Fast and Accurate Approximations of the Optimal Transport in Semi-Discrete and Discrete Settings*

Pankaj K. Agarwal† Sharath Raghvendra‡ Pouyan Shirzadian§ Keegan Yao†

Abstract

Given a d-dimensional continuous (resp. discrete) probability distribution μ and a discrete distribution ν, the semi-discrete (resp. discrete) optimal transport (OT) problem asks for computing a minimum-cost plan to transport mass from μ to ν; we assume n to be the number of points in the support of the discrete distributions. In this paper, we present three approximation algorithms for the OT problem with strong provable guarantees.

(i) *Additive approximation for semi-discrete OT:* For any parameter $\varepsilon > 0$, we present an algorithm that computes a semi-discrete transport plan $\tilde{\tau}$ with cost $\text{¢}(\tilde{\tau}) \leq \text{¢}(\tau^*) + \varepsilon$ in $n^{O(d)} \log \frac{\mathrm{D}}{\varepsilon}$ time; here, τ^* is the optimal transport plan, D is the diameter of the supports of μ and ν, and we assume we have access to an oracle that outputs the mass of μ inside a constant-complexity region in $O(1)$ time. Our algorithm works for several ground distances including the L_p-norm and the squared-Euclidean distance.

(ii) *Relative approximation for semi-discrete OT:* For any parameter $\varepsilon > 0$, we present an algorithm that computes a semi-discrete transport plan $\tilde{\tau}$ with cost $\text{¢}(\tilde{\tau}) \leq (1+\varepsilon)\text{¢}(\tau^*)$ in $n \log(n) \cdot (\varepsilon^{-1} \log\log n)^{O(d)}$ expected time; here, τ^* is the optimal transport plan, and we assume we have access to an oracle that outputs the mass of μ inside an orthogonal box in $O(1)$ time, and the ground distance is any L_p norm.

(iii) *Relative approximation for discrete OT:* For any parameter $\varepsilon > 0$, we present a Monte-Carlo algorithm that computes a transport plan $\tilde{\tau}$ with an expected cost $\text{¢}(\tilde{\tau}) \leq (1+\varepsilon)\text{¢}(\tau^*)$ under any L_p norm in $n \log(n) \cdot (\varepsilon^{-1} \log\log n)^{O(d)}$ time; here, τ^* is an optimal transport plan and we assume that the spread of the supports of μ and ν is polynomially bounded.

1 Introduction

Optimal transport (OT) is a powerful tool for comparing probability distributions and computing maps between them. Put simply, the optimal transport problem deforms one distribution to the other with smallest possible cost. Classically, the OT problem has been extensively studied within the mathematics, statistics, and operations research [38, 39, 49]. In recent years, OT has seen rapid rise in various machine learning and computer vision applications as a meaningful metric between distributions and has been extensively used in generative models [20, 26, 44], robust learning [21], supervised learning [29, 36], computer vision applications [12, 27], variational inference [7], blue noise generation [19, 42], and parameter estimation [14, 35]. These applications have led to developing efficient algorithms for OT; see the book [41] for review of computational OT.

In the *geometric OT* problem, the supports are (possibly infinite) point sets in $\mathbb{R}^d$ and the cost of transporting unit mass between two points is some L_p distance between them. In this paper, we design simple, efficient approximation algorithms for the semi-discrete and discrete geometric OT problems assuming d is a constant.

Let μ be a continuous probability distribution (i.e., density) defined over a compact bounded support $A \subset \mathbb{R}^d$, and let ν be a discrete distribution, where the support of ν, denoted by B, is a set of n points in $\mathbb{R}^d$. Let $\mathrm{d}(\cdot,\cdot)$ be the ground distance between a pair of points in $\mathbb{R}^d$. A coupling $\tau\colon A \times B \to \mathbb{R}_{\geq 0}$ is called a *transport plan* for μ and ν if for all $a \subseteq A$, $\sum_{b\in B} \tau(a,b) = \mu(a)$ (where $\mu(a)$ is the mass of μ inside a) and for all $b \in B$, $\int_A \tau(a,b)\,da = \nu(b)$. The cost of the transport plan τ is given by $\text{¢}(\tau) := \int_A \sum_{b\in B} \mathrm{d}(a,b)\tau(a,b)\,da$. The goal is to find a minimum-cost (semi-discrete) transport plan satisfying μ and ν[1]. For any parameter $\varepsilon > 0$, a transport plan τ between μ and ν is called ε-*close* if the cost of τ is within an additive error of ε from the cost of the optimal transport plan τ^*, i.e., $\text{¢}(\tau) \leq \text{¢}(\tau^*) + \varepsilon$. A $(1+\varepsilon)$-*approximate OT plan*, or simply ε-*OT plan*, is a transport plan τ with $\text{¢}(\tau) \leq (1+\varepsilon)\text{¢}(\tau^*)$.

*The full version of the paper can be accessed at https://arxiv.org/abs/2311.02172

†Department of Computer Science, Duke University.
‡Department of Computer Science, North Carolina State University.
§Department of Computer Science, Virginia Tech.

[1]Apparently the semi-discrete OT was introduced by Cullen and Purser [18] without reference to optimal transport.

The problem of computing semi-discrete OT between μ and ν reduces to the problem of finding a set of weights $y : B \to \mathbb{R}_{\geq 0}$ so that, for any point $b \in B$, the Voronoi cell of b in the additively weighted Voronoi diagram has a mass equal to $\nu(b)$, i.e., $\mathrm{Vor}(b) = \{x \in \mathbb{R}^d \mid \mathrm{d}(x,b) - y(b) \leq \mathrm{d}(x,b') - y(b'), \forall b' \in B\}$, $\mu(\mathrm{Vor}(b)) = \nu(b)$, and the mass of μ in $\mathrm{Vor}(b)$ is transported to b; see [11]. One can thus define an optimal semi-discrete transport plan by describing the weights of points in B. For arbitrary distributions, weights can have large bit (or algebraic) complexity, so our goal will be to compute the weights accurately up to $s = O(\log \varepsilon^{-1})$ bits, which in turn will return an ε-close semi-discrete OT plan.

If μ is also a discrete distribution with support A, a *discrete transport plan* is $\tau \colon A \times B \to \mathbb{R}_{\geq 0}$ that assigns the mass transported along each edge $(a,b) \in A \times B$ such that $\sum_{b \in B} \tau(a,b) = \mu(a)$ for each point $a \in A$ and $\sum_{a \in A} \tau(a,b) = \nu(b)$ for each point $b \in B$. The cost of τ is given by $¢(\tau) = \sum_{(a,b) \in A \times B} \tau(a,b)\mathrm{d}(a,b)$. The *discrete OT problem* asks for a transport plan τ with the minimum cost. We refer to such plan as an *OT plan*.

Related work. The discrete optimal transport problem under any metric can be modeled as an uncapacitated minimum-cost flow problem and can be solved in strongly polynomial time of $O((m + n \log n) n \log n)$ time using the algorithm by Orlin [40]. Using recent techniques [45], it can be solved in $m^{1+o(1)}\mathrm{poly}\log(\Delta)$ time, where Δ depends on the spread of $A \cup B$ and the maximum demand. The special case where all points have the same demand is the widely studied *minimum-cost bipartite matching* problem. There is extensive work on the design of near-linear time approximation for the optimal transport and related matching problems [4, 8, 12, 23, 30, 43, 46]. The near-linear time algorithms by Khesin *et. al.* [30] and Fox and Lu [23] for computing an ε-OT plan use minimum-cost-flow (MCF) solvers (e.g. [47]) as a black box and numerically precondition their minimum-cost flow instance using geometry [23, 30, 47]. The work of Zuzic [50] describes a multiplicative-weights update (MWU) based boosting method for minimum-cost flows using an approximate primal-dual oracle as a black box, which replaces the preconditioner used in [30, 47]. All these algorithms are Monte Carlo algorithms and have running time of $n(\varepsilon^{-1} \log n)^{O(d)}$. Recently, Agarwal *et. al.* [1] presented an $n(\varepsilon^{-1} \log n)^{O(d)}$-time deterministic algorithm for computing an ε-approximate bipartite matching in $\mathbb{R}^d$. A Monte-Carlo ε-approximation algorithm for matching with run time $n \log^4 n(\varepsilon^{-1} \log \log n)^{O(d)}$ was presented in [2]. Very recently, Fox and Lu proposed a deterministic algorithm for ε-OT with run time of $O(n\varepsilon^{-(d+2)} \log^5 n \log \log n)$ [24].

