EFFICIENT ALGORITHMS FOR COMPUTING A PARTIAL p-WASSERSTEIN BARYCENTER IN METRIC SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Given a collection of k probability distributions over a total of n points in a space Ω equipped with a pairwise distance function d, the task of computing an average probability distribution with respect to the optimal transport cost is known in the literature as the *Wasserstein barycenter problem*. However, like OT plans, Wasserstein barycenters are also sensitive to the presence of outliers in each distribution. The *α-partial Wasserstein barycenter problem* for $\alpha \in (0, 1]$, which addresses this issue, asks to compute a measure of α mass for which the average unbalanced OT cost from the input distributions to this measure is minimized.

In this paper, we design provably efficient approximation algorithms for the α-partial Wasserstein barycenter problem. Our first algorithm computes an approximate fixed-support α-partial Wasserstein barycenter in arbitrary spaces in roughly $mn \cdot \delta^{-2}$ time, where m is the size of the fixed support and δ is a small approximation error in the mass. The second algorithm improves the run time in the free-support setting for metric spaces by a factor of m at the expense of an additional ε approximation error in the cost of the OT plans. We additionally prove that an exact solution to the α-partial Wasserstein barycenter problem can be computed in roughly $n^{O(d)}$ time.

1 INTRODUCTION

The optimal transport (OT) problem, at a high level, asks to deform one probability measure into another with the minimum total work. It is a natural method for defining a metric over distributions, called the OT metric, using the ground geometry of the supports (Villani et al., 2008), and has found widespread application in machine learning (Arjovsky et al., 2017; Kusner et al., 2015; Lambert et al., 2022; Rubner et al., 2000; Tong et al., 2024). In many machine learning applications, given a large collection of distributions equipped with the OT metric, the goal is to compute its Wasserstein barycenter, a single representative distribution for the collection that minimizes the average OT distance to the input distributions. The Wasserstein barycenter is notable for its ability to express more insightful average distributions than the ones based on other distances such as KL divergence and total variation in many scenarios.

Wasserstein barycenters are, however, sensitive to the presence of outliers in input distributions. Two general approaches have been proposed to mitigate the impact of outliers on Wasserstein barycenters. The first approach (Le et al., 2021; Wang et al., 2024; Yang & Ding, 2024) requires the target barycenter ν to be a probability distribution, i.e. $\nu(Y) = 1$, and the OT plans from each input distribution to ν are outlier-robust. The problem with this approach, however, is that robust OT plans from each of the input distributions can independently choose which mass in ν is considered an outlier mass with respect to that input distribution. That is, each robust OT plan can transport to a different subset of the barycenter distribution, which may lead to undesired results; see e.g. Figure 1 and refer to Appendix A for additional comparison of the two approaches. The second approach (Kitagawa & Pass, 2015) requires the *target measure* ν itself to be "robust" by restricting its total mass to be α for some $\alpha \in (0, 1]$, and computes unbalanced OT plans from each input distribution to ν. We refer to this version as the *α-partial Wasserstein barycenter* problem, or the α-PWB problem for short. This formulation ensures that all k unbalanced transport plans transport α mass onto the same measure. In this sense, the second approach better captures the geometric centroid nature of the original Wasserstein barycenter. While this model notably produces more geometrically insightful

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

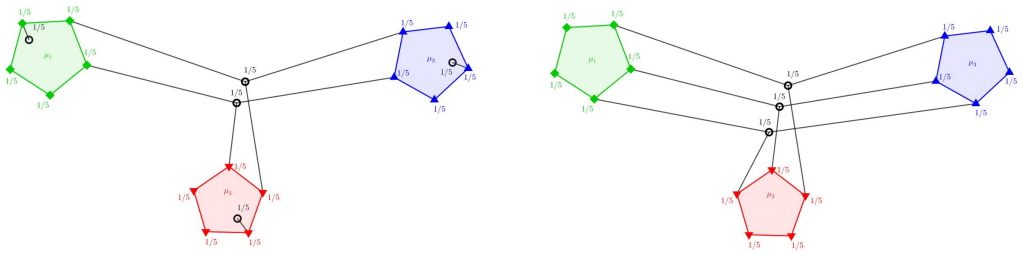


Figure 1: (Left) An example consisting of three uniform distributions (blue, red, green) on five points in $\mathbb{R}^2$, where the first approach to outlier robust Wasserstein barycenters performs poorly. The total mass in the barycenter distribution (black) is 1 and each plan only transports an $\alpha = \frac{3}{5}$ fraction of mass to the barycenter. To minimize the cost, one can add $\frac{1}{5}$ mass of the robust barycenter near each input distribution, which is considered outlier mass (a total of $\frac{2}{5}$) with respect to the other two input distributions. (Right) the partial Wasserstein barycenter on the same example. The barycenter has a measure of $\alpha = \frac{3}{5}$ and all of its mass is transported to by some mass in each input distribution.

outlier-robust Wasserstein barycenters, its multimarginal constraints (i.e. the same amount of mass is transported from each input distribution to any point of the barycenter) are more complicated, which makes the problem computationally more challenging, and no efficient algorithm with provable guarantees are known—see Section 1.2 below. In this paper, we design the first provably efficient approximation algorithm for the α-partial Wasserstein barycenter problem.

1.1 Problem Definition

Optimal transport. Let Ω be a potentially infinite-sized set (e.g. $\mathbb{R}^d$), and let $X, Y \subseteq \Omega$ be two finite subsets of Ω. Let μ and ν be two measures over X and Y, respectively, such that $1 = \mu(X) \geq \nu(Y) > 0$. We call any function $f\colon X \times Y \to \mathbb{R}_{\geq 0}$ a *flow* or a *plan*. If $\sum_{y \in Y} f(x, y) \leq \mu(x)$ for all $x \in X$ and $\sum_{x \in X} f(x, y) = \nu(y)$ for all $y \in Y$, then f is called an *unbalanced transport plan*. In the event that $\mu(X) = \nu(Y)$, then $\sum_{y \in Y} f(x, y) = \mu(x)$ for any unbalanced transport plan f, and it is called a *transport plan* from μ to ν; $f(x, y)$ is the mass being transported from x to y.

Given a cost function $\mathsf{d}\colon \Omega \times \Omega \to \mathbb{R}_{\geq 0}$, the cost of transporting one unit of mass between a pair of points, and a parameter $p \geq 1$, we define the *W_p-cost of a flow* (or a plan) f as the total cost of transporting the mass from μ to ν following the plan f, i.e., $\cent_p(f) := \sum_{(x,y) \in X \times Y} f(x, y)\mathsf{d}^p(x, y)$. Then the *$W_p$ optimal transport (OT) problem* asks to compute a transport plan from μ to ν with the minimum W_p-cost. We will call such a plan a W_p OT plan. In the event that d is a metric, then $w_p(\mu, \nu) := \min_{\tau^*} \left(\cent_p(\tau^*)\right)^{\frac{1}{p}}$ is called the *W_p-distance* between μ and ν, and W_p is also a metric.

Wasserstein barycenter. Suppose $X_1, \ldots, X_k$ are k finite subsets of Ω. Let $\boldsymbol{\mu} = \langle \mu_1, \ldots, \mu_k \rangle$ be a collection of probability distributions, where μ_i has support X_i for each $i \leq k$. The *p-Wasserstein barycenter problem* asks to compute a new probability measure ν (referred to as the *Wasserstein barycenter*) over some $Y \subseteq \Omega$ and transport plans $\boldsymbol{f} = \langle f_1, \ldots, f_k \rangle$ from μ_i to ν (i.e. $\sum_{x \in X_i} f_i(x, y) = \nu(y)$ for all $y \in Y$ and $i \leq k$) such that $\cent_p(\boldsymbol{f}) = \frac{1}{k} \sum_{i=1}^{k} \cent_p(f_i)$ is minimized. Given an additional value of $\alpha \in (0, 1]$, the *α-partial p-Wasserstein barycenter problem* (α-PWB problem for short assuming p is fixed) asks to compute a new measure ν over some $Y \subseteq \Omega$ such that $\nu(Y) := \sum_{y \in Y} \nu(y) = \alpha \leq 1$ and unbalanced transport plans $\boldsymbol{f} = \langle f_1, \ldots, f_k \rangle$ from μ_i to ν such that $\cent_p(\boldsymbol{f})$ is minimized. We refer to the measure ν as the *α-partial Wasserstein barycenter (α-PWB)*. Set $w_p(\boldsymbol{f}) = \left(\cent_p(\boldsymbol{f})\right)^{\frac{1}{p}}$. It is known that the support of ν is a finite set. See, for example, analysis of Anderes et al. (2016) for a proof of this statement.

For a parameter $\varepsilon > 0$, we define an ε-relative (resp. ε-additive) approximate α-PWB to be a measure ν over Ω and unbalanced transport plans $\boldsymbol{f} = \langle f_1, \ldots, f_k \rangle$ such that $\nu(\Omega) = \alpha$ and $w_p(\boldsymbol{f}) \leq (1 + \varepsilon) w_p(\boldsymbol{f}^*)$ (resp. $w_p(\boldsymbol{f}) \leq w_p(\boldsymbol{f}^*) + \varepsilon \cdot \operatorname{diam}(X)$ for $X = \bigcup_{i=1}^{k} X_i$), where $\boldsymbol{f}^*$ is the optimal unbalanced transport plan to the α-PWB $\boldsymbol{\nu}^*$. Given parameters $\varepsilon > 0$ and $\delta \in (0, \alpha)$, we define a *δ-relaxed ε-approximate α-partial barycenter* (or a *(δ, ε)-approximate α-PWB* for brevity) to

be a measure ν over Ω such that $\alpha - \delta \leq \nu(\Omega) \leq \alpha$ and unbalanced transport plans $\boldsymbol{f} = \langle f_1, \ldots, f_k \rangle$ such that $w_p(\boldsymbol{f}) \leq (1+\varepsilon) \cdot \min_{\nu^*, \boldsymbol{f^*}} w_p(\boldsymbol{f^*})$ where $\boldsymbol{f}^*$, as above, is the optimal unbalanced transport plan to the α-PWB $\boldsymbol{\nu}^*$. Note that $(0, \varepsilon)$-approximate α-PWB is the same as ε-relative approximate α-PWB. Furthermore, we observe that the notion of (δ, ε)-approximate α-PWB is in some sense stronger than δ-additive approximate α-PWB because a $(\delta, 0)$-approximate α-PWB ν' immediately yields a $\delta^{1/p}$-additive approximate α-PWB by transporting $\alpha - \nu'(\Omega) \leq \delta$ mass at the cost of at most $\delta^{1/p} \cdot \mathrm{diam}(X)$ from each input distribution in a straightforward manner.

1.2 Related Work

Using the recent min-cost flow algorithms, a W_p-OT plan between two discrete distributions can be computed in near quadratic time (Chen et al., 2025), but these algorithms are too complicated to be of practical use. Driven by the need for scalable algorithms, machine learning research on OT has primarily focused on designing simple and efficient approximation algorithms and several of them are known (Agarwal et al., 2024; Altschuler et al., 2017; Cuturi, 2013; Dvurechensky et al., 2018; Fox & Lu, 2023; Jambulapati et al., 2019; Lahn et al., 2023; Lévy & Schwindt, 2018; Solomon et al., 2015). As discussed above, optimal transport costs are vulnerable to the presence of outliers, and therefore numerous approaches have been presented to mitigate the impact of outliers (Agarwal et al., 2025b; Balaji et al., 2020; Chapel et al., 2020; Ma et al., 2025; Mukherjee et al., 2021; Nietert et al., 2022; Raghvendra et al., 2024; Séjourné et al., 2023).

In contrast to the discrete OT problem, the Wasserstein barycenter problem is known to be NP hard in $\mathbb{R}^d$ if d is not fixed (Altschuler & Boix-Adsera, 2022). The best known algorithm in $\mathbb{R}^d$ runs in $\left(n^d \log U\right)^{O(1)}$ time (Altschuler & Boix-Adsera, 2021) using the ellipsoid method, where $n = \sum_{i=1}^{k} |X_i|$. If one fixes a candidate support set Y of m points in advance, the so-called *fixed-support barycenter problem*, then computing the lowest cost distribution over Y is a linear program that can be solved in $(n+m)^{O(1)}$ time. Choosing the candidate support Y to be the set of $N = \prod_{i=1}^{k} |X_i| = O(n^k)$ centroids in (Ω, d^p) for each k-tuple in $\prod_{i=1}^{k} X_i$ results in the optimal barycenter. There exists a natural trade-off between the size of Y and the quality of the resulting fixed-support barycenter.

For $p = 1$, a $(1+\varepsilon)$-approximate W_1 barycenter in $\mathbb{R}^d$ over a candidate support of size $n\varepsilon^{-O(d)}$ can be computed in $n\left(\frac{\log n}{\varepsilon^d}\right)^{O(1)}$ time (Agarwal et al., 2025a). However for $p > 1$, still no efficient relative-approximation algorithm is known. A $O(\delta)$-additive approximation algorithm can be obtained by choosing a $\delta\Delta$-grid of δ^{-d} points and computing the fixed-support Wasserstein barycenter with this grid as the support. Following this approach, fast additive-approximation algorithms for the general fixed-support Wasserstein barycenter problem have been proposed (Benamou et al., 2015; Cuturi & Doucet, 2014; Dvurechensky et al., 2018; Ge et al., 2019; Kroshnin et al., 2019; Lin et al., 2020; Takezawa et al., 2022). The fastest among these algorithms is the iterative Bregman projection (IBP) method (Benamou et al., 2015) that computes a δ-additive approximate fixed-support Wasserstein barycenter in $\widetilde{O}\left(mn \cdot \delta^{-2p}\right)$ time. Using the $\delta\Delta$-grid, this results in a $\widetilde{O}\left(nd + \delta^{-2d-2p} \cdot k\right)$ time algorithm. In addition to these fixed-support Wasserstein barycenter algorithms, there are a slew of alternative approaches to computing approximate Wasserstein barycenters which either prove additive approximation guarantees for free support Wasserstein barycenters (Borgwardt & Patterson, 2020; Carlier et al., 2015; Claici et al., 2018; Li et al., 2020) or serve as efficient and practically impactful heuristics (Bonneel et al., 2015; Halmos et al., 2024; Le et al., 2019; Puccetti et al., 2020).

As discussed above, two different approaches have been proposed for the "outlier-robust Wasserstein barycenter". The first approach (Cheng & Liu, 2026; Le et al., 2021; Wang et al., 2024; Yang & Ding, 2024)—which models the barycenter as a probability distribution, takes an outlier-robust optimal transport cost and computes the barycenter with respect to this cost function—is notable for the ability to reduce the problem to the classical barycenter problem. For example, the λ-robust OT cost of Mukherjee et al. (2021) is equivalent to replacing each edge cost $c(u, v)$ with $\min\{c(u, v), \lambda\}$ and computing the classical OT cost. Therefore, under the first model for outlier-robust Wasserstein barycenters, an additive δ-approximation can be computed in roughly $mn \cdot \delta^{-2p}$ time.

The second approach, which we referred to as the *α-partial Wasserstein barycenter (α-PWB) problem*, treats the barycenter measure itself as outlier-robust and computes classical (partial) OT plans from

each input distribution to the barycenter distribution. The α-PWB problem has technical advantages over the first formulation. However, none of the algorithms for the first formulation of the outlier-robust Wasserstein barycenter problem provably extend to the partial barycenter problem. Kitagawa & Pass (2015) prove some structural properties of partial barycenter but do not present any algorithm. Le et al. (2022) describe an additive δ-approximation algorithm for the partial barycenter problem that runs in $O\left(Nk^3\delta^{-2p}\right)$ time in any finite metric space. To the best of our knowledge, this represents the state-of-the-art for computing partial Wasserstein barycenters. Although a few heuristics are known for the partial Wasserstein barycenter problem, including Benamou et al. (2015) and Cances et al. (2026), proving their performance has remained an open problem and sparked the work of Le et al. (2022). Furthermore, we are unaware of any ε-relative approximation algorithm or exact algorithm for computing the α-partial barycenter that runs in $N \cdot \left(\frac{k}{\varepsilon}\right)^{O(1)}$ time; using a black box LP solver, the best known exact algorithm runs in $N^{O(1)}$ time.

1.3 Our Contributions

Our primary contribution is the first efficient, polynomial-time (δ, ε)-approximation algorithm for the α-partial Wasserstein barycenter problem. Let (X_i, μ_i), $1 \le i \le k$, be k discrete probability distributions. Set $X = \bigcup_{i=1}^{k} X_i$, $\boldsymbol{\mu} = \langle \mu_1, \ldots, \mu_k \rangle$, and $n = \sum_{i=1}^{k} |X_i|$. For brevity, we will use $(X, \boldsymbol{\mu})$ to denote these k distributions. We refer to n as the *size* of $(X, \boldsymbol{\mu})$, and we define the *spread* to be $\Delta := \Delta(X, \boldsymbol{\mu}) = \max\left\{n, \max_i \frac{\max_{x_i \in X_i} \mu_i(x_i)}{\min_{x_i \in X_i} \mu_i(x_i)}, \frac{\max_{a,b \in X} \mathsf{d}(a,b)}{\min_{a \neq b \in X} \mathsf{d}(a,b)}\right\}$, i.e., the maximum of the size, the spread of mass of each distribution, and the spread of pairwise distances in X. We first design an efficient algorithm for the fixed-support setting.

