances. We finally refer to Backurs et al. (2020) for a broader comparison of linear OT methods. All the algorithms used in the experiments were implemented in C++.

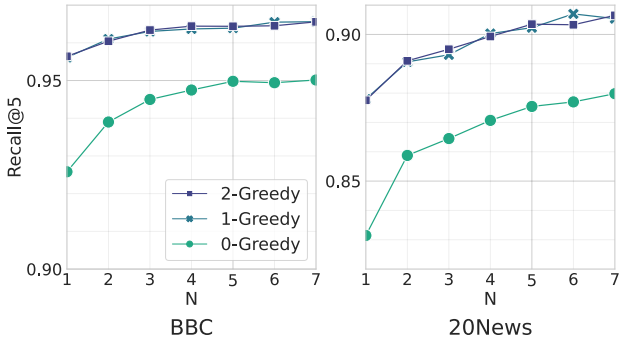

Figure 3: Recall@5 of k-Greedy when using the minimum Wasserstein estimate over N random projections.

5.2 Experiments on text datasets

Figure 2 shows the performances of tree methods for different kinds of tree embeddings. It clearly shows that, unlike Flowtree, the running time of our algorithms is independent of the tree height. We also observe that deeper Clustertree embeddings provide more accurate Wasserstein approximations in terms of correlation and recall@5; therefore, using deeper embeddings is useful, and k-Greedy makes it possible without additional running time. When the vocabulary is large (e.g., in Amazon and 20News datasets), 1-Greedy is even slightly faster than Flowtree on Quadtree while providing similar—and even slightly better—approximation quality. Finally, the 1D-ICT algorithm is less accurate than the other methods and benefits less of an increased tree height. Yet, its computations are significantly faster even on shallow trees, evidencing the relevance of this method for fast Wasserstein approximations.

The results obtained with particular tree embeddings are shown in Table 3 together with the results of sliced algorithms. It shows that (i) Clustertree enjoys higher accuracy than Quadtree, and (ii) Flowtree suffers from

Table 3: Correlation, Recall@5 and average running time of the different methods, and standard deviation over the 20 random initializations. Our proposed methods are marked in bold. h denotes the Clustertree height.

DATASET	METHOD	EMBEDDING	CORR.	R@5	TIME
	TWD	Quadtree	0.886 ± 0.000	$90.7 \pm 0.2\,\%$	2.3 ms
	Flowtree	Quadtree	0.927 ± 0.000	$95.5 \pm 0.0\,\%$	3.6 ms
	1D-ICT	**Clustertree(h=5)**	$\mathbf{0.939 \pm 0.003}$	$\mathbf{94.8 \pm 0.6\,\%}$	**2.9 ms**
	1-Greedy	**Clustertree(h=5)**	$\mathbf{0.954 \pm 0.004}$	$\mathbf{96.9 \pm 0.4\,\%}$	**4.5 ms**
TWITTER	Flowtree	Clustertree(h=5)	0.953 ± 0.004	$96.7 \pm 0.5\,\%$	4.7 ms
	Proj. Wass.	Sliced projection	0.687 ± 0.023	$45.9 \pm 1.6\,\%$	2.8 ms
	1D-ICT	**Sliced projection**	$\mathbf{0.922 \pm 0.003}$	$\mathbf{94.1 \pm 0.6\,\%}$	**2.9 ms**
	1-Greedy	**Sliced projection**	$\mathbf{0.934 \pm 0.003}$	$\mathbf{95.8 \pm 0.5\,\%}$	**4.5 ms**
	TWD	Quadtree	0.747 ± 0.024	$83.4 \pm 3.8\,\%$	2.0 ms
	Flowtree	Quadtree	0.948 ± 0.004	$96.0 \pm 0.5\,\%$	16.8 ms
	1D-ICT	**Clustertree(h=25)**	$\mathbf{0.863 \pm 0.011}$	$\mathbf{94.4 \pm 0.6\,\%}$	**5.6 ms**
	1-Greedy	**Clustertree(h=25)**	$\mathbf{0.961 \pm 0.003}$	$\mathbf{97.2 \pm 0.4\,\%}$	**22.9 ms**
BBC	Flowtree	Clustertree(h=25)	0.962 ± 0.003	$97.2 \pm 0.5\,\%$	40.5 ms
	Proj. Wass.	Sliced projection	0.798 ± 0.009	$44.8 \pm 1.8\,\%$	3.7 ms
	1D-ICT	**Sliced projection**	$\mathbf{0.886 \pm 0.009}$	$\mathbf{92.5 \pm 0.6\,\%}$	**4.7 ms**
	1-Greedy	**Sliced projection**	$\mathbf{0.939 \pm 0.002}$	$\mathbf{95.6 \pm 0.5\,\%}$	**21.4 ms**
	TWD	Quadtree	0.667 ± 0.058	$55.2 \pm 5.5\,\%$	13.6 ms
	Flowtree	Quadtree	0.775 ± 0.017	$82.2 \pm 1.2\,\%$	54.2 ms
	1D-ICT	**Clustertree(h=25)**	$\mathbf{0.792 \pm 0.029}$	$\mathbf{76.9 \pm 1.5\,\%}$	**20.3 ms**
	1-Greedy	**Clustertree(h=25)**	$\mathbf{0.819 \pm 0.019}$	$\mathbf{85.3 \pm 1.2\,\%}$	**50.5 ms**
AMAZON	Flowtree	Clustertree(h=25)	0.815 ± 0.022	$84.7 \pm 1.9\,\%$	105.4 ms
	Proj. Wass.	Sliced projection	0.626 ± 0.023	$24.0 \pm 1.3\,\%$	18.1 ms
	1D-ICT	**Sliced projection**	$\mathbf{0.701 \pm 0.013}$	$\mathbf{70.7 \pm 1.4\,\%}$	**21.3 ms**
	1-Greedy	**Sliced projection**	$\mathbf{0.755 \pm 0.011}$	$\mathbf{80.9 \pm 1.1\,\%}$	**52.4 ms**
	TWD	Quadtree	0.633 ± 0.040	$64.0 \pm 7.4\,\%$	13.6 ms
	Flowtree	Quadtree	0.873 ± 0.010	$88.6 \pm 0.8\,\%$	136.8 ms
	1D-ICT	**Clustertree(h=25)**	$\mathbf{0.824 \pm 0.007}$	$\mathbf{84.1 \pm 0.7\,\%}$	**40.0 ms**
	1-Greedy	**Clustertree(h=25)**	$\mathbf{0.907 \pm 0.007}$	$\mathbf{89.8 \pm 0.8\,\%}$	**130.3 ms**
20NEWS	Flowtree	Clustertree(h=25)	0.910 ± 0.012	$90.7 \pm 0.8\,\%$	283.8 ms
	Proj. Wass.	Sliced projection	0.722 ± 0.010	$26.6 \pm 1.2\,\%$	25.0 ms
	1D-ICT	**Sliced projection**	$\mathbf{0.775 \pm 0.010}$	$\mathbf{81.2 \pm 0.9\,\%}$	**30.0 ms**
	1-Greedy	**Sliced projection**	$\mathbf{0.850 \pm 0.004}$	$\mathbf{87.7 \pm 0.7\,\%}$	**121.3 ms**

longer running time than the other methods on Clustertrees. Using 1-Greedy instead of Flowtree, we improved the running time while preventing the correlation and recall from degrading much. Moreover, the 1D-ICT algorithm is three to four times faster than both Flowtree and Quadtree. Although the correlation and recall are smaller, the performances are still significantly better than the original TWD method and the Projected Wasserstein. Therefore, this algorithm provides an intermediate time-accuracy tradeoff between TWD and Flowtree.

Finally, when using the sliced projections, the performances of 1D-ICT and k-Greedy algorithms are slightly smaller than those with tree embeddings, although the decrease is rather moderate. Moreover, 1D-ICT is significantly better than the Projected Wasserstein distance while taking a similar amount of time to compute. Figure 3 shows that k-Greedy results can further be improved by aggregating the results over several projections; even with as few as $N = 5$ estimates, the NN recall is significantly improved and reaches the values previously obtained with tree embeddings. It makes these algorithms particularly relevant when no tree embedding is available, i.e., in situations where the vocabulary is not fixed or when the number of Wasserstein estimates is too small to amortize the tree construction.