Many known algorithms for semi-discrete OT compute an ε-close transport plan using the first- and second-order numerical solvers [11, 13, 17, 19, 31, 32, 34, 39]. These algorithms start with an initial set of weights for points in B and iteratively improve the weights until the mass inside the Voronoi cell of any point $b \in B$ is an additive factor ε away from $\nu(b)$. One can use these solvers to compute an ε-close transport plan by executing $\mathrm{poly}(n, 1/\varepsilon)$ iterations. Each iteration requires computation of several weighted Voronoi diagrams, each of which takes $n^{O(d)}$ time. Another widely used approach is to draw samples from the continuous distribution and convert the semi-discrete OT problem to a discrete instance [25]; however, due to sampling errors, this approach provides an additive approximation. Van Kreveld *et. al.* [48] presented a $(1+\varepsilon)$-approximation OT algorithm for the restricted case when the continuous distribution is uniform over a collection of simple geometric objects (e.g. segments, simplices, etc.), by sampling roughly n^2 points. Their running time is roughly $n^2\varepsilon^{-O(d)}\mathrm{poly}\log(n)$.

Our contributions. We present three new algorithms for the semi-discrete and discrete OT problems. Our first result (Section 2) is a cost-scaling algorithm that computes an ε-close transport plan for a semi-discrete instance in $n^{O(d)} \log(\mathrm{D}/\varepsilon)$ time, assuming that we have access to an oracle that, given a constant complexity region φ, returns $\mu(\varphi)$:

THEOREM 1.1. *Let μ be a continuous distribution defined on a compact bounded set $A \subset \mathbb{R}^d$ for some fixed $d \geq 1$, ν a discrete distribution with a support $B \subset \mathbb{R}^d$ of size n, and $\varepsilon > 0$ a parameter. Suppose there exists an* ORACLE *which, given a constant complexity region φ, returns $\mu(\varphi)$ in Q time. Then an ε-close transport plan can be computed in $Qn^{O(d)} \log(\frac{\mathrm{D}}{\varepsilon})$ time, where D is the diameter of $A \cup B$.*

To the best of our knowledge, our algorithm is the first one to compute an ε-close transport plan in time that is polynomial in both n and $\log(\varepsilon^{-1})$. Earlier algorithms had an $\varepsilon^{-O(1)}$ factor in the run time[2]. Our algorithm not only computes an ε-close transport plan, it also finds the optimal dual weights within an additive error of

[2]Mérigot and Thibert had conjectured that an algorithm for computing an ε-close OT for semi-discrete setting with runtime $(n \log \varepsilon^{-1})^{O(1)}$ might follow using a scaling framework [37, Remark 24]. Our result proves their conjecture in the affirmative.

ε, i.e., it computes optimal dual-weights up to $O(\log \varepsilon^{-1})$ bits of accuracy. Our algorithm works for any ground distance where the bisector of two points under the distance function $\mathrm{d}(\cdot,\cdot)$ is an algebraic variety of constant degree. Consequently, it works for several important distances, including the L_p-norm and the squared-Euclidean distance. The previous best-known algorithm by Kitagawa [31] for the semi-discrete OT has an execution time $n^{\Omega(d)}\mathrm{D}/\varepsilon$; furthermore, their algorithm only approximates the cost and does not necessarily provide any guarantees for the transport plan or the dual weights of B.

For each scale δ, our algorithm starts with a set of dual weights assigned to B and constructs an instance of discrete OT by using the arrangement of $4n+1$ shifts of the Voronoi cell of each point in B. This discrete instance, which is of size $n^{O(d)}$, is then solved using a primal-dual solver. The optimal dual weights for this discrete instance are then used to refine the dual weights of B. These refined dual weights act as the starting dual weights for the next scale $\delta/2$. Starting with $\delta = \mathrm{D}$, our algorithm executes $O(\log(\mathrm{D}/\varepsilon))$ scales and stops when $\delta \leq \varepsilon$. In order to show that the semi-discrete transport plan computed in scale δ is δ-close, we introduce a set of exponentially many δ-feasibility constraints and show that any transport plan that satisfies these is a δ-close transport plan. We then show that, in scale δ, the semi-discrete OT plan and the duals computed by our polynomial time algorithm satisfies all of these exponentially many constraints and therefore, is δ-close.

Our second result (Section 3) is another approximation algorithm for the semi-discrete setting whose running time is near-linear in n but the dependence on ε increases to $\varepsilon^{-O(d)}$.

THEOREM 1.2. *Let μ be a continuous distribution defined on a compact set $A \subset \mathbb{R}^d$ for some fixed $d \geq 1$, ν a discrete distribution with a support $B \subset \mathbb{R}^d$ of size n, and $\varepsilon > 0$ a parameter. Suppose there exists an* ORACLE *that given an axis-aligned box $\square$, returns $\mu(\square)$ in Q time. Then a $(1+\varepsilon)$-approximate OT plan can be computed in $O(n\varepsilon^{-3d-2}(\log^5(n)\log(\log n)+Q))$ time. If the spread of B is polynomially bounded, a $(1+\varepsilon)$-approximate OT plan can be computed in $O(n\varepsilon^{-4d-5}(\log(n)\log^{2d+5}(\log n)+Q))$ time with probability at least $\frac{1}{2}$.*

Similar to many previous algorithms (see e.g. [48]), we also discretize the continuous distribution and use a discrete OT algorithm. Our main contribution is a clever sampling strategy that works for arbitrary density, that guarantees relative ε-approximation of the OT cost, and that requires to choose only $n\varepsilon^{-O(d)}$ samples. Earlier approaches guaranteed additive error, worked for restricted cases, or required a much larger set of samples.

Our final result (Section 4) is a new $(1+\varepsilon)$-approximation algorithm for the discrete transport problem.

THEOREM 1.3. *Let μ and ν be two discrete distributions with finite support sets $A, B \subset \mathbb{R}^d$, respectively, for some fixed $d \geq 1$, and $\varepsilon > 0$ a parameter. Set $|A \cup B| = n$ and assume that the spread of $A \cup B$ is $n^{O(1)}$. Then a transport plan between μ and ν can be computed in $O\left(n\varepsilon^{-2d-5}\log(n)\log^{2d+5}(\log n)\right)$ time that is an ε-close OT plan with probability at least $\frac{1}{2}$.*

Notwithstanding the recent deterministic algorithm by Fox and Lu [24] for computing an ε-OT plan in $O(n\varepsilon^{-d-2}\log^5(n)\log(\log n))$ time, our result is of independent interest. The running time is slightly better than in [24] with respect to $\log n$, though of course their algorithm is deterministic. But our main contribution is a simple greedy, Monte-Carlo primal-dual $O(\log\log n)$-approximation algorithm that is geometric and that runs in $O(n\log\log n)$ time. By plugging our algorithm into the multiplicative-weight-update (MWU) method as in [50], we obtain a $(1+\varepsilon)$-approximation algorithm. We believe the derandomization technique of Lu and Fox can be applied to our algorithm, but one has to check all the technical details.

Proofs of some of the technical lemmas are omitted here and can be found in the full version [3].

2 Computing a Highly Accurate Semi-Discrete Optimal Transport

Given a continuous distribution μ over a compact bounded set $A \subset \mathbb{R}^d$, a discrete distribution ν over a set $B \subset \mathbb{R}^d$ of n points, and a parameter $\varepsilon > 0$, we present a cost-scaling algorithm for computing an ε-close transport plan from μ to ν. We first describe the overall framework, then provide details of the algorithm and analyze its efficiency, and finally prove its correctness.

In our algorithm, we use a black-box primal-dual discrete OT solver PD-OT(μ',ν') that given two discrete distributions μ' and ν' defined over two point sets A' and B', returns a transport plan σ from μ' to ν' and a dual weight $y(v)$ for each point $v \in A' \cup B'$ such that for any pair $(a,b) \in A' \times B'$,

$$y(b) - y(a) \leq \mathrm{d}(a,b), \tag{2.1}$$

$$y(b) - y(a) = \mathrm{d}(a,b) \quad \text{if } \sigma(a,b) > 0. \tag{2.2}$$

Standard primal-dual methods [33] construct a transport plan while maintaining (2.1) and (2.2). For concreteness, we use Orlin's algorithm [40] that runs in $O(|A' \cup B'|^3)$ time.

