Computing All Optimal Partial p-Wasserstein Matchings on the Line

Sebastian Angrick
Karlsruhe Institute of Technology
sebastian.angrick@kit.edu

Jacobus Conradi
University of Copenhagen
jaco@di.ku.dk

Mónika Csikós
Université Paris Cité, IRIF
csikos@irif.fr

Niko Hastrich
ETH Zurich
niko.hastrich@inf.ethz.ch

Danny Mittal
University of Maryland, College Park
dannymittal@gmail.com

André Nusser
Université Côte d'Azur, CNRS, Inria
andre.nusser@cnrs.fr

Krzysztof Onak
Boston University
konak@bu.edu

Sharath Raghvendra
North Carolina State University
skraghve@ncsu.edu

Abstract

For $p \geq 1$, the p-Wasserstein distance measures the minimum cost of transporting probability mass between distributions, where moving unit mass between two points costs the pth power of their distance. For discrete distributions in one dimension, full transport is especially simple: after sorting, mass is matched in order along the line. By contrast, partial and unbalanced transport on the line remains much less understood. Recently, Chapel and Tavenard [ICLR'25] showed that, for $p = 1$, all optimal partial transport plans between distributions supported on n points, with uniform mass at each point, can be computed in $O(n \log n)$ time by exploiting the metric structure of the cost. For $p > 1$, this structure no longer applies, and existing approaches require $\Omega(n^2)$ time. Our main contribution is an FFT-based data structure for balanced-interval transport queries, which bypasses this quadratic bottleneck and yields for any finite $p \geq 1$, an $O(p\, n \log^2 n)$-time algorithm for computing all optimal partial p-Wasserstein transports costs on the line. We also provide an open-source C++ implementation that outperforms the state-of-the-art baseline on a range of synthetic instances. Finally, we show that computing all optimal partial ∞-Wasserstein transport costs is (up to poly-logarithmic factors) equivalent to $(\min, +)$-convolution, for which the fastest currently claimed algorithm requires $\Omega(n^{1.96})$ time.

1 Introduction

Optimal transport characterizes the difference between probability distributions μ and ν by measuring the minimum transportation cost with which we can transform μ into ν. The p-Wasserstein distance, a particular formulation of optimal transport, is defined as the pth root of the minimum cost of transporting mass from μ to ν, where moving an amount of mass δ from a to b costs $\delta\, d(a, b)^p$ and d denotes the underlying metric. This metric is used widely for comparing distributions and point sets [26]. Its appeal stems from its mathematical flexibility, its sensitivity to the underlying geometry, and the fact that optimal transport plans provide explicit correspondences between points, aiding interpretability and model transparency. These features have made the p-Wasserstein distance a central tool in machine learning and statistics [21, 29].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In many practical settings the data may be noisy, contain outliers, or exhibit only partial overlap, making it undesirable to transport all available mass. This has motivated substantial interest in the *partial* and *unbalanced* variants of optimal transport. In these settings, the problem is not only to determine the transport plan, but also to decide how much mass should participate in the transport, while identifying and excluding points that are better treated as outliers. A natural way to search for the ideal trade-off is through the *optimal transport profile* (or OT-profile), which maps each fraction $\alpha \in [0, 1]$ to the minimum cost of a transport plan that moves total mass α. OT-profiles have been used in applications such as noise filtering and positive-unlabeled learning [12, 18, 22] and are a key component in the definition of the robust partial Wasserstein metric [24], a robust variant of the p-Wasserstein distance.

In general, the OT-profile may have high representation complexity. We focus on the discrete uniform setting where each distribution is supported on n points of mass $1/n$. In this setting, transporting mass k/n amounts to selecting k vertex-disjoint pairs, each containing one point from each distribution; we call such a set of pairs a *k-matching*. For $1 \leq p < \infty$, the p-Wasserstein cost[1] of a k-matching M is the sum of the transport costs over its pairs, $\sum_{(a,b)\in M} d(a, b)^p$. For $p = \infty$, the cost is the maximum distance among its matched pairs, that is, $\max_{(a,b)\in M} d(a, b)$. Thus, the OT-profile is fully determined by the sequence of costs $C_1, \ldots, C_n$, where C_k is the minimum cost of any k-matching.

From an algorithmic perspective, computing a full optimal transport plan and computing the OT-profile have the same best-known asymptotic complexity in general settings. For arbitrary real-valued costs, both can be computed in $O(n^3)$ time: the Hungarian algorithm [17] constructs a minimum cost full matching by successively computing minimum cost k-matchings for $k = 1, \ldots, n$, and therefore produces the entire OT-profile as a byproduct [22]. In low-dimensional geometric settings, the full transport plan as well as the OT-profile can be computed in near-quadratic time using $O(n^2)$ queries to a dynamic bichromatic closest pair data structure [3, 28]. Thus, near-quadratic time is a natural baseline even for low-dimensional geometric instances. However, such running times can still be prohibitive for large-scale optimal transport problems.

The one-dimensional setting is a notable exception to this scalability challenge. Here, the transport cost between two points a and b is simply $|a - b|^p$, and full optimal transport can be computed by sorting the points and matching them in order, giving a near-linear-time algorithm for all $p \geq 1$ [21, 27, 29].

The difficulty of high-dimensional optimal transport has motivated Wasserstein-based alternatives that reduce computation to one-dimensional problems. A central example is the sliced Wasserstein distance, which averages Wasserstein costs after projecting distributions onto random lines [19, 21, 23]. Sliced Wasserstein inherits many useful geometric properties of Wasserstein distance while being significantly more scalable, precisely because one-dimensional OT is much easier to compute. Analogously, sliced partial Wasserstein distances were introduced to handle noisy real-world datasets [6].

However, the computational simplicity of full one-dimensional optimal transport does not automatically extend to its partial and unbalanced versions [6, 13]: efficiently computing the entire OT-profile remains nontrivial. Recently, [13] showed that for $p = 1$, the OT-profile on the line can be computed in $O(n \log n)$ time for distributions supported on n points with uniform mass, matching the complexity of computing full optimal transport in this setting. For all $p > 1$, however, their approach requires $\Theta(n^2)$ time. This raises a natural question:

For $p > 1$, can the OT-profile be computed with the same efficiency as full one-dimensional optimal transport, or does a quadratic barrier arise?

The difficulty of larger values of p can already be seen in the following simple example. Consider two point sets $B, R \subset \mathbb{R}$ with $B = \{b_1, \ldots, b_n\}$ and $R = \{r_1, \ldots, r_n\}$ with $b_1 < \cdots < b_n < r_1 < \cdots < r_n$. For $p = 1$, an optimal k-matching pairs the last k points of B with the first k points of R. Since any bijection between $\{b_{n-k+1}, \ldots, b_n\}$ and $\{r_1, \ldots, r_k\}$ has the same 1-Wasserstein cost, we can impose the following nested structure on the matching: $M_k = \{(b_{n-i+1}, r_i) : i = 1, \ldots, k\}$. Moving from M_k to M_{k+1} simply requires adding the edge (b_{n-k}, r_{k+1}) without modifying existing pairs. Hence, across all k only $O(n)$ distinct edges appear.

[1] For simplicity of presentation, we work with the pth power of the classical Wasserstein distance, which does not impact the optimality of a matching.

For $p > 1$, convexity breaks this degeneracy: the optimal pairing is now unique. Even though the min-cost k-matching still matches the last k points of B to the first k points of R, we now have to match them in order, i.e., the min-cost k-matching is $M_k = \{(b_{n-k+i}, r_i) : i = 1, \ldots, k\}$. Note that increasing the cardinality from k to $k+1$ forces every edge to shift in order to obtain the matching $M_{k+1} = \{(b_{n-(k+1)+i}, r_i) : i = 1, \ldots, k+1\}$. Thus M_k and M_{k+1} share *no* edges. The sequence $M_1, M_2, \ldots, M_n$ therefore contains $\Theta(n^2)$ distinct edges. Consequently, any approach that explicitly maintains the partial matchings incurs quadratic total complexity for $p > 1$.

1.1 Our Contribution.

In this work, we show that for any fixed $p \in \mathbb{N}$, the OT-profile under the p-Wasserstein cost can be computed in $O(p\, n \log^2 n)$ time. We complement this upper bound with an open-source `C++` implementation that outperforms the state-of-the-art on a broad range of synthetic instances. Finally, for $p = \infty$, we show that computing the OT-profile is computationally equivalent (up to logarithmic factors) to computing the $(\min, +)$-convolution. All asymptotic costs are computed in the Word-RAM model.

Upper Bounds. We show the following theorem, constituting our main result.