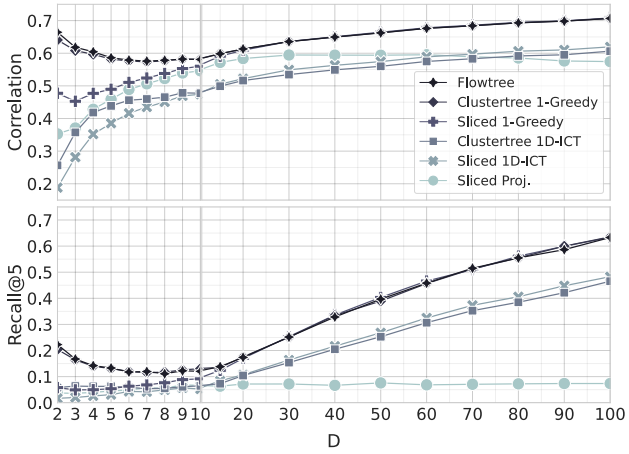

Figure 4: Evaluation of the algorithms on random point sets in $\mathbb{R}^d$, for different dimensions d (average over 15 random initializations).

5.3 Experiments on synthetic datasets

We next evaluate these algorithms on random Gaussian datasets, in order to assess their performances on less structured inputs. The results in Figure 4 show the relationship between the dimension and performance. We clearly observe that all the methods tested here perform better in higher dimensions. When the dimension exceeds $d = 20$, Clustertree 1-Greedy, Sliced 1-Greedy, and Flowtree are equivalent. Similarly, 1D-ICT is equivalent with both Sliced and Clustertrees embeddings when the dimension is large. A significant performance gap subsists between 1-Greedy and 1D-ICT methods across all input dimensions. Overall, these experiments show that our methods are mainly relevant in high-dimension, e.g., for language processing applications.

6 CONCLUSION

In this work, we demonstrated the interest of greedy-based strategies to compute fast and accurate 1-Wasserstein distance estimates. Our first proposed method, the k-Greedy algorithm, exhibits performance levels comparable to Flowtree when applied to text datasets. Our second method, the 1D-ICT method, is less accurate but still reliable on text data while being particularly fast. Both algorithms demonstrate improved scalability with respect to the tree height. Using sliced projections instead of tree embeddings, these methods still furnish reliable approximations without needing any data preprocessing or freezing the vocabulary. Finally, experiments on synthetic datasets seem to show the relevance of our methods in more general high-dimensional settings, and we hope that the ideas introduced here may prove their usefulness in other applications of optimal transport as well.

Acknowledgements

MY was supported by MEXT KAKENHI Grant Number 20H04243. HZ was supported in part by a research grant from the Amazon-Illinois Center on AI for Interactive Conversational Experiences (AICE). GH thanks OIST for the hospitality during his internship.

References

Kubilay Atasu and Thomas Mittelholzer. Linear-complexity data-parallel earth mover's distance approximations. In *International Conference on Machine Learning*, pages 364–373. PMLR, 2019.

Kubilay Atasu, Thomas Parnell, Celestine Dünner, Manolis Sifalakis, Haralampos Pozidis, Vasileios Vasileiadis, Michail Vlachos, Cesar Berrospi, and Abdel Labbi. Linear-complexity relaxed word mover's distance with gpu acceleration. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 889–896. IEEE, 2017.

David Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13(4):475–493, 1983.

Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Scalable nearest neighbor search for optimal transport. In *International Conference on Machine Learning*, pages 497–506. PMLR, 2020.

Robert Bamler and Stephan Mandt. Dynamic word embeddings. In *International Conference on Machine Learning*, pages 380–389. PMLR, 2017.

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using Lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–12, 2011.

Georgios-Ioannis Brokos, Prodromos Malakasiotis, and Ion Androutsopoulos. Using centroids of word embeddings and word mover's distance for biomedical document retrieval in question answering. *arXiv preprint arXiv:1608.03905*, 2016.

Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *Advances in neural information processing systems*, 30, 2017.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems*, 26, 2013.

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

Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 377–384, 2006.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. Supervised word mover's distance. *Advances in Neural Information Processing Systems*, 29, 2016.

Jacob C Kimmel, Lolita Penland, Nimrod D Rubinstein, David G Hendrickson, David R Kelley, and Adam Z Rosenthal. Murine single-cell rna-seq reveals cell-identity-and tissue-specific trajectories of aging. *Genome Research*, 29(12):2088–2103, 2019.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced variants of wasserstein distances. *Advances in Neural Information Processing Systems*, 32, 2019.

Bruno Leclerc. Description combinatoire des ultramétriques. *Mathématiques et Sciences humaines*, 73:5–37, 1981.

Wuchen Li, Ernest K Ryu, Stanley Osher, Wotao Yin, and Wilfrid Gangbo. A parallel method for earth mover's distance. *Journal of Scientific Computing*, 75 (1):182–197, 2018a.

Wuchen Li, Penghang Yin, and Stanley Osher. Computations of optimal transport distance with fisher information regularization. *Journal of Scientific Computing*, 75:1581–1595, 2018b.

Tianyi Lin, Nhat Ho, and Michael Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pages 3982–3991. PMLR, 2019.

Jialin Liu, Wotao Yin, Wuchen Li, and Yat Tin Chow. Multilevel optimal transport: a fast approximation of wasserstein-1 distances. *arXiv preprint arXiv:1810.00118*, 2018.

Guillaume Mahey, Laetitia Chapel, Gilles Gasso, Clément Bonet, and Nicolas Courty. Fast optimal transport through sliced wasserstein generalized geodesics. *arXiv preprint arXiv:2307.01770*, 2023.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pages 666–704, 1781.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

Andrea Ottolini and Stefan Steinerberger. Greedy matching in optimal transport with concave cost. *arXiv preprint arXiv:2307.03140*, 2023.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*, pages 460–467. IEEE, 2009.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision: Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29–June 2, 2011, Revised Selected Papers 3*, pages 435–446. Springer, 2012.

Julien Rabin, Sira Ferradans, and Nicolas Papadakis. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4852–4856. IEEE, 2014.

Edward M Reingold and Robert E Tarjan. On a greedy heuristic for complete matching. *SIAM Journal on Computing*, 10(4):676–681, 1981.

Mark Rowland, Jiri Hron, Yunhao Tang, Krzysztof Choromanski, Tamas Sarlos, and Adrian Weller. Orthogonal estimation of wasserstein distances. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 186–195. PMLR, 2019.

Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

Meyer Scetbon and Marco Cuturi. Low-rank optimal transport: Approximation, statistics and debiasing. *Advances in Neural Information Processing Systems*, 35:6802–6814, 2022.

Meyer Scetbon, Marco Cuturi, and Gabriel Peyré. Low-rank sinkhorn factorization. In *International Conference on Machine Learning*, pages 9344–9354. PMLR, 2021.

Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

Sameer Shirdhonkar and David W Jacobs. Approximate earth mover's distance in linear time. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

Shashank Sonkar, Andrew E Waters, and Richard G Baraniuk. Attention word embedding. *arXiv preprint arXiv:2006.00988*, 2020.

Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33:19276–19289, 2020.

Yuki Takezawa, Ryoma Sato, and Makoto Yamada. Supervised tree-wasserstein distance. In *International Conference on Machine Learning*, pages 10086–10095. PMLR, 2021.

Alexander Y Tong, Guillaume Huguet, Amine Natik, Kincaid MacDonald, Manik Kuchroo, Ronald Coifman, Guy Wolf, and Smita Krishnaswamy. Diffusion earth mover's distance and distribution embeddings. In *International Conference on Machine Learning*, pages 10336–10346. PMLR, 2021.

Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer, 2009.

Wei Wang, Dejan Slepčev, Saurav Basu, John A Ozolek, and Gustavo K Rohde. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International Journal of Computer Vision*, 101:254–269, 2013.

Ruicheng Xian, Lang Yin, and Han Zhao. Fair and optimal classification via post-processing. In *International Conference on Machine Learning*, 2023.