2.1 The Scaling Framework. The algorithm works in $O(\log(D\varepsilon^{-1}))$ rounds, where D is the diameter of $A \cup B$. In each round, we have a parameter $\delta > 0$ that we refer to as the *current scale*, and we also maintain a dual weight $y(b)$ for every point $b \in B$. Initially, in the beginning of the first round, $\delta = \mathrm{D}$ and $y(b) = 0$ for all $b \in B$. Execute the following steps $s = \lceil \log_2(\mathrm{D}\varepsilon^{-1}) \rceil$ times[3].

(i) *Construct a discrete OT instance:* Using the current values of dual weights of B, as described below, construct a discrete distribution $\hat{\mu}_\delta$ with a support set X_δ, where $|X_\delta| = n^{O(d)}$, and define a (discrete) distance function $\mathrm{d}_\delta : X_\delta \times B \to \{0, \ldots, 4n+1\}$.

(ii) *Solve OT instance:* Compute an optimal transport plan between discrete distributions $\hat{\mu}_\delta$ and ν using the procedure PD-OT$(\hat{\mu}_\delta, \nu)$. Let σ_δ be the coupling and $\hat{y} : B \to \mathbb{R}$ be the dual weights returned by the procedure.

(iii) *Update dual weights:* $y(b) \leftarrow y(b) + \delta\hat{y}(b)$ for each point $b \in B$.

(iv) *Update scale:* $\delta \leftarrow \delta/2$.

Our algorithm terminates when $\delta \le \varepsilon$. We now describe the details of step (i) of our algorithm, which is the only non-trivial step. Let $y(\cdot)$ be the dual weights of B at the start of scale δ.

Constructing a discrete OT instance. We construct the discrete instance by constructing a family of Voronoi diagrams and overlaying some of their cells. For a weighted point set $P \subset \mathbb{R}^d$ with weights $w : P \to \mathbb{R}$ and a distance function $\mathrm{d} : \mathbb{R}^d \times P \to \mathbb{R}_{\ge 0}$, we define the *weighted distance* from a point $p \in P$ to any point $x \in \mathbb{R}^d$ as $\mathrm{d}_w(x,p) = \mathrm{d}(x,p) - w(p)$. For a point $p \in P$, its *Voronoi cell* is $\mathrm{Vor}_w(p) = \{x \in \mathbb{R}^d \mid \mathrm{d}_w(x,p) \le \mathrm{d}_w(x,p'), \forall p' \in P\}$, and the *Voronoi diagram* $\mathrm{VD}_w(P)$ is the decomposition of $\mathbb{R}^d$ induced by Voronoi cells; see [22].

For $i \in [1, 4n+1]$ and a point $b \in B$, we define a Voronoi cell V_b^i using a weight function $w_i : B \to \mathbb{R}_{\ge 0}$, as follows. We set $w_i(b) = y(b) + i\delta$ and $w_i(b') = y(b')$ for all $b' \ne b$. We set $V_b^i = \mathrm{Vor}_{w_i}(b)$ in $\mathrm{VD}_{w_i}(B)$. By construction, $V_b^1 \subseteq V_b^2 \subseteq \cdots \subseteq V_b^{4n+1}$. Set $\mathcal{V}_b = \{V_b^i \mid i \in [1, 4n+1]\}$ and $\mathcal{V} = \bigcup_{b \in B} \mathcal{V}_b$ (See Figure 1(a)). Let $\mathcal{A}(\mathcal{V})$ be the *arrangement* of $\mathcal{V}$, the decomposition of $\mathbb{R}^d$ into (connected) cells induced by $\mathcal{V}$; each cell of $\mathcal{A}(\mathcal{V})$ is the maximum connected region lying in the same subset of regions of $\mathcal{V}$ [5].

For each cell φ in $\mathcal{A}(\mathcal{V})$, we choose a representative point r_φ arbitrarily and set its mass to $\hat{\mu}_\delta(r_\varphi) = \mu(\varphi)$, where for any region ρ in $\mathbb{R}^d$, $\mu(\rho) = \int_\rho \mu(a)\, da$ is the mass of μ inside ρ (Here we assume the mass to be 0 outside the support A of μ). Set $X_\delta = \{r_\varphi \mid \varphi \in \mathcal{A}(\mathcal{V})\}$. The resulting mass distribution on X_δ is $\hat{\mu}_\delta$.

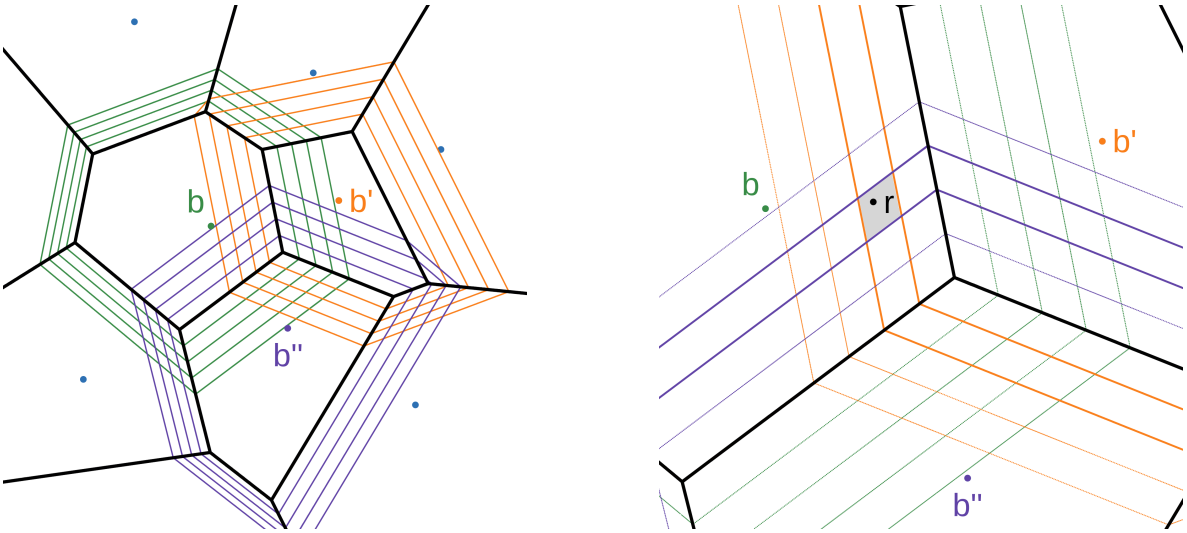
The (discrete) distance $\mathrm{d}_\delta(r, b)$ between any point $b \in B$ and a point $r \in X_\delta$ is defined as

$$\mathrm{d}_\delta(r,b) = \begin{cases} 0, & \text{if } r \in V_b^1, \\ i, & \text{if } r \in V_b^{i+1} \setminus V_b^i, \quad i \in [1, 4n], \\ 4n+1, & \text{if } r \notin V_b^{4n+1}. \end{cases}$$

See Figure 1(b). Since each V_b^i is defined by n algebraic surfaces of constant degree, assuming the bisector of two points under the distance function $\mathrm{d}(\cdot,\cdot)$ is an algebraic variety of constant degree, $\mathcal{A}(\mathcal{V})$ has $n^{O(d)}$ cells and a point in every cell of $\mathcal{A}(\mathcal{V})$ can be computed in $n^{O(d)}$ time [15]. Hence, $|X_\delta| = n^{O(d)}$. This completes the construction of $X_\delta, \hat{\mu}_\delta$, and d_δ.

Computing a semi-discrete transport plan. At the end of any scale δ, we compute a δ-close transport plan τ_δ from the discrete transport plan σ_δ as follows: For any edge $(r_\varphi, b) \in X_\delta \times B$, we arbitrarily transport $\sigma_\delta(r_\varphi, b)$ mass from the points inside the region φ to the point b. A simple construction of such transport plan is to set, for any region φ, any point $a \in \varphi$, and any point $b \in B$, $\tau_\delta(a,b) = \frac{\mu(a)}{\hat{\mu}_\delta(r_\varphi)} \sigma_\delta(r_\varphi, b)$. Our algorithm will only compute the transport plan at the end of the last scale, i.e., $\delta \le \varepsilon$.

