A SCALABLE CONSTANT-FACTOR APPROXIMATION ALGORITHM FOR W_p OPTIMAL TRANSPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

Let (X, d) be a metric space and let μ, ν be discrete distributions supported on finite point sets $A, B \subseteq X$. For any $p \in [1, \infty]$, the W_p-distance between μ and ν, $W_p(\mu, \nu)$, is defined as the p-th root of the minimum cost of transporting the mass from μ to ν, where moving a unit of mass from $a \in A$ to $b \in B$ incurs a cost of $d^p(a, b)$. We give a (Las Vegas) randomized algorithm that always computes a $(4 + \varepsilon)$-approximate W_p optimal-transport (OT) plan in $O(n^2 + (n^{3/2}\varepsilon^{-1} \log^2 \Delta)^{1+o(1)})$ expected time, for all $p \in [1, \infty]$, where $\varepsilon > 0$ is an arbitrarily small constant and Δ is the spread of $A \cup B$. The best previous result achieved an $O(\log n)$-approximation in $O(pn^2)$ time, but only for constant p. Our algorithm significantly improves the approximation factor and, importantly, is the first quadratic-time method that extends to the W_∞-distance. In contrast, additive approximation methods such as Sinkhorn are efficient only for constant p and fail to handle $p = \infty$. Finally, we show that obtaining a relative approximation factor better than 2 in $O(n^2)$ time would resolve the long-standing open problem of computing a perfect matching in an arbitrary bipartite graph in quadratic time.

1 INTRODUCTION

Let μ and ν be discrete probability distributions supported on sets A and B, respectively, with $|A| + |B| = n$. For each pair $(a, b) \in A \times B$, let $\mathsf{d}(a, b)$ denote the ground distance between a and b. A *transport plan* is a function $\sigma : A \times B \to \mathbb{R}_{\geq 0}$ that assigns a mass to each pair (a, b) such that $\sum_{b \in B} \sigma(a, b) \leq \mu(a)$ and $\sum_{a \in A} \sigma(a, b) \leq \nu(b)$. Given a parameter $p \geq 1$, suppose the cost of moving a unit of mass from a point $a \in A$ to a point $b \in B$ is given by $\mathsf{d}(a, b)^p$. The W_p cost of any transport plan σ between μ and ν is defined as

$$w_p(\sigma) := \left(\sum_{a \in A, b \in B} \sigma(a, b) \times \mathsf{d}(a, b)^p \right)^{1/p}.$$

The W_p cost above for finite p extends naturally to the W_∞ cost of a transport plan σ, defined as

$$w_\infty(\sigma) := \lim_{p \to \infty} w_p(\sigma) = \max_{a, b \in A \times B \,:\, \sigma(a,b) > 0} \mathsf{d}(a, b).$$

In the W_p optimal transport (OT) problem, we wish to compute the transport plan σ^* that transports the entire mass and has the smallest W_p cost. We refer to the cost $w_p(\sigma^*)$ as the W_p-*distance* and denote it by $W_p(\mu, \nu)$. If μ and ν are uniform distributions, then we refer to the W_p OT problem as the W_p *matching* problem.

The W_p distances, for varying values of p, possess several appealing properties that make it favorable in applications. OT plans under the W_1 distance measure total displacement and are therefore useful to capture structural properties such as semantic relationships from word embeddings (Kusner et al. (2015)). OT plans arising from the W_2 distance have nice structural qualities such as monotonicity (Brenier (1991); Aurenhammer et al. (1998)) and translation invariance (Cohen & Guibas (1999)), and tend to preserve the geometry of the distributions. Furthermore, recent work in machine learning and topological data analysis uses W_∞ distance to establish consistency and convergence properties of topological summaries (Vishwanath et al. (2020); Damrich et al.

(2024)), and to design topological layers in neural networks (Kim et al. (2020)). Due to these favorable properties, W_p distances has been used in applications across machine learning (Chang et al. (2023); Chuang et al. (2022)), computer vision (Backurs et al. (2020); Lai et al. (2022)), and natural language processing (Alvarez-Melis & Jaakkola (2018); Yurochkin et al. (2019)).

From an algorithmic standpoint, the exact computation of the W_p distance between discrete distributions can be formulated as a minimum-cost flow (MCF) problem, which can be solved in $n^{2+o(1)}$ time using recent advances in MCF algorithms (Chen et al. (2022)). While these results mark important theoretical progress, they are highly complicated, making them unsuitable for practical implementations. Indeed, even the simpler task of designing a truly quadratic-time exact algorithm for deciding whether a dense graph admits a perfect matching remains a longstanding open problem in graph theory (Behnezhad et al. (2024)).

Because exact algorithms remain expensive, research has shifted toward scalable approximation methods. A seminal result by Charikar (2002) introduced an $O(\log n)$-approximation for W_1 by embedding the ground metric into a hierarchically well-separated tree; a greedy transport procedure on the tree yielded an exact solution in $O(n^2)$ time, producing an overall $O(\log n)$-approximation. This work opened the door to more refined methods, and subsequent efforts have developed near-linear-time $(1+\varepsilon)$-approximation algorithms under additional assumptions, such as when the ground distance is Euclidean in fixed dimensions (Agarwal et al. (2022; 2024); Fox & Lu (2023)), and more recently, sub-quadratic algorithms for higher-dimensional Euclidean settings (Andoni & Zhang (2023); Beretta et al. (2025)). However, none of these techniques extend naturally to the case $p \geq 2$. Building on this line of work, Lahn et al. (2025) recently presented a relative $O(\log n)$-approximation algorithm for any finite $p \geq 2$, with runtime $O\left(n^2 \log U \log \Delta \log n\right)$, where $\log U$ is the bit-length of the input probabilities and Δ is the spread of $A \cup B$ (the ratio of its largest to smallest nonzero pairwise distance).