Theorem 1.1 (k-partial p-Wasserstein Distance)**.** *Let $B, R \subset \mathbb{R}$ be two point sets of size n. For any $p \in \mathbb{N}$, the k-partial p-Wasserstein distance between B and R can be computed for all $k \in [n]$ in a total of $O(p\, n \log^2 n)$ time using $O(n \log n)$ space.*

Similar to the approach of [13], our algorithm can be viewed as a one-dimensional specialization of the Hungarian algorithm. It maintains a collection of at most n disjoint balanced intervals, each containing an equal number of red and blue points, that compactly represents the current minimum-cost partial matching. Interestingly, a minimum-cost $(k+1)$-matching can be obtained from a minimum-cost k-matching by modifying only a few intervals. Thus, although all partial matchings together may contain $\Theta(n^2)$ distinct edges, their structure admits an $O(n)$-space representation.

The main remaining task is to identify which intervals to modify. This reduces to evaluating minimum matching costs inside balanced intervals, which is handled by our main technical contribution, summarized by the following theorem.

Theorem 1.2. *Let $B, R \subset \mathbb{R}$ be point sets of size n each. For any $p \in \mathbb{N}$, in $O(p\, n \log^2 n)$ time we can construct a data structure of size $O(n \log n)$ that, given a balanced query interval I (i.e., $|I \cap R| = |I \cap B|$), returns the p-Wasserstein cost of a minimum-cost matching between $I \cap R$ and $I \cap B$ in $O(\log n)$ time.*

Pairing red and blue points in order yields a minimum-cost matching inside any balanced interval. In such a matching, if the first red point in the interval has index k and the first blue point has index $k+t$, then every matched pair has the same *shift* t. The data structure of Theorem 1.2 exploits this fact with a one-dimensional range tree. Each node corresponding to a contiguous range of points $S \subset R \cup B$ stores matching-cost information for all $O(|S|)$ relevant shifts between red and blue points in S. Given a balanced query interval, we determine its shift t from its leftmost red and blue points, and recover its optimal matching cost by aggregating the precomputed shift-t values over the $O(\log n)$ maximal range-tree nodes into which the interval decomposes. This aggregation is nontrivial: matching costs are not generally decomposable over disjoint subproblems. Here, decomposability follows from the one-dimensional order structure together with balancedness; see Section 2. The remaining challenge is preprocessing all shift-values without quadratic work. We encode the costs for all shifts at a node as coefficients of carefully chosen polynomials; expanding $|a-b|^p$ and using FFT computes these values efficiently. Summed over the range tree, this gives $O(p\, n \log^2 n)$ preprocessing time and $O(\log n)$ query time.

Implementation. We complement our theoretical results with an open-source `C++` implementation. Compared with the state-of-the-art implementation of [13] on a broad collection of synthetic instances, our implementation is competitive throughout and orders of magnitude faster on most large inputs, with speedups that increase as the input size grows.

Lower Bounds. Our upper bound applies to every finite $p \in \mathbb{N}$, but not to $p = \infty$. For this case, we prove that computing all k-partial ∞-Wasserstein matching costs cannot be performed asymptotically faster than computing the $(\min, +)$-convolution. A classical hypothesis in fine-grained complexity stated that $(\min, +)$-convolution cannot be computed in truly sub-quadratic

time. This hypothesis was very recently refuted by Anthropic as reported by Alman and Vassilevska Williams [5]; still, our reduction shows that finding all k-partial ∞-Wasserstein matching costs is a computationally challenging problem. In particular, finding a near-linear algorithm would imply finding a near-linear algorithm for $(\min,+)$-convolution. A known reduction [15] shows that any $n^{2-\varepsilon+o(1)}$-time algorithm for 3SUM can be transformed into an $n^{2-\varepsilon/2+o(1)}$-time algorithm for $(\min,+)$-convolution. At the time of writing, this gives an upper bound of $O(n^{1.9825})$ for an algorithm with a human-readable proof [11] and $O(n^{1.9602})$ for an algorithm with a proof in Lean [8].

Although currently no formal lower bound establishes a separation, our reduction suggests a substantial gap between computing full transport and computing the OT-profile. For every $\varepsilon > 0$, full ∞-Wasserstein transport can be computed exactly in $O(n^{4/3+\varepsilon})$ time even in two dimensions [10], while achieving a comparable running time for the OT-profile in one dimension would be a major breakthrough for the $(\min,+)$-convolution problem.

We note that our technique for computing the OT-profile is directly applicable to the case $p = \infty$ by replacing FFT with $(\max,-)$-convolution, which is equivalent to computing the $(\min,+)$-convolution. Together, this shows that (up to poly-logarithmic factors) computing the OT-profile is computationally equivalent to computing the $(\min,+)$-convolution. To state this equivalence, let $\widetilde{O}(f)$ suppress poly-logarithmic factors depending on the input size, i.e., $\widetilde{O}(f(I)) = \bigcup_{i\in\mathbb{N}} O(f(I)\log^i(|I|))$ for an instance I and its encoding size $|I|$.

Theorem 1.3. *Let $\varepsilon > 0$. There is an algorithm computing all k-partial ∞-Wasserstein matching costs between two sets B and R of polynomially bounded integer points in time $\widetilde{O}(n^{2-\varepsilon})$ if and only if there is an algorithm computing the $(\min,+)$-convolution of arrays of size n and polynomially bounded entries in time $\widetilde{O}(n^{2-\varepsilon})$.*

In Appendix C, we give two additional lower bounds for a related data-structure setting: given $p \in \mathbb{N}$ and point sets $B, R \subset \mathbb{R}$, answer queries that for intervals $I_B, I_R \subset \mathbb{R}$ with $|B \cap I_B| = |R \cap I_R|$ return the p-Wasserstein distance between $B \cap I_B$ and $R \cap I_R$. Interestingly, this setting is harder than the setting with a single query interval, unless an almost-quadratic matrix multiplication algorithm for boolean matrices exists.

2 Data Structure for Balanced Interval Matching

In this section we describe the data structure from Theorem 1.2 supporting queries of the form: given a balanced interval I, i.e., an interval such that $|B \cap I| = |R \cap I|$, return the cost of a minimum-cost complete matching between $B \cap I$ and $R \cap I$. We denote this value by $w^p(I)$. This data structure constitutes our main technical contribution and allows us, using the algorithm described in Section 3, to efficiently compute minimum-cost k-matchings for all $k = 1, ..., n$ and $p \geq 1$.

The overall structure. Let $r_1, \ldots, r_n$ and $b_1, \ldots, b_n$ be the red and blue points in non-decreasing order, and let $p_1, \ldots, p_{2n}$ denote the merged sorted order of all points. Our data structure is a standard binary range tree $\mathcal{T}$ over the index range $[1, 2n]$. Each node $S = [a, b]$ has midpoint $\mathsf{mid}_S = \lfloor (a + b)/2 \rfloor$, left child $\mathsf{left}_S = [a, \mathsf{mid}_S]$, and right child $\mathsf{right}_S = [\mathsf{mid}_S + 1, b]$. A node $S = [a, b]$ corresponds to contiguous ranges $r_j, \ldots, r_{j'}$ and $b_k, \ldots, b_{k'}$ of red and blue points. To keep notation lightweight, we also write $r, b \in S$ for $r \in R$ and $b \in B$ to denote that their global indices in the merged order of the points lie in the index range S.

For a pair (r_j, b_k), we call $k - j$ its *shift*. Thus, the shifts of pairs contained in S lie in $[\, k - j',\ k' - j\,]$, which we call the range of *relevant shifts* for S. For a nonempty balanced interval I with leftmost red and blue points r_j and b_k, the *shift of I* is $k - j$; since the order-preserving minimum-cost matching in I matches its red and blue points in order, all its pairs have this shift, see Figure 1.

We augment each node $S = [a, b]$ of the tree with $O(|S|)$ values: For each relevant shift d, we store

- the total cost of all pairs (r_j, b_{j+d}) at shift d with r_j in left_S and b_{j+d} in right_S: $W(\mathsf{left}_S, \mathsf{right}_S, d) \coloneqq \sum_{(r_j, b_{j+d}) \in \mathsf{left}_S \times \mathsf{right}_S} |r_j - b_{j+d}|^p$, and
- the total cost of all pairs (r_j, b_{j+d}) at shift d with r_j in right_S and b_{j+d} in left_S: $W(\mathsf{right}_S, \mathsf{left}_S, d) \coloneqq \sum_{(r_j, b_{j+d}) \in \mathsf{right}_S \times \mathsf{left}_S} |r_j - b_{j+d}|^p$, and
- the total cost of pairs contained in S at shift d: $W(S, d) \coloneqq \sum_{(r_j, b_{j+d}) \in S \times S} |r_j - b_{j+d}|^p$.

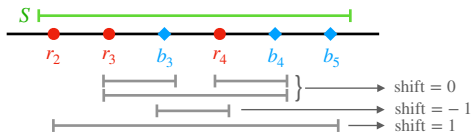


Figure 1: Balanced intervals in S and their shift values. The range of relevant shifts is $[-1, 3]$.