[3] Computing an ε-close transport plan requires $O(\log(\mathrm{D}/\varepsilon))$ iterations. When the goal, on the other hand, is to obtain accurate dual weights up to $O(\log \varepsilon^{-1})$ bits, we need to execute our algorithm for $O(\log(n\mathrm{D}/\varepsilon))$ iterations. See Section 2.3.



(a) (b)

Figure 1: (a) The (expanded) Voronoi cells $V_\cdot^i$ of three points $b, b', b'' \in B$, (b) A region $\varphi \in \mathcal{A}(\mathcal{V})$ (highlighted in gray) with a representative point $r \in X_\delta$, where $\mathrm{d}_\delta(b, r) = 0$ since $r \in V_b^1$, $\mathrm{d}_\delta(r, b') = 1$ since $r \in V_{b'}^2 \setminus V_{b'}^1$, and $\mathrm{d}_\delta(r, b'') = 2$ since $r \in V_{b''}^3 \setminus V_{b''}^2$. The ground distance $\mathrm{d}(\cdot, \cdot)$ in this figure is squared Euclidean.

Efficiency analysis. Our algorithm runs $O(\log(\mathrm{D}\varepsilon^{-1}))$ scales, where in each scale, it constructs a discrete OT instance in $n^{O(d)}$ time and solves the OT instance using a polynomial-time primal-dual OT solver. Since the size of the discrete OT instance is $n^{O(d)}$, solving it also takes $n^{O(d)}$ time, resulting in a total execution time of $n^{O(d)} \log(\mathrm{D}\varepsilon^{-1})$ for our algorithm.

2.2 Proof of Correctness. In the discrete setting, cost scaling algorithms obtain an ε-close transport plan that satisfies (2.2) and an additive ε relaxation of (2.1). For our proof, we extend these relaxed feasibility conditions to a semi-discrete setting and show that the transport plan computed by our algorithm for a scale δ satisfies these conditions. We use the relaxed feasibility conditions to show that our transport plan is δ-close. Thus, our algorithm returns an ε-close transport plan from μ to ν at the end of the last scale ($\delta \le \varepsilon$).

δ-feasible transport plan. For points $B = \{b_1, b_2, \ldots, b_n\}$, let $w = \langle w_1, \ldots, w_n \rangle$ be an n-dimensional vector representing a weight assignment to the points in B. We say that the vector w is *valid* if each w_i is a non-negative integer multiple of δ and bounded by $(8n + 2)\mathrm{D}$. Consider the set $\mathbb{W}_\delta$ of all valid vectors, i.e., $\mathbb{W}_\delta = (\delta\mathbb{Z} \cap [0, (8n + 2)\mathrm{D}])^n$. For any δ, consider a decomposition of the support A of the continuous distribution μ into a set of regions $\mathscr{A}_\delta$, where each region ϱ in $\mathscr{A}_\delta$ satisfies the following condition:

(P1) Any two points x and y in ϱ have the same weighted nearest neighbor in B with respect to any valid weight vector $w \in \mathbb{W}_\delta$,

where a point b is a *weighted nearest neighbor* of a point $a \in A$ with respect to weights $w(\cdot)$ if $\mathrm{d}_w(a, b) = \min_{b' \in B} \mathrm{d}_w(a, b')$. For a valid vector $w \in \mathbb{W}_\delta$, let $\mathrm{VD}_w(B)$ denote the weighted Voronoi diagram constructed for the points in B with weights w. The partitioning $\mathscr{A}_\delta$ is simply the overlay of all weighted Voronoi diagrams $\mathrm{VD}_w(B)$ across all valid weight vectors $w \in \mathbb{W}_\delta$ (See Figure 2). For each region $\varrho \in \mathscr{A}_\delta$, let r_ϱ denote an arbitrary representative point inside ϱ.

For any point $b \in B$, let $y(b)$ be the dual weight of b, where $y(b)$ is a non-negative integer multiple of δ. For each region $\varrho \in \mathscr{A}_\delta$, we derive a dual weight $y_\delta(r_\varrho)$ for its representative point as follows. Let $b_\varrho \in B$ be the weighted nearest neighbor of r_ϱ with respect to weights $y(\cdot)$. We set the dual weight of r_ϱ as

$$y_\delta(r_\varrho) \leftarrow y(b_\varrho) - \mathrm{d}(r_\varrho, b_\varrho) - \delta. \tag{2.3}$$

We say that a transport plan τ from μ to ν along with the set of dual weights $y(\cdot)$ for points in B is *δ-feasible* if,

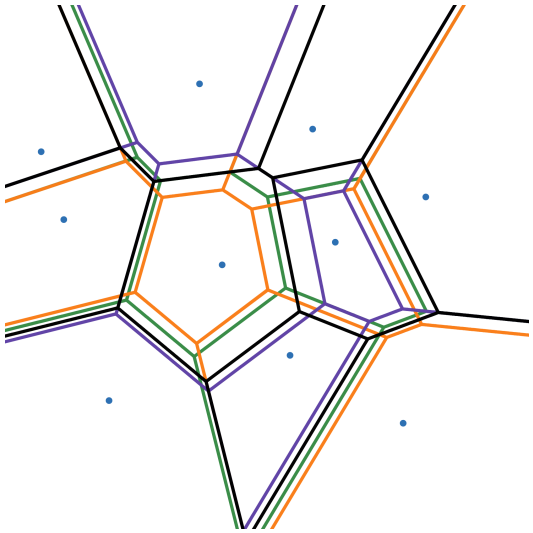


Figure 2: The weighted Voronoi diagrams for four different weight vectors in $\mathbb{W}_\delta$. The ground distance in this figure is squared Euclidean.

for each point $b \in B$ and each region $\varrho \in \mathscr{A}_\delta$,

$$y(b) - y_\delta(r_\varrho) \leq \mathrm{d}(r_\varrho, b) + \delta, \tag{2.4}$$

$$y(b) - y_\delta(r_\varrho) \geq \mathrm{d}(r_\varrho, b) \qquad \text{if } \tau(\varrho, b) > 0. \tag{2.5}$$

In the following lemma, we show that any δ-feasible transport plan $\tau, y(\cdot)$ from μ to ν is δ-close.

LEMMA 2.1. *Suppose $\tau, y(\cdot)$ is any δ-feasible transport plan from μ to ν and let τ^* denote any optimal transport plan from μ to ν. Then, $\mathfrak{c}(\tau) \leq \mathfrak{c}(\tau^*) + \delta$.*

Let $y(\cdot)$ denote the set of dual weights maintained by our algorithm at the beginning of scale δ. For any point $b \in B$ and any region $\varrho \in \mathscr{A}_\delta$, we define a *slack* on condition (2.4) for the pair (ϱ, b), denoted by $s_\delta(\varrho, b)$, as

$$s_\delta(\varrho, b) := \left\lfloor \frac{\mathrm{d}(r_\varrho, b) + \delta - y(b) + y_\delta(r_\varrho)}{\delta} \right\rfloor \delta.$$

Next, we show that for each scale δ, the semi-discrete transport plan τ_δ and dual weights $(y + \delta\hat{y})(\cdot)$ for the points in B computed by our algorithm at the end of the scale is a δ-feasible transport plan.

δ-feasibility of the computed transport plan. We begin by relating the decomposition $\mathscr{A}_\delta$ to the partitioning $\mathcal{A}(\mathcal{V})$ that is constructed in step (i) of our algorithm. We also relate the distance d_δ computed by our algorithm to the slacks s_δ.

In any scale δ, it can be shown that for each point $b \in B$ and each $i \in [1, 4n+1]$, the i-expansion V_b^i can be seen as the Voronoi cell of b in the weighted Voronoi diagram constructed with respect to some valid weight vector in $\mathbb{W}_\delta$. Hence, by the construction of $\mathscr{A}_\delta$, each region $\varrho \in \mathscr{A}_\delta$ completely lies inside some region $\varphi \in \mathcal{A}(\mathcal{V})$, i.e., each region in $\mathcal{A}(\mathcal{V})$ consists of a collection of regions in $\mathscr{A}_\delta$.

We observe that for each point $b \in B$, all regions with a zero slack to b lie inside the 1-expansion V_b^i, all regions with a slack $i\delta$ to b, for $i \in [1, 4n]$, lie inside the region sandwiched between i and $i+1$ expansions of the weighted Voronoi cell of b, and all regions ϱ with a slack $> 4n\delta$ to b are outside V_b^{4n+1}. Using this observation, in the next lemma, we establish a connection between the slacks and the distances d_δ.