One influential direction of work was introduced by Cuturi (2013), who proposed entropic regularization of OT. It guarantees solutions within an additive error of $\varepsilon\Delta$, where Δ denotes the maximum ground distance between points in $A \cup B$. Although weaker than a relative $(1+\varepsilon)$-approximation, it applies across all metrics and inspired a series of additive approximation algorithms, including parallelizable variants (Altschuler et al. (2017); Dvurechensky et al. (2018); Jambulapati et al. (2019); Lahn et al. (2019; 2023)). Nonetheless, their runtimes remain on the order of $n^2/\varepsilon^{O(1)}$ for W_1 and worsen to $n^2/\varepsilon^{O(p)}$ for larger p, with no extension to the case $p = \infty$.

Our results. In this paper, we present two constant-factor approximation algorithms for the W_p problem. These are the first truly quadratic-time (assuming $\Delta + U = 2^{O(n^{1/8})}$) approximation algorithms for the W_p problem over *any ground metric*, applicable to all $p \in [1, \infty]$, including $p = \infty$. This improves the $O(\log n)$-approximation of Lahn et al. (2025) to a constant factor, while extending the guarantee to every p.

Theorem 1.1. *Let μ and ν be two discrete distributions supported on a set of n points in an arbitrary metric space. Let $p \in [1, \infty]$ be a parameter, and let $\varepsilon > 0$ be an arbitrarily small constant. A $(4+\varepsilon)$-approximate OT plan under the W_p metric can be computed in $O(n^2 + (n^{3/2}\varepsilon^{-1} \log^2 \Delta \log U)^{1+o(1)})$ expected time, where Δ is the spread of the support set and U is the ratio of the max to min probability in μ or ν.*

Our main technical contribution is a technique for approximating $\mathsf{d}(\cdot,\cdot)^p$, inspired by Bourgain's multi-level sampling (Bourgain (1985)). This multi-level sampling has inspired a sequence of results based on clustering points, much like the approach we adopt in this paper. Broadly, these results fall into two categories: spanner constructions and distance oracles for metric spaces. The spanners (Har-Peled et al. (2023); Baswana & Sen (2007); Cohen (1998)) are typically designed to approximate the underlying metric, whereas in our setting we seek to approximate $\mathsf{d}^p(\cdot,\cdot)$, which is not a metric. This distinction forces us to design a clustering scheme and a *directed* spanner that is not strongly connected, but in which shortest paths nevertheless preserve $\mathsf{d}^p(\cdot,\cdot)$ within a factor of $(4+\varepsilon)^p$. Using a minimum-cost flow algorithm on the directed spanner we construct will imply Theorem 1.1.

Since the interior point method in Chen et al. (2022) does not admit a practical implementation, we also design a simple combinatorial algorithm for the W_p matching problem, i.e., when μ and ν are uniform distributions. A parallel line of work instead leverages such clusterings to de-

sign distance oracles, i.e., data structures that allow efficient querying of distances between two points (Thorup & Zwick (2005); Mendel & Naor (2007); Awerbuch et al. (1998)). However, these oracles cannot be used to answer bichromatic closest-pair (BCP) queries, which are central to our setting. To overcome this limitation, we tailor our clustering scheme to efficiently support both weighted nearest-neighbor (WNN) and weighted bichromatic closest-pair queries. Importantly, although answering pairwise-distance queries are expensive in our framework, our algorithms avoid them entirely and instead rely solely on BCP and WNN queries. Using these data structures, we obtain a simpler combinatorial algorithm to compute a $(4+\varepsilon)$-approximate W_p matching.

Theorem 1.2. *Let A and B be two point sets of size n each in an arbitrary metric space, and let $p \in [1,\infty]$ be a parameter. A $(4+\varepsilon)$-approximate W_p matching of A and B can be computed in $O(n^2\varepsilon^{-2}\log^2\Delta)$ expected time, and an $(8+\varepsilon)$-approximate W_p matching of A and B can be computed in $O(n^2 + n^{5/3}\varepsilon^{-2}\log^2\Delta)$ time.*

To our knowledge, there are no known practical and implementable approximation algorithms for the W_∞-matching problem that run in $o(n^{2.5})$ time.

Next, we establish conditional lower bounds that suggest our results cannot be significantly improved without a major breakthrough in the graph matching problem, namely, computing a perfect matching in any graph in $O(n^2)$ time.

Theorem 1.3. *If there exists a quadratic-time algorithm that achieves a $(2-\varepsilon)$-relative approximation or $\Delta/2-\varepsilon$ additive approximation for the W_∞-matching problem, where Δ is the diameter of the point set and $\varepsilon > 0$ is a constant, then a perfect matching in an arbitrary graph can be computed in $O(n^2)$ time if one exists.*

We conclude with a primitive implementation of our simple combinatorial algorithm alongside some experimental results suggesting that the algorithm computes good quality W_p-matchings for $p \in [1,\infty]$ in Section 4. While we prove that the approximation factor of our algorithm is $(4+\varepsilon)$ in the worst case, our experimental results indicate that our algorithm computes even better approximations of W_p-matchings in practice.

2 Distance Approximation and Proximity Queries

Let P be a set of points, and let $\mathsf{d}\colon P\times P\to\mathbb{R}_{\geq 0}$ be a metric. We describe a clustering based distance function that approximates $\mathsf{d}(\cdot,\cdot)$, similar to the methods for constructing k-spanners and distance oracles discussed in Section 1, and that can be represented using roughly $n^{3/2}$ space (as opposed to $O(n^2)$ space to store all pairwise distances), and we use it to construct a spanner and to maintain bichromatic closest pairs.

We present only a two-layered clustering in the main text. Similar to the prior works, this layered clustering approach can be generalized to a k-level clustering. Extending to k-level clustering has a reduced number of clusters in which any point is expected to participate, at the expense of an increased stretch factor of the data structure. We provide more details about the k-level clustering in Appendix B.