Theorem 1. *Let Ω be a point set equipped with a cost function $\mathsf{d}\colon \Omega^2 \to \mathbb{R}_{\ge 0}$. Let $(X, \boldsymbol{\mu}) = \{(X_1, \mu_1), \ldots, (X_k, \mu_k)\}$ be a family of k discrete probability distributions with $\sum_{i=1}^{k} |X_i| = n$ and spread $\Delta > 1$, let $Y \subseteq \Omega$ be a set of m points, and let $\alpha \in (0, 1], \delta \in (0, \alpha)$ be two parameters. A $(\delta, 0)$-approximate fixed-support α-partial W_p barycenter of $(X, \boldsymbol{\mu})$ with support set Y can be computed in $n(m+k)\delta^{-2}\log^{O(1)}\Delta$ time.*

We next show that our algorithm can be further refined for the free-support setting in $\mathbb{R}^d$ to shave off a factor of m in the runtime at the expense of some approximation error in the cost.

Theorem 2. *Let $(X, \boldsymbol{\mu})$ be a family of k discrete probability distributions in a metric space (Ω, d) with $\sum_{i=1}^{k} |X_i| = n$ and spread Δ, and let $\alpha \in (0, 1], \delta \in (0, \alpha)$ be parameters.*

1. *If (Ω, d) is d-dimensional Euclidean space, then for any parameter $\varepsilon > 0$, an (δ, ε)-approximate α-partial W_p barycenter can be computed in $nk\left(\varepsilon^{-d}\delta^{-1}\log\Delta\right)^{O(1)}$ time.*
2. *If (Ω, d) is d-dimensional Euclidean space, then for any parameter $c > 1$, an $(\delta, O(c))$-approximate α-partial W_p barycenter can be computed in $n^{1+\frac{1}{c^2}}k\left(\delta^{-1}\log\Delta\right)^{O(1)}$ time.*
3. *For any $c > 1$, an $(\delta, O(c))$-approximate α-partial W_p barycenter can be computed in $n^2\log^{O(1)}\Delta + n^{1+\frac{1}{c}}k\left(\delta^{-1}\log\Delta\right)^{O(1)}$ time.*

We present two algorithms to obtain Theorems 1 and 2. The first algorithm iteratively builds up a measure ν by pushing more flow from each input distribution to the support of ν. In each iteration, we maintain a set of weights and compute a k-tuple of points $\boldsymbol{x} = (x_1, \ldots, x_k) \in \prod_{i=1}^{k} X_i$ such that the average weighted cost from each x_i to the centroid $\mathsf{ctr}(\boldsymbol{x})$ of $\boldsymbol{x}$ is minimized. We then increase $\nu(\mathsf{ctr}(\boldsymbol{x}))$ by a small amount (equivalently increase each $f_i(x_i, \mathsf{ctr}(\boldsymbol{x}))$ by the same small amount) and increment the weights of $x_1, \ldots, x_k$ so that they are less likely to be called upon in future iterations. We prove that if this procedure is repeated enough times, then the resulting measure ν is the desired approximate α-partial barycenter. In contrast to IBP algorithms, which notably struggle to prove such a guarantee for the partial barycenter problem (Le et al., 2022), our algorithm always preserves the multi-marginal constraints $f_i(X_i, y) = f_j(X_j, y) = \nu(y)$ for all distributions i, j and barycenter support points y. By maintaining this invariant, the convergence analysis of our algorithm becomes an easy consequence of standard results in convex optimization (Arora et al., 2012; Boyd & Vandenberghe, 2004), and we prove the runtime to be roughly mn. We note that the simplicity of our algorithm and its analysis are both of independent interest.

The second algorithm, described in Section 3, clusters the points in the support of each distribution and represents each cluster with a single point, thereby working with a compact representation of each distribution. We then use the above algorithm to compute a set of k unbalanced transport plans. A side effect of clustering is that a point may appear in multiple clusters, so we may transport more mass from a point. As such, we relax the demand constraint of each point, and we use another level of iterative procedure (based on the multiplicative-weight-update method) to gradually tighten the demand constraint so that we have an approximate unbalanced transport plan when the algorithm terminates. The exact implementation of the algorithm depends on the underlying setting. This two-level approach yields Theorem 2.

As an additional result, we show how the α-partial Wasserstein barycenter problem can in fact be solved exactly in polynomial time in constant dimensional Euclidean spaces. As in Altschuler & Boix-Adsera (2021) for computing the classical barycenter, we also rely on the ellipsoid algorithm.

Theorem 3. *Let $(X_1, \mu_1) \dots, (X_k, \mu_k)$ be k discrete probability distributions in $\mathbb{R}^d$ with total size of n and spread Δ, $p \geq 1$ a constant, $\mathsf{d}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ a distance function such that the bisector of two points under d^p is an algebraic variety of constant degree, and $\alpha \in (0, 1]$ a parameter. Then an α-partial p-Wasserstein barycenter can be computed in $\left(n^d \log \Delta\right)^{O(1)}$ time.*

2 Approximation Algorithm for the Partial Barycenter Problem

In this section, we present the first approximation algorithm for the fixed-support α-partial Wasserstein barycenter problem (Theorem 1). Our algorithm consists of two stages. We first perform a binary search on the cost of the barycenter and argue that the decision procedure for the partial barycenter problem can be reformulated as a packing LP. Then, we present an iterative algorithm that computes an approximate solution to this packing LP.

Binary search. We perform binary search on the cost of the optimal α-partial barycenter over the search interval $[0, \Delta]$ with an initial guess of Δ. Given a guess $g > 0$ of the optimal cost[1] $\min_{\nu, \boldsymbol{f}} \mathfrak{c}_p(\boldsymbol{f})$, the decision problem determines whether (a) g is *large*, i.e., there exists a measure ν with mass $\nu(Y) \in (\alpha - \delta, \alpha]$ and set of unbalanced transport plans $\boldsymbol{f}$ from each μ_i to ν such that $\mathfrak{c}_p(\boldsymbol{f}) \leq g$, or (b) g is *small*, i.e., for every measure ν and set of unbalanced transport plans $\boldsymbol{f}$ from each μ_i to ν such that $\mathfrak{c}_p(\boldsymbol{f}) \leq g$, one has $\nu(Y) < \alpha$. If both of these cases are true, then the decision procedure can return either answer. In the first case, it also returns such a choice of ν and $\boldsymbol{f}$. If g is too large (resp. small), we decrease (resp. increase) the guess g. The binary search terminates when the size of the search interval is at most δ times the left end of the search interval, i.e. the largest guess deemed too small. We return the measure $\frac{\nu}{1+\delta}$ and unbalanced transport plans $\frac{\boldsymbol{f}}{1+\delta}$, scaled versions of those computed for the right end of the search interval, i.e. the smallest guess deemed too large as the desired approximate partial barycenter, to have a cost of at most the left end of the search interval.

The decision problem can be formulated as a packing LP problem[2] as shown in Figure 2(i). It determines the largest measure ν on Y and the corresponding unbalanced transport plans from the input distributions to ν such that the total cost of the transport plans is at most g.

An iterative decision procedure. We now describe an iterative algorithm, based on the *multiplicative weight update method* Arora et al. (2012), for computing an approximate solution to the above packing LP. For simplicity, we describe a somewhat inefficient implementation of the algorithm. At the end, we sketch how to make it efficient and provide its details in Appendix C.

Let $\mathbb{X} = \prod_{i=1}^{k} X_i$. We use $\boldsymbol{x} = (x_1, \dots, x_k)$ to denote a k-tuple from $\mathbb{X}$. We will maintain weights $\psi_i^t(x)$, for each point $x \in X_i$, and ψ_0^t which essentially indicate how much mass has been transported from x and how costly the current set of unbalanced transport plans are. We also maintain flow functions $f_i\colon X_i \times Y \to [0, 1]$ for $1 \leq i \leq k$. In each iteration, we will choose a k-tuple $\boldsymbol{x} \in \mathbb{X}$

[1] We need to check if $g = 0$ results in a feasible approximate α-partial barycenter. The guess $g = 0$ is only a good guess if α mass can be constructed by k-tuples of masses, one from each distribution, which are all co-located. This can be done in linear time in a greedy manner.

[2] To be precise, the LP is not exactly a packing LP. However, in the process of solving this LP we will maintain the invariant that the equality constraints are always preserved.

$$\begin{aligned} &\max \ \ \nu(Y) \\ &\text{s.t. } f_i(x,Y) \le \mu_i(x) \ \ \forall i \le k, x \in X_i, \\ &\quad f_i(X_i,y) = \nu(y) \ \ \forall i \le k, y \in Y, \\ &\quad \text{¢}_p(\boldsymbol{f}) \le g, \quad f_i, \nu \ge 0. \end{aligned}$$

(i)

$$\begin{aligned} &\max \ \ \nu(Y) \\ &\text{s.t. } f_i(\widetilde{x}_j, B_j) \le \mu_i(\widetilde{x}_j) \ \ \forall i \le k, (A_j, B_j) \in \Pi_i, \\ &\quad f_i(\widetilde{X}_i, y) = \nu(y) \ \ \forall i \le k, y \in Y, \\ &\quad f_i(\widetilde{x}_j, y) = 0 \ \ \forall i, (A_j, B_j) \in \Pi_i, y \notin B_j, \\ &\quad \text{¢}_p(\boldsymbol{f}) \le g, \quad \text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}) \le \frac{g}{\delta}, \quad f_i, \nu \ge 0. \end{aligned}$$

(ii)

Figure 2: The progression of the packing LP formulation for the partial barycenter decision problem: (i) the packing LP studied in Section 2, the decision variant of the partial barycenter problem; (ii) the relaxed partial barycenter problem invoked in the algorithm of Section 3, demand constraints change to WSPD pair capacities and we add a new constraint to make sure the dual objective of the flow is bounded by $\frac{g}{\delta} = \text{Þ}(\boldsymbol{\phi})$.

whose weighted cost to its centroid $\mathsf{ctr}(\boldsymbol{x}) = \arg\min_{y \in Y} \left\{ \sum_{i=1}^{k} \mathsf{d}^p(x_i, y) \right\}$ is minimized. We increase the flow by a small (but the same) amount from each point x_i of $\boldsymbol{x}$ to the centroid of $\boldsymbol{x}$ and increase its weight $\psi_i^t(x_i)$. We also update the weight of ψ_0^t to account for the increased cost. We terminate once either the transport cost exceeds g or there exists a point $x \in X$ whose mass is exhausted.

We now describe the algorithm in more detail. Define an update parameter $\eta = \frac{\delta}{4}$ and constraint overcommitment threshold $T = O(\eta^{-2} \log n)$. Set an iteration counter $t = 1$. Initialize the constraint weights as $\psi_i^1(x_i) = 1$ for all $1 \le i \le k$ and $x_i \in X_i$ and $\psi_0^1 = 1$. Additionally initialize flows $f_i(x, y) = 0$ for all pairs $(x, y) \in X_i \times Y$.

Define the *weighted cost* $\ell_i^t(x_i, y) = \mathsf{d}^p(x_i, y) + \frac{\psi_i^t(x_i) \cdot g}{\psi_0^t \cdot \mu_i(x_i)}$ for each $i \le k$ and every $x_i \in X_i, y \in Y$. For any $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathbb{X}$ and $y \in Y$, we will use the notation $\mathsf{d}^p(\boldsymbol{x}, y) = \sum_{i=1}^{k} \mathsf{d}^p(x_i, y)$ and $\ell^t(\boldsymbol{x}, y) = \sum_{i=1}^{k} \ell_i^t(x_i, y)$. For each tuple $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathbb{X}$, we define the centroid of $\boldsymbol{x}$ as $\mathsf{ctr}(\boldsymbol{x}) := \arg\min_{y \in Y} \{\mathsf{d}^p(\boldsymbol{x}, y)\}$. We additionally define $f_i(x, Y) = \sum_{y \in Y} f_i(x, y)$ for all $1 \le i \le k$ and $x \in X_i$.

While both (1) $f_i(x_i, Y) < T \cdot \mu_i(x_i)$ for all $i \le k$ and $x_i \in X_i$, and (2) $\text{¢}_p(\boldsymbol{f}) < T \cdot g$, we repeat the following.

Let $\boldsymbol{x}^t = (x_1^t, \ldots, x_k^t)$ be the tuple in $\mathbb{X}$ minimizing $\ell^t(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))$. Define

$$\overline{\lambda_0^t} = \left\lceil \frac{\text{¢}_p(\boldsymbol{f})}{g} \right\rceil \cdot g - \text{¢}_p(\boldsymbol{f}), \quad \lambda_0^t = \frac{\overline{\lambda_0^t}}{\mathsf{d}^p(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))}, \quad \lambda_i^t = \left\lceil \frac{f_i(x_i^t, Y)}{\mu_i(x_i^t)} \right\rceil \mu_i(x_i^t) - f_i(x_i^t, Y) \quad (1)$$

for $1 \le i \le k$; λ_i^t is the total amount of mass one can commit to the edge $(x_i^t, \mathsf{ctr}(\boldsymbol{x}))$ for the flow $f_i(x_i^t, Y)$ through x_i^t to reach the next integer multiple of $\mu_i(x_i^t)$, $\overline{\lambda_0^t}$ denotes how far away the costs of the current flows are from the next integer multiple of g, and λ_0^t denotes the total amount of mass that can be transported through each edge $(x_i^t, \mathsf{ctr}(\boldsymbol{x}^t))$ to attain a cost of $\overline{\lambda_0^t}$. Then for the flow update parameter $\lambda^t = \min\{\lambda_0^t, \min_i \{\lambda_i^t\}\}$, we increment the flows and update the weights as follows:

$$\begin{aligned} f_i(x_i^t, \mathsf{ctr}(\boldsymbol{x}^t)) &\leftarrow f_i(x_i^t, \mathsf{ctr}(\boldsymbol{x}^t)) + \lambda^t && \text{for all } 1 \le i \le k, \\ \psi_i^{t+1}(x_i^t) &\leftarrow \psi_i^t(x_i^t) \cdot \left(1 + \eta \cdot \frac{\lambda^t}{\mu_i(x_i^t)}\right) && \text{for all } 1 \le i \le k, \\ \psi_0^{t+1} &\leftarrow \psi_0^t \cdot \left(1 + \eta \cdot \frac{\lambda^t \cdot \mathsf{d}^p(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))}{g}\right) \end{aligned}$$

We keep the weights $\psi_i^{t+1}(x) = \psi_i^t(x)$ for all $x \ne x_i^t$ in X_i under each distribution $i \le k$, i.e. for all points whose outgoing flow does not increase in iteration t. Then increment t and return to the beginning of the while loop.

Once $f_i(x_i, Y) = T \cdot \mu_i(x_i)$ for some $i \le k$ and $x_i \in X_i$ or $\text{¢}_p(\boldsymbol{f}) = T \cdot g$, we stop and return the flows $\frac{\boldsymbol{f}}{T} = \left\langle \frac{f_1}{T}, \ldots, \frac{f_k}{T} \right\rangle$ and measure ν such that $\nu(y) = \frac{f_i(X_i, y)}{T}$ for some $i \le k$ and every $y \in Y$

($f_i(X_i, y) = f_j(X_j, y)$ for all pairs i, j). Note that $\nu(Y) = \frac{1}{T}\sum_{t\geq 1}\lambda^t$. Conclude that g is small if $\nu'(Y) < \alpha - \delta$, and g is large otherwise.

Analysis. We first prove the correctness of the algorithm and then analyze its runtime. For any time $t \geq 1$ let $\Psi^t = \psi_0^t + \sum_{i=1}^k \sum_{x\in X_i} \psi_i^t(x)$ be the total weight at time t. Let $F = \sum_{i=1}^k \sum_{x_i, y} f_i(x_i, y)$ be the total amount of flow computed by the algorithm, let $\nu(Y) = \frac{F}{T}$ be the measure computed by the algorithm, and let $\nu_g^*(Y) = \arg\max_{\nu'} \nu'(Y)$ be the optimal solution of the decision problem, i.e., the maximum measure that can be transported from each of the input distributions at the total cost of at most g. The following lemma states the main inequality we use to prove the correctness of the algorithm.

Lemma 1. *There exists a constant $0 < c \leq 2$ such that*

$$(1-\eta) - \frac{c\ln n}{T\eta} \leq \frac{1}{T}\sum_{t\geq 1}\frac{\lambda^t}{\Psi^t}\cdot\left(\frac{\psi_0^t}{g}\cdot \ell_i^t(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))\right) \leq \frac{\nu(Y)}{\nu_g^*(Y)}.$$

The proof of Lemma 1 is given in Appendix B. By substituting the values of $\eta = \frac{\delta}{4}$ and $T = O(\eta^{-2}\log n)$ in Lemma 1, we immediately obtain that $\frac{\nu(Y)}{\nu_g^*(Y)} \geq 1 - \frac{\delta}{2}$. Hence, we obtain the following.