A shift d is relevant for S if there exists a pair (r_i, b_{i+d}) with both endpoints in S. If S contains red points $r_j, \ldots, r_{j'}$ and blue points $b_k, \ldots, b_{k'}$, the relevant shifts are precisely those satisfying $k - j' \le d \le k' - j$. For shifts d that are not relevant for S, the values are defined to be zero, which we store implicitly. As a side note, we remark that for any shift d at least one of $W(\mathsf{left}_S, \mathsf{right}_S, d)$ and $W(\mathsf{right}_S, \mathsf{left}_S, d)$ is zero. As each node S stores $O(|S|)$ values, the total size of the data structure is in $O(n \log n)$.

We first show how to answer the desired queries using this data structure and then we explain how we can actually build the data structure with the correct costs.

Answering Queries. Let $I \subseteq [1, 2n]$ be a nonempty balanced interval, and let d be the shift of its order-preserving minimum-cost matching.[2] For any node $S \in \mathcal{T}$, we define the cost $\mathrm{Cost}(I \cap S, d)$ of all pairs in $S \cap I$ at shift d via the following recursive procedure (refer to Figure 2):

- If $S \subseteq I$, return $W(S, d)$.
- If I intersects both children of S, return

$$W(\mathsf{left}_S, \mathsf{right}_S, d) + W(\mathsf{right}_S, \mathsf{left}_S, d) + \mathrm{Cost}(I \cap \mathsf{left}_S, d) + \mathrm{Cost}(I \cap \mathsf{right}_S, d).$$

- If $I \cap S \subseteq \mathsf{left}_S$, return $\mathrm{Cost}(I \cap \mathsf{left}_S, d)$.
- If $I \cap S \subseteq \mathsf{right}_S$, return $\mathrm{Cost}(I \cap \mathsf{right}_S, d)$.

To answer a query, we compute $\mathrm{Cost}(I \cap S_0, d)$, where $S_0 = [1, 2n]$ is the root of the tree $\mathcal{T}$. We now prove that this recursive definition indeed results in the values that we want to report.

Lemma 2.1. *Let $I \subseteq [1, 2n]$ be a nonempty balanced interval, and let d be the shift of its order-preserving minimum-cost matching. Then $\mathrm{Cost}(I \cap S_0, d) = w^p(I)$, where $S_0 = [1, 2n]$ is the root of $\mathcal{T}$.*

Proof. We show that there is an optimal matching within I such that every pair of shift d contributing to this matching is counted exactly once by the recursion, and that no other pair is counted.

The only point requiring justification is the use of the full terms $W(\mathsf{left}_S, \mathsf{right}_S, d)$ and $W(\mathsf{right}_S, \mathsf{left}_S, d)$ when I intersects both children of a node S. We claim that in this case every shift-d pair crossing between left_S and right_S is fully contained in I. Indeed, suppose $R \cap I = \{r_\alpha, \ldots, r_{\alpha+m-1}\}$ and $B \cap I = \{b_\beta, \ldots, b_{\beta+m-1}\}$ for some α, β, and m. Since I is balanced, there is a minimum-cost matching inside I that matches points in order, so its shift is $d = \beta - \alpha$. Now consider a shift-d pair (r_j, b_{j+d}) with $r_j \in \mathsf{left}_S$ and $b_{j+d} \in \mathsf{right}_S$. If $j < \alpha$, then $j + d < \beta$, so b_{j+d} lies before the first blue point of I and hence before I in the merged order. This contradicts $b_{j+d} \in \mathsf{right}_S$, since I intersects both children of S. Similarly, if $j > \alpha + m - 1$, then r_j lies after the last red point of I and hence after I in the merged order, contradicting $r_j \in \mathsf{left}_S$. Hence $\alpha \le j \le \alpha + m - 1$, and therefore also $\beta \le j + d \le \beta + m - 1$. Thus both points of the pair (r_j, b_{j+d}) lie in I. The case of a pair from right_S to left_S is symmetric.

Now consider the recursion at a node S. If $S \subseteq I$, the procedure returns $W(S, d)$, which is exactly the total cost of all shift-d pairs contained in S. If I intersects both children of S, then every shift-d pair in $I \cap S$ is of exactly one of the following four types: contained in left_S, contained in right_S, crossing from left_S to right_S, or crossing from right_S to left_S. The first two types are handled recursively, and the last two are counted by $W(\mathsf{left}_S, \mathsf{right}_S, d)$ and $W(\mathsf{right}_S, \mathsf{left}_S, d)$. By the arguments above, these cross terms count no pairs outside I. If I intersects only one child, the recursion only descends into that child, so again no pair is lost or counted twice.

[2] As a preliminary step in the query algorithm, d must be determined. This can be done easily by maintaining a binary search tree over each of the red and blue points, then given I, querying said binary search tree to determine the leftmost red point r_j in I and leftmost blue point b_k in I, after which we can simply take $d = k - j$.

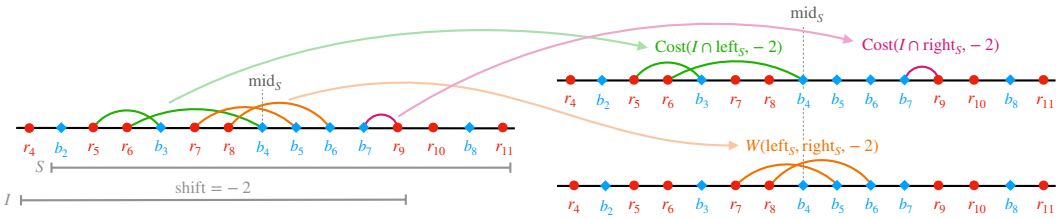


Figure 2: The order-preserving minimum-cost matching of I has shift -2. We decompose the edges of this matching that are contained in $S \cap I$ into three disjoint matchings of cost $\mathrm{Cost}(I \cap \mathrm{left}_S, d)$, $\mathrm{Cost}(I \cap \mathrm{right}_S, d)$, and $W(\mathrm{left}_S, \mathrm{right}_S, d)$. Note that $W(\mathrm{right}_S, \mathrm{left}_S, -2) = 0$.

Thus, every shift-d pair contained in I is counted exactly once, and no shift-d pair outside I is counted. Since d is the shift of an optimal matching inside the balanced interval I, these are precisely the pairs of that matching. Finally, as $I \subseteq S_0 = [1, 2n]$, we have $\mathrm{Cost}(I \cap S_0, d) = w^p(I)$. □

Since the recursion follows the standard range-query traversal in a binary tree over $[1, 2n]$, it has at most two active root-to-leaf search paths. All other visited nodes are fully contained in I and terminate immediately. Hence the recursion visits $O(\log n)$ nodes and spends $O(1)$ time at each node. Thus, the query time is $O(\log n)$.

Constructing the data structure. Let $S = [a, b]$ be a node of the range tree. Suppose the red points in left_S are $r_u, \ldots, r_v$ in the global red order, and the blue points in right_S are $b_s, \ldots, b_t$ in the global blue order. For any shift d, the value $W(\mathrm{left}_S, \mathrm{right}_S, d)$ is the sum of $|b_{j+d} - r_j|^p$ over all j with $j \in [u, v]$ and $j + d \in [s, t]$. Since all red points in left_S lie to the left of all blue points in right_S, we always have $b_{j+d} > r_j$, so $|b_{j+d} - r_j|^p = (b_{j+d} - r_j)^p$. Expanding $(b_{j+d} - r_j)^p$, we obtain $(b_{j+d} - r_j)^p = \sum_{c=0}^{p} \binom{p}{c} (-1)^c r_j^c \, b_{j+d}^{p-c}$, and hence

$$W(\mathrm{left}_S, \mathrm{right}_S, d) = \sum_{\substack{j \in [u,v] \\ j+d \in [s,t]}} (b_{j+d} - r_j)^p = \sum_{c=0}^{p} \binom{p}{c} (-1)^c \left(\sum_{\substack{j \in [u,v] \\ j+d \in [s,t]}} r_j^c \, b_{j+d}^{p-c} \right).$$

Thus, to obtain $W(\mathrm{left}_S, \mathrm{right}_S, d)$, it suffices to compute, for each $c \in \{0, \ldots, p\}$, the sums

$$\sum_{\substack{j \in [u,v],\, k \in [s,t] \\ k - j = d}} r_j^c \, b_k^{p-c}$$

for all relevant shifts d. These sums are entries of a convolution: for $A = (r_u^c, r_{u+1}^c, \ldots, r_v^c)$ and $B = (b_t^{p-c}, b_{t-1}^{p-c}, \ldots, b_s^{p-c})$, the sum above is the entry $(A * B)[t - u - d]$, where $*$ denotes the discrete convolution. Hence, for all relevant shifts d these values can be computed simultaneously by one convolution in $O((b-a)\log(b-a))$ time via FFT [14, 20]. Repeating this for every $c = 0, \ldots, p$, the values $W(\mathrm{left}_S, \mathrm{right}_S, d)$ for all relevant shifts are computed in $O(p(b-a)\log(b-a))$ time. For more details, refer to Appendix B. We compute $W(\mathrm{right}_S, \mathrm{left}_S, d)$ symmetrically. As the sum of the lengths of all node ranges in $\mathcal{T}$ is $O(n \log n)$, the total construction time is $O(pn \log^2 n)$.