We begin with a few notations. Given a point $x\in P$ and a subset $Q\subseteq P$, the distance from x to Q is defined as $\mathsf{d}(x,Q)=\min_{q\in Q}\mathsf{d}(x,q)$. For a point $q\in P$ and subset $Q\subseteq P$, define the *Voronoi region* of q to be

$$V(q,Q) := \{y\in P \mid \mathsf{d}(y,q) < \mathsf{d}(y,Q)\}.$$

That is, $V(q,Q)$ consists of the points in P for which q is closer than any point in Q.

Two-layered clustering. We construct a two-layered clustering of points of P. Set $P_0=P$. Next, we choose a subset $P_1\subseteq P_0$ by sampling each point in P_0 independently with probability $n^{-1/2}$. The expected size of P_1 is $\mathbb{E}[|P_1|]=n^{1/2}$.

Let $\Delta = \max_{p,q\in P}\mathsf{d}(p,q)$ be the diameter of P. Without loss of generality, assume $\min_{p,q}\mathsf{d}(p,q)=1$ implying the metric space (P,d) also has spread Δ. We choose $\varepsilon>0$ to be a sufficiently small constant. Set $t=\lceil\log_{(1+\frac{\varepsilon}{4})}\Delta\rceil$, $r_0=0$, and $r_i=(1+\frac{\varepsilon}{4})^i$ for $1\leq i\leq t$. We generate two types of clusters: (i) For each $q\in P_0\setminus P_1$ and for every $i\leq t$,

define $C_q[i] = \{x \in V(q, P_1) \mid \mathsf{d}(x, q) \leq r_i\}$. (ii) For each $q \in P_1$ and for every $i \leq t$, define $C_q[i] = \{x \in P_0 \mid \mathsf{d}(x, q) \leq r_i\}$. We refer to i as the *index* of the cluster $C_q[i]$. Let $\mathcal{C} = \{C_q[i] \mid q \in P_0, i \leq t\}$ be the collection of all clusters. Note that a point $p \in P$ may belong to many clusters. The number of clusters that contain p is called the *degree* of p and is denoted as $\deg_{\mathcal{C}}(p)$. While the degree of any particular point may be as large as n in the worst case, we prove that the expected degree of each point in P is much smaller.

Lemma 2.1. $\mathbb{E}[\deg_{\mathcal{C}}(p)] = O(n^{1/2}\varepsilon^{-1}\log\Delta)$ *for all* $p \in P$.

Proof. We note that for any $0 \leq i \leq j \leq t$ and for any $q \in P$, $C_q[i] \subseteq C_q[j]$. There are at most $O(\varepsilon^{-1}\log\Delta)$ different values of i. Therefore it suffices to prove for any $p \in P$, the number of points $q \in P$ where $p \in C_q[t]$ is $O(n^{1/2})$ in expectation.

Fix an arbitrary $p \in P_0$. For points $q \in P_0 \setminus P_1$ we have that p can only participate in clusters centered at q if $\mathsf{d}(p, q) < \mathsf{d}(p, P_1)$. Let $w_1, \ldots, w_s$ be the points of P_0 ordered by non-decreasing distance to p. If $w_j \in P_0 \setminus P_1$ and $p \in C_{w_j}[t]$, then it must be the case that $w_1, \ldots, w_{j-1} \notin P_1$. We sampled the points P_1 independently from P_0 with probability $n^{-1/2}$, so we have

$$\Pr\left[p \in C_{w_j}[t]\right] \leq \prod_{t<j} \Pr\left[w_t \notin P_1\right] = \left(1 - \frac{1}{\sqrt{n}}\right)^j.$$

The expected number of points in $P_0 \setminus P_1$ with a cluster containing p is then

$$\sum_{s \leq n} \mathbb{1}(w_s \in P_0 \setminus P_1) \cdot \Pr\left[p \in C_{w_s}[t]\right] \leq \sum_{s \leq n} \left(1 - \frac{1}{\sqrt{n}}\right)^s \leq \sqrt{n}.$$

We additionally note that $\mathbb{E}[|P_1|] = \sqrt{n}$. Therefore, $\mathbb{E}[\deg_{\mathcal{C}}(p)] \leq 2\sqrt{n}$. □

Cluster-induced distance approximation. We define the distance function, $\mathsf{d}_{\mathcal{C}} \colon P \times P \to \mathbb{R}_{\geq 0}$ based on the clustering. For any pair of points $x, y \in P$, let i be the smallest index of a cluster that contains both x and y. Then we set $\mathsf{d}_{\mathcal{C}}(x, y) = 2r_i$.

Lemma 2.2. $\mathsf{d}(x, y) \leq \mathsf{d}_{\mathcal{C}}(x, y) < (4 + \varepsilon)\mathsf{d}(x, y)$.

Proof. We say two points x and y are *separated* by P_1 if there are points $a, b \in P_1$ such that $\mathsf{d}(x, a) < \mathsf{d}(x, y)$ and $\mathsf{d}(y, b) < \mathsf{d}(x, y)$. Without loss of generality, let $\mathsf{d}(x, a) \leq \mathsf{d}(y, a)$. Then $y \in C_a[i]$ for i such that $\mathsf{d}(y, a) \leq r_i = (1 + \frac{\varepsilon}{4})^i < (1 + \frac{\varepsilon}{4})\mathsf{d}(y, a)$. So we have

$$\mathsf{d}_{\mathcal{C}}(x, y) = 2r_i = 2\left(1 + \frac{\varepsilon}{4}\right)^i < 2\left(1 + \frac{\varepsilon}{4}\right)\mathsf{d}(y, a) \leq 4\left(1 + \frac{\varepsilon}{4}\right)\mathsf{d}(x, y).$$

If x and y are not separated then either $x \in C_y$ or $y \in C_x$. Without loss of generality, assume $x \in C_y[i]$. Then we have $\mathsf{d}_{\mathcal{C}}(x, y) = 2r_i = 2\left(1 + \frac{\varepsilon}{4}\right)^i < 2\left(1 + \frac{\varepsilon}{4}\right)\mathsf{d}(x, y)$. □

2.1 Proximity Queries

Next, we show that the clustering constructed above can be used for answering some proximity queries, which will be crucial for our OT plan computation.