Lemma 2. *For a given cost value $g \geq 0$ and set Y, the decision procedure computes a measure ν over Y such that $\nu(Y) \geq \nu_g^*(Y) - \delta$, where $\nu_g^*(Y)$ is the optimal solution of the decision problem.*

Finally, by choosing the stopping criteria of the binary search to be when the width of the search interval is roughly δ times the width of the search interval, we conclude that the binary search terminates in $O(\log\Delta)$ many steps. After rescaling the transport plans computed for the upper bound guess to have cost equal to the lower bound guess, the binary search successfully concludes with a cost value g bounded above by the cost of the α-partial W_p barycenter. If the search interval is small enough, then we conclude that the measure still has total mass at least $\alpha - \delta$. Hence, the algorithm computes a $(\delta, 0)$-approximate α-partial W_p barycenter.

To analyze the runtime of the algorithm, we need the following straightforward lemma.

Lemma 3. *The decision algorithm terminates within $O(nT) = O(n\delta^{-2}\log n)$ steps.*

Each step of the algorithm does the following tasks: (1) compute $\underline{\boldsymbol{x}}^t$, (2) add transported mass from x_i^t to $\mathsf{ctr}(\boldsymbol{x}^t)$, and (3) increase the weights $\psi_i^t(x_i)$ and ψ_0^t based on the transported mass to $\boldsymbol{x}^t$. The most expensive part in each step of the decision procedure is computing $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \ell^t(\boldsymbol{x}, \mathsf{ctr}(\boldsymbol{x}))$; the other two steps can be done in $O(k)$ time. A naive implementation takes $O(mn)$ time to compute $\boldsymbol{x}^t$. Therefore, by Lemma 3, the decision algorithm takes $O(n^2 m\delta^{-2}\log n)$ time. We shave off an n factor from the runtime by approximating ℓ^t with another function $\underline{\ell}^t \colon \mathbb{X}\times Y \to \mathbb{R}_{\geq 0}$ where for $x \in X_i$ and $y \in Y$, $\underline{\ell}_i^t(x, y) = \mathsf{d}^p(x, y) + \frac{\psi_i^t(x)}{\mu_i(x)}\cdot\frac{g}{\underline{\psi}_0^t}$, and $\underline{\psi}_i^t$, $\underline{\psi}_0^t$ are approximations of ψ_i^t, ψ_0^t obtained by updating the weights in a lazy manner so that $\underline{\psi}_i^t(x) \leq \psi_i^t(x) \leq (1+2\eta)\underline{\psi}_i^t(x)$ and $\underline{\psi}_0^t \leq \psi_0^t \leq (1+2\eta)\underline{\psi}_0^t$, respectively. In Appendix C, we describe a dynamic data structure that reports $\underline{\boldsymbol{x}}^t = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \underline{\ell}^t(\boldsymbol{x}, \mathsf{ctr}(\boldsymbol{x}))$ in constant time and maintains $\underline{\boldsymbol{x}}^t$ in amortized $O(m\log n)$ time under updates to the weights $\underline{\psi}_i^t$ and $\underline{\psi}_0^t$. We prove in Lemma 8 that lazy weight updates only distort Lemma 1 by a $(1+2\eta)$ factor. The consequence of the data structure is that each step of the algorithm can be implemented in amortized $O(k + m\log n)$ time. Combined with Lemmas 2 and 3, we obtain Theorem 1.

3 A Faster Algorithm

In this section, we show how we can improve the runtime of the above algorithm in terms of m, at the expense of a small approximation error, when $X \subset \mathbb{R}^d$ and the support of ν can be arbitrary points in $\mathbb{R}^d$, by choosing a support Y of ν of $\widetilde{O}(n)$ points (see Appendix D) and by working with a compact representation of the input distributions. The algorithm consists of three levels. At the outer level, we perform binary search on the guess of the cost g identical to Section 2. At the inner level, for a fixed

guess g we compute a solution to the decision variant of the packing LP. To solve the packing LP more efficiently, we work with a relaxed version of the packing LP formed by compressing edges and pushing constraints from vertices to edges, which may overcommit some points, referred to as *congestion*. We therefore add a middle level, which we refer to as the *congestion-reducing procedure*, that assigns weights to the input points that penalize them for having large congestion, gradually bringing down the congestion at each point to almost one after enough iterations. Each iteration of the middle loop executes the inner loop and updates the weights. We now describe the inner and middle level procedures in detail.

Inner-level compressed packing LP algorithm. Let (Ω, d) be a metric space and let $X, Y \subseteq \Omega$ be two point sets. For parameters $\varepsilon \geq 0$ and $\rho \geq 1$, define an (ε, ρ)*-pair decomposition* of $X \times Y$ as a set $\Pi = \{(A_1, B_1), \ldots, (A_r, B_r)\}$ such that:

1. for every $j \leq r$, $A_j \subseteq X$ and $B_j \subseteq Y$;
2. for every $x \in X, y \in Y$ there exists a pair $(A_j, B_j) \in \Pi$ where $x \in A_j$, $y \in B_j$, and
$$(1+\varepsilon)^{-1} \cdot \max_{z,w \in A_j \times B_j} \mathsf{d}(z,w) \leq \mathsf{d}(x,y) \leq (1+\varepsilon) \cdot \min_{z,w \in A_j \times B_j} \mathsf{d}(z,w),$$
3. every $x \in X \cup Y$ appears in at most ρ pairs of Π, i.e. $|\{(A_j, B_j) \in \Pi : x \in A_j \cup B_j\}| \leq \rho$.

We note that every pair of sets X, Y of size n and distance function d admits a trivial $(0, n)$-pair decomposition, which is simply the set $X \times Y$. Well-known trade-offs have been studied in the past that significantly reduce ρ at the cost of a relatively small increase in ε. For example: in d-dimensional Euclidean space, one can construct either a $(\varepsilon, \varepsilon^{-d} \log \Delta)$-pair decomposition in $O(n(\varepsilon^{-d} + \log n))$ time (Callahan & Kosaraju, 1995) or a $\left(O\left(\varepsilon^{-\frac{1}{2}}\right), n^{\varepsilon} \log \Delta\right)$-pair decomposition in $O(n^{1+\varepsilon} \log \Delta)$ time (Andoni & Indyk, 2006; Har-Peled et al., 2013) for any $\varepsilon > 0$; in any metric space, a $(4\varepsilon^{-1}, n^{\varepsilon} \log \Delta)$-pair decomposition can be constructed in $O(n^2)$ time for any $\varepsilon > 0$ (Agarwal et al., 2026).

Given a weight function $\phi \colon X \to \left[0, \frac{\Delta^2}{\delta}\right]$, which will be set by the middle level congestion-reduction procedure below, we will modify the pair decomposition to also group points by similar weights. To construct this modified pair decomposition, we simply take the unweighted pair decomposition and divide each set A_j in the pair (A_j, B_j) into groups such that the weights within a group are equal up to a $(1+\varepsilon)$ factor, therefore dividing each pair (A_j, B_j) into $O\left(\frac{\log \Delta}{\varepsilon}\right)$ many pairs.

We use the pair decomposition on weighted points to compress the packing LP from Section 2 and show how the same algorithm solves the compressed LP more efficiently. Suppose for each $i \leq k$, we compute a (ε, ρ)-pair decomposition Π_i between X_i and Y with respect to d^p and cluster each pair by similar weights as described above. For every $(A_j, B_j) \in \Pi_i$, we choose an arbitrary point of A_j, call it the *center* of A_j and denote it as $\widetilde{x}_j$. We set $\mu_i(\widetilde{x}_j) = \mu_i(A_j)$. Furthermore, for any pair $(x, y) \in A_j \times B_j$, we set its distance as $\widetilde{\mathsf{d}}(x, y) := \mathsf{d}(\widetilde{x}_j, \widetilde{y}_j)$ where $\widetilde{y}_j$ is some arbitrary fixed point of B_j.[3] Then $\widetilde{\mathsf{d}}$ serves as a $(1+\varepsilon)$-approximation of d, and it suffices to compute a $(\delta, 0)$-approximate partial Wasserstein barycenter with respect to $\widetilde{\mathsf{d}}^p(\cdot, \cdot)$. Set $\widetilde{X}_i = \{\widetilde{x}_j : (A_j, B_j) \in \Pi_i\}$. Our goal is to compute a flow function $f_i \colon \widetilde{X}_i \times Y \to [0, 1]$ with the constraint that a mass can be transported from $\widetilde{x}_j$ only to the points of B_j, i.e., $f_i(\widetilde{x}_j, y) = 0$ for all $y \notin B_j$. Given a collection of weights $\boldsymbol{\phi}$ and flows $\boldsymbol{f}$, we define the weight costs $\text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}) = \sum_{i=1}^{k} \sum_{x \in X_i} f_i(x, Y) \cdot \phi_i(x)$ and $\text{Þ}(\boldsymbol{\phi}) = \sum_{i=1}^{k} \sum_{x \in X_i} \phi_i(x) \cdot \mu_i(x)$. If $\boldsymbol{f}$ is a collection of transport plans, then $\text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}) = \text{Þ}(\boldsymbol{\phi})$. Consider the packing LP in Figure 2(ii), a generalization of the packing LP in Figure 2(i), that guarantees weaker edge capacities on the pairs of the pair decomposition rather than vertex demands and the weighted cost of the flow is small.

We adapt the algorithm of Section 2 as follows. In each iteration, we choose a k-tuple $\widetilde{\boldsymbol{x}} = (\widetilde{x}_1, \ldots, \widetilde{x}_k) \in \prod_{i=1}^{k} \widetilde{X}_i$, where for each $i \leq k$, $\widetilde{x}_i$ is the center point of A_i for some pair $(A_i, B_i) \in \Pi_i$ such that $B = \bigcap_{i=1}^{k} B_i \neq \varnothing$. Among such choices, we choose the point in B and the corresponding tuple $\widetilde{\boldsymbol{x}} = (\widetilde{x}_i, \ldots, \widetilde{x}_k)$ that minimizes the average weighted distance, now weighted

[3] If (x, y) appears in multiple pairs (A_j, B_j), we arbitrarily choose one of them to define $\widetilde{\mathsf{d}}(x, y)$.

by *both* $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ (due to the extra constraint $\text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}) \leq \frac{g}{\delta}$), to the $\widetilde{x}_i$'s, then update all ψ weights in an analogous manner. The algorithm can be implemented efficiently, as described in Appendix F. We show in Appendix F that the algorithm computes a measure ν such that $\nu(Y) \geq \max_{\nu'} \nu'(Y) - \delta$, where the maximum is taken over all feasible measures to the LP in Figure 2(ii) (i.e. of cost at most g), in $O(nk\rho\delta^{-2} \log^{O(1)} \Delta)$ time.

Given a flow $f_i \colon \widetilde{X}_i \times Y \to [0, 1]$, we convert it into a flow $f_i \colon X_i \times Y \to [0, 1]$ by distributing the flow from $\widetilde{x}_j$ among the points of A_j. Since a point $x \in X_i$ is in at most ρ pairs, $f_i(x, Y) \leq \rho \cdot \mu_i(x)$ after the redistribution. We therefore call the above LP a *relaxed partial barycenter problem*, and ρ the *congestion* of the relaxed partial barycenter problem.

Our goal is, however, to solve the original LP with unit congestion. We again use a MWU based procedure, referred to as the *congestion-reduction procedure*, that solves a weighted version of the packing LP at each step and updates the weight. We thus have two nested levels of MWU procedures. Suppose the cost of the flow is fixed to g. If the guess g is large, the MWU procedure pushes the congestion of each point $x \in X$ down from ρ to $(1 + \delta)$. Otherwise, the algorithm terminates at some step stating that the cost of transporting α mass exceeds g.

Middle-level congestion-reducing MWU procedure. For each $i \leq k$, we will maintain a weight function $\phi_i \colon X \to \mathbb{R}_{\geq 0}$. The weight functions $\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_k \rangle$ are maintained to satisfy the invariant $\text{Þ}(\boldsymbol{\phi}) = \frac{g}{\delta}$. In particular, no single weight can be larger than $\frac{\Delta^2}{\delta}$ under the invariant. Then define the *weighted distance* $\mathsf{d}_{\phi_i}(u, v) = \mathsf{d}^p(u, v) + \phi_i(u)$. For a flow function $\boldsymbol{f} = \langle f_1, \ldots, f_k \rangle$, we will let $¢_{\boldsymbol{\phi}}(\boldsymbol{f})$ be the cost of $\boldsymbol{f}$ with respect to the weighted distances d_{ϕ_i}. Initialize the weights $\phi_i^1(x) = \frac{g}{\delta k}$ for all $i \leq k$ and $x \in X_i$.

Let $\eta = \Theta(\delta)$ and let $\sigma = O(\rho\eta^{-2} \log \Delta)$. For $t = 1, \ldots, \sigma$, repeat the following. We solve the weighted variant of the packing LP for the relaxed barycenter problem, As shown in Figure 2(ii).

Suppose the algorithm for the weighted packing LP returns in step t a measure ν^t and a flow function $\boldsymbol{f}^t = \langle f_1^t, \ldots, f_k^t \rangle$. The same analysis as in Section 2 (see Appendix B) implies that if the algorithm returns a measure ν^t with $\nu^t(Y) < \alpha - \delta$, then we conclude that g is too small, terminate the current congestion-reduction procedure, update the value of g (in the binary search procedure), and restart the congestion-reduction procedure. Otherwise (i.e. $\nu^t(Y) \geq \alpha - \delta$), we update the weights as follows. We first rescale the mass to be at most α by updating $f_i^t \leftarrow \min\left\{1, \frac{\alpha}{\nu(Y)}\right\} \cdot f_i^t$ and $\nu^t \leftarrow \min\left\{1, \frac{\alpha}{\nu(Y)}\right\} \cdot \nu^t$. Then, we apply the weight update $\phi_i^{t+1}(x) \leftarrow \phi_i^t(x) \cdot \left(1 + \eta \cdot \frac{f_i^t(x,Y)}{\rho \cdot \mu_i(x)}\right)$ on this rescaled flow for all $i \leq k$ and $x \in X_i$. After updating all weights, we normalize the weights $\phi_i^{t+1}(x) \leftarrow \phi_i^{t+1}(x) \cdot \frac{g}{\delta \cdot \text{Þ}(\boldsymbol{\phi}^{t+1})}$ for all i to maintain the invariant $\text{Þ}(\boldsymbol{\phi}^{t+1}) = \frac{g}{\delta}$.

This concludes the congestion reducing algorithm. At the end, we obtain the flows $f_i = \frac{1}{\sigma} \sum_t f_i^t$ for each $1 \leq i \leq k$ and the measure $\nu = \frac{1}{\sigma} \sum_t \nu^t$. The flows and measure certify that g is large. Moreover, by the constraint of the packing LP solved in each iteration, it is immediate that under the approximate distances $\widetilde{\mathsf{d}}^p$, we have $¢_p(\boldsymbol{f}) \leq g$. We then show that the congestion of the resulting flow is almost one (see Appendix E for proof).

Lemma 4. *If the congestion-reducing MWU algorithm returns a flow $\boldsymbol{f}$ and a measure ν, then $f_i(x, Y) \leq (1 + \delta) \cdot \mu_i(x)$ for all $i \leq k$ and $x \in X_i$.*

Because the flows are not feasible unbalanced transport plans, we return the rescaled measure $\frac{\nu}{1+\delta}$ and flows $\frac{\boldsymbol{f}}{1+\delta}$. Combining Lemma 4 with binary search (as in Section 2) proves Theorem 2.

4 Conclusion

In this paper, we present the first provably efficient approximation algorithms for the partial p-Wasserstein barycenter problem, which runs in quadratic time with respect to n and linear time with respect to k, in contrast to the IBP algorithms of Le et al. (2022). We give faster algorithms with respect to n at the expense of some additional approximation factor. Future research directions include: can the algorithm be parallelized, and is there a pure relative-approximation algorithm with a similar runtime.

AI USE STATEMENT

In this work, we have not used generative AI tools for any technical research and idea generation, and all substantive writing was conducted by the authors. We have only used generative AI tools to (1) help with typos and other small writing errors in the final submission, and (2) speed up the process of generating code to test our algorithm (not currently included in the submission) and create figures on real MNIST data in the appendix (Figure 3). We emphasize that we did not use generative AI for any substantive work in the submission. Our main contributions are theoretical and entirely generated by human effort.

We have reviewed all AI-assisted work. The authors have thoroughly reviewed any grammatical errors and typos raised by AI assistance, as well as the AI-aided figure in the appendix. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work is primarily a theoretical contribution to the problem of averaging distributions under the Wasserstein distance amidst the presence of outlier data points. We do not see any ethical concerns with this submission.

REPRODUCIBILITY STATEMENT

Since the contributions of our work are provided in theorem statements, the mathematics can be reproduced through the proofs and exposition of our work and the references therein.

REFERENCES

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 35th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 4514–4529, 2024.

Pankaj K Agarwal, Sharath Raghvendra, Pouyan Shirzadian, and Keegan Yao. Efficient approximation algorithm for computing Wasserstein barycenter under Euclidean metric. In *Proceedings of the 36th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 4809–4826, 2025a.

Pankaj K Agarwal, Sharath Raghvendra, Pouyan Shirzadian, and Keegan Yao. Efficient algorithms for robust and partial semi-discrete optimal transport. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*, 2025b.

Pankaj K Agarwal, Oliver Chubet, Sharath Raghvendra, and Keegan Yao. A scalable constant-factor approximation algorithm for W_p optimal transport. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.

Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1961–1971, 2017.

Jason M Altschuler and Enric Boix-Adsera. Wasserstein barycenters can be computed in polynomial time in fixed dimension. *Journal of Machine Learning Research*, 22(44):1–19, 2021.