Finally, the values $W(S, d)$ are computed bottom-up using the recurrence

$$W(S, d) = W(\mathrm{left}_S, d) + W(\mathrm{left}_S, \mathrm{right}_S, d) + W(\mathrm{right}_S, \mathrm{left}_S, d) + W(\mathrm{right}_S, d),$$

which requires $O(|S|)$ time per node S and $O(n \log n)$ time overall, implying Theorem 1.2.

3 A Near-Linear Time Algorithm for k-Partial p-Wasserstein Distance

Using the data structure introduced in Section 2, we now present an $O(pn \log^2 n)$-time algorithm to compute minimum-cost k-matchings for all $k = 1, \ldots, n$ on the line. We note that this general algorithmic framework follows the classical Hungarian algorithm applied to points on a line. A similar approach has previously been used by, e.g., [13]. We begin by reviewing some of the basic properties of matchings needed to describe our algorithm.

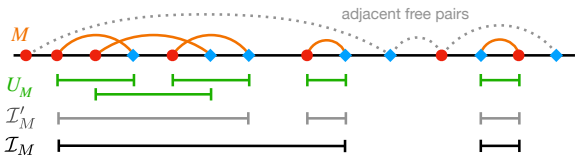


Figure 3: Construction of $\mathcal{I}_M$ and adjacent free pairs.

A vertex $v \in R \cup B$ is said to be *free* with respect to a matching M if no edge of M is incident to v. Let R_F and B_F denote the sets of free red and blue points, respectively. A path P in the complete bipartite graph on $R \cup B$ is called *alternating* (with respect to M) if its edges alternate between those in M and those not in M. An alternating path is *augmenting* if its two endpoints are free vertices, one in R_F and the other in B_F. Flipping a matching along such a path removes the matching edges in P and adds the non-matching edges of P; when P is augmenting, this operation is called an *augmentation*, and it increases the size of the matching by one.

For any alternating path or cycle P with respect to M, we define its *net cost* to be

$$\Phi(P) \;=\; \sum_{(a,b)\in P\setminus M} |a-b|^p \;-\; \sum_{(a,b)\in P\cap M} |a-b|^p.$$

The net cost $\Phi(P)$ is exactly the change in the total cost of the matching when M is replaced by the symmetric difference $M \oplus P$. The next lemma shows that augmenting a minimum-cost k-matching along a minimum net-cost augmenting path gives a minimum-cost $(k+1)$-matching.

Lemma 3.1. *Let M_k be a minimum-cost k-matching. Among all minimum-cost $(k+1)$-matchings, let M_{k+1} be one that maximizes the number of edges it shares with M_k. Then the symmetric difference $M_k \oplus M_{k+1}$ consists of a single augmenting path with respect to M_k, and this path has minimum net cost among all augmenting paths for M_k.*

We defer the proof to Appendix A.1. The classical Hungarian algorithm begins with the empty matching and, using a primal-dual framework, repeatedly identifies a minimum net-cost augmenting path and augments the matching along it. In this way, it successively constructs minimum-cost k-matchings for all $k = 1, \ldots, n$.

Next, we introduce a compact representation of minimum-cost partial matchings on the line. This reduces computing the net-cost of an augmenting path to two queries to our data structure.

Interval Representation. Consider a minimum-cost k-matching M between the blue points B and the red points R. Let $B_F \subseteq B$ and $R_F \subseteq R$ denote the free (unmatched) points with respect to M. Each matching edge $(b, r) \in M$ naturally defines the interval $I_{b,r} := [\min\{b, r\},\ \max\{b, r\}]$. Edges in the minimum-cost matching for points on the line satisfy the following property.

Observation 3.2. *No free point lies in the interior of any interval $I_{b,r}$ of an edge $(b, r) \in M$.*

If a free point $p \in (b, r)$ existed, replacing (b, r) by (p, r) or (b, p) would strictly shorten the edge and reduce the total cost of M, contradicting its minimality.

Given a set of disjoint intervals $\mathcal{I}$, we say that two intervals $I, J \in \mathcal{I}$ are *adjacent* if there is no other interval of $\mathcal{I}$ lying strictly between them in the ordering of the real line. Let $U_M := \bigcup_{(b,r)\in M}\{I_{b,r}\}$ be the set of all edge-intervals, see Figure 3 for an example. We describe the *interval representation* $\mathcal{I}_M$ of M in two steps. First, let $\mathcal{I}'_M$ be the set of disjoint intervals, one spanning each connected component of $\bigcup U_M$. We then obtain $\mathcal{I}_M$ by repeatedly merging adjacent intervals of $\mathcal{I}'_M$ as follows: if $I, J \in \mathcal{I}'_M$ are adjacent, with $I = [x_1, x_2]$ and $J = [x_3, x_4]$ and $x_2 \le x_3$, and the interval $[x_1, x_4]$ contains no free points, we replace I and J by the single interval $[x_1, x_4]$. We continue this process until no further merge is possible. We refer to any pair of free blue and red points $(b, r) \in B_F \times R_F$ as an *adjacent free pair* if no other free point is contained in $I_{b,r}$.

For any minimum-cost matching M, the set of intervals $\mathcal{I}_M$ satisfies the following properties:

I *Balancedness:* Every interval in $\mathcal{I}_M$ contains the same number of blue and red points. Indeed, the intervals in $\mathcal{I}'_M$, the connected components of U_M, are balanced by construction, and merging adjacent disjoint balanced intervals with no free point between them preserves balancedness.

II *Characterization of matched points:* A point x is matched by M if and only if x lies in some interval of $\mathcal{I}_M$. All matched points lie in U_M, and by Observation 3.2 no free point lies in the interior of any edge-interval; the merging step also does not introduce free points.

III *Monotone structure within intervals:* Inside each interval $I \in \mathcal{I}_M$, the matching M pairs the i-th smallest blue point with the i-th smallest red point. All matching edges lie within intervals of $\mathcal{I}_M$. If this monotone pairing failed inside some I, replacing the edges of M in I by the order-preserving minimum-cost matching on the points of I would yield a cheaper matching, contradicting the optimality of M.

IV *At most one interval between adjacent free points:* For every adjacent free pair (b, r), the interval $I_{b,r}$ contains at most one interval of $\mathcal{I}_M$. Otherwise, since $I_{b,r}$ contains no free points, the contained intervals would have been merged.

These properties make $\mathcal{I}_M$ a compact and convenient representation of the matching M. For any balanced interval I, let M_I denote the order-preserving minimum-cost matching on the points in I, and define $w^p(I) := \sum_{(b',r') \in M_I} |b' - r'|^p$ to be the matching cost. Using this notation, the net-cost of an augmentation step can be expressed entirely in terms of interval costs.

Lemma 3.3. *Let M be a minimum-cost k-matching, and let P be a minimum net-cost augmenting path with respect to M. Then P has endpoints at an adjacent free pair (b, r), and its net-cost is*

$$\Phi(P) = \Phi(b, r) := \begin{cases} |b - r|^p, & \text{if } I_{b,r} \text{ contains no interval of } \mathcal{I}_M, \\ w^p(I_{b,r}) - w^p(I), & \text{if } I \in \mathcal{I}_M \text{ lies in } I_{b,r}. \end{cases}$$

See Appendix A.2 for the proof. We now present our algorithm, which uses this characterization to identify a minimum net-cost augmenting path using only two interval-cost queries.

The Algorithm. Our algorithm incrementally constructs minimum-cost matchings of sizes $1, 2, \ldots, n$. At all times it maintains the current matching M through its interval representation $\mathcal{I}_M$, the free sets $B_F \subseteq B$ and $R_F \subseteq R$, and the set of adjacent free pairs together with their net-costs.

Initially, we set $M = \emptyset$, $\mathcal{I}_M = \emptyset$, $B_F = B$, and $R_F = R$. Each consecutive red-blue pair is adjacent, and since no intervals are present, each adjacent pair (b, r) receives the initial weight $\Phi(b, r) = |b - r|^p$. Each adjacent pair is added to a min-heap $\mathcal{Q}$ with its weight as the key value. We build the data structure $\mathcal{D}$ from Section 2 on the point sets R and B.

The algorithm now iteratively transforms a minimum-cost k-matching into a minimum-cost $(k+1)$-matching via the following steps.