Directed spanner. Let $A, B \subseteq P$ be two disjoint subsets of P, let $\mathsf{d} \colon P \times P \to \mathbb{R}_{\geq 0}$ be a metric, and let $p \in [1, \infty)$. We construct a graph $G = (V, E)$ and a set of edge weights $w_p \colon E \to \mathbb{R}_{\geq 0}$ such that the shortest path from any $a \in A$ to any $b \in B$ in the graph G with respect to weights w_p is approximately $\mathsf{d}^p(a, b)$.

For each cluster $C \in \mathcal{C}$, we create two vertices a_C, b_C. Set $V = A \cup B \cup \{a_C, b_C \mid C \in \mathcal{C}\}$. For each cluster $C \in \mathcal{C}$, we add the following three sets of edges to E:

(i) Add the edge $a_C \to b_C$ and set $w_p(a_C \to b_C) = (2r_i)^p$ if the index of C is i.

(ii) For every $a \in A \cap C$, we add the edge $a \to a_C$ and set $w_p(a \to a_C) = 0$.

(iii) For every $b \in B \cap C$, we add the edge $b_C \to b$ and set $w_p(b_C \to b) = 0$.

Clearly $|V| = O(n\varepsilon^{-1} \log \Delta)$ since $|\mathcal{C}| = O(n\varepsilon^{-1} \log \Delta)$. Since the expected degree of each point is $O(\sqrt{n}\varepsilon^{-1} \log \Delta)$, the expected number of edges is $O(n^{3/2}\varepsilon^{-1} \log \Delta)$. Define $\mathsf{d}_{G,p} \colon A \times B \to \mathbb{R}_{\geq 0}$ as the shortest path distance in G with respect to edge weights w_p.

Lemma 2.3. *The weighted graph G with weights w_p satisfies $\mathsf{d}^p(a,b) \leq \mathsf{d}_{G,p}(a,b) \leq (4+\varepsilon)^p \cdot \mathsf{d}^p(a,b)$ for all $a, b \in A \times B$ and for any $p \in [1, \infty)$.*

Weighted nearest neighbor. Let P be a point set, and let $A \subseteq P$. Given a weight function $w \colon A \to \mathbb{R}_{\geq 0}$, define the *weighted distance* $\mathsf{d}_w \colon A \times P \to \mathbb{R}$ as $\mathsf{d}_w(a,p) = \mathsf{d}_{\mathcal{C}}(a,p) - w(a)$. Our goal is to maintain the weighted nearest neighbor in A for every $p \in P$, i.e. $\mathrm{NN}_w(p) = \arg\min_{a \in A} \mathsf{d}_w(a,p)$, as points (of P) are inserted into and deleted from A.

We build the above clustering $\mathcal{C}$ on the entire set P. For each point $p \in A$, we store the list of clusters to which it belongs. For each $C \in \mathcal{C}$ such that $p \in P$, we maintain the points of $A \cap C$ in a max-heap using their weights. Let a_C be the point of $A \cap C$ stored at the root of the heap. If the index of C is i, we set $\phi_C = 2r_i - w(a_C)$. Next, we store the set $\mathsf{X} = \{(a_C, C) \mid C \in \mathcal{C}\}$ in a min-heap H using ϕ_C as the key. The first element a_C of the pair stored at the root of H is the desired nearest neighbor $\mathrm{NN}_w(p)$. Insertion or deletion of a point is straightforward. Omitting the details, we state that the expected update time is $O(n^{1/2}\varepsilon^{-1} \log(n) \log \Delta)$.

Dynamic bichromatic closest pair. Let P be a point set, and let $A, B \subseteq P$ be two disjoint point sets. Given a weight function $w \colon A \cup B \to \mathbb{R}_{\geq 0}$, we define the *weighted distance* $\mathsf{d}_w \colon A \times B \to \mathbb{R}$ as

$$\mathsf{d}_w(a,b) = \mathsf{d}_{\mathcal{C}}(a,b) - w(a) + w(b).$$

Our goal is to maintain $\mathrm{BCP}_w(A,B) = \arg\min_{(a,b) \in A \times B} \mathsf{d}_w(a,b)$ as points are inserted into and deleted from A and B. We only insert the points of P to A or B. We describe a simple data structure to maintain $\mathrm{BCP}_w(A,B)$.

As for NN queries, we build the above clustering $\mathcal{C}$ on the entire set P. For each $C \in \mathcal{C}$, we maintain the points of $B \cap C$ in a min-heap using their weights as the key, and we maintain the points of $A \cap C$ in a max-heap using their weights. Let a_C (resp. b_C) be the point of $A \cap C$ (resp. $B \cap C$) stored at the root of the heap. If the index of C is i, we set

$$\phi_C = 2r_i - w(a_C) + w(b_C).$$

Next, we store the set $\mathsf{X} = \{(a_C, b_C) \mid C \in \mathcal{C}\}$ in a min-heap H using ϕ_C as the key. The pair stored at the root of H is the desired pair $\mathrm{BCP}_w(A,B)$.

The following observation is critical to the design of the BCP data structure.