Makoto Yamada, Yuki Takezawa, Ryoma Sato, Han Bao, Zornitsa Kozareva, and Sujith Ravi. Approximating 1-Wasserstein distance with trees. *arXiv preprint arXiv:2206.12116*, 2022.

Karren Dai Yang, Karthik Damodaran, Saradha Venkatachalapathy, Ali C Soylemezoglu, GV Shivashankar, and Caroline Uhler. Predicting cell lineages using autoencoders and optimal transport. *PLoS computational biology*, 16(4):e1007828, 2020.

Han Zhao. Costs and benefits of fair regression. *Transactions on Machine Learning Research*, 2022.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*, 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes]

(b) The license information of the assets, if applicable. [Not Applicable]

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A DETAILED ANALYSIS OF GREEDY OPTIMAL TRANSPORT

In this section, we provide further theoretical and empirical results concerning the performances of greedy OT, whose procedure is recalled in Algorithm 1.

A.1 Approximation bounds

We first provide several theoretical results about greedy OT adapted from the discrete optimization literature. For maximum matching problems, i.e. when we aim to maximize $\langle C, \Pi \rangle$ rather than minimizing it, the greedy algorithm is known to provide a 1/2-approximation of the maximal value; see e.g. (Avis, 1983, Theorem 4) for a concise proof. Proposition A translates this result in the optimal transport context.

Proposition A. *Let $C \in \mathbb{R}_+^{n \times m}$ be a cost matrix and $\mu \in \mathbb{R}_+^n, \nu \in \mathbb{R}_+^m$ two probability vectors. Let $L_C(\mu, \nu)$ be the Optimal Transport cost between μ and ν. Let $\tilde{\Pi}$ be the greedy matching given by Algorithm 1, $\tilde{L}_C(\mu, \nu) = \langle C, \tilde{\Pi} \rangle$ the greedy cost approximation, and $\|C\|_\infty = \max_{i,j} C_{i,j}$. The following bound holds :*

$$L_C(\mu, \nu) \leq \tilde{L}_C(\mu, \nu) \leq \frac{1}{2} \left(L_C(\mu, \nu) + \|C\|_\infty \right).$$

In the metric case, denoting $d_M = \max_{i,j} d(x_i, y_j)$, we have for $p \leq 1$:

$$W_p(\mu, \nu) \leq \tilde{W}_p(\mu, \nu) \leq \frac{1}{2} \left(W_p(\mu, \nu) + TV(\mu, \nu) \cdot d_M \right).$$

Greedy matching has also been studied for Minimum Matching problems. The worst case bound has been determined in Reingold and Tarjan (1981) and is recalled in Proposition B for the Monge transportation problem.

Proposition B (Reingold and Tarjan (1981))**.** *Let $(\mathcal{X}, d)$ be a metric space and $\nu, \mu \in \mathcal{M}_+^1(\mathcal{X})$. In the case where $|\mathrm{Supp}(\mu)| = |\mathrm{Supp}(\nu)| = n$ and where μ, ν are uniform on their support, then the greedy 1-Wasserstein distance approximate $\tilde{W}$ satisfies the following inequality:*

$$\tilde{W} \leq \left(\frac{4}{3} n^{\log_2(3/2)} - 1 \right) \cdot W.$$

Moreover, the bound is tight: if n is a power of 2, there exists a metric space $(\mathcal{X}, d)$ and two distributions μ, ν such that $|\mathrm{Supp}(\mu)| = |\mathrm{Supp}(\nu)| = n$ such that the bound is reached.

Both results show that the worst-case performances of greedy OT are very poor. Indeed, the bound of Proposition A involves the diameter $\|C\|_\infty$ of the cost matrix that can be significantly larger than the Optimal Transport cost; therefore, this bound provides a limited control on the greedy performances for Wasserstein approximations. The approximation ratio of Proposition B increases polynomially in the number of points, meaning that the worst-case performances of the greedy algorithm degrade quickly with the size of the input.

When the two inputs are sufficiently close, however, greedy OT computes the true OT solution, as stated in Proposition C.

Proposition C. *Let $p \leq 1$. Let $(\mathcal{X}, d)$ be a finite metric space and $\mu, \nu \in \mathcal{M}_+^1(\mathcal{X})$. If there exists a mapping $T : \mathcal{X} \to \mathcal{X}$ such that $\nu = T_\sharp \mu$ and that satisfies:*

$$\max_{x \in \mathcal{X}} d(x, T(x)) \leq \frac{1}{2} \cdot \min_{\substack{x, x' \in \mathrm{Supp}(\mu) \\ x \neq x'}} d(x, x'),$$

then $W_p(\mu, \nu) = \tilde{W}_p(\mu, \nu)$.

A.2 Experiments on random datasets

Our experiments on text datasets showed the strong empirical accuracy of greedy matching in real cases.

The approximation quality of greedy matching for 1-Wasserstein distance is lower for random inputs than

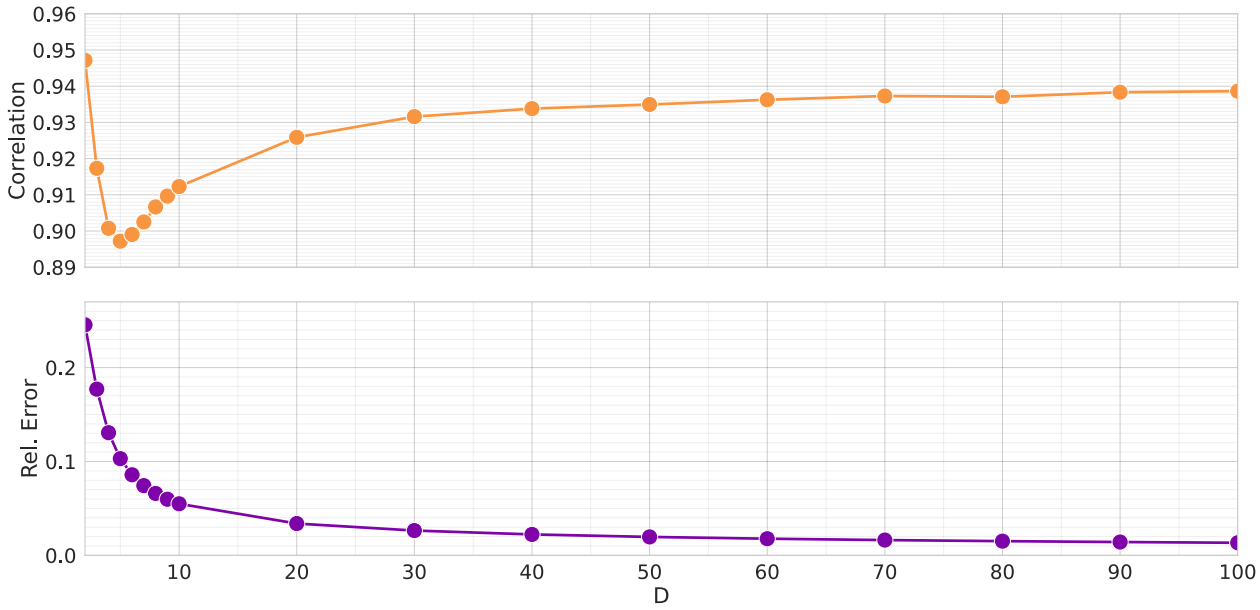

Figure A: Evaluation of the Greedy Wasserstein distance on random inputs. The datasets contain $n = 100$ points sampled from a standard Gaussian distribution in $\mathbb{R}^d$, with varying values of d.

with real datasets (Figure A). However, the correlation remains larger than 0.9 across the vast majority of the tested dimensions. Moreover, the relative error decreases and the correlation increases with the input dimension. Therefore, greedy OT seems mainly relevant in high dimensions.

Interestingly, greedy OT is less correlated with true Wasserstein distances for small dimensions (dim. 3 to 10). The reason for this correlation drop is unclear and may indicate that 1-Wasserstein distances have a more complex geometry across this range of dimensions.

B DETAILED ALGORITHMIC IMPLEMENTATION

This section is dedicated to the implementation details of the algorithms and procedures introduced in our paper.