1. **Select a minimum-cost augmentation.** From the min-heap $\mathcal{Q}$, choose the adjacent free pair (b, r) minimizing $\Phi(b, r)$.
2. **Update the interval representation and free sets.** By Property IV of $\mathcal{I}_M$, at most one interval of $\mathcal{I}_M$ lies inside $I_{b,r}$; remove it if present, and insert $I_{b,r}$. Merge adjacent intervals in $\mathcal{I}_M$. Finally, mark b and r as matched and remove them from B_F and R_F.
3. **Update adjacency.** Remove all adjacent pairs involving b or r. At most one new adjacent free pair (b', r') is created.
4. **Update net-costs.** For any newly created adjacent free pair (b', r'), we query $\mathcal{D}$ to compute $\Phi(b', r') = w^p(I_{b',r'}) - w^p(I)$, where I is the unique interval of $\mathcal{I}_M$ that lies inside $I_{b',r'}$.

Repeating this process for n iterations yields interval representations and costs of the minimum-cost matchings of all sizes $1, \ldots, n$.

Running time. Building the data structure $\mathcal{D}$ takes $O(pn \log^2 n)$ time. The remaining initialization, including the construction of the initial priority queue of adjacent free pairs, takes $O(n \log n)$ time. Each iteration performs a constant number of priority-queue operations and interval insertions and deletions, as well as a constant number of interval-cost queries to $\mathcal{D}$, each taking time $O(\log n)$. The n iterations therefore take $O(n \log n)$ time. Thus, the construction of $\mathcal{D}$ dominates the running time. Overall, this implies Theorem 1.1.

4 Experimental Evaluation

To validate the efficiency and practicality of our approach, we compare our algorithm against the algorithm given in [13], with $p = 2$.

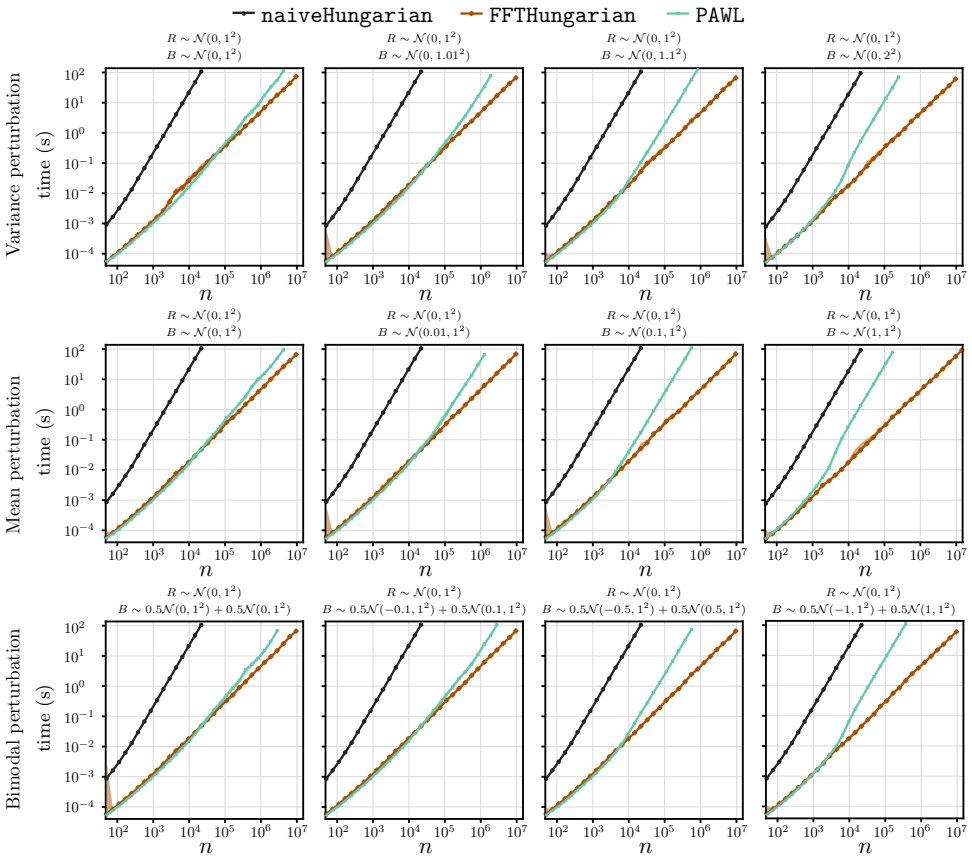


Figure 4: Running-time comparison of our naive (black) and FFT-based (orange) implementations with the implementation from Chapel and Tavenard [13] (green) on synthetic data sets. Curves show median running times; shaded regions show the 10th–90th percentile range across runs. Rows correspond to distribution families, with increasing distributional difference from left to right.

Implementation Details. We consider three implementations. The first is a naive Python baseline, `naiveHungarian`, which computes the cost of an interval with M points in $O(M)$ time. The second is the implementation `PAWL`[3] of [13], which follows the same principle but is substantially optimized. Our implementation `FFTHungarian`[4] is written in `C++` and follows the description in Section 3, with one practical modification: for intervals containing fewer than 2048 points, we compute the cost naively; otherwise, we query our data structure in $O(\log n)$ time. The threshold was chosen to approximately balance the two running times.

Experimental Setup. We evaluate the algorithms on synthetic instances that are obtained by drawing point sets of a certain size from given continuous distributions. The distributions from which the two input point sets are drawn are either identical or moderately different. We use samples of varying sizes from the following distributions: (i) two normal distributions $\mathcal{N}(0, 1^2)$ and $\mathcal{N}(0, \sigma^2)$ with different standard deviations $\sigma \in \{1, 1.01, 1.1, 2\}$, (ii) two normal distributions $\mathcal{N}(0, 1^2)$ and $\mathcal{N}(\mu, 1^2)$ with different means $\mu \in \{0, 0.01, 0.1, 1\}$, and (iii) one unimodal distribution $\mathcal{N}(0, 1^2)$ and one bimodal distribution $\frac{1}{2}\mathcal{N}(-\mu, 1^2) + \frac{1}{2}\mathcal{N}(\mu, 1^2)$ with $\mu \in \{0, 0.1, 0.5, 1\}$. In all cases the distributions have substantial overlap, which is favorable to the augmenting-path-based baselines. Each instance–algorithm pair was run 3–20 times, depending on its running time, with a 60-second cutoff; we report the 10th percentile, median, and 90th percentile running times. All experiments were conducted on an Apple MacBook Air with an M4 processor and 24 GB of RAM.

Evaluation. The running times are shown in Figure 4. We observe that `FFTHungarian` is significantly faster than `PAWL` whenever $n \geq 10\,000$ and the two distributions are not identical. We can also clearly

[3]This implementation is written in Python, but uses `numba/njit` for performance-critical computations.
[4]The code is available at `https://github.com/JacobusTheSecond/wasserstein`.

see the difference in asymptotic behavior of the two implementations: even when the standard deviation or mean differs by only 10%, the running time of `PAWL` approaches its quadratic worst-case behavior, while `FFTHungarian` continues to scale near-linearly. Across all tested regimes, `FFTHungarian` is never substantially slower than `PAWL` and becomes increasingly faster as the input size grows. Overall, for large samples from even slightly different distributions, `FFTHungarian` is the only implementation that remains practically viable. This regime is central in applications, where the goal is to compare samples from unknown distributions rather than from exactly identical ones.

5 Lower Bounds

Proof of Theorem 1.3. Assume there is an algorithm computing $(\min, +)$-convolution of arrays of size n and polynomially bounded entries in time $\widetilde{O}(n^{2-\varepsilon})$. It is easy to see that we can compute $(\max, -)$-convolution in the same time. For the partial ∞-Wasserstein problem, we adapt the data structure of Section 2 by replacing sums of edge costs with maxima. In particular, we set

$$W(S, d) = \max\big\{W(\mathrm{left}_S, d), W(\mathrm{left}_S, \mathrm{right}_S, d), W(\mathrm{right}_S, \mathrm{left}_S, d), W(\mathrm{right}_S, d)\big\}$$

for every node S, defining maxima over empty collections of edges to be zero. By reversing the order of the points in left_S, we can compute $W(\mathrm{left}_S, \mathrm{right}_S, d)$ and $W(\mathrm{right}_S, \mathrm{left}_S, d)$ using one $(\max, -)$-convolution each, so we can compute all k-partial ∞-Wasserstein distances in time $\widetilde{O}(n^{2-\varepsilon})$. To compute the OT-profile, use the interval representation of Section 3 with priorities $q(b, r) = w_\infty(I_{b,r})$, implicitly re-matching each selected interval in order. The correctness of this choice follows from the next claim.

Claim 5.1. *[Proof in Appendix A.3] For the maintained optimal k-matching of cost C_k, $k < n$, let $\lambda_k := \min\{q(b, r) : (b, r)$ adjacent free$\}$. Then $C_{k+1} = \max\{C_k, \lambda_k\}$.*

Each iteration creates at most one new adjacent free pair, while surviving priorities remain unchanged. Thus the $O(n)$ queries and heap operations take $O(n \log n)$ time, giving $\widetilde{O}(n^{2-\varepsilon})$ time.