LEMMA 2.2. *For any region $\varphi \in \mathcal{A}(\mathcal{V})$, any region $\varrho \in \mathscr{A}_\delta$ inside φ, and any point $b \in B$, if $\mathrm{d}_\delta(r_\varphi, b) \leq 4n$, then $s_\delta(\varrho, b) = \mathrm{d}_\delta(r_\varphi, b)\delta$. Furthermore, if $\mathrm{d}_\delta(r_\varphi, b) = 4n+1$, then $s_\delta(\varrho, b) \geq (4n+1)\delta$.*

Recall that X_δ denotes the set of representative points of the regions in $\mathcal{A}(\mathcal{V})$ and $\hat{\mu}_\delta$ is the discrete distribution over X_δ computed by our algorithm at step (i). In the following lemma, we show that any optimal transport plan σ^* from $\hat{\mu}_\delta$ to ν under distance function d_δ does not transport mass on edges $(r_\varphi, b) \in X_\delta \times B$ with cost $\mathrm{d}_\delta(r_\varphi, b) > 4n$.

LEMMA 2.3. *For any scale δ, let σ^* be any optimal transport plan from $\hat{\mu}_\delta$ to ν. For any point $b \in B$ and any region $\varphi \in \mathcal{A}(\mathcal{V})$, if σ^* transports mass from r_φ to b, then $\mathrm{d}_\delta(r_\varphi, b) \leq 4n$.*

Proof. Let $\tau_{2\delta}, y(\cdot)$ be the 2δ-feasible transport plan computed by our algorithm at scale 2δ. Let $\sigma_{2\delta}$ denote a transformation of $\tau_{2\delta}$ into a discrete transport plan from $\hat{\mu}_\delta$ to ν by simply setting, for each region $\varphi \in \mathcal{A}(\mathcal{V})$, $\sigma_{2\delta}(r_\varphi, b) := \tau_{2\delta}(\varphi, b)$. Let σ^* be any optimal transport plan from $\hat{\mu}_\delta$ to ν, where the cost of each edge (r_φ, b) is set to $\mathrm{d}_\delta(r_\varphi, b)$. Define a directed graph $\mathcal{G}$ on the vertex set $X_\delta \cup B$ as follows. For any pair $(r, b) \in X_\delta \times B$, if $\sigma^*(r, b) > \sigma_{2\delta}(r, b)$, then we add an edge, called a *forward edge*, directed from r to b with a capacity $\sigma^*(r, b) - \sigma_{2\delta}(r, b)$; otherwise, if $\sigma^*(r, b) < \sigma_{2\delta}(r, b)$, then we add an edge, called a *backward edge*, directed from b to r with a capacity $\sigma_{2\delta}(r, b) - \sigma^*(r, b)$. This completes the construction of the directed graph.

For the sake of contradiction, suppose there is a pair $(r^*, b^*) \in X_\delta \times B$ with $\mathrm{d}_\delta(r^*, b^*) > 4n$ such that $\sigma^*(r^*, b^*) > 0$. As shown in the full version of our paper, the edge (r^*, b^*) is a forward edge and is contained in a simple directed cycle $C = \langle b_1, r_1, \ldots, b_k, r_k, b_{k+1} = b_1 \rangle$ in $\mathcal{G}$. Note that by the construction of $\mathcal{G}$, the edges of C alternate between forward and backward edges. Define the cost of the cycle C as

$$w(C) := \sum_{i=1}^{k} \mathrm{d}_\delta(r_i, b_{i+1}) - \sum_{i=1}^{k} \mathrm{d}_\delta(r_i, b_i);$$

i.e., the cost of C is simply the total distance of its forward edges minus the total distance of its backward edges. Since σ^* is an optimal transport plan from $\hat{\mu}_\delta$ to ν, any directed cycle C on $\mathcal{G}$ has a non-positive cost. Since C is a simple cycle, the length of C is at most $2n$. Furthermore, as shown in the full version of our paper, any backward edge has a distance at most 4, i.e., for each $i \in [1, k], \mathrm{d}_\delta(r_i, b_i) \leq 4$. Finally, by construction, all edges have a non-negative distance. Therefore,

$$w(C) = \sum_{i=1}^{k} \mathrm{d}_\delta(r_i, b_{i+1}) - \sum_{i=1}^{k} \mathrm{d}_\delta(r_i, b_i) \geq \mathrm{d}_\delta(r^*, b^*) - \sum_{i=1}^{k} 4 \geq \mathrm{d}_\delta(r^*, b^*) - 4n > 0,$$

which is a contradiction of the fact that all simple cycles have a non-positive cost. Hence, σ^* cannot transport mass on edges (r^*, b^*) with distance $\mathrm{d}_\delta(r^*, b^*) > 4n$. ∎

Let $\sigma_\delta, \hat{y}(\cdot)$ be the optimal transport plan from $\hat{\mu}_\delta$ to ν computed at step (ii) of our algorithm, and recall that τ_δ is the transport plan from μ to ν computed at the end of scale δ. In the following lemma, we show that $\tau_\delta, (y + \delta\hat{y})(\cdot)$ is a δ-feasible transport plan.

LEMMA 2.4. *For each scale δ, let $(y + \delta\hat{y})(\cdot)$ denote the set of dual weights for points in B computed at step (iii) of our algorithm. Then, the transport plan $\tau_\delta, (y + \delta\hat{y})(\cdot)$ is a δ-feasible transport plan.*

Proof. Let $y_\delta(\cdot)$ denote the set of dual weights derived for the representative points of regions in $\mathscr{A}_\delta$ using Equation (2.3) at the beginning of scale δ. Consider a set of dual weights y'_δ that assigns, for each region $\varrho \in \mathscr{A}_\delta$ inside a region $\varphi \in \mathcal{A}(\mathcal{V})$, a dual weight $y'_\delta(r_\varrho) := y_\delta(r_\varrho) + \delta\hat{y}(r_\varphi)$. In what follows, we show that the transport plan τ_δ along with dual weights $(y + \delta\hat{y})(\cdot)$ and $y'_\delta(\cdot)$ satisfy δ-feasibility conditions (2.4) and (2.5). Using this, in the full version of our paper, we show that by reassigning the dual weights of the representative points of regions in $\mathscr{A}_\delta$ as in Equation (2.3), the δ-feasibility conditions (2.4) and (2.5) remain satisfied; hence, we conclude that $\tau, (y + \delta\hat{y})(\cdot)$ is δ-feasible, as claimed.

For any region $\varphi \in \mathcal{A}(\mathcal{V})$, any region $\varrho \in \mathscr{A}_\delta$ inside φ, and any point $b \in B$,

- by Lemma 2.2, $\mathrm{d}_\delta(r_\varphi, b)\delta \le s_\delta(\varrho, b)$. Combining with feasibility condition (2.1),

$$\begin{aligned}(y+\delta\hat{y})(b) - y'_\delta(r_\varrho) &= (y(b)+\delta\hat{y}(b)) - (y_\delta(r_\varrho)+\delta\hat{y}(r_\varphi))\\ &= (y(b)-y_\delta(r_\varrho)) + \delta(\hat{y}(b)-\hat{y}(r_\varphi))\\ &\le (y(b)-y_\delta(r_\varrho)) + \mathrm{d}_\delta(r_\varphi,b)\delta \le (y(b)-y_\delta(r_\varrho)) + s_\delta(\varrho,b)\\ &\le (y(b)-y_\delta(r_\varrho)) + (\mathrm{d}(r_\varrho,b)+\delta-y(b)+y_\delta(r_\varrho))\\ &= \mathrm{d}(r_\varrho,b)+\delta,\end{aligned}$$

  leading to δ-feasibility condition 2.4.

- if $\tau_\delta(\varrho, b) > 0$, then σ_δ transports mass from r_φ to b, i.e., $\sigma_\delta(r_\varphi, b) > 0$. In this case, by Lemma 2.3, $\mathrm{d}_\delta(r_\varphi, b) \le 4n$ and by Lemma 2.2, $s_\delta(\varrho, b) = \mathrm{d}_\delta(r_\varphi, b)\delta$. Combining with feasibility condition (2.2),

$$\begin{aligned}(y+\delta\hat{y})(b) - y'_\delta(r_\varrho) &= (y(b)+\delta\hat{y}(b)) - (y_\delta(r_\varrho)+\delta\hat{y}(r_\varphi))\\ &= (y(b)-y_\delta(r_\varrho)) + \delta(\hat{y}(b)-\hat{y}(r_\varphi))\\ &= (y(b)-y_\delta(r_\varrho)) + \mathrm{d}_\delta(r_\varphi,b)\delta = (y(b)-y_\delta(r_\varrho)) + s_\delta(\varrho,b)\\ &\ge (y(b)-y_\delta(r_\varrho)) + (\mathrm{d}(r_\varrho,b)-y(b)+y_\delta(r_\varrho))\\ &= \mathrm{d}(r_\varrho,b),\end{aligned}$$

  leading to δ-feasibility condition 2.5.