B.1 1D embedding of tree leaves

Algorithm A provides the procedure that implements the one-dimensional embedding for trees. It is based on a recursive procedure that ensures that the gap between the embedding of two successive nodes is larger than the maximum gap between two nodes with a smaller lowest common ancestor.

Remark. *In practice, the maximum value of the function f output by Algorithm A grows exponentially with the height of the embedded tree. More precisely, for*

Algorithm A: 1-dimensional tree embedding.

input : A rooted tree $\mathcal{T}$ of leaves $\mathcal{X}$.
output: A mapping $f : \mathcal{X} \to \mathbb{R}$.

$f = 0$
TREEEMBED$(\mathcal{T}, k = 0)$
return f

treeEmbed $\mathcal{T}, k$:
  $r =$ root of $\mathcal{T}$
  if r *is leaf* **then**
    $f(r) = k$
    return k
  else
    $i = 0$
    $i_{max} =$ (degree of r) $- 1$
    $\Delta k_{old} = 0$
    foreach *children* $\tilde{\mathcal{T}}$ *of* r **do**
      $k' =$ TREEEMBED$(\tilde{\mathcal{T}}, k)$
      $\Delta k = k' - k$
      $k = k'$
      if $i > 0$ **then**
        foreach *leaves* x *of* $\tilde{\mathcal{T}}$ **do**
          $f(x) = f(x) + \max(\Delta k_{old}, \Delta k)$;
        $k = k + \max(\Delta k_{old}, \Delta k)$
      if $i \neq i_{max}$ **then** $k = k + 1$;
      $\Delta k_{old} = \Delta k$
      $i = i + 1$
    return k

Algorithm B: One-dimensional nearest neighbor computation.

input : Points $x_1, \dots, x_n \in \mathcal{X}$ and $y_1, \dots, y_m \in \mathcal{X}$ sorted by values of f, mapping function $f : \mathcal{X} \to \mathbb{R}$.
output: $\tau : \{1, \dots, n\} \to \{1, \dots, m\}$ giving the index of nearest neighbors of the x_i among the set of y_j.

```
j = 1
for i = 1 to n do
    while j < m and
      |f(y_{j+1}) − f(x_i)| < |f(y_j) − f(x_i)| do
        j = j + 1
    τ(i) = j
return τ
```

a tree of height h with maximum degree l, the function f is bounded by $f_{max} = (2k-1)^h$. Arithmetic operations involving integers between $[0, M]$ have an asymptotic running time of $O(\log(M))$: therefore, theoretically, the algorithms presented in this part also scale linearly with h. However, in practice, the cost of arithmetic operations is significantly lower than the other elementary operations involved in the algorithms studied here, and the exponential growth of f_{max} has a negligible effect on the complexity in practice. This is yet an important observation for the implementation of these algorithms since integer overflows are likely to happen when using an inadequate data type.

B.2 1D Nearest Neighbor Search

Using the fact that nearest neighbor mappings are non-decreasing in one dimension, we can implement the NN list search in linear time by scanning both lists simultaneously in increasing order, as presented in Algorithm B. This algorithm computes the nearest neighbors of $x_1, \dots, x_n$ among the points $y_1, \dots, y_m$ in $O(n+m)$ time (i.e. an amortized complexity of $O(1)$ for one single point NN search).

B.3 k-Greedy algorithm

The exact implementation of the k-greedy method, given in Algorithm C, relies on pointer lists that allow to quickly scan over the unmatched points of the distributions μ and ν. The vectors $\overrightarrow{P}_x$, $\overrightarrow{P}_y$ and $\overleftarrow{P}_y$ contain the indices of the next unmatched element of each point of $x_1, \dots, x_n$ and $y_1, \dots, y_m$ in both increasing and decreasing order of f. This makes it possible to update the nearest neighbour mapping τ very efficiently, ensuring the linear complexity of the k-Greedy algorithm. This implementations runs in $O((k+1)(n+m)\log(n))$ time.

Algorithm C: One-dimensional k-Greedy algorithm (detailed implementation).

input : Points $x_1, \dots, x_n \in \mathcal{X}$ and $y_1, \dots, y_m \in \mathcal{X}$ sorted by values of f, weight vectors $\mu \in \mathbb{R}^n$, $\nu \in \mathbb{R}^m$, mapping function $f : \mathcal{X} \to \mathbb{R}$, $k \geq 0$.
output: The 1-Wasserstein estimate $\tilde{W}$.

```
W̃ = 0
→P_x = (1, ..., n)
→P_y = (1, ..., m)
←P_y = (1, ..., m)
τ = nearest neighbor mapping between x_1, ..., x_n
  and y_1, ..., y_m for the function f.
for k̃ = 0 to k do
    E_x = ∅
    i = 1
    while i < n do
        i_r = i
        while μ_{i_r} = 0 do
            i_r = →P_x[i_r + 1]
        →P_x[i] = i_r
        i = i_r
        Add i to E_x
    Sort E_x by increasing value of |f(x_i) − f(y_{τ(i)})|
    for i ∈ E_x do
        label 0
        u = min(μ_i, ν_{τ(i)})
        W̃ = W̃ + u · d(x_i, y_{τ(i)})
        μ_i = μ_i − u
        ν_{τ(i)} = ν_{τ(i)} − u
        j_l = τ(i)
        j_r = τ(i)
        if ν_{τ(i)} = 0 then
            while ν_{j_l} = 0 do
                ←P_y[j_l] = τ(i)
                j_l = ←P_y[j_l − 1]
            while ν_{j_r} = 0 do
                →P_y[j_r] = τ(i)
                j_l = →P_y[j_r + 1]
            ←P_y[τ(i)] = j_l
            →P_y[τ(i)] = j_r
        if μ_i = 0 then
            →P_x[i] = →P_x[i + 1]
        else
            if |f(x_i) − f(y_{j_l})| < |f(x_i) − f(y_{j_r})|
              then τ(i) = j_l  else τ(i) = j_r
            if k̃ = k then goto 0
return W̃
```

C PROOFS

In this section, we provide proofs of the propositions stated in the main article and in this supplementary material.

C.1 Approximation bounds of Greedy Optimal Transport

Proposition A. *Let $C \in \mathbb{R}_+^{n\times m}$ be a cost matrix and $\mu \in \mathbb{R}_+^n, \nu \in \mathbb{R}_+^m$ two probability vectors. Let $L_C(\mu,\nu)$ be the Optimal Transport cost between μ and ν. Let $\tilde{\Pi}$ be the greedy matching given by Algorithm 1, $\tilde{L}_C(\mu,\nu) = \langle C, \tilde{\Pi}\rangle$ the greedy cost approximation, and $\|C\|_\infty = \max_{i,j} C_{i,j}$. The following bound holds :*

$$L_C(\mu,\nu) \leq \tilde{L}_C(\mu,\nu) \leq \frac{1}{2}\left(L_C(\mu,\nu) + \|C\|_\infty\right).$$

In the metric case, denoting $d_M = \max_{i,j} d(x_i, y_j)$, we have for $p \leq 1$:

$$W_p(\mu,\nu) \leq \tilde{W}_p(\mu,\nu) \leq \frac{1}{2}\left(W_p(\mu,\nu) + TV(\mu,\nu)\cdot d_M\right).$$

Proof of Proposition A. The lower bound $L_C \leq \tilde{L}_C$ is a consequence of the feasibility of the greedy matching, i.e. $\tilde{\Pi}$ satisfies the constraints of the Optimal Transport.

In order to prove the upper bound, let us first remark that

$$\tilde{L}_C(\mu,\nu) = \int_0^{\|C\|_\infty} \sum_{i,j} \tilde{\Pi}_{i,j}\cdot \mathbf{1}_{C_{i,j}>\lambda} \mathrm{d}\lambda \qquad (1)$$

and

$$L_C(\mu,\nu) = \int_0^{\|C\|_\infty} \sum_{i,j} \Pi_{i,j}\cdot \mathbf{1}_{C_{i,j}>\lambda} \mathrm{d}\lambda. \qquad (2)$$

We will now fix a value $\lambda \in [0, \|C\|_\infty]$ and try to bound the value of $\sum_{i,j} \tilde{\Pi}_{i,j}\cdot \mathbf{1}_{C_{i,j}\leq\lambda}$.