Jason M Altschuler and Enric Boix-Adsera. Wasserstein barycenters are NP-hard to compute. *SIAM Journal on Mathematics of Data Science*, 4(1):179–203, 2022.

Ethan Anderes, Steffen Borgwardt, and Jacob Miller. Discrete Wasserstein barycenters: Optimal transport for discrete data. *Mathematical Methods of Operations Research*, 84:389–409, 2016.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 459–468, 2006.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 214–223, 2017.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Robust optimal transport with applications in generative modeling and domain adaptation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 12934–12944, 2020.

Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*. Springer, 2006.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51:22–45, 2015.

Steffen Borgwardt and Stephan Patterson. Improved linear programs for discrete barycenters. *Informs Journal on Optimization*, 2(1):14–33, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Paul B Callahan and S Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

Adrien Cances, Quentin Mérigot, and Luca Nenna. Particle method for a nonlinear multimarginal optimal transport problem. *arXiv preprint arXiv:2603.25584*, 2026.

Guillaume Carlier, Adam Oberman, and Edouard Oudet. Numerical methods for matching for teams and Wasserstein barycenters. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(6): 1621–1642, 2015.

Laetitia Chapel, Mokhtar Z Alaya, and Gilles Gasso. Partial optimal transport with applications on positive-unlabeled learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 2903–2913, 2020.

Li Chen, Rasmus Kyng, Yang Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *Journal of the ACM*, 72 (3):1–103, 2025.

Zixiong Cheng and Hang Liu. Robust wasserstein barycenter. *arXiv preprint arXiv:2603.07563*, 2026.

Sebastian Claici, Edward Chien, and Justin Solomon. Stochastic Wasserstein barycenters. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 999–1008, 2018.

Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2292–2300, 2013.

Marco Cuturi and Arnaud Doucet. Fast computation of Wasserstein barycenters. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 685–693, 2014.

Pavel Dvurechensky, Darina Dvinskikh, Alexander Gasnikov, César A Uribe, and Angelia Nedic. Decentralize and randomize: faster algorithm for Wasserstein barycenters. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 10783–10793, 2018.

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

Dongdong Ge, Haoyue Wang, Zikai Xiong, and Yinyu Ye. Interior-point methods strike back: solving the Wasserstein barycenter problem. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 6894–6905, 2019.

Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

Peter Halmos, Xinhao Liu, Julian Gold, and Benjamin J Raphael. Low-rank optimal transport through factor relaxation with latent coupling. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 114374–114433, 2024.

Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 804–809, 2013.

Arun Jambulapati, Aaron Sidford, and Kevin Tian. A direct $(1/\varepsilon)$ iteration parallel algorithm for optimal transport. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11359–11370, 2019.

Jun Kitagawa and Brendan Pass. The multi-marginal optimal partial transport problem. In *Forum of Mathematics, Sigma*, volume 3, pp. e17. Cambridge University Press, 2015.

Alexey Kroshnin, Nazarii Tupitsa, Darina Dvinskikh, Pavel Dvurechensky, Alexander Gasnikov, and Cesar Uribe. On the complexity of approximating Wasserstein barycenters. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3530–3540, 2019.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 957–966, 2015.

Nathaniel Lahn, Sharath Raghvendra, and Kaiyi Zhang. A combinatorial algorithm for approximating the optimal transport in the parallel and MPC settings. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 21675–21686, 2023.

Marc Lambert, Sinho Chewi, Francis Bach, Silvère Bonnabel, and Philippe Rigollet. Variational inference via wasserstein gradient flows. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 14434–14447, 2022.

Khang Le, Huy Nguyen, Quang Minh Nguyen, Tung Pham, Hung Bui, and Nhat Ho. On robust optimal transport: computational complexity and barycenter computation. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pp. 21947–21959, 2021.

Khang Le, Huy Nguyen, Khai Nguyen, Tung Pham, and Nhat Ho. On multimarginal partial optimal transport: Equivalent forms and computational complexity. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 4397–4413, 2022.

Tam Le, Viet Huynh, Nhat Ho, Dinh Phung, and Makoto Yamada. Tree-Wasserstein barycenter for large-scale multilevel clustering and scalable bayes. *arXiv preprint arXiv:1910.04483*, 2019.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, pp. 424–433, 2014.

Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *Proceedings of the 56th annual IEEE Symposium on Foundations of Computer Science*, pp. 230–249, 2015.

Bruno Lévy and Erica L Schwindt. Notions of optimal transport theory and how to implement them on a computer. *Computers & Graphics*, 72:135–148, 2018.

Lingxiao Li, Aude Genevay, Mikhail Yurochkin, and Justin M Solomon. Continuous regularized Wasserstein barycenters. *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pp. 17755–17765, 2020.

Tianyi Lin, Nhat Ho, Xi Chen, Marco Cuturi, and Michael Jordan. Fixed-support Wasserstein barycenters: Computational hardness and fast algorithm. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, pp. 5368–5380, 2020.

Yiming Ma, Hang Liu, Davide La Vecchia, and Metthieu Lerasle. Inference via robust optimal transportation: theory and methods. *International Statistical Review*, 2025.

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*, pp. 7850–7860, 2021.

Sloan Nietert, Ziv Goldfeld, and Rachel Cummings. Outlier-robust optimal transport: Duality, structure, and statistical analysis. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 11691–11719, 2022.

Giovanni Puccetti, Ludger Rüschendorf, and Steven Vanduffel. On the computation of Wasserstein barycenters. *Journal of Multivariate Analysis*, 176:104581, 2020.

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*, pp. 41867–41885, 2024.

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

Thibault Séjourné, Gabriel Peyré, and François-Xavier Vialard. Unbalanced optimal transport, from theory to numerics. *Handbook of Numerical Analysis*, 24:407–471, 2023.

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):1–11, 2015.

Yuki Takezawa, Ryoma Sato, Zornitsa Kozareva, Sujith Ravi, and Makoto Yamada. Fixed support tree-sliced Wasserstein barycenter. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 1120–1137, 2022.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, pp. 1–34, 2024.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

Xu Wang, Jiawei Huang, Qingyuan Yang, and Jinpeng Zhang. On robust Wasserstein barycenter: The model and algorithm. In *Proceedings of the 24th SIAM International Conference on Data Mining*, pp. 235–243, 2024.

Qingyuan Yang and Hu Ding. Approximate algorithms for k-sparse wasserstein barycenter with outliers. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, pp. 5316–5325, 2024.

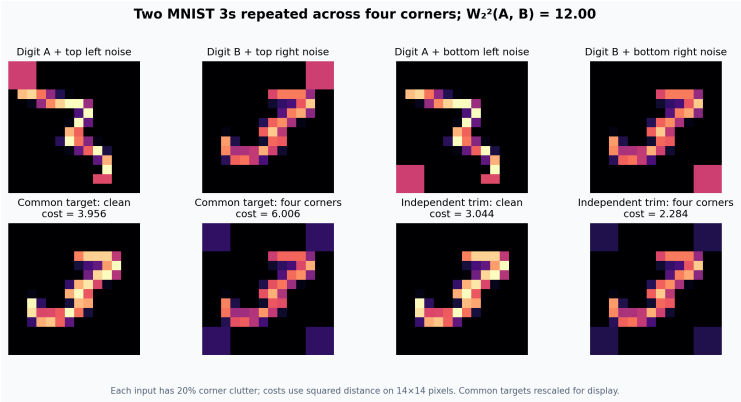


Figure 3: In this figure, we highlight on real MNIST data how the addition of noise can still result in noise appearing in the resulting robust Wasserstein barycenter under the first approach. On the top row are the input distributions with 20% noise in each corner. On the bottom row are the two proposed robust barycenter distributions.

A Additional Motivating Examples

We dedicate this section to briefly discuss two more motivating examples for the partial Wasserstein barycenter formulation.

As discussed in Section 1, when several observations contain shared structure and observation-specific noise, we want a representative that every observation helps explain. This arises when combining aligned point-cloud scans with different background clutter or sensor measurements with different artifacts. If each input may transport to a different portion of a probability barycenter, that barycenter can retain clutter from one observation because the others simply ignore it. A partial Wasserstein barycenter avoids this ambiguity by allowing each input to discard some mass while requiring all retained mass to transport to the same measure.

To see why the common measure matters, consider three uniform distributions on $\{a, b\}$, $\{b, c\}$, and $\{c, a\}$. Suppose each input transports $\alpha = \frac{2}{3}$ of its mass. A formulation that lets each input use a different portion of a probability barycenter assigns zero cost to the uniform measure on $\{a, b, c\}$: each input transports to its two support points and ignores the third. But no point is present in all three inputs. In the partial barycenter formulation, every input must transport to the entire same measure of mass $\frac{2}{3}$. Since no location is shared by all three inputs, every solution has a positive cost. We visualize this example in Figure 4.

To motivate this phenomenon with real data, we constructed four observations from two distinct MNIST images of the digit 3, A and B. The images have a relatively large squared optimal transport cost of 11.995 on a 14×14 pixel grid. Each observation contains 80% digit mass and 20% noise in one corner; the noise occupies a different corner in each observation. The constructed noisy MNIST images and the candidate robust barycenter distributions can be seen in Figure 3.

We compared a clean candidate based on B with a candidate that places 5% mass in each corner. Under our partial barycenter formulation, every observation must transport to the same entire target of mass 0.8. The clean candidate costs 3.956, while the four-corner candidate costs 6.006. Under the alternative formulation, each observation can use a different 0.8-mass portion of a probability target. The ranking reverses: the four-corner candidate costs 2.284, while the clean candidate costs 3.044. Each observation can use its own corner and ignore the other three. Feasible transport plans and LP dual bounds certify both comparisons. These are comparisons between the two candidates, not claims that either is globally optimal.

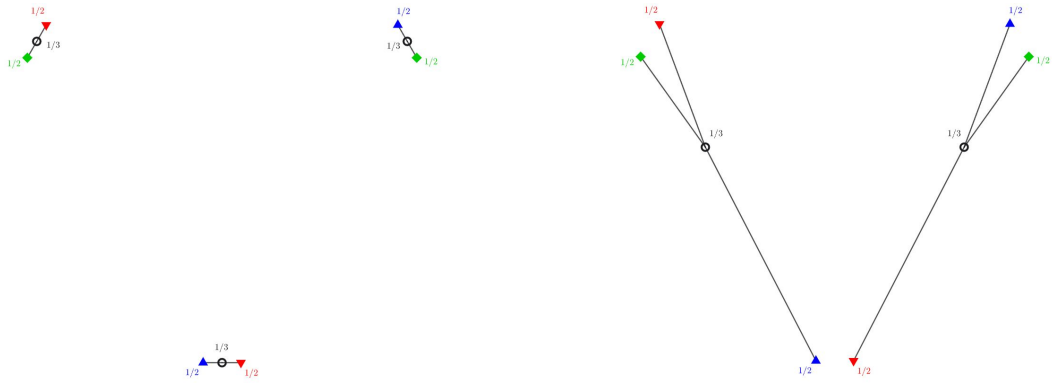


Figure 4: (Left) The unbalanced transport plans between the (almost) collocated points and the barycenter under the robust OT metric are independent of each other, resulting in a very low cost. (Right) The partial Wasserstein barycenter successfully highlights that the three input distributions are distinct.

B Analysis of the Algorithm from Section 2

In this appendix, we provide the proofs of Lemmas 1 and 3. The proof of Lemma 1 is split into Lemmas 5 and 6—the former proves the lower bound and the latter proves the upper bound.

Lemma 5.

$$\frac{1}{T}\sum_{t\geq 1}\frac{\lambda^t}{\Psi^t}\cdot\left(\frac{\psi_0^t}{g}\cdot\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t))\right)\geq(1-\eta)-\frac{c\ln n}{T\eta}.$$

Proof. We prove the lemma using Theorem 2.5 in Arora et al. (2012). To formulate the left hand side in the terminology of Arora et al. (2012), we define the following:

$$m_0^t=\frac{\lambda^t\cdot\mathsf{d}^p(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t))}{g}\qquad m_i^t(x)=\begin{cases}0, & \text{if } x\neq x_i^t,\\ \frac{\lambda^t}{\mu_i(x_i^t)}, & \text{if } x=x_i^t\end{cases}\qquad\forall x\in X_i, i\in[k],$$

$$p_0^t=\frac{\psi_0^t}{\Psi^t}\qquad p_i^t(x)=\frac{\psi_i^t(x)}{\Psi^t}\qquad\forall x\in X_i, i\in[k].$$

By definition, $m_0^t,p_0^t,m_i^t(x),p_i^t(x)\in[0,1]$. Set $\boldsymbol{m}^t=\langle m_0^t,m_i^t(x)|_{1\leq i\leq k,x\in X_i}\rangle$ and $\boldsymbol{p}^t=\langle p_0^t,p_i^t(x)|_{1\leq i\leq k,x\in X_i}\rangle$. Then since $\ell^t(\boldsymbol{x},y)=\mathsf{d}^p(\boldsymbol{x},y)+\sum_{i=1}^k\frac{\psi_i^t(x_i)}{\mu_i(x_i)}\cdot\frac{g}{\psi_0^t}$ for all $\boldsymbol{x}=(x_1,\ldots,x_k)\in\mathbb{X}$ and $y\in Y$, a simple algebraic manipulation implies that

$$\sum_{t\geq 1}\frac{\lambda^t}{\Psi^t}\left(\frac{\psi_0^t}{g}\cdot\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t))\right)=\sum_{t\geq 1}\boldsymbol{m}^t\cdot\boldsymbol{p}^t.\tag{2}$$

By applying Theorem 2.5 in Arora et al. (2012) and using the fact that $m_0^t,m_i^t(x)\in[0,1]$, we obtain

$$\sum_{t\geq 1}\boldsymbol{m}^t\cdot\boldsymbol{p}^t\geq(1-\eta)\max\left\{\sum_{t\geq 1}m_0^t,\max_{1\leq k,x\in X_i}\sum_{t\geq 1}m_i^t(x)\right\}-\frac{\ln(n+1)}{\eta}.\tag{3}$$

Let $\boldsymbol{f}=\langle f_1,\ldots,f_k\rangle$ be the final flow computed by the algorithm. Then

$$\sum_{t\geq 1}m_0^t=\frac{\mathfrak{C}_p(\boldsymbol{f})}{g}\leq T\text{ and }\sum_{t\geq 1}m_i^t(x)=\frac{f_i(x,Y)}{\mu_i(x)}\leq T\text{ for all }1\leq i\leq k, x\in X_i\tag{4}$$

by the termination condition of the algorithm. The lemma now follows immediately by combining (2), (3) and (4). □

Lemma 6.

$$\frac{1}{T}\sum_{t\geq 1}\frac{\lambda^t}{\Psi^t}\cdot\left(\frac{\psi_0^t}{g}\cdot\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t))\right)\leq\frac{\nu(Y)}{\nu_g^*(Y)}.$$

Proof. Let $\boldsymbol{f}^*=\langle f_1^*,\ldots,f_k^*\rangle$ be the optimal flow function of the decision problem (corresponding to the measure ν_g^*). Since $\boldsymbol{f}^*$ is a feasible flow, the LP constraints (cf. Figure 2(i)) imply that

$$\frac{\text{¢}_p(\boldsymbol{f}^*)}{g}\leq 1\quad\text{ and }\quad\frac{f_i^*(x,Y)}{\mu_i(x)}\leq 1\text{ for all }1\leq i\leq k, x\in X_i. \tag{5}$$

Hence, we can rewrite Ψ^t as:

$$\begin{aligned}\Psi^t&=\psi_0^t+\sum_{i=1}^k\sum_{x\in X_i}\psi_i^t(x)\\&\geq\psi_0^t\cdot\frac{\text{¢}_p(\boldsymbol{f}^*)}{g}+\sum_{i=1}^k\sum_{x\in X_i}\psi_i^t(x)\cdot\frac{f_i^*(x,Y)}{\mu_i(x)} &(6)\\&=\frac{\psi_0^t}{g}\sum_{i=1}^k\sum_{x\in X_i,y\in Y}f_i^*(x,y)\left(\mathsf{d}^p(x,y)+\frac{\psi_i^t(x)}{\mu_i(x)}\cdot\frac{g}{\psi_0^t}\right)\\&=\frac{\psi_0^t}{g}\sum_{i=1}^k\sum_{x\in X_i,y\in Y}f_i^*(x,y)\cdot\ell_i^t(x,y), &(7)\end{aligned}$$

where (6) follows from (5) and (7) follows from the definition of ℓ_i^t. Recall that in each iteration we choose $\boldsymbol{x}^t$ to satisfy

$$\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t))=\min_{\boldsymbol{x}\in\mathbb{X},y\in Y}\ell^t(\boldsymbol{x},y).$$

Therefore, the cheapest way to transport $\nu_g^*(Y)$ mass from all input distributions to Y, ignoring the demand constraint μ_i, such that the same amount of flow is sent from all input distributions to each point $y\in Y$, is to set

$$f_i(x,y)=\begin{cases}\nu_g^*(Y), & \text{if } x=x_i^t, y=\mathsf{ctr}(\boldsymbol{x}^t),\\ 0, & \text{otherwise.}\end{cases}$$

Hence, we conclude that

$$\sum_{i=1}^k\sum_{x\in X_i,y\in Y}f_i^*(x,y)\cdot\ell_i^t(x,y)\geq\nu_g^*(Y)\cdot\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t)), \tag{8}$$

and therefore

$$\Psi^t\geq\frac{\psi_0^t}{g}\cdot\nu_g^*(Y)\cdot\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t)). \tag{9}$$

We remark that (8) is the only place where we use the fact that $\boldsymbol{x}^t=\arg\min_{\boldsymbol{x}\in\mathbb{X}}\ell^t(\boldsymbol{x},\mathsf{ctr}(\boldsymbol{x}))$ in each iteration. Substituting (9) into the left-hand side of the lemma statement, we obtain

$$\sum_{t\geq 1}\frac{\lambda^t}{\Psi^t}\cdot\left(\frac{\psi_0^t}{g}\cdot\ell^t(\boldsymbol{x}^t,\mathsf{ctr}(\boldsymbol{x}^t))\right)\leq\sum_{t\geq 1}\frac{\lambda^t}{\nu_g^*(Y)}=\frac{\sum_{t\geq 1}\lambda^t}{\nu_g^*(Y)}=\frac{T\cdot\nu(Y)}{\nu_g^*(Y)}.$$

This completes the proof of the lemma. □

Lemmas 5 and 6 together prove Lemma 1. Next, we prove Lemma 3.