Lemma 2.4. *Let (a^*, b^*) be the pair stored at the root of H. Then $\mathsf{d}_w(a^*, b^*) = \min_{a,b \in A \times B} \mathsf{d}_w(a,b)$.*

A similar claim also appears in Lahn et al. (2025) as Lemma 2.3. We include the proof in Appendix C, for completeness. We also note that the BCP data structure can be updated efficiently.

Lemma 2.5. *Let P be a set of n points in a metric space. Let $A, B \subseteq P$ be two weighted point sets. A weighted BCP data structure under the distance function d_w can be maintained in $O(\sqrt{n}\varepsilon^{-1} \log \Delta \log n)$ expected time per insertion and deletion.*

3 Algorithms for W_p

In this section, we use the collection of clusters and data structures constructed in Section 2 to design two efficient algorithms for the optimal transport problem.

3.1 Minimum-Cost Flow Based Algorithm

Let μ and ν be discrete distributions with support sets A and B; let $|A| + |B| = n$. We compute an approximate W_p-OT as follows. First assume $p \geq 1$ is a finite value. Let $G = (V, E)$ be the directed graph constructed on $A \cup B$ described in Section 2.1, and let w_p be the corresponding

weight function on E. We add a source vertex s and sink vertex t to the graph G. We also add an edge $s \to a$ for every $a \in A$ with weight $w_p(s \to a) = 0$ and an edge $b \to t$ for every $b \in B$ with weight $w_p(b \to t) = 0$. This addition gives the graph a single source and single sink to run minimum cost flow. Next, we assign capacities to each edge as follows: For each cluster $C \in \mathcal{C}$ and corresponding edge $a_C \to b_C$ in E, assign a capacity of $u(a_C \to b_C) = 1$. Additionally, for each $a, b \in C$ we assign the capacity $u(a \to a_C) = u(b_C \to b) = 1$. Finally, for each $a \in A$ and $b \in B$, we assign the source and sink edge capacities as $u(s \to a) = \mu(a)$ and $u(b \to t) = \nu(b)$.

We compute the capacitated min-cost max- flow f^* in this directed graph using the algorithm by Chen et al. (2022) in $(n^{3/2}\epsilon^{-1}\log^2\Delta)^{1+o(1)}\log U$ expected time. Using the minimum cost flow f^*, we compute a transport plan σ where $w_p(\sigma) \leq \left(\sum_{e\in E} f^*(e)w_p(e)\right)^{1/p}$ as follows. Initially, $\sigma(a,b) = 0$ for all $a \in A, b \in B$. While the total flow from s to t is positive, find any path $\pi = s \to a \to a_C \to b_C \to b \to t$ where $f^*(e) > 0$ for every edge e on the path π and increment $\sigma(a,b)$ by $\lambda = \min\{f^*(e) \mid e \in \pi\}$. Additionally decrement $f^*(e)$ by λ for every edge $e \in \pi$. We repeat until f^* is zero everywhere. This concludes the construction of the transport plan σ. It follows naturally from Lemma 2.3 that σ is an approximate transport plan with respect to the W_p distance. Since G has $O(n^{3/2}\varepsilon^{-1}\log\Delta)$ edges, the overall expected runtime of constructing σ from f^* is $O(n^{3/2}\varepsilon^{-1}\log\Delta)$. This proves Theorem 1.1 for $p \in [1,\infty)$.

For $p = \infty$, we proceed as follows. We maintain the same source and sink vertices s, t as well as the same edge capacities as above. We then compute a sequence of maximum flows instead of a single minimum cost flow in G, and perform binary search on the radii of the clusters. By construction, there are at most $O(\varepsilon^{-1}\log\Delta)$ different values of r_i. For a fixed $1 \leq i \leq O(\varepsilon^{-1}\log\Delta)$, define the graph G_i to be the graph G with all edges of cost $w_1(a_C \to b_c) > 2r_i$ removed. Compute a maximum flow f_i in G_i from s to t in $(n^{3/2}\varepsilon^{-1}\log\Delta)^{1+o(1)}\log U$ expected time using the algorithm of Chen et al. (2022). If $\sum_{a\in A} f_i(s \to a) = 1$, then conclude that $W_\infty(\mu,\nu) \leq 2r_i$ and decrease i. Otherwise, conclude that $W_\infty(\mu,\nu) > 2r_i$ and increase i.

Let i^* be the smallest value of i such that $\sum_{a\in A} f_i(s \to x) = 1$. Then we compute a transport plan σ from f_{i^*} as above in the case when $p < \infty$. Initially, $\sigma(a,b) = 0$ for all $a \in A, b \in B$. While the total flow from s to t is positive, find any path $\pi = s \to a \to a_C \to b_C \to b \to t$ where $f_{i^*}(e) > 0$ for every edge e on the path π and increment $\sigma(a,b)$ by $\lambda = \min\{f_{i^*}(e) \mid e \in \pi\}$. Additionally decrement $f_{i^*}(e)$ by λ for every edge $e \in \pi$. We repeat until f_{i^*} is zero everywhere. This concludes the construction of the transport plan σ. It follows naturally from Lemma 2.3 that σ is an approximate transport plan with respect to the W_∞ distance. Similar to the algorithm for finite p, we observe that the overall expected runtime of constructing σ from f_{i^*} is $O(n^{3/2}\varepsilon^{-1}\log\Delta)$. This proves Theorem 1.1 for $p = \infty$.

3.2 A Simpler Matching Algorithm

We present a significantly simpler, combinatorial algorithm that runs in $\tilde{O}(n^2)$ time and computes a minimum–cost matching under the W_p metric between two point sets A and B, each of size n. The algorithm selects an appropriate parameter δ and simulates a single scale of the Gabow–Tarjan cost–scaling framework for bipartite matching, with all steps executed efficiently via a bichromatic closest–pair data structure. Given the parameter δ, we begin by defining scaled costs as a scaled version of the p-th power of the proxy distance $\hat{c}(a,b) = \lceil \frac{1}{\delta}\, \mathrm{d}_C^p(a,b) \rceil$. The algorithm proceeds with these integer costs $\hat{c}(a,b)$.