Let us assume that $\tilde{\Pi} \neq \Pi$. Then, there exists a set of indices $i_1, j_1, \ldots, i_N, j_N$ such that each i_k is distinct, each j_k is distinct, and for each $k \in \{1,\ldots,N\}$,with the convention $j_{N+1} = j_1$:

$$\tilde{\Pi}_{i_k,j_k} > \Pi_{i_k,j_k} \quad \text{and} \quad \Pi_{i_k,j_{k+1}} > \tilde{\Pi}_{i_k,j_{k+1}}. \qquad (3)$$

The existence of a cyclical permutation satisfying Equation 3 is a direct consequence of the constraints of optimal transport: if there exists a couple (i,j) such that $\Pi_{i,j} > \tilde{\Pi}_{i,j}$, then there must be an other index k such that $\Pi_{i,l} < \tilde{\Pi}_{i,l}$ since that for each value of i,

$$\sum_{l=1}^m \Pi_{i,l} = \sum_{l=1}^m \tilde{\Pi}_{i,l}.$$

Therefore, we can build such a cycle recursively by alternating between edges where $\Pi_{i,j} > \tilde{\Pi}_{i,j}$ and where $\tilde{\Pi}_{i,j} > \Pi_{i,j}$.

Let $\tilde{\Gamma} = \{(i_k, j_k)\}$, $\Gamma = \{(i_k, j_{k+1})\}$, $I = \{i_k\}$ and $J = \{j_k\}$. For each (i_k, j_{k+1}), we have either:

$$C_{i_k,j_{k+1}} \geq C_{i_k,j_k} \quad \text{or} \quad C_{i_k,j_{k+1}} \geq C_{i_{k+1},j_{k+1}}.$$

Indeed, the first inequality of Equation 3 states that $\tilde{\Pi}_{i_k,j_k} > \Pi_{i_k,j_k}$; since that $\Pi_{i_k,j_k} \geq 0$, we must have $\tilde{\Pi}_{i_k,j_k} > 0$. Similarly, $\tilde{\Pi}_{i_k+1,j_{k+1}} > 0$. But $\tilde{\Pi}$ is a greedy matching: when the couple (i_k, j_{k+1}) is processed by the algorithm, all the possible remaining mass is attributed to $\tilde{\Pi}_{i_k,j_{k+1}}$, and either $\mu_{i_k} = 0$ or $\nu_{j_{k+1}} = 0$ after that iteration of the algorithm. Hence, if both (i_k, j_k) and (i_{k+1}, j_{k+1}) were processed after (i_k, j_{k+1}), then we would have either $\tilde{\Pi}_{i_k,j_k} = 0$ or $\tilde{\Pi}_{i_k+1,j_{k+1}} = 0$, leading to a contradiction.

Therefore, if $C_{i_k,j_{k+1}} \geq \lambda$, then either $C_{i_k,j_k} \leq \lambda$ or $C_{i_{k+1},j_{k+1}} \leq \lambda$. Hence, the number of couples (i_k, j_{k+1}) such that $C_{i_k,j_{k+1}} \geq \lambda$ is at most twice the number of couples (i_k, j_k) satisfying $C_{i_k,j_k} \leq \lambda$, i.e.

$$\sum_{k=1}^N \mathbf{1}_{C_{i_k,j_k}\leq\lambda} \geq \frac{1}{2}\sum_{k=1}^N \mathbf{1}_{C_{i_k,j_{k+1}}\leq\lambda}.$$

Denoting by $\omega > 0$ the minimal value of $|\Pi - \tilde{\Pi}|$ on both $\tilde{\Gamma}$ and Γ, the previous inequality can be rewritten as:

$$\sum_{i,j} \omega \mathbf{1}_{(i,j)\in\tilde{\Gamma}}\cdot \mathbf{1}_{C_{i,j}\leq\lambda} \geq \frac{1}{2}\sum_{i,j} \omega \mathbf{1}_{(i,j)\in\Gamma}\cdot \mathbf{1}_{C_{i,j}\leq\lambda}.$$

Moreover, the matrices Π' and $\tilde{\Pi}'$ defined as

$$\forall i,j, \quad \Pi'_{i,j} = \Pi_{i,j} - \omega \mathbf{1}_{(i,j)\in\tilde{\Gamma}}$$
$$\text{and} \quad \tilde{\Pi}'_{i,j} = \tilde{\Pi}_{i,j} - \omega \mathbf{1}_{(i,j)\in\Gamma}$$

are valid matching matrices between the (unnormalized) weight vectors μ' and ν' whose coefficients are defined as

$$\mu'_i = \mu_i - \omega \mathbf{1}_{i\in I} \quad \text{and} \quad \nu'_j = \nu_j - \omega \mathbf{1}_{j\in J},$$

in the sense that

$$\Pi' \mathbf{1}_n = \mu', \quad \Pi'^T \mathbf{1}_m = \nu'$$

and

$$\tilde{\Pi}' \mathbf{1}_n = \mu', \quad \tilde{\Pi}'^T \mathbf{1}_m = \nu'.$$

The matrix $\tilde{\Pi}'$ is still a greedy matching between these unnormalized weight vectors, so we can iteratively apply the same argument until $\Pi' = \tilde{\Pi}'$, and eventually prove:

$$\sum_{i,j} \tilde{\Pi}_{i,j}\cdot \mathbf{1}_{C_{i,j}\leq\lambda} \geq \frac{1}{2}\sum_{k=1}^N \Pi_{i,j}\cdot \mathbf{1}_{C_{i,j}\leq\lambda}. \qquad (4)$$

Going back to Equations 1 and 2, we obtain:

$$\begin{aligned}\tilde{L}_C(\mu,\nu) &= \int_0^{\|C\|_\infty} \left(1 - \sum_{i,j} \tilde{\Pi}_{i,j} \mathbf{1}_{C_{i,j}\leq\lambda}\right) \mathrm{d}\lambda \\ &\leq \int_0^{\|C\|_\infty} \left(1 - \frac{1}{2}\sum_{i,j} \Pi_{i,j} \mathbf{1}_{C_{i,j}\leq\lambda}\right) \mathrm{d}\lambda \\ &\leq \int_0^{\|C\|_\infty} \left(\frac{1}{2} + \frac{1}{2}\sum_{i,j} \Pi_{i,j} \mathbf{1}_{C_{i,j}>\lambda}\right) \mathrm{d}\lambda \\ &\leq \frac{1}{2}\left(\|C\|_\infty + L_C(\mu,\nu)\right).\end{aligned}$$

In the metric case when $p \leq 1$, the triangular inequality makes it optimal to match $x \in \mathrm{Supp}(\mu)$ and $y \in \mathrm{Supp}(\nu)$ together as soon as $x = y$. The greedy matching also assigns identical points together, so we can always find an optimal matching Π such that $\Pi_{i,j} = \tilde{\Pi}_{i,j}$ for each (i,j) with $x_i = y_j$. Since that Equation 3 only involves edges satisfying $\Pi_{i,j} \neq \tilde{\Pi}_{i,j}$, Equation 4 becomes:

$$\sum_{i,j} \tilde{\Pi}_{i,j} \cdot \mathbf{1}_{0<d(x_i,y_j)\leq\lambda} \geq \frac{1}{2}\sum_{k=1}^{N} \Pi_{i,j} \cdot \mathbf{1}_{0<d(x_i,y_j)\leq\lambda}.$$

Moreover, $\tilde{\Pi}$ is minimal for the cost $C_{i,j} = \mathbf{1}_{x_i\neq y_j}$; therefore:

$$\sum_{i,j} \Pi_{i,j} \cdot \mathbf{1}_{x_i\neq y_j} = \sum_{i,j} \tilde{\Pi}_{i,j} \cdot \mathbf{1}_{x_i\neq y_j} = TV(\mu,\nu),$$

the first equality being a consequence of the fact that $\Pi_{i,j} = \tilde{\Pi}_{i,j}$ when $x_i = y_j$. Then, for $\lambda > 0$,

$$\begin{aligned}&\sum_{i,j} \tilde{\Pi}_{i,j} \cdot \mathbf{1}_{d(x_i,y_j)>\lambda} \\ &\qquad = \sum_{i,j} \tilde{\Pi}_{i,j} \cdot (1 - \mathbf{1}_{0<d(x_i,y_j)\leq\lambda} - \mathbf{1}_{x_i=y_j}) \\ &\qquad = \sum_{i,j} \tilde{\Pi}_{i,j} \cdot (\mathbf{1}_{x_i\neq y_j} - \mathbf{1}_{0<d(x_i,y_j)\leq\lambda}) \\ &\qquad = TV(\mu,\nu) - \sum_{i,j} \tilde{\Pi}_{i,j} \cdot \mathbf{1}_{0<d(x_i,y_j)\leq\lambda}.\end{aligned}$$