Lemma 3. *The decision algorithm terminates within $O(nT)=O(n\delta^{-2}\log n)$ steps.*

Proof. In every step of the algorithm, we increase $f_i(x_i^t,\mathsf{ctr}(\boldsymbol{x}^t))$ by λ^t, defined as the maximum amount to increase the flows until either (a) one of the points x_i^t has an outgoing flow equal to the next integer multiple of its demand, or (b) the total cost of the flow reaches the next integer multiple of the guess g—see (1) and the definition of λ^t thereafter. The algorithm terminates precisely when the first one of these terms hits a magnitude of T times its LP constraint, i.e., either a point $x\in X_i$ for some $1\leq i\leq k$ satisfies $f_i(x,Y)=T\cdot\mu_i(x)$ or $\text{¢}_p(\boldsymbol{f})=T\cdot g$. This must happen within at most $(n+1)T$ iterations. □

C Efficient Implementation and its Analysis

We now describe an efficient implementation of the algorithm described in Section 2. As mentioned in Section 2, we approximate the function $\ell^t : \mathbb{X} \times Y \to \mathbb{R}_{\geq 0}$ with a function $\underline{\ell}^t : \mathbb{X} \times Y \to \mathbb{R}_{\geq 0}$ as follows. Instead of using ψ_0^t and ψ_i^t, which are updated frequently, we use $\underline{\psi_0^t}$ and $\underline{\psi_i^t}$ that are *lazy* versions of ψ_0^t and ψ_i^t and updated much less frequently, as described below. For $1 \leq i \leq k$ and $x \in X_i, y \in Y$, we define

$$\underline{\ell_i^t}(x, y) = \mathsf{d}^p(x, y) + \frac{\underline{\psi_i^t}(x)}{\mu_i(x)} \cdot \frac{g}{\underline{\psi_0^t}}$$

and for $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathbb{X}$, we set $\underline{\ell}^t(\boldsymbol{x}, y) = \sum_{i=1}^k \underline{\ell_i^t}(x_i, y)$. Note that updating $\underline{\psi_i^t}(x)$ only affects $\underline{\ell_i^t}(x, y)$ for all $y \in Y$ but updating $\underline{\psi_0^t}$ affects $\underline{\ell_i^t}(x, y)$ for all $i \leq k$ and pairs $(x, y) \in X_i \times Y$. In each iteration, we update only one of the weights, and each of them is updated at most T times; see below for details.

For each $i \leq k$ and $y \in Y$, we maintain a heap $H_{i,y}$ that stores X_i and uses $\underline{\ell_i^t}(x, y)$ as the priority of $x \in X_i$ at time t. Let $\overline{x}_{i,y}^t = \arg\min_{x \in X_i} \underline{\ell_i^t}(x, y)$ be the minimum priority element of X_i with respect to y in the heap $H_{i,y}$. For each $y \in Y$, let $\overline{\boldsymbol{x}}_y^t = (\overline{x}_{1,y}^t, \ldots, \overline{x}_{k,y}^t)$. We maintain a global priority queue H on Y where the priority of $y \in Y$ is $\underline{\ell}^t(\overline{\boldsymbol{x}}_y^t, y)$. Set $\overline{y}^t = \arg\min_{y \in Y} \underline{\ell}^t(\overline{\boldsymbol{x}}_y^t, y)$ to be the minimum priority element and let $\underline{\boldsymbol{x}}^t = \overline{\boldsymbol{x}}_{\overline{y}^t}^t$. Then

$$\underline{\boldsymbol{x}}^t = \arg \min_{\boldsymbol{x} \in \mathbb{X}, y \in Y} \underline{\ell}^t(\boldsymbol{x}, y).$$

We now describe how to proceed at each step. Initially, for $t = 1$, we set $\underline{\psi_0^t} = \psi_0^t$ and $\underline{\psi_i^t}(x) = \psi_i^t(x)$ for all $i \leq k$ and $x \in X_i$. At each time t, the above data structure returns $\underline{\boldsymbol{x}}^t$. We set $\boldsymbol{x}^t = \underline{\boldsymbol{x}}^t$ and proceed as described in Section 2, i.e., compute λ^t and update the flow function and weights. In addition, we do the following. Let $\boldsymbol{x}^t = (x_1^t, \ldots, x_k^t)$.

(i) If $\lambda^t = \lambda_i^t$ for some $i \geq 1$, then after updating $\psi_i^{t+1}(x_i^t)$, we set $\underline{\psi_i^{t+1}}(x_i^t) = \psi_i^{t+1}(x_i^t)$. We do not update $\underline{\psi_i^{t+1}}(x)$ for any other $x \in X_i \setminus \{x_i^t\}$ or $\underline{\psi_j^{t+1}}$ for $j \neq i$. Since $\underline{\psi_i^{t+1}}(x_i^t)$ has changed, we update the priority of x_i^t in $H_{i,y}$ for all $y \in Y$. If this step also changes $\underline{\ell}^t(\overline{\boldsymbol{x}}_y^t, y)$ for some $y \in Y$, then we update the priority of each such y in the global priority queue H. This step takes $O(m \log(n + m))$ time.

(ii) If $\lambda^t = \lambda_0^t$, then after updating ψ_0^{t+1}, we set $\underline{\psi_0^{t+1}} = \psi_0^{t+1}$. We reconstruct all priority queues $H_{i,y}$ for all $1 \leq i \leq k$ and $y \in Y$, as well as the global priority queue H. This step takes $O(mn)$ time.

We remark that for each x_i in X, the weight $\underline{\psi_i^t}(x_i)$ is updated at most $T = O(\eta^{-2} \log n)$ many times, precisely when the total flow from x_i reaches the next integer multiple of $\mu_i(x_i)$. In an analogous manner, if $\lambda_0^t = \lambda^t$, then after updating the value of ψ_0^{t+1} we update $\underline{\psi_0^{t+1}} \leftarrow \psi_0^{t+1}$. Then similarly $\underline{\psi_0^t}$ is updated at most $T = O(\eta^{-2} \log n)$ many times. Hence, the total time spent in updating the data structure over all iterations is $O(mn \cdot \eta^{-2} \log^2(m + n))$.

We now argue that Lemma 2 still holds. As mentioned above, the only place in the analysis where we used the fact that the algorithm chooses $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x} \in \mathbb{X}, y \in Y} \ell^t(\boldsymbol{x}, y)$ is in the proof of Lemma 6. We first prove that $\underline{\psi_0^t}$ and $\underline{\psi_i^t}$ are approximations of ψ_0^t and ψ_i^t, respectively.

Lemma 7. *For any $t \geq 1$,*

$$\underline{\psi_i^t}(x) \leq \psi_i^t(x) \leq (1 + 2\eta) \cdot \underline{\psi_i^t}(x) \;\; \forall i \leq k, x \in X_i \quad \textit{and} \quad \underline{\psi_0^t} \leq \psi_0^t \leq (1 + 2\eta) \cdot \underline{\psi_0^t}.$$

Proof. We note that $\underline{\psi_i^t}(x_i) \leq \psi_i^t(x_i)$ is always true since $\psi_i^{t+1}(x_i) \geq \psi_i^t(x_i)$ for all $i \leq k$, $x_i \in X_i$ and iterations t. The analogous observation allows us to conclude $\underline{\psi_0^t} \leq \psi_0^t$. It suffices to prove the upper bounds.

Let $i \leq k$ be an arbitrary index. We will assume $i \neq 0$; the same argument will extend to $i = 0$. Let $x_i \in X_i$ be arbitrarily chosen, and let $t > 0$ be an arbitrary iteration number. If $\underline{\psi_i^t}(x_i) = \psi_i^t(x_i)$, then we are done. Assume not, and hence $\underline{\psi_i^t}(x_i) < \psi_i^t(x_i)$. We note that the update $\underline{\psi_i^{t+1}}(x_i) \leftarrow \psi_i^{t+1}(x_i)$ happens precisely when $\lambda^t = \lambda_i^t$ by definition. And by definition of λ^t and λ_i^t, this event occurs when $f_i(x_i, Y)$ increases to the next integer multiple of $\mu_i(x_i)$. Let $t_0 < t$ be the most recent time before iteration t that $\underline{\psi_i^{t_0}}(x_i) = \psi_i^{t_0}(x_i)$, and let $t_0 \leq t_1, \ldots, t_r \leq t$ be the iterations such that $x_i = x_i^{t_j}$ for all $1 \leq j \leq r$, i.e., the iterations when x_i is the point from the support of distribution i selected in the minimizing k-tuple. Then we observe that

$$0 \leq \sum_{j=1}^{r} \lambda^{t_j} \leq \mu_i(x_i).$$

Moreover, by definition of the multiplicative weight update rule in the algorithm, we have

$$\frac{\psi_i^t(x_i)}{\underline{\psi_i^t}(x_i)} = \prod_{j=1}^{r} \left(1 + \eta \frac{\lambda^{t_j}}{\mu_i(x_i)}\right).$$

Using $\eta \leq 1$ and $\sum_{j=1}^{r} \frac{\lambda^{t_j}}{\mu_i(x_i)} \leq 1$, we deduce

$$\prod_{j=1}^{r} \left(1 + \eta \frac{\lambda^{t_j}}{\mu_i(x_i)}\right) \leq \exp\left(\eta \cdot \sum_{j=1}^{r} \frac{\lambda^{t_j}}{\mu_i(x_i)}\right) \leq \exp(\eta) \leq (1 + 2\eta).$$

□

We now state a slightly weaker version of Lemma 1.

Lemma 8. *There exists a constant $0 < c \leq 2$ such that*

$$(1 - \eta) - \frac{c \ln n}{T\eta} \leq \frac{1}{T} \sum_{t \geq 1} \frac{\lambda^t}{\Psi^t} \cdot \left(\frac{\psi_0^t}{g} \cdot \ell^t(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))\right) \leq (1 + 2\eta) \cdot \frac{\nu(Y)}{\nu_g^*(Y)}.$$

Proof. Since the proof of Lemma 5 does not use the fact that $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x} \in \mathbb{X}, y \in Y} \ell^t(\boldsymbol{x}, \mathsf{ctr}(\boldsymbol{x}))$, Lemma 5 still applies and the lower bound still holds. We prove the upper bound by following the same argument as in 6 and use Lemma 7 to write ψ_i^t in terms of $\underline{\psi_i^t}$ as follows:

$$\begin{aligned}
\frac{\psi_0^t}{g} \cdot \ell^t(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t)) &= \psi_0^t \cdot \frac{\mathsf{d}^p(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))}{g} + \sum_{i=1}^{k} \frac{\psi_i^t(x_i^t)}{\mu_i(x_i^t)} \\
&\leq (1 + 2\eta) \cdot \left(\underline{\psi_0^t} \cdot \frac{\mathsf{d}^p(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))}{g} + \sum_{i=1}^{k} \frac{\underline{\psi_i^t}(x_i^t)}{\mu_i(x_i^t)}\right) \\
&\leq (1 + 2\eta) \cdot \left(\frac{\underline{\psi_0^t}}{g} \cdot \underline{\ell}^t(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))\right)
\end{aligned}$$

and

$$\begin{aligned}
\Psi^t &= \psi_0^t + \sum_{i=1}^{k} \sum_{x \in X_i} \psi_i^t(x) \\
&\geq \underline{\psi_0^t} + \sum_{i=1}^{k} \sum_{x \in X_i} \underline{\psi_i^t}(x) \\
&\geq \underline{\psi_0^t} \cdot \frac{\boldsymbol{\mathfrak{c}}_p(\boldsymbol{f}^*)}{g} + \sum_{i=1}^{k} \sum_{x \in X_i} \underline{\psi_i^t}(x) \cdot \frac{f_i^*(x, Y)}{\mu_i(x)} \\
&= \frac{\underline{\psi_0^t}}{g} \sum_{i=1}^{k} \sum_{x \in X_i, y \in Y} f_i^*(x, y) \left(\mathsf{d}^p(x, y) + \frac{\underline{\psi_i^t}(x)}{\mu_i(x)} \cdot \frac{g}{\underline{\psi_0^t}}\right)
\end{aligned}$$

$$= \frac{\underline{\psi}_0^t}{g} \sum_{i=1}^{k} \sum_{x \in X_i, y \in Y} f_i^*(x,y) \cdot \underline{\ell}_i^t(x,y).$$

That is, substituting $\underline{\psi}_i^t$, $\underline{\psi}_0^t$ and $\underline{\ell}^t$ in place of ψ_i^t, ψ_0^t and ℓ^t only incurs an extra $(1+2\eta)$ factor. Now the proof follows by the same argument as in the proof of Lemma 6. □

Finally, substituting the values of η and T and choosing the constant sufficiently large in the value of T, we obtain

$$\begin{aligned}\nu(Y) &\geq \frac{1}{1+\frac{\delta}{2}} \cdot \left(1 - \frac{\delta}{4} - \frac{\delta}{8}\right) \cdot \nu_g^*(Y) \\ &\geq \frac{1}{1+\frac{\delta}{2}} \cdot \left(1 - \frac{3\delta}{8}\right) \cdot \nu_g^*(Y) \\ &\geq (1-\delta) \cdot \nu_g^*(Y).\end{aligned}$$

Hence, Lemma 2 still holds. Putting everything together, we obtain Theorem 1.

D Choosing a Good Fixed Support for the Free-Support PWB Problem

In this section, we reduce the free-support partial barycenter problem to the fixed-support problem at the cost of a small relative approximation error. Let (X_i, μ_i), $1 \leq i \leq k$, be k discrete probability distributions in a metric space (Ω, d). The following lemma suggests that one can efficiently construct a candidate fixed support set Y for a given instance of the free-support partial barycenter problem.

Lemma 9. *Let $(X_1, \mu_1), \ldots, (X_k, \mu_k)$ be k discrete probability distributions in a metric space (Ω, d), let $X = \bigcup_{i=1}^{k} X_i$, and let $\alpha \in (0,1]$ be a parameter. Let ν^* and $\boldsymbol{f}^*$ denote the optimal α-partial barycenter and unbalanced transport plans of $(X, \boldsymbol{\mu})$. Then:*

1. *there exists a measure ν, with support set X and $\nu(X) = \alpha$, and a set of unbalanced transport plans $\boldsymbol{f}$ from the input distributions to ν such that $w_p(\boldsymbol{f}) \leq 2 \cdot w_p(\boldsymbol{f}^*)$.*
2. *if (Ω, d) is d-dimensional Euclidean space, then for any $\varepsilon > 0$, a set Y of size $|Y| = n\varepsilon^{-d} \log \Delta$ can be computed in $O(n\varepsilon^{-d} \log \Delta)$ time such that there exist a measure ν over Y, where $\nu(Y) = \alpha$, and a set of unbalanced transport plans $\boldsymbol{f}$ from the input distributions to ν such that $w_p(\boldsymbol{f}) \leq (1+\varepsilon) \cdot w_p(\boldsymbol{f}^*)$.*

Proof. The first claim follows from the triangle inequality. Let ν^* be an optimal partial Wasserstein barycenter and let y be an element of the support of ν^*. Then there exist $x_1, \ldots, x_k$ such that $x_i \in Y_i$ and $f_i(x_i, y) > 0$ for each $i \leq k$. Let $x = \arg\min_{i \in [k]} \mathsf{d}(x_i, y)$. Then we note that $x \in X$, and additionally

$$\mathsf{d}(x_i, x) \leq \mathsf{d}(x_i, y) + \mathsf{d}(x, y) \leq 2 \cdot \mathsf{d}(x_i, y)$$

for all $i \leq k$ since x was chosen to be the minimum distance point from y. From this observation, we deduce

$$\sum_{i=1}^{k} \mathsf{d}^p(x_i, x) \leq \sum_{i=1}^{k} (2 \cdot \mathsf{d}(x_i, x))^p = 2^p \cdot \sum_{i=1}^{k} \mathsf{d}^p(x_i, x).$$

The second claim is an immediate consequence of Lemma 3.1 in Agarwal et al. (2024). □

E Analysis of the Congestion-Reducing Procedure in Section 3

Before diving into the analysis of the faster algorithm, we give a brief high-level overview of the difference between the algorithms of Sections 2 and 3.