□

2.3 Computing Optimal Dual Weights. In this section, we show that in addition to computing an ε-close transport cost in the semi-discrete setting, our algorithm can also compute the set of dual weights for the points in B accurately, up to $O(\log \varepsilon^{-1})$ bits. To obtain such accurate set of dual weights, we execute our algorithm for $O(\log(n\mathrm{D}/\varepsilon))$ iterations so that the final value of δ when the algorithm terminates is at most $\varepsilon/5n$. In the following, we show that the dual weight computed for each point in B at the last scale is ε-close to the optimal dual weight value.

Note that any edge in the graph constructed in Step (i) of our algorithm has a cost at most $4n + 1$. Consequently, in Step (ii), the largest dual weight returned by the primal-dual solver is at most $4n + 1$[4] and in Step (iii), the dual weight of any point $b \in B$ changes by at most $(4n+1)\delta$. Since the dual weight of b becomes the optimal dual weight in the limit, to bound the difference between the current dual weight and the optimal, it suffices if we bound the total change in the dual weights for all scales after scale $\delta \le \varepsilon/5n$. The difference between the optimal dual weight and the current dual weight is at most

$$(4n+1)\sum_{i=1}^{\infty} \delta/2^i = (4n+1)\delta \le (4n+1)(\varepsilon/5n) \le \varepsilon.$$

Therefore, after $O(\log(n\mathrm{D}/\varepsilon))$ iterations of the algorithm, the difference in the optimal dual weight $y(b)$ and the current dual weight of b is at most ε.

3 Approximation Algorithm for Semi-Discrete Optimal Transport

In this section, we present our second approximation algorithm for the semi-discrete setting that computes an ε-OT plan in $n\varepsilon^{-O(d)}\mathrm{poly}\log(n)$ expected time. We begin by describing a few notations that help us in presenting the algorithm. Let μ, ν, A, and B be the same as above. For any point $b \in \mathbb{R}^d$ and any $r \ge 0$, let $D(b, r)$ denote the Euclidean ball of radius r centered at b. Any pair of point sets $P, Q \subset \mathbb{R}^d$ is called *ε-well separated* if $\max\{\mathrm{diam}(P), \mathrm{diam}(Q)\} \le \varepsilon \cdot \min_{(p,q)\in P\times Q} \|p-q\|$. Given a set S of n points in $\mathbb{R}^d$ and a parameter ε, a collection $W = \{(P_1, Q_1), \ldots, (P_k, Q_k)\}$ is an *ε-well separated pair decomposition* (ε-WSPD) of S if *(i)* each pair (P_i, Q_i) is ε-well separated, and *(ii)* for any distinct $p, q \in S$, there exists a pair $(P_i, Q_i) \in W$ where $p \in P_i$ and $q \in Q_i$. Given

[4]Any set of dual weights returned by the algorithm can be translated by a fixed value so that the smallest dual weight becomes 0. Assuming this, it is easy to see that the largest dual weight is $4n + 1$.

a point set $B \subset \mathbb{R}^d$ and a hypercube $\square$, we say that $\square$ is ε-*close to* $b \in B$ if $\max_{a\in\square} \|b-a\| \le \varepsilon \min_{b'\neq b\in B} \|b'-b\|$. For any parameter $\delta > 0$, let $\mathbb{G}_\delta$ denote an axis-aligned grid of side-length δ with a vertex at the origin, i.e., $\mathbb{G}_\delta := [0,\delta]^d + \mathbb{Z}^d$. In the remainder of this section, we present our algorithm and analyze its correctness and efficiency.

3.1 Algorithm. Here is a brief overview of our algorithm. Let H be a hypercube of side-length $\frac{4}{\varepsilon}\mathrm{diam}(B)$ centered at one of the points of B. First, we partition H into a collection of hypercubes such that for each $b \in B$ and all hypercubes $\square$ except the ones that are $(\beta_2\varepsilon)$-close to b for some constant $\beta_2 > 0$ (see below), the following condition holds: for all $p, q \in \square$, $\|b-p\| \le (1+\varepsilon)\|b-q\|$. If a hypercube $\square$ is $(\beta_2\varepsilon)$-close to $b \in B$, then we greedily route the mass of μ inside $\square$ to b. We then construct a discretization $\hat{\mu}$ of the remaining mass from μ by collapsing the mass $\mu(\square)$ of each cell $\square$ to its center point $c_\square$. We compute an ε-OT plan σ from $\hat{\mu}$ to ν using the algorithm described in Section 4 and transform σ into a semi-discrete transport plan τ_σ by dispersing the mass transportation throughout each hypercube, as described in Section 2.1. We now describe the algorithm in more detail.

Construction of hypercubes. First, we construct an $(\frac{\varepsilon}{4})$-WSPD W of B using the algorithm by Callahan and Kosaraju [16]. For every pair $(B_1, B_2) \in W$, we construct a set of hypercubes by closely following the construction of an approximate Voronoi diagram [10], as follows. Let $b_1 \in B_1$ and $b_2 \in B_2$ denote arbitrary representative points of B_1 and B_2, respectively. Let β_0, β_1 be some fixed constants. For any integer $i = 0, \ldots, t = \beta_1 \log(\varepsilon^{-1})$, define $\delta_i = 2^i \frac{\beta_0\varepsilon}{2\sqrt{d}}\|b_1 - b_2\|$ and let $\mathcal{G}_i(B_1, B_2)$ denote the set of hypercubes of the grid $\mathbb{G}_{\varepsilon\delta_i}$ intersecting $D(b_1, \delta_i) \cup D(b_2, \delta_i)$. For any cell $\square \in \mathcal{G}_i(B_1, B_2)$, if there exists a child cell $\square' \subset \square$ in $\mathcal{G}_{i-1}(B_1, B_2)$, then we replace $\square$ with its 2^d children cells to keep all hypercubes interior disjoint. Set

$$\mathcal{G} = \bigcup_{(B_1,B_2)\in W} \bigcup_{i=0}^{t} \mathcal{G}_i(B_1, B_2).$$

Transporting local mass. For any point $b \in B$ and some sufficiently small constant $\beta_2 > 0$, define its local neighborhood to be

$$\mathcal{N}_\varepsilon(b) = \{\square \in \mathcal{G} : \square \text{ is } (\beta_2\varepsilon)\text{-close to } b\}.$$

For each $b \in B$, we transport the mass locally as follows. If $\nu(b) > 0$ and there exists a hypercube $\square \subseteq \mathcal{N}_\varepsilon(b)$ with $\mu(\square) > 0$, we transport $\min\{\mu(\square), \nu(b)\}$ mass from $\square$ to b. If $\mu(\square) \le \nu(b)$, we set $\nu(b) = \nu(b) - \mu(\square)$, delete $\square$ from $\mathcal{G}$, and repeat the above step. If $\mu(\square) > \nu(b)$, we set $\nu(b) = 0$ and scale the mass in $\square$ down so that $\mu(\square) = \mu(\square) - \nu(b)$. This process stops when either $\nu(b) = 0$ or no cell of $\mathcal{G}$ lies inside $\mathcal{N}_\varepsilon(b)$.