Similarly,

$$\sum_{i,j} \Pi_{i,j} \cdot \mathbf{1}_{d(x_i,y_j)>\lambda} = TV(\mu,\nu) - \sum_{i,j} \Pi_{i,j} \cdot \mathbf{1}_{0<d(x_i,y_j)\leq\lambda}$$

Plugging this equation in Equations 1 and 2, we eventually obtain the following inequality:

$$\tilde{W}_p(\mu,\nu) \leq \frac{1}{2}(W_p(\mu,\nu) + d_M \cdot TV(\mu,\nu)).$$

□

Proposition C. *Let $p \leq 1$. Let $(\mathcal{X}, d)$ be a finite metric space and $\mu, \nu \in \mathcal{M}_+^1(\mathcal{X})$. If there exists a mapping $T : \mathcal{X} \to \mathcal{X}$ such that $\nu = T_\sharp\mu$ and that satisfies:*

$$\max_{x\in\mathcal{X}} d(x, T(x)) \leq \frac{1}{2} \cdot \min_{\substack{x,x'\in\mathrm{Supp}(\mu)\\ x\neq x'}} d(x,x'),$$

then $W_p(\mu,\nu) = \tilde{W}_p(\mu,\nu)$.

Proof of Proposition C. Let $x \in \mathrm{Supp}(\mu)$ and $y = T(x)$. Then, for each $x' \in \mathrm{Supp}(\mu)$:

$$\begin{aligned}d(x',y) &\geq d(x,x') - d(x,y) \\ &\geq 2 \cdot d(x, T(x)) - d(x,y) \\ &\geq d(x,y).\end{aligned}$$

Therefore, for each $x \in \mathrm{Supp}(\mu)$, $y = T(x)$ is the nearest neighbor of x in $\mathrm{Supp}(\nu)$. Since that $\nu = T_\sharp\mu$, the greedy algorithm exactly matches the points x with their image $T(x)$, resulting in a feasible matching. The inequality proven above also implies that the map T is optimal for the Monge transportation. It shows that the optimal matching is exactly the greedy matching, hence the result. □

C.2 Connection between Tree Optimal Transport and Greedy Matching

The equivalence between tree OT and greedy matching is rather classical, and is related to the properties of minimum flows over trees (minimum flow being an optimization problem equivalent to optimal transport). However, proofs are difficult to find in the literature. Therefore, we provide here a formal and complete proof here. These proofs mainly rely on the cyclical monotony of optimal matching.

We first recall the Proposition 1, a standard property of ultrametric that will be useful in the next proofs.

Proposition 1 (Leclerc (1981))**.** *Let $(\mathcal{X}, d)$ be an ultrametric space. Then, there exists a tree $\mathcal{T}$ and a weight function $w : \mathcal{E} \to \mathbb{R}_+$ such that $d = d_{\mathcal{T},w}$. Reciprocally, if $\mathcal{T}$ is a tree, then there is a weight function $w : \mathcal{E} \to \mathbb{R}_+$ such that $d_{\mathcal{T},w}$ is an ultrametric on $\mathcal{X}$.*

Proposition 2. *For each tree $\mathcal{T}$, the set of optimal matchings associated to the metric $d_{\mathcal{T},w}$ is identical for any weight function w on its edges: for any $w, w' : \mathcal{E} \to \mathbb{R}_+$ and $\mu, \nu \in \mathcal{M}_+^1(\mathcal{X})$, we have:*

$$\operatorname*{arg\,min}_{\Pi\in\mathcal{U}(\mu,\nu)} \langle d_{\mathcal{T},w}, \Pi\rangle = \operatorname*{arg\,min}_{\Pi\in\mathcal{U}(\mu,\nu)} \langle d_{\mathcal{T},w'}, \Pi\rangle.$$

The core idea in the proof of Proposition 2 is that a non-optimal matching always matches together two

pairs of nodes whose shortest paths overlap in the tree. Therefore, for any weight function on the edges exchanging the pairings will always improve the cost of the matching.

Proof of Proposition 2. Let $\mathcal{T} = (\mathcal{X}, \mathcal{V}, \mathcal{E})$ be a tree, let $x_1, \dots, x_n$ and $y_1, \dots, y_m$ two set of points of $\mathcal{X}$, and $\mu \in \mathbb{R}^n$, $\nu \in \mathbb{R}^m$ two probability vectors. Let Π be a feasible matching between μ and ν that is not optimal for the distance $d_{\mathcal{T},w}$ (for some weight function w). We will now show that it cannot be optimal for any other distance $d_{\mathcal{T},w'}$ on the same tree as well.

If Π is not optimal with respect to $d_{\mathcal{T},w}$, then it is not cyclically monotone; see e.g. (Villani, 2009, Chapter 5). It means that there exists a set of indices $(i_1, j_i), \dots, (i_N, j_N) \in \mathrm{Supp}(\Pi)$ such that:

$$\sum_{k=1}^{N} d_{\mathcal{T},w}\left(x_{i_k}, y_{j_k}\right) > \sum_{k=1}^{N} d_{\mathcal{T},w}\left(x_{i_k}, y_{j_{k+1}}\right), \quad (5)$$

with the convention $j_{N+1} = j_1$. Moreover, the definition of tree metrics implies that for any $(x, y) \in \mathcal{X}$:

$$d_{\mathcal{T},w}(x, y) = \sum_{e \in \mathcal{E}} w_e \mathbf{1}_{e \in \mathcal{P}(x,y)} \quad (6)$$

where $\mathcal{P}(x, y) \subset \mathcal{E}$ denotes the shortest path between x and y in $\mathcal{T}$. Since that $\mathcal{T}$ is a tree, the shortest path is unique and is independent of the edges weights. Equation 5 can now be rewritten as follow:

$$\sum_{e \in \mathcal{E}} w_e \sum_{k=1}^{N} \mathbf{1}_{e \in \mathcal{P}(x_{i_k}, y_{j_k})} > \sum_{e \in \mathcal{E}} w_e \sum_{k=1}^{N} \mathbf{1}_{e \in \mathcal{P}(x_{i_k}, y_{j_{k+1}})}.$$

Therefore, there is an $e \in \mathcal{E}$ such that:

$$\sum_{k=1}^{N} \mathbf{1}_{e \in \mathcal{P}(x_{i_k}, y_{j_k})} > \sum_{k=1}^{N} \mathbf{1}_{e \in \mathcal{P}(x_{i_k}, y_{j_{k+1}})}. \quad (7)$$

Let e be such an edge, and let $\mathcal{T}_1, \mathcal{T}_2$ the two connected components of $\mathcal{T} \backslash \{e\}$. For any $x, y \in \mathcal{X}$, $e \in \mathcal{P}(x, y)$ if and only if x and y belong to distinct connected components of $\mathcal{T} \backslash \{e\}$, i.e.:

$$\mathbf{1}_{e \in \mathcal{P}(x,y)} = \mathbf{1}_{x \in \mathcal{T}_1 \cap y \in \mathcal{T}_2} + \mathbf{1}_{x \in \mathcal{T}_2 \cap y \in \mathcal{T}_1}.$$

Equation 7 then becomes:

$$\sum_{k=1}^{N} \left(\mathbf{1}_{x_{i_k} \in \mathcal{T}_1 \cap y_{j_k} \in \mathcal{T}_2} + \mathbf{1}_{x_{i_k} \in \mathcal{T}_2 \cap y_{j_k} \in \mathcal{T}_1} \right) > \sum_{k=1}^{N} \left(\mathbf{1}_{x_{i_k} \in \mathcal{T}_1 \cap y_{j_{k+1}} \in \mathcal{T}_2} + \mathbf{1}_{x_{i_k} \in \mathcal{T}_2 \cap y_{j_{k+1}} \in \mathcal{T}_1} \right). \quad (8)$$