Conversely, assume that there is an algorithm computing all k-partial ∞-Wasserstein matching costs between two sets B and R of polynomially bounded integer points in time $\widetilde{O}(n^{2-\varepsilon})$. As an intermediate step, we reduce computing the $(\min, +)$-convolution on arbitrary instances to computing the $(\max, -)$-convolution on structured instances.

Lemma 5.2. *[Proof in Appendix A.4] Let $\varepsilon > 0$. If there is an algorithm taking an increasing integer array $(X)_{i=0}^{n-1}$ and a decreasing integer array $(Y)_{i=0}^{n-1}$ with polynomially bounded entries and $\max_j Y[j] < \min_i X[i]$ as input and outputting their $(\max, -)$-convolution in time $\widetilde{O}(n^{2-\varepsilon})$, then there is an algorithm computing the $(\min, +)$-convolution of arrays of size n and polynomially bounded entries in time $\widetilde{O}(n^{2-\varepsilon})$.*

Let X and Y satisfy the input conditions of Lemma 5.2 and let Z be their $(\max, -)$-convolution. By Lemma 5.2, it suffices to compute their $(\max, -)$-convolution in time $\widetilde{O}(n^{2-\varepsilon})$. Let $k \in [n]$ and consider the k-partial ∞-Wasserstein distance between X and Y. Since $\max_j Y[j] \le \min_i X[i]$ and as X is increasing and Y is decreasing, an optimal k-matching matches the k smallest values of X to the k largest values of Y in increasing order. Thus, the ∞-Wasserstein cost of this k-matching is $\max_{i+j=k-1} |X[i] - Y[j]| = Z[k-1]$, and thus the $(\max, -)$-convolution of X and Y can be computed in time $\widetilde{O}(n^{2-\varepsilon})$. □

Acknowledgments

This research was initiated during the Workshop "Massive Data Models and Computational Geometry II" held at the University of Bonn in September 2025.

Funding in direct support of this work:

- Jacobus Conradi is funded by the Carlsberg Foundation, grant CF24-1929.
- Mónika Csikós was supported by the program "Investissement d'Avenir" launched by the French Government and implemented by ANR, with the reference ANR-18-IdEx-0001 as part of its program Emergence for the project VC-GRAPHES.
- Niko Hastrich received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No. 850979).
- Danny Mittal was supported by a travel grant from NSF 2115677, and is also partially supported by DARPA expMath, ONR MURI 2024 award on Algorithms, Learning, and Game Theory, Army-Research Laboratory (ARL) grant W911NF2410052, NSF AF:Small grants 2218678, 2114269, 2347322.
- André Nusser is supported by the France 2030 investment plan managed by the ANR as part of the Initiative of Excellence of Université Côte d'Azur with reference number ANR-15-IDEX-01.
- Work by Sharath Raghvendra is supported by an NSF Grant CCF-2514753.

References

[1] Amir Abboud, Nick Fischer, Zander Kelley, Shachar Lovett, and Raghu Meka. New graph decompositions and combinatorial boolean matrix multiplication algorithms. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*, pages 935–943, 2024.

[2] Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemysław Uznański, and Daniel Wolleb-Graf. Faster algorithms for all-pairs bounded min-cuts. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:15, 2019.

[3] Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal of Computing*, 29(3), dec 1999.

[4] Shyan Akmal and Ce Jin. An efficient algorithm for all-pairs bounded edge connectivity. *Algorithmica*, 86(5):1623–1656, 2024.

[5] Josh Alman and Virginia Vassilevska Williams. Truly subquadratic 3SUM and truly subcubic APSP via triangles in sparse lopsided graphs, 2026. arXiv:2610.06783.

[6] Yikun Bai, Bernhard Schmitzer, Matthew Thorpe, and Soheil Kolouri. Sliced optimal partial transport. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 13681–13690, 2023.

[7] Aaron Bernstein, Sayan Bhattacharya, Nick Fischer, Peter Kiss, and Thatchaphol Saranurak. Separations between oblivious and adaptive adversaries for natural dynamic graph problems. In *Proceedings of the 37th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 5669–5687, 2026.

[8] 3SUM – Big O Speedrun. `https://bigospeedrun.com/3sum`, 2026. Accessed 2026-11-06.

[9] Itai Boneh, Dvir Fried, Shay Golan, Matan Kraus, and Ely Porat. Hamming distance oracles. In *Proceedings of the 37th Annual Symposium on Combinatorial Pattern Matching, CPM*, volume 369 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:12, 2026.

[10] Sergio Cabello, Siu-Wing Cheng, Otfried Cheong, and Christian Knauer. Geometric matching and bottleneck problems. In *Proceedings of the 40th International Symposium on Computational Geometry, SoCG*, volume 293 of *LIPIcs*, pages 31:1–31:15, 2024.

[11] Timothy M. Chan, Reina Itakura, Zhiyuan Ma, and Yuancheng Yu. 3SUM made simpler (without matrix multiplication), k-SUM hypothesis refuted for odd k, and real 3SUM/APSP derandomized, 2026. arXiv:2610.42012.

[12] Laetitia Chapel, Mokhtar Z Alaya, and Gilles Gasso. Partial optimal tranport with applications on positive-unlabeled learning. *Advances in Neural Information Processing Systems, NeurIPS*, 33:2903–2913, 2020.

[13] Laetitia Chapel and Romain Tavenard. One for all and all for one: Efficient computation of partial Wasserstein distances on the line. In *Proceedings of the 13th International Conference on Learning Representations, ICLR*, 2025.

[14] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[15] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. *ACM Trans. Algorithms*, 15(1):14:1–14:25, 2019.

[16] Pawel Gawrychowski and Przemyslaw Uznanski. Towards unified approximate pattern matching for Hamming and L_1 distance. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming ICALP*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:13, 2018.

[17] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[18] Debarghya Mukherjee, Aritra Guha, Justin M Solomon, Yuekai Sun, and Mikhail Yurochkin. Outlier-robust optimal transport. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 7850–7860. PMLR, 2021.

[19] Khai Nguyen. An introduction to sliced optimal transport: foundations, advances, extensions, and applications. *Foundations and Trends in Computer Graphics and Vision*, 17(3-4):171–391, 11 2025.

[20] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[21] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11(5-6):355–607, 2019.

[22] Abhijeet Phatak, Sharath Raghvendra, Chittaranjan Tripathy, and Kaiyi Zhang. Computing all optimal partial transports. In *Proceedings of the 11th International Conference on Learning Representations, ICLR*, 2023.

[23] 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, SSVM, Revised Selected Papers*, volume 6667, pages 435–446, 2011.

[24] Sharath Raghvendra, Pouyan Shirzadian, and Kaiyi Zhang. A new robust partial p-Wasserstein-based metric for comparing distributions. In *Proceedings of the 41st International Conference on Machine Learning, ICML*, 2024.

[25] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011.

[26] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International J. of Comput. Vision*, 40(2):99–121, 2000.

[27] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications. Springer International Publishing, 2015.

[28] Pravin M Vaidya. Geometry helps in matching. *SIAM Journal of Computing*, 18(6):1201–1225, 1989.

[29] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

A Omitted Proofs

A.1 Proof of lemma 3.1

Proof. The symmetric difference of two matchings is a vertex-disjoint union of alternating cycles and paths. Observe that $M_k \oplus M_{k+1}$ cannot contain a cycle or non-augmenting path, otherwise we would either contradict the optimality of M_k or M_{k+1} or contradict the choice of M_{k+1}. If $M_k \oplus M_{k+1}$ contains at least 2 augmenting paths, it must contain a path P which is augmenting for M_k and a path P' which is augmenting for M_{k+1} such that $\text{cost}(M_{k+1}) = \text{cost}(M_{k+1} \oplus P \oplus P')$, otherwise either M_k or M_{k+1} was not optimal. However $|(M_{k+1} \oplus P \oplus P') \cap M_k| > |M_{k+1} \cap M_k|$ which would contradict the choice of M_{k+1}. □

A.2 Proof of lemma 3.3

Proof. Let M be a minimum-cost k-matching and let P be a minimum net-cost augmenting path with respect to M. Let $b \in B_F$ and $r \in R_F$ be its endpoints, and write $I_{b,r} = [\min\{b, r\}, \max\{b, r\}]$.

First, we show that b and r form an adjacent pair of free points. Consider the matching $M' = M \oplus P$. By Lemma 3.1, both M and M' are minimum-cost matchings of sizes k and $k+1$, respectively, and every edge of P belongs either to M or to M'. By Observation 3.2, no edge of a minimum-cost matching can contain a free point in its interior. If there were a free point strictly between b and r, then some edge of P would have to cross over that free point, contradicting Observation 1. Hence there is no free point in the interior of $I_{b,r}$, and (b, r) is an adjacent free pair by definition.