Overview of the algorithm. As described in Section 2, the computational bottleneck of the algorithm in each iteration is the requirement to maintain a (approximate) minimum cost k-tuple. The reason why this step requires $\Omega(m)$ time currently is because when the weight of a point is updated, we must also update priority queues for every $y \in Y$. To mitigate this cost, we choose to compress (x, y) pairs of comparable costs for $x \in X_i, y \in Y$ using well-separated pair decompositions (WSPDs); the WSPD clusters pairs with similar pairwise distances. However, when formulating the packing LP as a flow over the WSPD to compress the total number of pairs, it is hard to extract the flow from each individual vertex while also preserving its demand constraints. For this reason, we defined the packing LP in Figure 2(ii), which relaxes the demand constraints on vertices and expresses them as capacity constraints on WSPD pairs. WSPDs are constructed so that each vertex has bounded degree ρ (see e.g. Callahan & Kosaraju (1995); Andoni & Indyk (2006); Agarwal et al. (2026), and therefore replacing vertex demands with edge capacities only distorts the demand feasibility conditions of the packing LP by a factor of ρ.

We use a multiplicative weight update algorithm to push the infeasibility of the LP from a factor of ρ to a factor of $(1 + \delta)$, which we called the *congestion-reducing algorithm*. In each iteration of the congestion-reducing algorithm, we solve a weighted version of the relaxed packing LP, which we describe in Figure 2(ii) and prove to be a relaxation of the original packing LP in a straightforward manner.

Lemma 10. *Any feasible solution to the LP of Figure 2(i) is also a feasible solution to the LP of Figure 2(ii).*

Proof sketch. We can write any unbalanced transport plan as the flow over the WSPD pairs. It suffices to show this is feasible. □

Notably, because the structure of the packing LP in Figure 2(ii) for a fixed guess g and set of weights $\boldsymbol{\phi}^t$ is nearly identical to the structure of the packing LP in Figure 2(i), the same multiplicative weight update algorithm from Section 2 can be used to compute its approximate solution in each iteration of the congestion-reducing procedure. We need to prove, however, that the use of the WSPD and capacity constraints in place of vertex demands improves the runtime of the simple algorithm in Section 2 by a significant factor.

Lemma 11. *A δ-approximate solution to the LP from Figure 2(ii), i.e. a feasible measure ν and flows $\boldsymbol{f}$ such that $\nu(Y) \in [\nu^*(Y) - \delta, \nu^*(Y)]$, can be computed in $O\left((n+m)(\rho+k)\left(\frac{\log(n+m)}{\delta}\right)^2\right)$ time.*

Appendix F is devoted to prove Lemma 11. Lemmas 10 and 11 imply that the measure ν the algorithm computes has a total mass of at least $\nu_g^*(Y) - \delta$.

Analysis of the algorithm. Since the congestion-reducing procedure runs for a total of $\sigma = O(\rho\eta^{-2}\log(nk))$ many iterations. Lemma 11 implies that the algorithm runs in $(n+m)\rho(\rho+k)\delta^{-4}\log^{O(1)}\Delta$ time.

For a collection of weights $\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_k \rangle$ and flows $\boldsymbol{f} = \langle f_1, \ldots, f_k \rangle$, let $\Phi(\boldsymbol{\phi}, \boldsymbol{f}) = \sum_{i=1}^{k} \sum_{x \in X_i, y \in Y} \phi_i(x) \cdot f_i(x, y)$ and $\Phi(\boldsymbol{\phi}) = \sum_{i=1}^{k} \sum_{x \in X_i} \phi_i(x) \cdot \mu_i(x)$. If $\boldsymbol{f}$ is a collection of transport plans, $\Phi(\boldsymbol{\phi}, \boldsymbol{f}) = \Phi(\boldsymbol{\phi})$. Throughout the algorithm, we maintained $\Phi(\boldsymbol{\phi}^t) = \delta^{-1}g$ for all iterations t. We will additionally define $\rho_i^t(x) = \frac{f_i^t(x,Y)}{\mu_i(x)}$ to be the congestion of any fixed $x \in X_i$ for each distribution $i \le k$ and iteration $t \ge 1$. Note that $\rho_i^t(x) \le \rho$ for all $i \le k$, $t \ge 1$ and $x \in X_i$.

We now prove Lemma 4. We first prove an analog of Lemma 5, namely, we prove that the congestion of any fixed point is comparable to the portion of the weighted costs contributed by the weights. The congestion bound will be applied to the analysis to prove Lemma 4.

Lemma 12. *Suppose $\eta \in (0, \frac{1}{2}]$. Then for any $1 \le j \le k$ and $z \in X_j$, we have*

$$\frac{\delta}{g}\sum_{t=1}^{\sigma}\Phi(\boldsymbol{\phi}^t, \boldsymbol{f}^t) \ge (1-\eta)\sum_{t=1}^{\sigma}\rho_j^t(z) - \frac{\rho\ln(nk)}{\eta}.$$

Proof. Analogous to the proof of Lemma 5, we prove the lemma using Theorem 2.5 in Arora et al. (2012). To formulate the left hand side in the terminology of Arora et al. (2012), we define the following:

$$m_i^t(x) = \frac{f_i^t(x,Y)}{\rho \cdot \mu_i(x)} = \frac{\rho_i^t(x)}{\rho}, \qquad \forall x \in X_i, i \in [k],$$

$$p_i^t(x) = \frac{\phi_i^t(x) \cdot \mu_i(x)}{\text{Þ}(\boldsymbol{\phi}^t)} \qquad \forall x \in X_i, i \in [k].$$

By definition, $m_0^t, p_0^t, m_i^t(x), p_i^t(x) \in [0,1]$. Set $\boldsymbol{m}^t = \langle m_i^t(x)|_{1\le i\le k, x\in X_i}\rangle$ and $\boldsymbol{p}^t = \langle p_i^t(x)|_{1\le i\le k, x\in X_i}\rangle$. Then a simple algebraic manipulation implies that

$$\begin{aligned}\sum_{t\ge 1} \boldsymbol{m}^t \cdot \boldsymbol{p}^t &= \sum_{t\ge 1}\sum_{i\le k}\sum_{x\in X_i} \frac{\rho_i^t(x)}{\rho} \cdot \frac{\phi_i^t(x)\cdot\mu_i(x)}{\text{Þ}(\boldsymbol{\phi}^t)} \\ &= \frac{\delta}{g\rho}\sum_{t\ge 1}\sum_{i\le k}\sum_{x\in X_i} f_i^t(x,Y)\cdot\phi_i^t(x) \\ &= \frac{\delta}{g\rho}\sum_{t\ge 1}\text{Þ}(\boldsymbol{\phi}^t,\boldsymbol{f}^t). \end{aligned} \tag{10}$$

Since $m_0^t, m_i^t(x) \in [0,1]$, Theorem 2.5 in Arora et al. (2012) implies that

$$\sum_{t\ge 1}\boldsymbol{m}^t\cdot\boldsymbol{p}^t \ge (1-\eta)\max_{1\le k, x\in X_i}\sum_{t\ge 1} m_i^t(x) - \frac{\ln(n+1)}{\eta}. \tag{11}$$

The lemma then follows from substituting the definition of m_i^t and combining 10 and 11. □

We now prove Lemma 4, the technical lemma, which we restate here.

Lemma 4. *If the congestion-reducing MWU algorithm returns a flow $\boldsymbol{f}$ and a measure ν, then $f_i(x,Y) \le (1+\delta)\cdot\mu_i(x)$ for all $i \le k$ and $x \in X_i$.*

Proof. Since each iteration of the middle level MWU algorithm obtains a flow and does not terminate early, we must have

$$\text{Þ}(\boldsymbol{\phi}^t,\boldsymbol{f}^t) \le \frac{g}{\delta}$$

for all iterations t by the constraint of any feasible solution to the LP of Figure 2(ii). Then taking the average over t implies

$$\frac{1}{\sigma}\cdot\frac{\delta}{g}\sum_t \text{Þ}(\boldsymbol{\phi}^t,\boldsymbol{f}^t) \le 1.$$

Using the bound from Lemma 12, we obtain

$$\frac{1-\eta}{\sigma}\sum_t \rho_j^t(z) - \frac{\rho\ln(nk)}{\sigma\eta} \le 1$$

for any choice of $1 \le j \le k$ and $z \in X_j$. Since

$$f_j(z,Y) = \frac{1}{\sigma}\sum_{t=1}^{\sigma} f_j^t(z,Y) = \frac{\mu_j(z)}{\sigma}\sum_{t=1}^{\sigma}\rho_j^t(z),$$

we obtain

$$\frac{f_j(z,Y)}{\mu_j(z)} = \frac{1}{1-\eta}\left[1 + \frac{\rho\ln(nk)}{\sigma\eta}\right] \le 1+\delta$$

because $\eta = O(\delta)$ and $\sigma = \Omega\left(\rho\eta^{-2}\ln(nk)\right)$. □

Combining Lemma 4 with Lemma 11, the observation that $\text{¢}_p(\boldsymbol{f}) \le g$ and binary search over g, we obtain Theorem 2.

F Efficient Implementation of the Algorithm for the Packing LP in Figure 2(ii)

We now describe an efficient implementation of the algorithm for approximately solving the packing LP from Figure 2(ii). The problem assumes a fixed collection of weights $\boldsymbol{\phi} = \langle \phi_1, \ldots, \phi_k \rangle$ and pushes the vertex demands to edge capacities. We re-derive the packing LP here, then describe an efficient algorithm for it, and finally prove its efficiency.

The Packing LP. For each $i \le k$, we construct a pair decomposition $\Pi_i = \{(A_j, B_j)\}$ of pairs $A_j \subseteq X_i$ and $B_j \subseteq Y$ that satisfies the following:

1. for each $(A_j, B_j) \in \Pi_i$ and for every $x_1, x_2 \in A_j$ and $y_1, y_2 \in B_j$,
$$\frac{\mathsf{d}^p(x_1, y_1)}{1+\varepsilon} \le \mathsf{d}^p(x_2, y_2) \le (1+\varepsilon) \cdot \mathsf{d}^p(x_1, y_1),$$
2. for each $(A_j, B_j) \in \Pi_i$ and every $x_1, x_2 \in A_j$,
$$\frac{\phi_i(x_1)}{1+\varepsilon} \le \phi_i(x_2) \le (1+\varepsilon) \cdot \phi_i(x_1).$$

Under this assumption on the pair decomposition, we are able to make the following claim about its stability under weight updates.

Lemma 13. *For every $i \le k$ and given real numbers $c_1, c_2 \ge 0$, define the function $\ell_i(x, y) = \mathsf{d}^p(x, y) + c_1 \cdot \phi_i(x) + c_2$. Then for any $(A_j, B_j) \in \Pi_i$ and any $x_1, x_2 \in X_i$, $y_1, y_2 \in Y$,*
$$\frac{\ell_i(x_1, y_1)}{1+\varepsilon} \le \ell_i(x_2, y_2) \le (1+\varepsilon) \cdot \ell_i(x_1, y_1).$$

Lemma 13 is sufficient to claim that under arbitrary weight updates performed by the MWU algorithm, every pair decomposition will remain a pair decomposition of the same quality. We now construct the pair decomposition in the following way. First, construct a (ε, ρ)-pair decomposition Π_i' between X_i and Y for each $i \le k$ with respect to $\mathsf{d}^p(\cdot, \cdot)$. Then, for each $(A_q', B_q') \in \Pi_i'$, we partition A_j into $\rho' = \varepsilon^{-O(1)} \log \Delta$ many sets $A_{q,1}, \ldots, A_{q,\rho'}$ where $A_{q,j} = \{x \in A_q : \phi_i(x) \in [\frac{\Delta^2}{2^j \delta}, \frac{\Delta^2}{2^{j-1}\delta})\}$. The new $(3\varepsilon, \rho\varepsilon^{-O(1)} \log \Delta)$-pair decomposition will be the set $\Pi_i = \{(A_{q,j}, B_q) : (A_q, B_q) \in \Pi_i'\}$.

For every $(A_j, B_j) \in \Pi_i$, we choose an arbitrary point of A_j, call it the *center* of A_j and denote it as $\widetilde{x}_j$. We set $\mu_i(\widetilde{x}_j) = \mu_i(A_j)$. Furthermore, for any pair $(x, y) \in A_j \times B_j$, we set its distance as $\mathsf{d}(\widetilde{x}_j, \widetilde{y}_j)$ where $\widetilde{y}_j$ is some arbitrary fixed point of B_j. Set $\widetilde{X}_i = \{\widetilde{x}_j : (A_j, B_j) \in \Pi_i\}$. Our goal is to compute a flow function $f_i \colon \widetilde{X}_i \times Y \to [0, 1]$ with the constraint that a mass can be transported from $\widetilde{x}_j$ only to the points of B_j, i.e., $f_i(\widetilde{x}_j, y) = 0$ for all $y \notin B_j$. We will let
$$\widetilde{X}_i(y) := \{\widetilde{x}_j : (A_j, B_j) \in \Pi_i \text{ and } y \in B_j\}$$
be the set of centers that can reach $y \in Y$ via some pair $(A_j, B_j) \in \Pi_i$ and define
$$\widetilde{\mathbb{X}}(y) = \prod_{i=1}^{k} \widetilde{X}_i(y)$$
to be the set of tuples that can reach y. Then let $\widetilde{\mathbb{W}} = \bigcup_{y \in Y} \left(\widetilde{\mathbb{X}}(y) \times \{y\}\right)$ denote the set of tuples and their reachable centroids from the candidate set Y.

$$\begin{aligned}
&\max \ \ \nu(Y) \\
\text{s.t. } &f_i(\widetilde{x}_j, B_j) \le \mu_i(\widetilde{x}_j) \ \ \forall i \le k, \pi_j = (A_j, B_j) \in \Pi_i, \\
&f_i(\widetilde{X}_i, y) = \nu(y) \ \ \forall i \le k, y \in Y, \\
&\mathfrak{C}_p(\boldsymbol{f}) \le g, \\
&\mathfrak{P}(\boldsymbol{\phi}, \boldsymbol{f}) \le \mathfrak{P}(\boldsymbol{\phi}), \\
&f_i, \nu \ge 0, \\
&f_i(\widetilde{x}_j, y) = 0 \ \ \forall i \le k, \widetilde{x}_j \notin \widetilde{X}_i(y).
\end{aligned}$$

Efficient Algorithm and Runtime Analysis. Similar to Appendix C, for all iterations $t \geq 1$ we will maintain weights ψ_*^t, $\psi_0^t \in \mathbb{R}_{\geq 0}$ and $\psi_i^t \colon \Pi_i \to \mathbb{R}_{\geq 0}$ that are updated in every iteration, and their lazily-updated counterparts $\underline{\psi}_*^t$, $\underline{\psi}_0^t$ and $\underline{\psi}_i^t$ which approximate the original weights. The new weights ψ_*^t and $\underline{\psi}_*^t$ will be used to capture how close the constraint $\text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}) \leq \text{Þ}(\boldsymbol{\phi})$ is from being satisfied. For each $i \leq k$ and $\pi \in \Pi_i$, the weight $\psi_i^t(\pi)$ captures how close the capacity of the WSPD pair is from being satisfied. The weights ψ_0^t and $\underline{\psi}_0^t$ are the same as in Section 2. We will additionally define $\Psi^t = \psi_0^t + \psi_*^t + \sum_{i=1}^k \sum_{\pi \in \Pi_i} \psi_i^t(\pi)$. For $1 \leq i \leq k$ and $\pi = (A, B) \in \Pi_i$ such that $\widetilde{x}$ is the center of A and $y \in B$, we define the weighted distance

$$\ell_{i,\boldsymbol{\phi}}^t(\widetilde{x}, y) = \mathsf{d}^p(\widetilde{x}, y) + \left[\frac{\psi_i^t(\pi)}{\mu_i(\widetilde{x})} + \frac{\phi_i(\widetilde{x}) \cdot \psi_*^t}{\text{Þ}(\boldsymbol{\phi})}\right] \cdot \frac{g}{\psi_0^t} \tag{12}$$

and its approximation

$$\underline{\ell}_{i,\boldsymbol{\phi}}^t(\widetilde{x}, y) = \mathsf{d}^p(\widetilde{x}, y) + \left[\frac{\underline{\psi}_i^t(\pi)}{\mu_i(\widetilde{x})} + \frac{\phi_i(\widetilde{x}) \cdot \underline{\psi}_*^t}{\text{Þ}(\boldsymbol{\phi})}\right] \cdot \frac{g}{\underline{\psi}_0^t}. \tag{13}$$

For $\boldsymbol{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_k) \in \widetilde{\mathbb{X}}(y)$, we set

$$\ell_{\boldsymbol{\phi}}^t(\boldsymbol{x}, y) = \sum_{i=1}^k \ell_{i,\boldsymbol{\phi}}^t(\widetilde{x}_i, y) \quad \text{and} \quad \underline{\ell}_{\boldsymbol{\phi}}^t(\boldsymbol{x}, y) = \sum_{i=1}^k \underline{\ell}_{i,\boldsymbol{\phi}}^t(\widetilde{x}_i, y).$$