However, for any $x, y \in \mathcal{X}$, we also have:

$$\mathbf{1}_{x \in \mathcal{T}_1 \cap y \in \mathcal{T}_2} = \mathbf{1}_{x \in \mathcal{T}_1} + \mathbf{1}_{y \in \mathcal{T}_2} - \mathbf{1}_{x \in \mathcal{T}_1 \cup y \in \mathcal{T}_2}.$$

Moreover, the events

$$\{x \in \mathcal{T}_1 \cup y \in \mathcal{T}_2\}$$

and

$$\{x \in \mathcal{T}_2 \cap y \in \mathcal{T}_1\}$$

are complementary, so:

$$\mathbf{1}_{x \in \mathcal{T}_1 \cup y \in \mathcal{T}_2} + \mathbf{1}_{x \in \mathcal{T}_2 \cap y \in \mathcal{T}_1} = 1.$$

This leads to the following equation:

$$\begin{aligned} \mathbf{1}_{x \in \mathcal{T}_1 \cap y \in \mathcal{T}_2} - \mathbf{1}_{x \in \mathcal{T}_2 \cap y \in \mathcal{T}_1} &= \mathbf{1}_{x \in \mathcal{T}_1} + \mathbf{1}_{y \in \mathcal{T}_2} - 1 \\ &= \mathbf{1}_{x \in \mathcal{T}_1} - \mathbf{1}_{y \in \mathcal{T}_1}. \end{aligned} \quad (9)$$

In Equation 9, the terms in x and y are linearly separated, so we can write:

$$\begin{aligned} &\sum_{k=1}^{N} \left(\mathbf{1}_{x_{i_k} \in \mathcal{T}_1 \cap y_{j_k} \in \mathcal{T}_2} - \mathbf{1}_{x_{i_k} \in \mathcal{T}_2 \cap y_{j_k} \in \mathcal{T}_1} \right) \\ &= \sum_{k=1}^{N} \left(\mathbf{1}_{x_{i_k} \in \mathcal{T}_1 \cap y_{j_{k+1}} \in \mathcal{T}_2} - \mathbf{1}_{x_{i_k} \in \mathcal{T}_2 \cap y_{j_{k+1}} \in \mathcal{T}_1} \right). \end{aligned} \quad (10)$$

Equations 8 and 10 imply that

$$\sum_{k=1}^{N} \mathbf{1}_{x_{i_k} \in \mathcal{T}_1 \cap y_{j_k} \in \mathcal{T}_2} > \sum_{k=1}^{N} \mathbf{1}_{x_{i_k} \in \mathcal{T}_1 \cap y_{j_{k+1}} \in \mathcal{T}_2}$$

and

$$\sum_{k=1}^{N} \mathbf{1}_{x_{i_k} \in \mathcal{T}_2 \cap y_{j_k} \in \mathcal{T}_1} > \sum_{k=1}^{N} \mathbf{1}_{x_{i_k} \in \mathcal{T}_2 \cap y_{j_{k+1}} \in \mathcal{T}_1}.$$

In particular, the left terms of both inequalities are non-zero, because they are strictly greater than a positive sum. Therefore, there must exist two indices k, l such that $x_{i_k} \in \mathcal{T}_1$, $y_{j_k} \in \mathcal{T}_2$, $x_{i_l} \in \mathcal{T}_2$ and $y_{j_l} \in \mathcal{T}_1$. The shortest paths between two elements of $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) is always contained in $\mathcal{T}_1$ (resp. $\mathcal{T}_2$); these indices hence satisfy:

$$\mathcal{P}(x_{i_k}, y_{j_l}) \cap \mathcal{P}(x_{i_l}, y_{j_k}) = \emptyset.$$

Moreover, both $\mathcal{P}(x_{i_k}, y_{j_k})$ and $\mathcal{P}(x_{i_l}, y_{j_l})$ contain the edge e, since that e is the only edge connecting $\mathcal{T}_1$ to $\mathcal{T}_2$. In particular, $\mathcal{P}(x_{i_l}, y_{j_l})$ and $\mathcal{P}(x_{i_k}, y_{j_k})$ both contain the least common ancestors $x_{i_k} \wedge y_{j_l}$ and $x_{i_l} \wedge y_{j_k}$, which implies the following inclusions:

$$\begin{aligned} &\mathcal{P}(x_{i_l}, x_{i_l} \wedge y_{j_k}) \subset \mathcal{P}(x_{i_l}, y_{j_l}); \\ &\mathcal{P}(x_{i_k} \wedge y_{j_l}, y_{j_l}) \subset \mathcal{P}(x_{i_l}, y_{j_l}); \\ &\mathcal{P}(x_{i_k}, x_{i_k} \wedge y_{j_l}) \subset \mathcal{P}(x_{i_k}, y_{j_k}); \\ &\mathcal{P}(x_{i_l} \wedge y_{j_k}, y_{j_k}) \subset \mathcal{P}(x_{i_l}, y_{j_l}). \end{aligned}$$

Since that

$$\mathcal{P}(x_{i_k}, y_{j_l}) = \mathcal{P}(x_{i_k}, x_{i_k} \wedge y_{j_l}) \cup \mathcal{P}(x_{i_k} \wedge y_{j_l}, y_{j_l})$$

and

$$\mathcal{P}(x_{i_l}, y_{j_k}) = \mathcal{P}(x_{i_l}, x_{i_l} \wedge y_{j_k}) \cup \mathcal{P}(x_{i_l} \wedge y_{j_k}, y_{j_k}),$$

we have that:

$$\mathcal{P}(x_{i_k}, y_{j_l}) \cup \mathcal{P}(x_{i_l}, y_{j_k}) \subsetneq \mathcal{P}(x_{i_k}, y_{j_k}) \cup \mathcal{P}(x_{i_l}, y_{j_l}). \tag{11}$$

Therefore, it is possible to strictly improve the matching Π by swapping (x_{i_k}, y_{j_k}) with (x_{i_k}, y_{j_l}) and (x_{i_l}, y_{j_l}) with (x_{i_l}, y_{j_k}). Since that the shortest path $\mathcal{P}(x, y)$ between any $x, y \in \mathcal{X}$ is independent of the tree weights, Equation 6 ensures that the swapped matching is strictly better for any tree distance $d_{\mathcal{T},w'}$, and that Π is not optimal for any $d_{\mathcal{T},w'}$. By contraposition, this proves that the set of optimal matchings is identical for any weight of the tree edges. □

Proposition 3. *If $(\mathcal{X}, d)$ is an ultrametric space, then the matching output by Algorithm 1 is optimal.*

Proof of Proposition 3. Let $(\mathcal{X}, d)$ an ultrametric space. Then, from Proposition 1 there exists a tree $\mathcal{T} = (\mathcal{X}, \mathcal{V}, \mathcal{E})$ and a weight function w such that $d = d_{\mathcal{T},w}$. Let $x_1, \dots, x_n \in \mathcal{X}$, $y_1, \dots, y_m \in \mathcal{X}$, $\mu \in \mathbb{R}^n$ and $\nu \in \mathbb{R}^m$ be two weight vectors, and $\tilde{\Pi}$ be the greedy matching output by Algorithm 1. Let assume that $\tilde{\Pi}$ is not optimal. Then, the proof of Proposition 2 shows that there exists $(i_1, j_1), (i_2, j_2) \in \mathrm{Supp}(\tilde{\Pi})$ that can be swapped to strictly improve the matching $\tilde{\Pi}$; that is, indices that satisfy:

$$d(x_{i_1}, y_{j_2}) + d(x_{i_2}, y_{j_1}) < d(x_{i_1}, y_{j_1}) + d(x_{i_2}, y_{j_2}). \tag{12}$$

By symmetry, let us assume that:

$$d(x_{i_1}, y_{j_2}) \leq d(x_{i_2}, y_{j_1}). \tag{13}$$

We will now prove that:

$$d(x_{i_1}, y_{j_2}) < \min(d(x_{i_1}, y_{j_1}), d(x_{i_2}, y_{j_2})). \tag{14}$$