Matchings and augmenting paths A *matching* M is a collection of vertex–disjoint edges. A vertex not incident to any edge of M is said to be *free*. A matching is *perfect* if no vertex is free. Given a matching M, an *alternating path* is a path whose edges alternate between those in M and those outside M. An *augmenting path* is an alternating path whose two endpoints are free. Augmenting along such a path flips the membership of its edges in M, thereby increasing the size of the matching by one.

1-feasible matching. Each vertex $v \in A \cup B$ is assigned an integer dual variable $y(v)$. A matching M and dual weights $y(\cdot)$ are 1-*feasible* if

$$y(a) + y(b) \;\leq\; \hat{c}(a,b) + 1 \qquad \text{for all } (a,b) \in A \times B, \tag{1}$$

$$y(a) + y(b) \;=\; \hat{c}(a,b) \qquad \text{for all } (a,b) \in M. \tag{2}$$

We define the *slack* of an edge (a, b) with respect to a matching M and dual weights $y(\cdot)$ as

$$s(a,b) \;=\; \begin{cases} 0, & \text{if } (a,b) \in M, \\ \hat{c}(a,b) - y(a) - y(b) + 1, & \text{if } (a,b) \notin M. \end{cases}$$

An edge is *admissible* if $s(a, b) = 0$, and the set of admissible edges forms the *admissible graph*

We initialize the matching $M = \emptyset$ and set all dual weights to zero, i.e., $y(v) = 0$ for every $v \in A \cup B$. Note that (M, y) is 1-feasible. Let $B_F = B$ be the free vertices of B with resepct to M. The algorithm maintains a 1-feasible pair (M, y) consisting of a matching M and dual weights $y(\cdot)$, and executes iterations. Each iteration has the *dual adjustment* step, which builds an augmenting path of admissible edges, and *augmentation* step, which computes a maximal set of vertex-disjoint augmenting paths and augments the matching along these paths to increase the size of the matching. Next, we describe the dual adjustment and the augmentation steps.

Dual adjustment via BCP-based Hungarian Search. The Hungarian search procedure runs a Dijkstra-style shortest path search using slacks as edge lengths. This search is implemented using bichromatic closest pair (BCP) queries, described in Section 2.1, with implicit dual updates. The search maintains a tree. Let $U \subseteq B$: the set of vertices of B already added to this search tree and let $V \subseteq A$ be the set of vertices of A not yet added to the search tree. Initially, U contains all free vertices in B, each with distance label $\ell_b = 0$, and $V = A$. Define effective weights

$$w(b) = y(b) - \ell_b \quad \text{for } b \in U, \qquad w(a) = y(a) \quad \text{for } a \in V.$$

At each iteration, select the edge

$$(a,b) = \arg\min_{a' \in V,\, b' \in U} \big\{ s(a',b') + \ell_{b'} \big\} = \arg\min_{a' \in V,\, b' \in U} \big\{ \mathsf{d}^p(a',b') - w(a') - w(b') \big\}.$$

This minimization is exactly a BCP query, which we answer using the data structure described in Section 2.1.

Remove a from V, set $\ell_a = \ell_b + s(a, b)$, and add it to the search tree. If a is free, a shortest augmenting path has been found. Otherwise, let b' be its matched partner; set $\ell_{b'} = \ell_a$, update $w(b') = y(b') - \ell_{b'}$, and insert b' into U.

This procedure simulates Hungarian search procedure without explicitly updating all duals: offsets are stored in the effective weights and automatically incorporated by BCP queries. When the search terminates at a free vertex $a^\star \in A$, let $\Delta = \ell_{a^\star}$. The dual weights are then updated as

$$y(a) \leftarrow y(a) + \ell(a) \quad \text{for all } a \in S, \qquad y(b) \leftarrow y(b) - \ell(b) \quad \text{for all } b \in T,$$

where $S \subseteq A$ and $T \subseteq B$ are the sets of vertices reached. In practice, these updates are never carried out explicitly. Instead, the effective weights $w(\cdot)$ store the necessary offsets, and BCP queries automatically incorporate them.

Augmentation step using weighted nearest neighbor. Once the dual adjustment phase reaches a free vertex in A, the search guarantees that there is at least one augmenting path in the admissible graph. The algorithm then finds a maximal set of vertex-disjoint augmenting paths by conducting a sequence of partial depth-first search (DFS): Start a DFS from each free point of B in a sequential manner. Let X be the set of points of A that have not yet been visited by any DFS. Initially $X = A$. The DFS alternates between unmatched admissible edges from B to A and matched edges from A back to B. Whenever the DFS is at a vertex $u \in B$, the next admissible edge can be retrieved by a *weighted nearest neighbor query*:

$$a = \arg\min_{a' \in X} \big\{ \mathsf{d}^p(u,a') - y(u) - y(a') \big\}.$$

We use the data structure of Section 2.1 to answer the NN query. We then check whether (u, a) is admissible. Thus, a single nearest neighbor query suffices to reveal the next admissible edge. a is removed from X. If a is matched to b'', then we extend the alternating path by adding (a, b'') to it and the DFS continues from b''. If the DFS reaches a free vertex a in A, an augmenting path is

identified. The algorithm then starts a DFS from a different free point of B. This step terminates when a DFS has been executed from each of the free points of B.

Finally, all discovered augmenting paths are flipped simultaneously to update the matching. For each augmenting path and for every vertex $b \in B$ lying on it, reduce the dual weight by one: $y(b) \leftarrow y(b) - 1$. This correction guarantees that all newly matched edges remain tight under the 1-feasible condition.

The algorithm alternates between dual adjustment and augmentation until all vertices are matched. The final pair (M, y) is a perfect matching and remains 1-feasible throughout the execution.