Next, we argue that all edges of P are contained inside the interval $I_{b,r}$. Suppose, for a contradiction, that some edge of P crosses either b or r. By Observation 1, this edge cannot be a matching edge of M. Without loss of generality, suppose that there is an edge that crosses r, and let (b', r') with $b' < r < r'$ be the first (starting from b) such edge of P. Along P, the subpath from r' to r is an alternating path of non-negative net-cost (otherwise, alternating along this subpath alone would reduce the cost of M, contradicting its optimality). Because r lies strictly between b' and r' on the line, we have $\|b' - r\|^p < \|b' - r'\|^p$. Replacing the subpath from b' to r in P by the edge (b', r) thus yields another augmenting path from b to r with net-cost strictly smaller than that of P, a contradiction.

Since P is contained in $I_{b,r}$, the only edges of M affected by the augmentation lie inside this interval. By the interval representation (Property IV), at most one interval $I \in \mathcal{I}_M$ is contained in $I_{b,r}$. If no such interval exists, then $I_{b,r}$ contains only the two free endpoints b and r, so P consists of the single edge (b, r) and $\Phi(P) = \|b - r\|^p$. If such an interval I exists, then inside I the matching M coincides with the optimal monotone matching of cost $w^p(I)$ (Property III), while after augmenting along P the matching inside $I_{b,r}$ becomes the optimal monotone matching on that larger interval, with cost $w^p(I_{b,r})$. Thus $\Phi(P) = w^p(I_{b,r}) - w^p(I)$. □

A.3 Proof of claim 5.1

Proof. Let M be the maintained k-matching of cost C_k. During the construction, we ensure that no matching edge spans a free point and that, for every adjacent free pair (b, r), all interior points of $I_{b,r}$ are matched inside that interval. Replacing the edges of M within $I_{b,r}$ by the order-preserving matching of $I_{b,r}$ adds one matched pair and preserves the above stated properties. Choosing a pair attaining λ_k therefore gives

$$C_{k+1} \leq \max\{C_k, \lambda_k\}.$$

For the reverse inequality, let N be an optimal $(k+1)$-matching and set $T := C_{k+1} \geq C_k$. The symmetric difference $M \oplus N$ contains an augmenting path whose edges all have length at most T. Among all augmenting paths of M whose edges all have length at most T, choose P minimizing its endpoint span in the merged order and, subject to this, its number of edges. Let b and r be its free blue and red endpoints, respectively.

Any free point between b and r must be crossed by a non-matching edge. Shortcutting that edge to the free point and retaining the appropriate prefix or suffix preserves the threshold and reduces the endpoint span. Thus (b, r) is an adjacent free pair.

We next show that P lies inside $I_{b,r}$. Without loss of generality, assume $b < r$ and traverse P from b to r. If P visits a point to the right of r, its first entering edge is an unmatched edge (b', r') with $b' < r < r'$. Replacing the suffix from b' to r by (b', r) preserves alternation and the threshold, since

$$|b' - r| \le |b' - r'| \le T,$$

while reducing the number of edges, contradicting the choice of P. Similarly, P cannot contain any point to the left of b.

Augmenting M along P completely matches $I_{b,r}$ using edges of length at most T, which therefore gives

$$\lambda_k \le w_\infty(I_{b,r}) \le T = C_{k+1}.$$

Together with $C_k \le C_{k+1}$, this proves the claim. □

A.4 Proof of lemma 5.2

Proof. For some $c \in \mathbb{N}$, let A and B be n-element integer sequences with elements in $[-n^c, n^c]$. Define their $(\min, +)$-convolution D by

$$D[k] \coloneqq \min_{\substack{0 \le i,j < n \\ i+j=k}} \big(A[i] + B[j]\big), \qquad 0 \le k < n.$$

Set $M \coloneqq n^c$ and

$$X[i] \coloneqq 3iM - A[i] \quad \text{and} \quad Y[j] \coloneqq -3(j+1)M + B[j].$$

Note that $X[i] \in [3iM - M, 3iM + M]$. Since $3iM + M < 3(i+1)M - M$, X is strictly increasing. Similarly, Y is strictly decreasing. Moreover,

$$\max_j Y[j] \le -2M < -M \le \min_i X[i],$$

so the separation condition holds. Let Z be the $(\max, -)$-convolution of X and Y. For all $k \in \{0, \dots, n-1\}$, we have

$$\begin{aligned}
Z[k] &= \max_{i+j=k} \big(X[i] - Y[j]\big) \\
&= \max_{i+j=k} \big(3iM - A[i] - (-3(j+1)M + B[j])\big) \\
&= 3(k+1)M + \max_{i+j=k} \big(-A[i] - B[j]\big) \\
&= 3(k+1)M - \min_{i+j=k} \big(A[i] + B[j]\big) \\
&= 3(k+1)M - D[k].
\end{aligned}$$

Therefore, given Z, we can recover D in linear time using $D[k] = 3(k+1)M - Z[k]$. Notice that all elements of X and Y are contained in $[-6n^{c+1}, 6n^{c+1}]$. Thus, any algorithm computing Z in time $\widetilde{O}(n^{2-\varepsilon})$ yields an algorithm computing D in time $\widetilde{O}(n^{2-\varepsilon})$. □

B Fast Fourier Transform

In this section, we elaborate on the specific way in which we apply the fast Fourier transform to compute $W(\mathrm{left}_S, \mathrm{right}_S, d)$.

Let $S = [a, b]$ be a node of the range tree. Suppose the red points in left_S are $r_u, \dots, r_v$ in the global red order, and the blue points in right_S are $b_s, \dots, b_t$ in the global blue order. For any shift d, the value $W(\mathrm{left}_S, \mathrm{right}_S, d)$ is the sum of $|b_{j+d} - r_j|^p$ over all j with $j \in [u, v]$ and $j + d \in [s, t]$. Since all red points in left_S lie to the left of all blue points in right_S, we always have $b_{j+d} > r_j$, so $|b_{j+d} - r_j|^p = (b_{j+d} - r_j)^p$.

We compute these values simultaneously for all relevant shifts d by expanding each term using the binomial identity

$$(b_{j+d} - r_j)^p = \sum_{c=0}^{p} \binom{p}{c} (-1)^c r_j^c \, b_{j+d}^{p-c}.$$

Thus, to obtain $W(\text{left}_S, \text{right}_S, d)$, it suffices to compute, for each $c \in \{0, \ldots, p\}$, the sum of $r_j^c \, b_k^{p-c}$ over all pairs (j, k) with $j \in [u, v]$, $k \in [s, t]$, and $k - j = d$.

To accomplish this efficiently, we construct polynomials encoding the sequences of red and blue values. For each c, define

$$P_c(x) = r_u^c x^{\,v-u} + r_{u+1}^c x^{\,v-u-1} + \cdots + r_v^c,$$

so that the ith red point from the left contributes to a monomial determined by its offset from v. Similarly define

$$Q_c(x) = b_s^{p-c} + b_{s+1}^{p-c} x + \cdots + b_t^{p-c} x^{\,t-s},$$

so that the ith blue point from the left contributes to degree i. The product $P_c(x)Q_c(x)$ can be computed in $O((b-a)\log(b-a))$ time using FFT. The coefficient of x^ℓ in this product equals

$$\sum_{\substack{j \in [u,v],\, k \in [s,t] \\ (v-j)+(k-s)=\ell}} r_j^c \, b_k^{p-c}.$$

The condition $(v - j) + (k - s) = \ell$ is equivalent to $k - j = \ell + s - v$. Thus the coefficient of x^ℓ aggregates exactly the contributions of all pairs (r_j, b_k) in $\text{left}_S \times \text{right}_S$ whose shift is $d = \ell + s - v$. Summing over all c with the appropriate binomial coefficients, we define

$$H(x) = \sum_{c=0}^{p} \binom{p}{c} (-1)^c \, P_c(x) Q_c(x).$$

By construction, the coefficient of x^ℓ in $H(x)$ is precisely $W(\text{left}_S, \text{right}_S, d)$ for $d = \ell + s - v$. Thus, to store all cross-term values for node S, we store the coefficients of $H(x)$ indexed by their corresponding shifts d.

The total time to compute $W(\text{left}_S, \text{right}_S, d)$ for all relevant d is $O((b-a)\log(b-a))$.

C Omitted Lower Bounds

C.1 Two Intervals Matching Problem

We show two more lower bounds. Although the data structure in our balanced-interval setting admits a near-linear-time solution for any fixed p, a slightly more general version of the data structure might not exist with similar performance bounds. In this more general formulation, the data structure is given $p \in \mathbb{N}$ and point sets $B, R \subset \mathbb{R}$, and it must answer queries of the following form: given two intervals $I_B, I_R \subset \mathbb{R}$ with $|B \cap I_B| = |R \cap I_R|$, return the p-Wasserstein distance between $B \cap I_B$ and $R \cap I_R$. We refer to this as the *Two Intervals Matching Problem*.

If only combinatorial[5] methods are allowed, we present a lower bound based on the Combinatorial Matrix Multiplication Conjecture. This conjecture states that no truly subcubic *combinatorial* matrix multiplication algorithm exists.