First, by ultrametric property, we have:

$$d(x_{i_2}, y_{j_1}) \leq \max(d(x_{i_2}, y_{j_2}), d(y_{j_2}, x_{i_1}), d(x_{i_1}, y_{j_1})). \tag{15}$$

Because of Equation 13, Equation 12 implies:

$$d(y_{j_2}, x_{i_1}) \leq \max(d(x_{i_2}, y_{j_2}), d(x_{i_1}, y_{j_1})).$$

Therefore, Equation 15 becomes:

$$d(x_{i_2}, y_{j_1}) \leq \max(d(x_{i_2}, y_{j_2}), d(x_{i_1}, y_{j_1})). \tag{16}$$

Similarly, by ultrametric property and using Equation 13, we have:

$$\begin{aligned} d(x_{i_1}, y_{j_1}) &\leq \max(d(x_{i_1}, y_{j_2}), d(y_{j_2}, x_{i_2}), d(x_{i_2}, y_{j_1})) \\ &\leq \max(d(y_{j_2}, x_{i_2}), d(x_{i_2}, y_{j_1})) \end{aligned} \tag{17}$$

and

$$\begin{aligned} d(x_{i_2}, y_{j_2}) &\leq \max(d(x_{i_2}, y_{j_1}), d(y_{j_1}, x_{i_1}), d(x_{i_1}, y_{j_2})) \\ &\leq \max(d(x_{i_2}, y_{j_1}), d(y_{j_1}, x_{i_1})). \end{aligned} \tag{18}$$

Let us further assume that:

$$d(x_{i_2}, y_{j_2}) < d(x_{i_1}, y_{j_1}).$$

Then, in order for Equation 17 to be satisfied, we must have:

$$d(x_{i_1}, y_{j_1}) \leq d(x_{i_2}, y_{j_1}).$$

Conversely, if we have:

$$d(x_{i_1}, y_{j_1}) < d(x_{i_2}, y_{j_2}),$$

Equation 18 implies:

$$d(x_{i_2}, y_{j_2}) \leq d(x_{i_2}, y_{j_1}).$$

Therefore, if $d(x_{i_1}, y_{j_1}) \neq d(x_{i_2}, y_{j_2})$, then:

$$d(x_{i_2}, y_{j_1}) \geq \max(d(x_{i_1}, y_{j_1}), d(x_{i_2}, y_{j_2}));$$

Combined with Equation 16, we obtain:

$$d(x_{i_2}, y_{j_1}) = \max(d(x_{i_1}, y_{j_1}), d(x_{i_2}, y_{j_2})),$$

and Equation 12 then implies Equation 14. The same inequality also holds when $d(x_{i_1}, y_{j_1}) = d(x_{i_2}, y_{j_2})$ as a direct consequence of Equation 12.

But Equation 14 contradicts the greedy property of $\tilde{\Pi}$. Indeed, because of Equation 14, the edge (x_{i_1}, y_{j_2}) should be then processed before edges (x_{i_1}, y_{j_1}) and (x_{i_2}, y_{j_2}). After processing, the remaining mass of either x_{i_1} or y_{j_2} would be zero, and either $(i_1, j_1) \notin \mathrm{Supp}(\tilde{\Pi})$ or $(i_2, j_2) \notin \mathrm{Supp}(\tilde{\Pi})$. This proves that the greedy matching for ultrametrics must be an optimal matching. □

Proposition 4. *Let $\mathcal{T}$ be a rooted tree and $\mathcal{X}$ its set of leaves. Then, there exists a mapping $f : \mathcal{X} \to \mathbb{R}$ that defines a distance $d_f(x, y) = |f(x) - f(y)|$ such that for each ultrametric d on $\mathcal{T}$ and for each $x, y, z \in \mathcal{X}$,*

$$d(x, y) < d(x, z) \implies d_f(x, y) < d_f(x, z).$$

In particular, the greedy matching on d_f provides an optimal matching for any tree distance on $\mathcal{T}$.

Proof of Proposition 4. We will show that Algorithm A provides a mapping f satisfying the property of the proposition.

Indeed, by construction, the function f output by Algorithm A satisfies the following separation property:

if $\tilde{\mathcal{T}}, \tilde{\mathcal{T}}'$ are two distinct subtrees of the root r of $\mathcal{T}$, and if $x, y \in \tilde{\mathcal{T}}$, $z \in \tilde{\mathcal{T}}'$, then we have that :

$$|f(x) - f(y)| < |f(x) - f(z)|. \tag{19}$$

Indeed, depending on the processing order of $\tilde{\mathcal{T}}$ and $\tilde{\mathcal{T}}'$ in the first foreach-loop of the algorithm, two cases may happen:

- $\tilde{\mathcal{T}}$ is processed before $\tilde{\mathcal{T}}'$: then, $|f(x) - f(z)|$ is strictly larger than the value of Δk_{old} computed right after the $\tilde{\mathcal{T}}$ iteration of the for-loop, wich in turn is larger than $|f(x) - f(y)|$, and Equation 19;
- $\tilde{\mathcal{T}}'$ is processed before $\tilde{\mathcal{T}}$: then, $|f(x) - f(z)|$ is strictly larger than the value of Δk computing during the $\tilde{\mathcal{T}}$ iteration, which is larger than $|f(x) - f(y)|$.

Hence, in both cases, Equation 19 is satisfied. By recursion, this property holds for any subtree of $\mathcal{T}$. Therefore, for each $x, y, z \in \mathcal{X}$, we have that:

$$(x \wedge y) \prec (x \wedge z) \implies |f(x) - f(y)| < |f(x) - f(z)|,$$

where $x \wedge y$ (resp. $x \wedge z$) denotes the lowest common ancestor of x and y (resp. x and z), and $(x \wedge y) \prec (x \wedge z)$ means that $x \wedge z$ is an ancestor of $x \wedge y$ in $\mathcal{T}$. Let now d be an ultrametric on rooted tree $\mathcal{T}$. Ultrametrics are always increasing by least common ancester order (Leclerc (1981)): for each $x, y, z \in \mathcal{X}$,

$$d(x, y) < d(x, z) \implies (x \wedge y) \prec (x \wedge z),$$

which in turns implies $d_f(x, y) < d_f(x, z)$. This proves the inequality of Proposition 4. Let now prove that if $\tilde{\Pi}$ is a greedy matching for d_f, then $\tilde{\Pi}$ is a greedy matching for d as well. Since $\tilde{\Pi}$ is greedy, there exists an ordered set of indices $(i_1, j_i), \ldots, (i_N, j_N)$ such that $\mathrm{Supp}(\tilde{\Pi}) = \{(i_1, j_i), \ldots, (i_N, j_N)\}$ and for each $u \in \{1, \ldots, n\}$, $v, w \geq u$, the following inequality holds:

$$d\left(x_{i_u}, y_{j_u}\right) \leq d\left(x_{i_v}, y_{j_w}\right).$$

In particular, for each $u \geq 2$,

$$d_f\left(x_{i_1}, y_{j_1}\right) \leq d_f\left(x_{i_1}, y_{j_u}\right)$$

and

$$d_f\left(x_{i_1}, y_{j_1}\right) \leq d_f\left(x_{i_u}, y_{j_1}\right).$$

Then, the inequality of Proposition 4 implies:

$$d\left(x_{i_1}, y_{j_1}\right) \leq d\left(x_{i_1}, y_{j_u}\right)$$

and

$$d\left(x_{i_1}, y_{j_1}\right) \leq d\left(x_{i_u}, y_{j_1}\right).$$

It means that $d(x_{i_1}, y_{j_1})$ is the smallest distance among the ones involving either x_{i_1} or y_{j_1}: there exists an ordering of $\mathrm{Supp}(\tilde{P}i)$ in increasing order of d such that (i_1, j_1) is the first pair containing i_1 or j_1. Therefore, there exists a greedy matching $\tilde{\Pi}'$ for d such that $\tilde{\Pi}'_{i_1,j_1} = \tilde{\Pi}_{i_1,j_1}$. By recursion, we eventually show that the matching $\tilde{\Pi}$ is a greedy matching for d. □