Discrete OT on remaining demand. Let μ' and ν' be the two distributions after transporting the local mass. Note that μ' and ν' are not necessarily probability distributions, i.e., the mass of each one of them might not add up to 1; however, the total mass in μ' equals that of ν'. Let $\mathcal{G}$ be the set of remaining hypercubes. Let $\hat{A} = \{c_\square : \square \in \mathcal{G}\} \cup \{c_0\}$ for some $c_0 \in A \setminus H$, where $c_\square$ denotes the center of $\square$. Define $\hat{\mu}(c_\square) = \int_\square \mu'(a)\, da$ for every hypercube $\square \in \mathcal{G}$ and let $\hat{\mu}(c_0) = \int_A \mu'(a)da - \sum_{\square\in\mathcal{G}} \hat{\mu}(c_\square)$. We compute a $(1+\beta_3\varepsilon)$-approximate discrete transport plan σ from $\hat{\mu}$ to ν', for some constant $\beta_3 < 1$, using the algorithm described in Section 4. We then convert σ into a semi-discrete transport plan τ_σ in a straightforward manner, similar to Section 2.1. We return a transport plan $\widetilde{\tau}$ obtained from combining τ_σ with the local mass transportation committed in the previous step in a straight-forward manner. It is easy to confirm that the transport plan $\widetilde{\tau}$ is a transport plan from μ to ν. This completes the description of our algorithm.

3.2 Proof of Correctness. In this section, we show that the transport plan computed by our algorithm is a $(1+\varepsilon)$-approximate transport plan from μ to ν. Recall that as a first step, our algorithm constructs a family $\mathcal{G}$ of hypercubes. In the following lemma, we enumerate useful properties of these hypercubes.

LEMMA 3.1. *There exist constants β_4, β_5 depending only on $\beta_0, \beta_1, \beta_2, \beta_3$ such that for each $\square \in \mathcal{G}$ the hypercube $\square$ satisfies at least one of the following two conditions:*

1. *For any two points $a_1, a_2 \in \square$ and any $b \in B$, $\|a_1 - b\| \le (1+\beta_4\varepsilon)\|a_2 - b\|$,*

2. There exists some $b \in B$ such that $\|a - b\| \leq \beta_5 \varepsilon \min_{b' \neq b} \|b' - b\|$ for all $a \in \square$.

We then use a simple triangle inequality argument similar to [6] to show that a greedy routing on $\mathcal{N}_\varepsilon(b)$ only incurs another small relative error.

LEMMA 3.2. *Let τ^* be an optimal transport plan between μ and ν, and let $\widetilde{\tau}$ be the transport plan returned by the algorithm. There exists a transport plan $\hat{\tau}$ such that (i) $\hat{\tau} = \widetilde{\tau}$ when restricted to $\bigcup_{b \in B} \mathcal{N}_\varepsilon(b)$, and (ii) there exists some constant β_6 depending only on $\beta_0, \beta_1, \beta_2, \beta_3$ where $¢(\hat{\tau}) \leq (1 + \beta_6 \varepsilon) ¢(\tau^*)$.*

We next show that any mass outside of H can be routed arbitrarily while incurring small relative error because any two points $b_1, b_2 \in B$ are approximately equidistant from any $a \in A \setminus H$.

LEMMA 3.3. *Let $\widetilde{\tau}$ be the semi-discrete transport plan constructed by our algorithm. Let τ be any arbitrary transport plan. Then for some constant $\beta_7 < 1$ depending only on $\beta_0, \beta_1, \beta_2, \beta_3$,*

$$\sum_{b \in B} \int_{A \setminus \bigcup_{\square \in \mathcal{G}} \square} \|a - b\| \cdot \widetilde{\tau}(a, b)\, da \leq (1 + \beta_7 \varepsilon) \sum_{b \in B} \int_{A \setminus \bigcup_{\square \in \mathcal{G}} \square} \|a - b\| \cdot \tau(a, b)\, da.$$

Finally, we consider the mass that lies inside H but does not lie in a cell of $\mathcal{G}$ that is $(\beta_2 \varepsilon)$-close to a point of B that has survived. We use the fact that all points within such a cell $\square$ of $\mathcal{G}$ are roughly at the same distance from a point of B, i.e. for any $p, q \in \square$ where $\mu'(\square) > 0$ and for any $b \in B$ where $\nu'(b) > 0$, $\|p - q\| \leq (1 + \beta_2 \varepsilon)\|q - b\|$.

LEMMA 3.4. *Let $\hat{\tau}'$ be a transport plan between μ' and ν' defined by $\hat{\tau}'(a, b) = \hat{\tau}(a, b)$ if $a \notin \mathcal{N}_\varepsilon(b)$ and $\hat{\tau}'(a, b) = 0$ otherwise. Then $¢(\tau_\sigma) \leq (1 + \beta_8 \varepsilon) ¢(\hat{\tau}')$ for some constant $\beta_8 > 0$ depending only on $\beta_0, \beta_1, \beta_2, \beta_3$.*

Lemmas 3.2-3.4 together imply that our algorithm returns an ε-OT plan.

LEMMA 3.5. *Let $\widetilde{\tau}$ be the transport plan computed by our algorithm, and let τ^* be an optimal transport plan between μ and ν. Then $¢(\widetilde{\tau}) \leq (1 + \varepsilon) ¢(\tau^*)$.*

3.3 Efficiency analysis. Callahan and Kosaraju [16] have shown that an $(\frac{\varepsilon}{4})$-WSPD W of S of size $O(n\varepsilon^{-d})$ can be constructed in $O(n(\varepsilon^{-d} + \log n))$ time. For each pair in W, our algorithm computes $O(\log \varepsilon^{-1})$ approximate balls, where for each approximate ball, our algorithm adds $O(\varepsilon^{-d})$ hypercubes to $\mathcal{G}$. Therefore, the collection $\mathcal{G}$ of hypercubes has size $O(n\varepsilon^{-2d} \log \varepsilon^{-1})$. Hence, partitioning the hypercube H takes $O(n(\log n + \varepsilon^{-2d} \log \varepsilon^{-1}))$ time. Furthermore, computing the mass of μ inside each hypercube take $O(n\varepsilon^{-2d} \log \varepsilon^{-1} Q)$ time. Finally, note that the discrete OT instance computed by our algorithm has size $O(n\varepsilon^{-2d} \log \varepsilon^{-1})$ and hence, can be solved in $O(n\varepsilon^{-4d-5} \log(n) \log^{2d+5}(\log n) \log(\varepsilon^{-1}))$ time using the algorithm in Section 4 when the spread of B is polynomially bounded, leading to Theorem 1.2.

4 A Near-Linear ε-Approximation Algorithm for Discrete OT

In this section, we present a randomized Monte-Carlo $(1+\varepsilon)$-approximation algorithm for the discrete OT problem. We now let μ, ν be two discrete distributions with support sets A and B, respectively, which are finite point sets in $\mathbb{R}^d$. Set $n = |A| + |B|$. We first present an overview of the algorithm, then provide details of the various steps, and finally analyze its correctness and efficiency. Our algorithm can be seen as an adaptation of the boosting framework presented by Zuzic [50] to the discrete OT problem; we present an $O(\log \log n)$-approximation algorithm for the discrete OT problem and then boost the accuracy of our algorithm using the multiplicative-weight-update method and compute a $(1 + \varepsilon)$-approximate OT plan.

4.1 Overview of the Algorithm. At a high level, we compute a hierarchical graph $\mathcal{G} = (V, E)$, where $V \supseteq A \cup B$ is a set of points in $\mathbb{R}^d$. The weight of an edge is the Euclidean distance between its endpoints. The construction of $\mathcal{G}$ is randomized, and $\mathcal{G}$ is a $(1 + \varepsilon)$-*spanner* in expectation, i.e., $d_{\mathcal{G}}(a, b)$, the shortest-path distance between $(a, b) \in A \times B$ in $\mathcal{G}$ satisfies the condition

$$\|a - b\| \leq \mathbb{E}[d_{\mathcal{G}}(a, b)] \leq (1 + \varepsilon)\|a - b\|.$$

We formulate the OT problem as a min-cost flow problem in $\mathcal{G}$ by setting $\eta(u) = \mu(u)$ if $u \in A$ and $\eta(u) = -\nu(u)$ if $u \in B$. Following a bottom-up greedy approach, we construct a flow $\sigma \colon V \to \mathbb{R}_{\geq 0}$ and dual weights $y : V \to \mathbb{R}$ that satisfy (C1) and (C2) with $\rho = a_1 \log \log n$, where $a_1 > 0$ is a constant:

(C1) $|y(u) - y(v)| \le \rho\|u - v\| \quad \forall (u,v) \in E,$

(C2) $\sum_{(u,v)\in E} \sigma(u,v)\|u - v\| \le \sum_{u\in V} y(u)\eta(u).$

The first condition guarantees the dual solution y is ρ-approximately feasible, while the second condition guarantees that y is non-trivial and the flow σ is a ρ-approximation. Using such a primal-dual solution, one can use multiplicative-weight-update method (MWU) [9] to boost a ρ-approximate flow into a $(1+\varepsilon)$-approximate flow on $\mathcal{G}$ by making $O(\rho^2\varepsilon^{-2}\log n)$ calls to our greedy primal-dual approximation algorithm. We also describe the MWU procedure in Section 4.4. Once a $(1+\varepsilon)$-approximate flow is obtained in $\mathcal{G}$, then one can simply shortcut paths in $\mathcal{G}$ to obtain an ε-OT plan; see e.g. [24]. We remark that a $(1+\varepsilon)$-spanner is not needed if only a $O(\log\log n)$-approximation is desired. An $O(\log\log n)$-OT plan can be constructed directly in $O(n\log\log n)$ time using our algorithm. We now describe the details of our algorithm.