Efficiency. We select the parameter δ so that the edge costs become integers and the optimal cost is scaled to $\Theta(n/\varepsilon)$. Scaling by δ preserves the true optimum, while rounding introduces at most an additive error of n. Moreover, the 1-feasible matching produced is itself within $+n$ of the rounded optimum (Gabow & Tarjan (1989)). Hence, the total deviation is at most $2n$, and whenever the rounded optimum is at least $2n/\varepsilon$, the resulting solution is guaranteed to be within a $(1+\varepsilon)$ factor of the true optimum.

Gabow and Tarjan showed that if the costs are integers and the value of the optimal solution is $O(n/\varepsilon)$, then a single scale of their algorithm converges in $O(\sqrt{n/\varepsilon})$ phases. In particular, when the optimal solution has value $2n/\varepsilon$, the algorithm terminates in $O(\sqrt{n/\varepsilon})$ phases, and combined with the error bounds above, produces a $(1+\varepsilon)$-approximation.

Each of the two steps—dual adjustment and augmentation—of a phase can be implemented using efficient geometric data structures. The dual adjustment step builds a weighted bichromatic closest pair (BCP) data structure on U and V and performs dynamic updates as U and V change. Since points are only added to U and deleted from V, the total number of updates cannot exceed $2n$, and therefore the total time spent in this step is bounded by $O(n)$ queries to the BCP data structure. The augmentation step builds a weighted nearest neighbor (WNN) data structure on all points of A, and as the points of A are visited by a depth-first search they are deleted from the structure. Thus, the augmentation step can also be implemented using $O(n)$ queries to the WNN data structure. Each of these queries is supported in $\tilde{O}(\sqrt{n})$ time (see Section 2.1), so the overall execution time per phase is $\tilde{O}(n^2)$.

Next, we describe how to choose a δ so that the optimal cost with the rounded costs is scaled to $\Theta(n/\varepsilon)$. Since the optimal cost can take values between 1 and $n\Delta$, we consider a sequence of scales

$$\delta_i \;=\; (1+\varepsilon)^i \cdot \frac{\varepsilon}{n}, \qquad i = 1, 2, \ldots, \left\lceil \log_{(1+\varepsilon)}(n\Delta) \right\rceil .$$

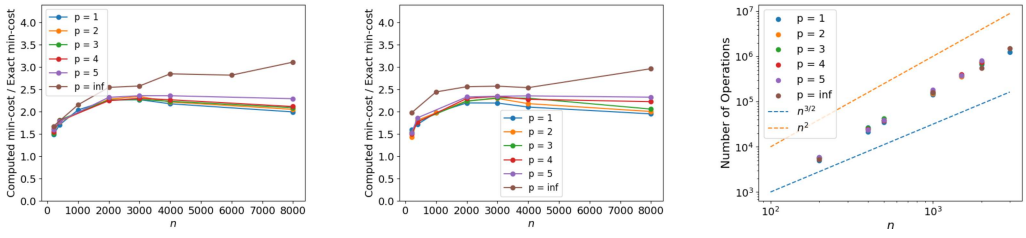
For each δ_i, we execute a single scale of the algorithm, each of which runs in $\tilde{O}(n^2)$ time. Among all executions that terminate within this bound, we return the matching of smallest cost. This proves Theorem 1.2.

4 Empirical Evaluation

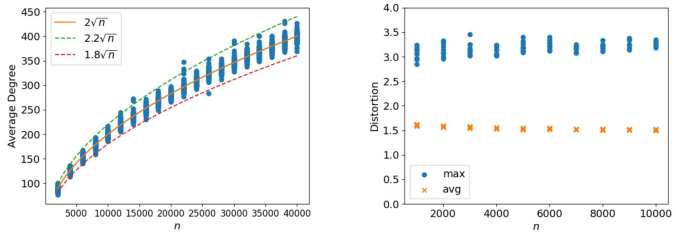
This section contains an empirical evaluation of the clustering method from Section 2 as well as the approximation factor obtained from the primal–dual algorithm of Section 3.2. Computations were performed on a computer with an 8-core Apple M1 CPU with 16GB RAM. Samples are drawn from uniform and truncated normal distributions on the unit cube in up to 10 dimensions.

Cluster distance accuracy. We first evaluate the quality of the clustering by comparing the induced cluster distances to the ground metric. For each value of n, we measure both the maximum and the average distortion across all pairs. Figure (1e) confirms that the worst-case distortion never exceeds the theoretical $(4+\varepsilon)$-approximation guarantee of Lemma 2.2. More importantly, the average distortion is often substantially smaller, typically close to a factor of 2. This suggests that in practice the effective approximation factor is significantly tighter than the worst-case analysis.

Clustering efficiency. Next, we examine the degree, ie. the number of clusters each point participates in. Figure (1d) shows that the observed averages closely track the theoretical bound of



(a) Approximation Ratio (Normal) (b) Approximation Ratio (Uniform) (c) Algorithm Efficiency



(d) Average Degree (e) Max and Average Distortion

Figure 1: Empirical evaluation of the 2-layer clustering and W_p-matching algorithm.

Lemma 2.1 for dimension $d \leq 10$, and both distributions. This indicates that the two-layer clustering is both space-efficient and stable across different settings.

Algorithm accuracy. To evaluate the accuracy of the primal–dual matching algorithm, we compare the computed matching cost to that obtained using the exact distance matrix. Figures (1b) and (1a) report the approximation ratio across values of $p \in \{1, 2, 3, 4, 5, \infty\}$. The ratios consistently remain well within the theoretical $(4 + \varepsilon)$ factor, with typical values close to 1.5–2, again suggesting that the empirical performance is considerably better than the worst-case analysis. This trend is stable across both uniform and normal distributions.