For each $i \leq k$ and $y \in Y$, we maintain a heap $H_{i,y}$ that stores $\widetilde{X}_i(y)$ and uses $\underline{\ell}_{i,\boldsymbol{\phi}}^t(x, y)$ as the priority of $x \in \widetilde{X}_i(y)$ at time t. We emphasize that $|\widetilde{X}_i(y)| \leq \rho$ for all $y \in Y$ and $i \leq k$. Let $\overline{x}_{i,y}^t = \arg\min_{x \in \widetilde{X}_i(y)} \underline{\ell}_{i,\boldsymbol{\phi}}^t(x, y)$ be the minimum priority element of $\widetilde{X}_i(y)$ with respect to y in the heap $H_{i,y}$. For each $y \in Y$, let $\overline{\boldsymbol{x}}_y^t = (\overline{x}_{1,y}^t, \ldots, \overline{x}_{k,y}^t)$. We maintain a global priority queue H on Y where the priority of $y \in Y$ is $\underline{\ell}_{\boldsymbol{\phi}}^t(\overline{\boldsymbol{x}}_y^t, y)$. Set $\overline{y}^t = \arg\min_{y \in Y} \underline{\ell}_{\boldsymbol{\phi}}^t(\overline{\boldsymbol{x}}_y^t, y)$ to be the minimum priority element and let $\underline{\boldsymbol{x}}^t = \overline{\boldsymbol{x}}_{\overline{y}^t}^t$. Then

$$\underline{\boldsymbol{x}}^t = \arg \min_{y \in Y, \boldsymbol{x} \in \widetilde{\mathbb{X}}(y)} \underline{\ell}_{\boldsymbol{\phi}}^t(\boldsymbol{x}, y).$$

We now describe how to proceed at each step. Initially, for $t = 1$, we set $\underline{\psi}_0^t = \psi_0^t = 1$, $\underline{\psi}_*^t = \psi_*^t = 1$, and $\underline{\psi}_i^t(\pi) = \psi_i^t(\pi) = 1$ for all $i \leq k$ and $\pi \in \Pi_i$. At each time t, the above data structure returns $\underline{\boldsymbol{x}}^t$. We set $\boldsymbol{x}^t = \underline{\boldsymbol{x}}^t$ and proceed as described in Section 2. Define

$$\begin{aligned}
\overline{\lambda_0^t} &= \left\lceil \frac{¢_p(\boldsymbol{f})}{g} \right\rceil \cdot g - ¢_p(\boldsymbol{f}), \\
\lambda_0^t &= \frac{\overline{\lambda_0^t}}{\mathsf{d}^p(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))}, \\
\overline{\lambda_*^t} &= \left\lceil \frac{\text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}^t)}{\text{Þ}(\boldsymbol{\phi})} \right\rceil \cdot \text{Þ}(\boldsymbol{\phi}) - \text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}^t), \\
\lambda_*^t &= \frac{\overline{\lambda_*^t}}{\phi(\boldsymbol{x}^t)}, \\
\lambda_i^t &= \left\lceil \frac{f_i(x_i^t, B_i^t)}{\mu_i(x_i^t)} \right\rceil \mu_i(x_i^t) - f_i(x_i^t, B_i^t) \quad \forall 1 \leq i \leq k,
\end{aligned}$$

where for each $i \leq k$, we let $\pi_i^t = (A_i^t, B_i^t)$ be the WSPD pair from Π_i such that x_i^t is the center of A_i^t and $\overline{y}^t \in B_i^t$. λ_i^t is the total amount of mass one can commit to the edge $(x_i^t, \overline{y}^t)$ for the flow $f_i(x_i^t, B_i)$ through the center x_i^t to reach the next integer multiple of $\mu_i(x_i^t)$, $\overline{\lambda_0^t}$ (resp. $\overline{\lambda_*^t}$) denotes how far away the costs (resp. dual costs $\text{Þ}(\boldsymbol{\phi}, \boldsymbol{f}^t)$) of the current flows are from the next integer multiple of g (resp. $\text{Þ}(\boldsymbol{\phi})$), and λ_0^t (resp. λ_*^t) denotes the total amount of mass that can be transported through each edge $(x_i^t, \mathsf{ctr}(\boldsymbol{x}^t))$ to attain a cost of $\overline{\lambda_0^t}$ (resp. dual cost of $\text{Þ}(\boldsymbol{\phi})$). Then for the flow

update parameter $\lambda^t = \min\{\lambda_0^t, \lambda_*^t, \min_i\{\lambda_i^t\}\}$, we increment the flows and update the weights as follows:

$$
\begin{aligned}
f_i(x_i^t, \overline{y}^t) &\leftarrow f_i(x_i^t, \overline{y}^t) + \lambda^t && \text{for all } 1 \le i \le k,\\
\psi_i^{t+1}(\pi_i^t) &\leftarrow \psi_i^t(\pi_i^t) \cdot \left(1 + \eta \cdot \frac{\lambda^t}{\mu_i(x_i^t)}\right) && \text{for all } 1 \le i \le k,\\
\psi_0^{t+1} &\leftarrow \psi_0^t \cdot \left(1 + \eta \cdot \frac{\lambda^t \cdot \mathsf{d}^p(\boldsymbol{x}^t, \mathsf{ctr}(\boldsymbol{x}^t))}{g}\right)\\
\psi_*^{t+1} &\leftarrow \psi_*^t \cdot \left(1 + \eta \cdot \frac{\lambda^t \cdot \phi(\boldsymbol{x}^t)}{\mathrm{P}(\boldsymbol{\phi})}\right).
\end{aligned}
$$

We keep the weights $\psi_i^{t+1}(\pi) = \psi_i^t(\pi)$ for all $\pi \neq \pi_i^t$ in Π_i under each distribution $i \le k$, i.e. for all WSPD pairs which do not observe an increased flow in iteration t.

In addition, we do the following. Let $\boldsymbol{x}^t = (x_1^t, \ldots, x_k^t)$.

(i) If $\lambda^t = \lambda_i^t$ for some $i \ge 1$, then after updating $\psi_i^{t+1}(\pi_i^t)$, we set $\underline{\psi}_i^{t+1}(\pi_i^t) = \psi_i^{t+1}(\pi_i^t)$. We do not update $\underline{\psi}_i^{t+1}(\pi)$ for any other $\pi \in \Pi_i \setminus \{\pi_i^t\}$ or $\underline{\psi}_j^{t+1}$ for $j \neq i$. Since $\underline{\psi}_i^{t+1}(\pi_i^t)$ has changed, we update the priority of x_i^t in $H_{i,y}$ for all $y \in Y$ *such that* $x_i^t \in \widetilde{X}_i(y)$ (equivalently for all y such that x_i^t is the center of A and $y \in B$, where $\pi_i^t = (A, B)$). If this step also changes $\overline{\boldsymbol{x}}_y^t$ and $\underline{\ell}_{\boldsymbol{\phi}}^t(\overline{\boldsymbol{x}}_y^t, y)$ for some $y \in Y$, then we update the priority of each such y in the global priority queue H. This step takes $O(|B| \log(n+m))$ time, where B denotes the subset of Y such that $\pi_i^t = (A, B)$. We note that $\sum_{i=1}^k \sum_{(A,B)\in\Pi_i} |B| \le \rho k \cdot |Y| = \rho k m$ since each Π_i is a (ε, ρ)-pair decomposition.

(ii) If $\lambda^t = \lambda_0^t$, then after updating ψ_0^{t+1}, we set $\underline{\psi}_0^{t+1} = \psi_0^{t+1}$. We reconstruct all priority queues $H_{i,y}$ for all $1 \le i \le k$ and $y \in Y$, as well as the global priority queue H. This step takes $O((n+m)\rho)$ time.

(iii) If $\lambda^t = \lambda_*^t$, then after updating ψ_*^{t+1}, we set $\underline{\psi}_*^{t+1} = \psi_*^{t+1}$. In the same manner as for ψ_0^t, we reconstruct all priority queues $H_{i,y}$ for all $1 \le i \le k$ and $y \in Y$, as well as the global priority queue H. This step takes $O((n+m)\rho)$ time.

We remark that for each π_j in Π_i, the weight $\underline{\psi}_i^t(\pi_j)$ is updated at most $T = O(\eta^{-2} \log n)$ many times, precisely when the total flow from the center $\widetilde{x}_j$ reaches the next integer multiple of $\mu_i(\widetilde{x}_j)$. By the above observation that $\sum_{i=1}^k \sum_{(A,B)\in\Pi_i} |B| \le rhokm$, the total time updating the data structures due to updates to the pair weights $\underline{\psi}_i^t(\pi_j)$ is $O(mk\rho\eta^{-2}\log^2(m+n))$. In an analogous manner, if $\lambda_0^t = \lambda^t$ (resp. $\lambda_*^t = \lambda^t$), then after updating the value of ψ_0^{t+1} (resp. ψ_*^{t+1}) we update $\underline{\psi}_0^{t+1} \leftarrow \psi_0^{t+1}$ (resp. $\underline{\psi}_*^{t+1} \leftarrow \psi_*^{t+1}$). Then similarly $\underline{\psi}_0^t$ and $\underline{\psi}_*^t$ are updated at most $T = O(\eta^{-2} \log n)$ many times each, and each update takes $O((n+m)\rho)$ time. Hence, the total time spent in updating the data structure over all iterations is $O((n+m)k\rho \cdot \eta^{-2}\log^2(m+n))$.

The runtime of the algorithm can be broken down into the following analysis. Each step of the algorithm does the following tasks: (1) compute $\underline{\boldsymbol{x}}^t$, (2) add transported mass from x_i^t to $\mathsf{ctr}(\boldsymbol{x}^t)$, and (3) increase the weights $\psi_i^t(x_i)$ and ψ_0^t based on the transported mass to $\boldsymbol{x}^t$. The most expensive part in each step of the decision procedure is computing $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \ell^t(\boldsymbol{x}, \mathsf{ctr}(\boldsymbol{x}))$, and we just proved above that all computation and maintenance of $\boldsymbol{x}^t$ takes a total of $O((n+m)k\rho \cdot \eta^{-2}\log^2(m+n))$ over all iterations. The other two steps can be done in $O(k)$ time per iteration.

Analysis of the Measure. We now argue that Lemma 2 still holds. As mentioned above, the only place in the analysis where we used the fact that the algorithm chooses $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x}\in\mathbb{X}, y\in Y} \ell^t(\boldsymbol{x}, y)$ is in the proof of Lemma 6. We first observe that $\underline{\psi}_0^t$, $\underline{\psi}_*^t$ and $\underline{\psi}_i^t$ are approximations of ψ_0^t, ψ_*^t and ψ_i^t, respectively.

Lemma 14. *For any $t \ge 1$,*

$$\underline{\psi}_i^t(\pi) \le \psi_i^t(\pi) \le (1+2\eta) \cdot \underline{\psi}_i^t(\pi) \;\; \forall i \le k, \pi \in \Pi_i,$$

$$\underline{\psi}_0^t \le \psi_0^t \le (1+2\eta)\cdot \underline{\psi}_0^t,$$
$$\underline{\psi}_*^t \le \psi_*^t \le (1+2\eta)\cdot \underline{\psi}_*^t.$$

Proof. The proof is identical to that of Lemma 7, except for the addition of the new weights $\underline{\psi}_*^t$ and ψ_*^t, which follow in an analogous manner. □

We now state a slightly weaker version of Lemma 1.

Lemma 15. *There exists a constant $0 < c \le 2$ such that*

$$(1-\eta) - \frac{c\ln n}{T\eta} \le \frac{1}{T}\sum_{t\ge 1}\frac{\lambda^t}{\Psi^t}\cdot\left(\frac{\psi_0^t}{g}\cdot \ell_{\boldsymbol{\phi}}^t(\boldsymbol{x}^t,\overline{y}^t)\right) \le (1+2\eta)\cdot\frac{\nu(Y)}{\nu_g^*(Y)}.$$

Proof. To prove the lower bound, we note that the same analysis of the lower bound in Lemmas 5 and 8 extends after defining

$$m_*^t = \frac{\lambda^t\cdot\phi(\boldsymbol{x}^t)}{\mathrm{P}(\boldsymbol{\phi})},\quad p_*^t = \frac{\psi_*^t}{\Psi^t}$$

and observing $\sum_{t\ge1} m_*^t = \frac{\mathrm{P}(\boldsymbol{\phi},\boldsymbol{f})}{\mathrm{P}(\boldsymbol{\phi})} \le T$ by the termination condition of the algorithm.

As for the upper bound, using Lemma 14 along with the observations (5) and $\frac{\mathrm{P}(\boldsymbol{\phi},\boldsymbol{f})}{\mathrm{P}(\boldsymbol{\phi})} \le 1$, and the definitions (12), (13) and $\Psi^t = \psi_0^t + \psi_*^t + \sum_{i=1}^k\sum_{\pi\in\Pi_i}\psi_i^t(\pi)$, we obtain

$$\frac{\psi_0^t}{g}\cdot\ell_{\boldsymbol{\phi}}^t(\boldsymbol{x}^t,\overline{y}^t) \le (1+2\eta)\cdot\left(\frac{\underline{\psi}_0^t}{g}\cdot\underline{\ell}_{\boldsymbol{\phi}}^t(\boldsymbol{x}^t,\underline{y}^t)\right)$$

and

$$\Psi^t \ge \frac{\underline{\psi}_0^t}{g}\sum_{i=1}^k\sum_{y\in Y, x\in\widetilde{X}_i(y)} f_i^*(x,y)\cdot\underline{\ell}_{i,\boldsymbol{\phi}}^t(x,y)$$

in an analogous manner to Lemma 8. Then the remainder of the proof again follows in the exact same manner as in Lemma 6. □

Finally, substituting the values of η and T and choosing the constant sufficiently large in the value of T, we obtain

$$\begin{aligned}\nu(Y) &\ge \frac{1}{1+\frac{\delta}{2}}\cdot\left(1-\frac{\delta}{4}-\frac{\delta}{8}\right)\cdot\nu_g^*(Y)\\ &\ge \frac{1}{1+\frac{\delta}{2}}\cdot\left(1-\frac{3\delta}{8}\right)\cdot\nu_g^*(Y)\\ &\ge (1-\delta)\cdot\nu_g^*(Y).\end{aligned}$$

Hence, Lemma 2 still holds. Putting everything together, we obtain Theorem 1.

G An Efficient Exact Algorithm

In this section, we prove Theorem 3. To do so, we first describe the polynomial time algorithm, then prove its correctness. The algorithm will be divided into three components.

1. **Outer level dual ellipsoid method:** The first component is an application of the ellipsoid method to solve the dual partial Wasserstein barycenter problem. In each iteration of the ellipsoid method, we are given a set of weights $\boldsymbol{\pi} = \langle\pi_1,\ldots,\pi_k\rangle$ for the dual problem and make one call to what is known in the literature as a *separation oracle*. The separation oracle returns $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x}\in\mathbb{X}}\left\{\sum_{i=1}^k \mathsf{d}^p(x_i,\mathsf{ctr}(\boldsymbol{x})) + \pi_i(x_i)\right\}$, where $\mathsf{ctr}(\boldsymbol{x}) := \arg\min_{y\in Y}\mathsf{d}^p(\boldsymbol{x}^t,y)$. We use $\boldsymbol{x}^t$ to guide the dual weights for the next iteration.

2. **Inner level dual separation oracle:** The second component is an efficient implementation of the separation oracle. We describe an algorithm that, given weights $\boldsymbol{\pi}$, computes $\boldsymbol{x}^t = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \left\{\sum_{i=1}^k \mathsf{d}^p(x_i, \mathsf{ctr}(\boldsymbol{x})) + \pi_i(x_i)\right\}$ in $n^{O(d)}$ time. At a high level, we compute the overlay of k weighted Voronoi diagrams, one for each input distribution, and brute force search for the minimizer among each region in this overlay. We prove that the complexity of this overlay is bounded by $n^{O(d)}$.
3. **Dual-guided primal algorithm:** The final component is a procedure which solves the fixed-support partial Wasserstein barycenter problem on the candidate support set $Y^* = \{\arg\min_{y\in\mathbb{R}^d} \mathsf{d}^p(\boldsymbol{x}^t, y) : t \geq 1\}$, the set of centroids for each $\boldsymbol{x}^t$ output from the call to the separation oracle in each iteration $t \geq 1$ of the dual ellipsoid method. We prove the new candidate support set Y^* is polynomially sized, and therefore we can run any polynomial time algorithm for the fixed-support partial Wasserstein barycenter problem on this support.

We note that components 2 and 3 of the algorithm, described above, are nearly identical to the analogous stages of the algorithm for the Wasserstein barycenter algorithm of Altschuler & Boix-Adsera (2021) and largely included to keep the algorithm self-contained. The primary observation of our work is that the dual *partial Wasserstein barycenter (PWB)* problem has enough of the same structure as the dual Wasserstein barycenter problem to use their dual separation oracle as a subroutine for the dual PWB problem.

G.1 Outer Level Dual Ellipsoid Method

We first recall standard results on the ellipsoid method. For any convex set $\mathcal{K} \subseteq \mathbb{R}^d$ and any point $p \in \mathbb{R}^d$, a *separation oracle* is an algorithm which either concludes that $p \in \mathcal{K}$ or returns a separating hyperplane (defined by a vector $v \in \mathbb{R}^d$ and a constant $c \in \mathbb{R}$ such that $\langle p, v\rangle \geq c$ and $\langle x, v\rangle < c$ for all $x \in \mathcal{K}$) which separates p from $\mathcal{K}$.