Theorem C.1. *Assuming the Combinatorial Matrix Multiplication Conjecture holds, for any $p > 1$ and $\varepsilon > 0$, the Two Intervals Matching Problem admits no combinatorial data structure with preprocessing time P and query time Q satisfying $P \cdot Q = O\big((|B| \cdot |R|)^{1-\varepsilon}\big)$.*

When non-combinatorial techniques are permitted, we can still rule out almost-linear preprocessing time with $n^{o(1)}$ query time unless boolean matrix multiplication can be performed in almost-quadratic time (i.e., $\omega = 2$), under the standard assumption that boolean matrix multiplication cannot be performed faster than real matrix multiplication (boolean matrix multiplication conjecture). More precisely, we obtain the following lower bound:

Theorem C.2. *Let $n \coloneqq |B| = |R|$ and $\varepsilon > 0$. Unless the Boolean Matrix Multiplication Hypothesis fails, no data structure for the Two Intervals Matching Problem with preprocessing time P and query time Q satisfies $P + nQ \in \Omega(n^{\omega/2-\varepsilon})$, where ω denotes the Matrix Multiplication Exponent.*

[5] Roughly speaking, these are algorithms that avoid algebraic techniques such as fast matrix multiplication or FFT-based methods.

Our lower bound is based on the following conjectures that have been stated many times (see, e.g., [1, 16, 25] and [2, 4, 7]):

Conjecture C.3 (Combinatorial Matrix Multiplication)**.** For any $\alpha, \beta, \gamma, \varepsilon > 0$, there is no combinatorial algorithm for multiplying an $n^\alpha \times n^\beta$ matrix with an $n^\beta \times n^\gamma$ matrix in time $O(n^{\alpha+\beta+\gamma-\varepsilon})$.

Conjecture C.4 (Boolean Matrix Multiplication)**.** For any $\varepsilon > 0$, there is no algorithm multiplying two boolean $n \times n$ matrices in time $O(n^{\omega-\epsilon})$.

As has been noted in [16], there is no *precise* definition of what combinatorial means. However, the algebraic tricks that are used in FFT as well as any currently known fast matrix multiplication algorithm are clearly non-combinatorial by the above conjecture.

Combinatorial Matrix Multiplication reduces to the Hamming Distance Oracle Problem [9], which is defined as follows. Given two binary strings S, T, preprocess them to support queries for the Hamming distance between a length-ℓ substring of S and a length-ℓ substring of T.

Theorem C.5 (by [9, Theorem 3])**.** *Unless the Combinatorial Matrix Multiplication Conjecture fails, there is no combinatorial data structure for the Hamming Distance Oracle Problem with query time Q and preprocessing time P such that $P \cdot Q \in O((|S| \cdot |T|)^{1-\varepsilon})$.*

Recall the *Two Intervals Matching Problem*. The data structure is given $p \in \mathbb{N}$ and two point sets $B, R \subset \mathbb{R}$ to preprocess. Now it should answer the following queries: Given two contiguous intervals $I_B, I_R \subset \mathbb{R}$ such that $|B \cap I_B| = |R \cap I_R|$, return the smallest matching cost between $B \cap I_B$ and $R \cap I_R$. We obtain the following lower bound:

Theorem C.1. *Assuming the Combinatorial Matrix Multiplication Conjecture holds, for any $p > 1$ and $\varepsilon > 0$, the Two Intervals Matching Problem admits no combinatorial data structure with preprocessing time P and query time Q satisfying $P \cdot Q = O\big((|B| \cdot |R|)^{1-\varepsilon}\big)$.*

Proof. In the following we use the notation $S + x \coloneqq \{s + x \mid s \in S\}$, where $S \subset \mathbb{R}$ and $x \in \mathbb{R}$. We first define the gadgets that we use to encode 0s and 1s:

$$G(0) = \{1, 4\}, \quad G(1) = \{2, 3\}.$$

To understand why we choose these gadgets, consider their p-Wasserstein distances when placed some distance M apart:

$$Y(M) \coloneqq w(G(0), G(0) + M) = w(G(1), G(1) + M) = 2M^p$$
$$Z(M) \coloneqq w(G(0), G(1) + M) = w(G(1), G(0) + M) = (M + 1)^p + (M - 1)^p > 2M^p.$$

Note that gadgets of the same type result in a smaller distance than matching gadgets of a different type. Hence, this can encode the Hamming Distance at the bit level.

Let $S \in \{0, 1\}^n$ and $T \in \{0, 1\}^m$ be the two input strings of the Hamming Distance Oracle Problem. Now we define the point sets $B, R \subset \mathbb{N}$ from the Two Intervals Matching Problem that we use to encode the input of the Hamming Distance Oracle Problem.

$$B \coloneqq \{G(S[i]) + 4i \mid i \in \{1, \ldots, n\}\}, \quad R \coloneqq \{G(T[i]) + 4i \mid i \in \{1, \ldots, m\}\} + 4n.$$

It remains to specify how we reduce the queries. A query for the Hamming Distance between $S[i..i + \ell - 1]$ and $T[j..j + \ell - 1]$ is transformed into a query to the Two Intervals Matching Problem with intervals $I_B = [4i + 1, 4(i + \ell)]$ and $I_R = [4j + 1, 4(j + \ell)]$. Since matching the points in order is optimal, this query returns a distance of the form $k \cdot Z(4|j - i|) + (\ell - k) \cdot Y(4|j - i|)$, where k is the Hamming Distance between $S[i \ldots i + \ell]$ and $T[j \ldots j + \ell]$.

This finishes the description of the reduction. As $|B| \in \Theta(|S|)$ and $|R| \in \Theta(|T|)$, the lower bound for the Hamming Distance Oracle Problem simply carries over. □

Now we turn our attention to lower bounds for data structures that may utilize non-combinatorial approaches. We notice that the reduction used by [9] to obtain the combinatorial lower bound for Hamming Distance Oracle can actually be directly used to obtain a non-combinatorial lower bound. Concretely, given two $n \times n$ boolean matrices A, B whose product C we want to compute, we can encode each of A and B in a binary string of length $O(n^2)$ using the same encoding as in [9]. Using a single Hamming distance query to S, T, we can thereby compute one entry of C; hence, we can

compute all of C using n^2 Hamming distance queries. Let $N \coloneqq n^2$. The above discussion implies that for the Hamming Distance Oracle problem on two strings of length $O(N)$, the total time for preprocessing time and answering N queries cannot be faster than $\Omega(n^{\omega-\varepsilon}) = \Omega(N^{\omega/2-\varepsilon})$ for every $\epsilon > 0$ unless the boolean matrix multiplication conjecture fails. Thus, with the same reduction that we applied to obtain Theorem C.1, we get our non-combinatorial lower bound.

Theorem C.2. *Let $n \coloneqq |B| = |R|$ and $\varepsilon > 0$. Unless the Boolean Matrix Multiplication Hypothesis fails, no data structure for the Two Intervals Matching Problem with preprocessing time P and query time Q satisfies $P + nQ \in \Omega(n^{\omega/2-\varepsilon})$, where ω denotes the Matrix Multiplication Exponent.*

At this point, we want to emphasize that for the Balanced Interval Matching Problem, we achieved near-linear preprocessing time and logarithmic query time. Theorem C.2 shows that this and even almost-linear preprocessing time and subpolynomial query time is impossible, unless matrix multiplication is possible in almost-quadratic time.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We describe our data structure and the resulting algorithm in Sections 2 and 3. We provide the experimental evaluation in Section 4 and our conditional lower bound in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We clearly state the running time and required space of our algorithm in the statements of Theorems 1.1 and 1.2. There are no further limitations within our one-dimensional setting. We state the one-dimensional setting throughout the paper and as early as the title.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All lemmas and theorems are stated with all assumptions. All such statements have a complete proof, either after the statement or in the appendix. If the proof is only included in the appendix, it is referred to at the lemma/theorem statement.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The entire code is submitted as supplemental material and also publicly available at `https://github.com/JacobusTheSecond/wasserstein`, and is the only software used for the experimental evaluation. The repository provides exhaustive install instructions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The entire code is publicly available at `https://github.com/JacobusTheSecond/wasserstein`, and is the only software used for the experimental evaluation. The repository provides exhaustive install instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The only hyperparameter chosen is the threshold of 2048 for the naive fallback in the implementation, and we explain why we chose this value.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We only report running times on log log plots as experimental evidence. For it, we discuss the methodology (median running time, 10th and 90th percentiles over multiple runs).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss the hardware on which our experiments were run on.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work is primarily theoretical and the data that is used for experiments is synthetic and created by the authors. We do not see any potential harm or copyright and data-usage issues.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: We do not expect that an algorithmic improvement to the k-partial p-Wasserstein distance implies any societal impacts. None of the areas described by the NeurIPS Code of Ethics are affected.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We only use the code of [13] for comparison to our approach. We appropriately cite their work in the paper and give a link to their code in our supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the code for our evaluation containing a `README.md` for documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.