Algorithm efficiency. We measure efficiency by the number of bichromatic closest pair (BCP) queries, which dominate the running time. As shown in Figure (1c), the query counts scale as predicted and remain nearly identical across all choices of p. Combined with the $\tilde{O}(n^2)$ per-query complexity, this provides strong empirical evidence that the algorithm runs in quadratic time and scales smoothly with problem size.

Summary. Overall, the experiments demonstrate that the proposed method is both theoretically grounded and empirically robust. While the theoretical analysis guarantees only a $(4 + \varepsilon)$ approximation, the observed approximation ratios are consistently much smaller, indicating that the algorithm is practically near-optimal. The clustering step is efficient in both time and space, and its distortions are far below the worst-case bound. Taken together, these results suggest that our approach is a practical alternative to additive methods such as Sinkhorn, particularly in regimes where existing techniques either fail to apply (e.g., $p = \infty$) or require higher-than-quadratic time.

REFERENCES

Pankaj K Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. Deterministic, near-linear ε-approximation algorithm for geometric bipartite matching. In *Proc. 54th Annual ACM Sympos. Theory of Comput.*, pp. 1052–1065, 2022.

Pankaj K Agarwal, Sharath Raghvendra, Pouyan Shirzadian, and Keegan Yao. Fast and accurate approximations of the optimal transport in semi-discrete and discrete settings. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 4514–4529. SIAM, 2024.

Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in Neural Information Processing Sys-*

tems 30, pp. 1964–1974, 2017.

David Alvarez-Melis and Tommi S Jaakkola. Gromov-wasserstein alignment of word embedding spaces. *arXiv preprint arXiv:1809.00013*, 2018.

Alexandr Andoni and Hengjie Zhang. Sub-quadratic $(1+\epsilon)$-approximate euclidean spanners, with applications. In *IEEE 64th Annual Symposium on Foundations of Computer Science*, pp. 98–112. IEEE, 2023.

Franz Aurenhammer, Friedrich Hoffmann, and Boris Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1):61–76, 1998.

Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1):263–277, 1998.

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

Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007.

Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Approximating maximum matching requires almost quadratic time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pp. 444–454, 2024.

Lorenzo Beretta, Vincent Cohen-Addad, Rajesh Jayaram, and Erik Waingarten. Approximating high-dimensional earth mover's distance as fast as closest pair. *arXiv preprint arXiv:2508.06774*, 2025.

Jean Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985.

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

Wanxing Chang, Ye Shi, and Jingya Wang. Csot: Curriculum and structure-aware optimal transport for learning with noisy labels. *Advances in Neural Information Processing Systems*, 36:8528–8541, 2023.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th Annual ACM Sympos. Theory of Comput.*, pp. 380–388, 2002.

Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Sympos. Found. of Comput. Sci.*, pp. 612–623. IEEE, 2022.

Ching-Yao Chuang, R Devon Hjelm, Xin Wang, Vibhav Vineet, Neel Joshi, Antonio Torralba, Stefanie Jegelka, and Yale Song. Robust contrastive learning against noisy views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16670–16681, 2022.

Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1998.

Scott Cohen and L Guibas. The earth mover's distance under transformation sets. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pp. 1076–1083. IEEE, 1999.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Sebastian Damrich, Philipp Berens, and Dmitry Kobak. Persistent homology for high-dimensional data based on spectral methods. *Advances in Neural Information Processing Systems*, 37:41954–42014, 2024.

Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In *International Conference on Machine Learning*, pp. 1367–1376. PMLR, 2018.

Emily Fox and Jiashuai Lu. A deterministic near-linear time approximation scheme for geometric transportation. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1301–1315. IEEE, 2023.

Harold N Gabow and Robert E Tarjan. Faster scaling algorithms for network problems. *SIAM J. on Comput.*, 18:1013–1036, October 1989. ISSN 0097-5397.

Sariel Har-Peled, Manor Mendel, and Dániel Oláh. Reliable spanners for metric spaces. *ACM Transactions on Algorithms*, 19(1):1–27, 2023.

Arun Jambulapati, Aaron Sidford, and Kevin Tian. A direct $\tilde{O}(1/\epsilon)$ iteration parallel algorithm for optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

Kwangho Kim, Jisu Kim, Manzil Zaheer, Joon Kim, Frédéric Chazal, and Larry Wasserman. Pllay: Efficient topological layer based on persistent landscapes. *Advances in Neural Information Processing Systems*, 33:15965–15977, 2020.

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

Nathaniel Lahn, Deepika Mulchandani, and Sharath Raghvendra. A graph theoretic additive approximation of optimal transport. In *Advances in Neural Information Processing Systems*, pp. 13813–13823, 2019.

Nathaniel Lahn, Sharath Raghvendra, and Kaiyi Zhang. A combinatorial algorithm for approximating the optimal transport in the parallel and mpc settings. *Advances in Neural Information Processing Systems*, 36:21675–21686, 2023.

Nathaniel Lahn, Sharath Raghvendra, Emma Saarinen, and Pouyan Shirzadian. Scalable approximation algorithms for p-wasserstein distance and its variants. In *Forty-second International Conference on Machine Learning*, 2025.

Zhengfeng Lai, Chao Wang, Sen-ching Cheung, and Chen-Nee Chuah. Sar: Self-adaptive refinement on pseudo labels for multiclass-imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4091–4100, 2022.

Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1): 1–24, 2005.

Siddharth Vishwanath, Kenji Fukumizu, Satoshi Kuriki, and Bharath K Sriperumbudur. Robust persistence diagrams using reproducing kernels. *Advances in Neural Information Processing Systems*, 33:21900–21911, 2020.

Mikhail Yurochkin, Sebastian Claici, Edward Chien, Farzaneh Mirzazadeh, and Justin M Solomon. Hierarchical optimal transport for document representation. *Advances in neural information processing systems*, 32, 2019.