Theorem 4 (Grötschel et al. (1981))**.** *Let $M, N \in \mathbb{N}$ be two integers, and let $A \in \mathbb{R}^{M\times N}, b \in \mathbb{R}^M$ and $c \in \mathbb{R}^N$ be such that every element is represented with at most $\log\Delta$ bits. Then the ellipsoid method computes a vertex solution to $\arg\min\{c^\top x : Ax \leq b\}$ in poly$(N, \log\Delta)$ time and poly$(N, \log\Delta)$ calls to the separation oracle for the convex set $\{x \in \mathbb{R}^N : Ax \leq b\}$.*

We use the ellipsoid method to solve the dual partial Wasserstein barycenter problem. Note again that the free-support Wasserstein barycenter problem is equivalent to the fixed-support Wasserstein barycenter problem over a candidate support set Y of size $|Y| = O(n^k)$.

$$\begin{aligned}
\min_{\nu, f_i} \sum_{i=1}^k \sum_{x\in X_i, y\in Y} & \mathsf{d}^p(x,y)\cdot f_i(x,y) \\
\text{s.t. } \sum_{y\in Y} & f_i(x,y) \leq \mu_i(x) \;\; \forall i \leq k, x \in X_i, \\
\sum_{x\in X_i} & f_i(x,y) = \nu(y) \;\; \forall i \leq k, x \in X_i, y \in Y, \\
\sum_{y\in Y} & \nu(y) = \alpha, \\
& \nu, f_i \geq 0.
\end{aligned}$$

The choice of appropriate Y is $Y = \left\{\arg\min_{y\in\mathbb{R}^d} \sum_{i=1}^k \mathsf{d}^p(x_i, y) : x_i \in X_i \; \forall i \leq k\right\}$. See e.g. Anderes et al. (2016) for a proof of this fact. Intuitively, if k points have their mass transported to a point of the barycenter distribution, then the cheapest location to place the mass is at the centroid of the k points.

We now observe that the dual partial Wasserstein barycenter problem is therefore:

$$\begin{aligned}
\max_{\lambda, \pi_i, \omega_i} \; & \alpha\lambda - \sum_{i=1}^k \sum_{x\in X_i} \pi_i(x)\mu_i(x) \\
\text{s.t. } & \omega_i(y) - \pi_i(x) \leq \mathsf{d}^p(x,y) \;\; \forall i \leq k, x \in X_i, y \in Y,
\end{aligned} \tag{14}$$

$$\lambda - \sum_i \omega_i(y) \leq 0 \;\; \forall y \in Y,$$
$$\pi_i \geq 0.$$

The weights $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_k \rangle$ and $\boldsymbol{\omega} = \langle \omega_1, \ldots, \omega_k \rangle$ can be viewed as dual weights to the classical OT problem from μ_i to the variable barycenter distribution ν, and λ is a new dual variable corresponding to the constraint that the total mass of the barycenter is equal to α.

We note that ω_i variables are only auxiliary variables used to bound λ in terms of π_i and do not appear in the dual objective function. By substitution of the dual constraints, we can deduce that an equivalent linear program over only the variables $\lambda, \pi_1, \ldots, \pi_k$ is:

$$\max_{\lambda, \pi_i} \alpha\lambda - \sum_{i=1}^{k} \sum_{x \in X_i} \pi_i(x)\mu_i(x) \tag{15}$$
$$\text{s.t. } \lambda \leq \min_{y \in Y} \left\{ \sum_{i=1}^{k} \min_{x_i \in X_i} \left\{ \mathsf{d}^p(x_i, y) + \pi_i(x_i) \right\} \right\},$$
$$\pi_i \geq 0.$$

That is, we can ignore the ω_i variables entirely and end up with the same solution for λ and π_i. We prove this claim formally in the following lemma.

Lemma 16. *If $(\lambda, \boldsymbol{\pi}, \boldsymbol{\omega})$ is an optimal solution to the linear program defined by (14), then $(\lambda, \boldsymbol{\pi})$ is an optimal solution to the linear program defined by (15). Furthermore, if $(\lambda, \boldsymbol{\pi})$ is an optimal solution to the linear program defined by (15), then $(\lambda, \boldsymbol{\pi}, \boldsymbol{\omega})$ where $\omega_i(y) = \sum_{i=1}^{k} \min_{x_i \in X_i} \{\mathsf{d}^p(x_i, y) + \pi_i(x_i)\}$ for all $i \leq k$ and $y \in Y$ is an optimal solution to the linear program defined by (14).*

Proof. We observe that the objective functions of the LPs (14) and (15) are identical and only depend on λ and $\boldsymbol{\pi}$. The result follows once we prove: (a) for any feasible solution $(\lambda, \boldsymbol{\pi}, \boldsymbol{\omega})$ to the linear program defined by (14), $(\lambda, \boldsymbol{\pi})$ is a feasible solution to the linear program defined by (15); (b) for any feasible solution $(\lambda, \boldsymbol{\pi})$ is an optimal solution to the linear program defined by (15), then $(\lambda, \boldsymbol{\pi}, \boldsymbol{\omega})$ where $\omega_i(y) = \sum_{i=1}^{k} \min_{x_i \in X_i} \{\mathsf{d}^p(x_i, y) + \pi_i(x_i)\}$ for all $i \leq k$ and $y \in Y$ is a feasible solution to the linear program defined by (14). This would imply that the feasible values of λ and $\boldsymbol{\pi}$ for the two linear programs are identical.

(a) Suppose $(\lambda, \boldsymbol{\pi}, \boldsymbol{\omega})$ is an arbitrary feasible solution to the linear program defined by (14). Then by feasibility, we observe that

$$\omega_i(y) \leq \mathsf{d}^p(x, y) + \pi_i(x) \;\; \forall i \leq k, x \in X_i, y \in Y,$$

and combining constraints yields

$$\omega_i(y) \leq \min_{x \in X_i} \{\mathsf{d}^p(x, y) + \pi_i(x)\}.$$

We also note by feasibility that

$$\lambda \leq \sum_i \omega_i(y) \;\; \forall y \in Y$$
$$\leq \sum_i \min_{x \in X_i} \{\mathsf{d}^p(x, y) + \pi_i(x)\} \;\; \forall y \in Y.$$

Combining constraints yields

$$\lambda \leq \min_{y \in Y} \left\{ \sum_{i=1}^{k} \min_{x_i \in X_i} \{\mathsf{d}^p(x_i, y) + \pi_i(x_i)\} \right\}.$$

(b) Suppose $(\lambda, \boldsymbol{\pi})$ is an arbitrary feasible solution to the linear program defined by (15). Define the additional variables $\omega_i(y) = \sum_{i=1}^{k} \min_{x_i \in X_i} \{\mathsf{d}^p(x_i, y) + \pi_i(x_i)\}$ for all $i \leq k$ and $y \in Y$. Then by feasibility of $(\lambda, \boldsymbol{\pi})$ and the definition of $\boldsymbol{\omega}$, we observe

$$\lambda \leq \min_{y \in Y} \left\{ \sum_i \min_{x \in X_i} \{\mathsf{d}^p(x, y) + \pi_i(x)\} \right\}$$

$$= \min_{y \in Y} \left\{ \sum_{i=1}^{k} \omega_i(y) \right\}$$
$$\leq \sum_{i=1}^{k} \omega_i(y) \ \ \forall y \in Y.$$

Moreover, by the definition of $\boldsymbol{\omega}$ we conclude that

$$\begin{aligned} \omega_i(y) - \pi_i(x) &= \min_{x' \in X_i} \{\mathsf{d}^p(x', y) + \pi_i(x')\} - \pi_i(x) \\ &\leq (\mathsf{d}^p(x, y) + \pi_i(x)) - \pi_i(x) \\ &= \mathsf{d}^p(x, y) \end{aligned}$$

for all $i \leq k$, $x \in X_i$ and $y \in Y$.

□

From combining Lemma 16 with Theorem 4, we conclude

Lemma 17. *The optimal vertex solution to the dual partial Wasserstein barycenter problem can be computed in poly*$(n, \log \Delta)$ *time and poly*$(n, \log \Delta)$ *calls to the separation oracle for the convex set*

$$\mathcal{K} = \left\{ (\lambda, \boldsymbol{\pi}) : \lambda \leq \min_{y \in Y} \left\{ \sum_{i=1}^{k} \min_{x_i \in X_i} \{\mathsf{d}^p(x_i, y) + \pi_i(x_i)\} \right\}, \pi_i \geq 0 \ \ \forall i \leq k \right\} \subseteq \mathbb{R}^{n+1}.$$

Next, we design a separation oracle for $\mathcal{K}$ which runs in $n^{O(d)}$ time.

G.2 Inner Level Dual Separation Oracle

The separation oracle is given a point $(\lambda, \boldsymbol{\pi})$ and asked whether $(\lambda, \boldsymbol{\pi})$ is an element of $\mathcal{K}$ or not. If $(\lambda, \boldsymbol{\pi}) \notin \mathcal{K}$, then we need to find a hyperplane which separates $(\lambda, \boldsymbol{\pi})$ from $\mathcal{K}$.

We note that since there are at most n variables, checking if $\pi_i \geq 0$ for all i can be done in $O(n)$ time. It suffices to design an algorithm which decides if

$$\lambda \leq \min_{y \in Y} \left\{ \sum_{i=1}^{k} \min_{x_i \in X_i} \{\mathsf{d}^p(x_i, y) + \pi_i(x_i)\} \right\}.$$

We do so by finding the tuple $\boldsymbol{x} \in \mathbb{X}$ such that

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}' \in \mathbb{X}} \left\{ \sum_{i=1}^{k} \mathsf{d}^p(x'_i, \mathsf{ctr}(\boldsymbol{x}')) + \pi_i(x'_i) \right\},$$

where again $\mathsf{ctr}(\boldsymbol{x}') := \arg\min_{y \in Y} \mathsf{d}^p(\boldsymbol{x}', y)$ for any $\boldsymbol{x}' \in \mathbb{X}$. To find the tuple $\boldsymbol{x}$, we crucially observe that for any $p \in \mathbb{R}^d$,

$$\min_{\boldsymbol{x}' \in \mathbb{X}} \left\{ \sum_{i=1}^{k} \mathsf{d}^p(x'_i, p) + \pi_i(x'_i) \right\} = \sum_{i=1}^{k} \min_{x'_i \in X_i} \left[\mathsf{d}^p(x'_i, p) + \pi_i(x'_i) \right].$$

That is, since $\mathbb{X} = \prod_{i=1}^{k} X_i$ is decomposable, the minimum sum of weighted distances from any k-tuple in $\mathbb{X}$ to p is exactly the sum of the minimum weighted distance from each X_i to p. From this insight, we compute the minimizer $\boldsymbol{x}$ as follows.

1. For each $i \in [k]$, we compute the weighted Voronoi diagram $\mathcal{V}_i$ of X_i with respect to distances $\mathsf{d}^p(\cdot, \cdot)$ and weights π_i. Each weighted Voronoi diagram $\mathcal{V}_i = \{V_{\pi_i}(x) : x \in X_i\}$ consists of the cells

$$V_{\pi_i}(x) = \left\{ p \in \mathbb{R}^d : \mathsf{d}^p(x, p) + \pi_i(x) \leq \mathsf{d}^p(x', p) + \pi_i(x') \ \ \forall x' \in X_i \right\}.$$

2. We compute the overlay of the weighted Voronoi diagrams

$$\Xi = \left\{ \bigcap_{i=1}^{k} V_{\pi_i}(x_i) \neq \varnothing : x_i \in X_i \ \ \forall i \in [k] \right\}.$$

   Then for each cell $\mathcal{C} \in \Xi$, we note that there exists a tuple $\boldsymbol{x}' = (x_1, \ldots, x_k) \in \mathbb{X}$ such that $\mathcal{C} = \bigcap_{i=1}^{k} V_{\pi_i}(x_i)$.

3. We iterate over each tuple $\boldsymbol{x}' = (x_1, \ldots, x_k)$ in $\mathbb{X}$ such that $\bigcap_{i=1}^{k} V_{\pi_i}(x_i) \in \Xi$ and choose $\boldsymbol{x}$ to be such that $\sum_{i=1}^{k} \mathsf{d}^p(x_i', \mathsf{ctr}(\boldsymbol{x}')) + \pi_i(x_i')$ is minimized.

Assuming the bisector of two points under the distance d^p is an algebraic variety of constant degree (e.g. $\mathsf{d}(x, y) = \|x - y\|_p$), it is known that (i) for each $i \in [k]$, $|\mathcal{V}_i| = |X_i|^{O(d)}$ and $\mathcal{V}_i$ can be computed in $|X_i|^{O(d)}$ time, and (ii) $|\Xi| = n^{O(d)}$ and all cells of Ξ can be computed in $n^{O(d)}$ time (Basu et al., 2006). Iterating over all of the cells and computing $\boldsymbol{x}$ then takes $O(dk|\Xi|)$ time since for each cell in Ξ, we compute the total weighted distance from the corresponding k-tuple in $\mathbb{X}$ to its centroid in Y. Therefore, the three steps above take $n^{O(d)}$ time.

To conclude the separation oracle, we simply check if

$$\lambda \leq \sum_{i=1}^{k} \mathsf{d}^p(x_i, \mathsf{ctr}(\boldsymbol{x})) + \pi_i(x_i),$$

where $\boldsymbol{x} = (x_1, \ldots, x_k)$. If so, then we return that $(\lambda, \boldsymbol{\pi})$ is contained in $\mathcal{K}$. If not, then let $b = \frac{1}{2}\left(\lambda + \sum_{i=1}^{k} \left(\mathsf{d}^p(x_i, \mathsf{ctr}(\boldsymbol{x})) + \pi_i(x_i)\right)\right)$. The hyperplane

$$H(\boldsymbol{x}, b) = \left\{ (\lambda', \boldsymbol{\pi}') \in \mathbb{R}^{n+1} : \sum_{i=1}^{k} \mathsf{d}^p(x_i, \mathsf{ctr}(\boldsymbol{x})) + \pi_i'(x_i) = b \right\}$$

defined by $\boldsymbol{x}$ and b then separates $(\lambda, \boldsymbol{\pi})$ from $\mathcal{K}$. We then have the following lemma.

Lemma 18. *There exists a separation oracle for $\mathcal{K}$ which runs in $n^{O(d)}$ time.*

G.3 Dual-Guided Primal Algorithm

We conclude with the algorithm which uses the dual ellipsoid method to compute a primal solution. We use the ellipsoid method described in Theorem 4 to compute an optimal dual solution. In the process of running the ellipsoid method, we make $T = \text{poly}(n, \log \Delta)$ calls to the separation oracle by Lemma 17. For each $t \leq T$, let $\boldsymbol{x}^t$ be the minimizer computed by the separation oracle in iteration t, and let $Y^* = \{\mathsf{ctr}(\boldsymbol{x}^t) : t \in [T]\} \subseteq Y$. We note that $|Y^*| = T$ is polynomially sized in n and $\log \Delta$. Then the primal linear program for the fixed-support partial Wasserstein barycenter problem

$$\begin{aligned}
\min_{\nu, f_i} \ & \sum_{i=1}^{k} \sum_{x \in X_i, y \in Y^*} \mathsf{d}^p(x, y) \cdot f_i(x, y) \\
\text{s.t.} \ & \sum_{y \in Y^*} f_i(x, y) \leq \mu_i(x) \ \ \forall i \leq k, x \in X_i, \\
& \sum_{x \in X_i} f_i(x, y) = \nu(y) \ \ \forall i \leq k, x \in X_i, y \in Y^*, \\
& \sum_{y \in Y^*} \nu(y) = \alpha, \\
& \nu, f_i \geq 0
\end{aligned}$$

has polynomial size in n and $\log \Delta$. We use a polynomial time linear program solver (e.g. Lee & Sidford (2014; 2015)) to compute an optimal fixed-support partial Wasserstein barycenter ν on Y^* and return ν. Hence, as an immediate consequence of the polynomial time algorithm for linear programs and Lemmas 17 and 18, the algorithm has a polynomial runtime.

Lemma 19. *The measure ν is computed by the algorithm in $(n^d \log \Delta)^{O(1)}$ time.*

It suffices to prove that the computed distribution ν is an optimal free-support partial Wasserstein barycenter.

Lemma 20. *The measure ν computed by the algorithm is an optimal α-partial Wasserstein barycenter.*

Proof. Let $\mathcal{L}$ be the set of constraints output by the separation oracle through the computation of the ellipsoid algorithm on the dual partial Wasserstein barycenter problem. Then the dual problem restricted to the constraints in $\mathcal{L}$ has the same optimal value as the original dual partial Wasserstein barycenter problem (the ellipsoid algorithm would have the same set of queries to the separation oracle). Denote the dual problem restricted to $\mathcal{L}$ as the restricted dual PWB problem.

The restricted dual PWB problem has a primal LP equivalent to the fixed-support PWB problem with support equal to Y^*. By strong duality, we note that (a) the restricted dual PWB problem has the same optimal value as the fixed-support PWB problem with support Y^* and (b) the dual PWB problem has the same optimal value as the PWB problem. Therefore, the PWB problem has the same optimal value as the fixed-support PWB problem with support Y^*. This implies the lemma since ν is an optimal solution to the fixed-support PWB problem with support Y^*. □

Then combining Lemmas 19 and 20 results in Theorem 3.