4.2 Constructing a spanner. We now define the construction of the graph $\mathcal{G}$, which is built upon a hierarchical partitioning of $\mathbb{R}^d$ and the tree $\mathcal{T}$ associated with it. Let $\bar{\varepsilon} = \beta\varepsilon$ for some small enough constant β. We only rescale ε to guarantee that the resulting transport plan is $(1+\varepsilon)$-approximate.

Hierarchical partitioning. For simplicity, we refer to all d-dimensional hypercubes as cells. For any cell $\square$, let $\ell_\square$ and $c_\square$ denote its side-length and center, respectively. Let $\Delta = \frac{\max_{p,q\in A\cup B}\|p-q\|}{\min_{p,q\in A\cup B}\|p-q\|}$ denote the *spread* of $A\cup B$. We assume $\Delta = n^{O(1)}$. Additionally, define $\mathbb{G}(\square,\ell)$ to be the grid that partitions $\square$ into new cells of side-length ℓ. Without loss of generality, assume $A\cup B \subseteq [0,\Delta]^d$.

Let $\square^*$ be a randomly shifted cell of side-length 2Δ containing all points in $A\cup B$, i.e., $\square^* = [0,2\Delta]^d - x$ for some x chosen uniformly at random from the hypercube $[0,\Delta]^d$. We construct a hierarchical partition of $\square^*$ as follows. We designate $\square^*$ as the root cell of $\mathcal{T}$. For any cell $\square$ of $\mathcal{T}$, define $n_\square := |(A\cup B)\cap\square|$ as the number of points of $A\cup B$ contained within $\square$. We construct $\mathcal{T}$ recursively as follows. If $n_\square \le \left(\bar{\varepsilon}^{-1}\log\log n\right)^d$, $\square$ is a leaf of $\mathcal{T}$. Otherwise, using the grid $\mathbb{G}_\square = \mathbb{G}\left(\square, \ell_\square/n_\square^{\frac{1}{3d}}\right)$, we partition $\square$ into smaller cells of side-length $\ell_\square/n_\square^{\frac{1}{3d}}$. We add all non-empty cells of $\mathbb{G}_\square$ to $\mathcal{T}$ as the children of $\square$ and denote them by $\mathsf{C}[\square]$. The height h of $\mathcal{T}$ is $h = O(\log\log n)$.

For any cell $\square$ of $\mathcal{T}$, we define a set of $O((\bar{\varepsilon}^{-1}dh)^d)$ equal-sized subcells as follows. Define $\delta_\square = \frac{\bar{\varepsilon}\ell_\square}{4dh}$ to be the side-length of the subcells of $\square$. We add all the cells of the grid $\mathbb{G}(\square,\delta_\square)$ that contain a point of $A\cup B$ as the subcells of $\square$ and denote the resulting family by $\mathsf{S}[\square]$.

Vertices and edges of the graph. The vertex set of $\mathcal{G}$ consists of the points $A\cup B$ plus the center point of all non-empty cells and subcells of $\mathcal{T}$. More precisely,

$$V = (A\cup B)\cup\bigcup_{\square\in\mathcal{T}}\{c_\square\}\cup\{c_\xi : \xi\in\mathsf{S}[\square]\}.$$

The edge set of $\mathcal{G}$ consists of two sets of edges per cell of $\mathcal{T}$.

1. If $\square$ is a non-leaf cell, let $\mathcal{I}_\square = c_\square\cup\left(\bigcup_{\square'\in\mathsf{C}[\square]} c_{\square'}\right)\cup\left(\bigcup_{\xi\in\mathsf{S}[\square]} c_\xi\right)$ be the set of points composed of the center of $\square$, centers of its children, and the centers of the subcells of $\square$. Otherwise, let $\mathcal{I}_\square = c_\square\cup((A\cup B)\cap\square)$. We construct a $(1+\bar{\varepsilon})$-spanner $\mathcal{S}_\square$ on $\mathcal{I}_\square$. We add all edges of $\mathcal{S}_\square$ to $\mathcal{G}$ and refer to them as *greedy edges*. Note that $|\mathcal{I}_\square| = |\mathsf{C}[\square]| + |\mathsf{S}[\square]| + 1 = O(n_\square^{1/3} + (h/\bar{\varepsilon})^d)$ for any non-leaf cell.
2. In addition, for any non-leaf cell $\square$, let $X_\square = \bigcup_{\square'\in\mathsf{C}[\square]}\bigcup_{\xi\in\mathsf{S}[\square']} c_\xi$ be the set of centers of the subcells of the children of $\square$. Let $\mathcal{S}'_\square$ be a $(1+\bar{\varepsilon})$-spanner constructed on the points in $X_\square$. We add all the edges of $\mathcal{S}'_\square$ to $\mathcal{G}$ and refer to them as *shortcut edges*.

Recall that the weight of every edge in $\mathcal{G}$ is the Euclidean distance between its endpoints. The greedy edges are the edges that our greedy algorithm uses to compute a flow, whereas the shortcut edges guarantee that the shortest-path distances in $\mathcal{G}$ are a $(1+\varepsilon)$-approximation of the Euclidean distances in expectation. We remark that the shortcut edges are used only when we apply the MWU method to obtain a $(1+\varepsilon)$-approximate OT plan.

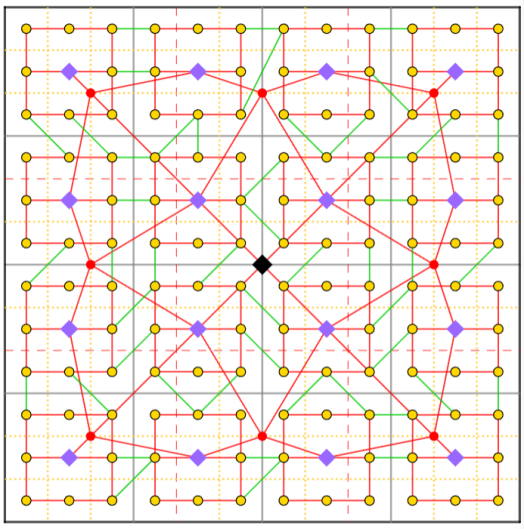


Figure 3: The hierarchical structure of the graph $\mathcal{G}$. The vertices of the graph are the centers of the cells (diamonds) and centers of subcells (disks). For any cell $\square$, the greedy edges form a spanner on its children and subcells (solid red lines) and the shortcut edges cross between distinct children (solid green lines).

For any pair $(a, b) \in A \times B$, let $\Pi_{a,b}$ be the shortest path in $\mathcal{G}$ from a to b, and let $\phi(a, b)$ be the cost of $\Pi_{a,b}$. The following lemma bounds the size of $\mathcal{G}$ and shows that the shortest path metric of $\mathcal{G}$, in expectation, $(1 + \varepsilon)$-approximates Euclidean distances.

LEMMA 4.1. *The graph $\mathcal{G}$ contains $O(nh)$ vertices and $O(n\varepsilon^{-d}h)$ edges. The max degree of any vertex in $\mathcal{G}$ is at most $O(\varepsilon^{-d} \log n)$. Furthermore, for any pair of points (a, b), $\phi(a, b) \geq \|a - b\|$ and $\mathbb{E}[\phi(a, b)] \leq (1 + \varepsilon)\|a - b\|$.*

4.3 Greedy Primal-Dual Algorithm. Given the graph $\mathcal{G} = (V, E)$ and a demand function $\eta \colon V \to \mathbb{R}$, we compute a primal-dual flow (σ, y) on $\mathcal{G}$ that routes the demand function η and satisfies the conditions (C1) and (C2). To do so, for every cell $\square$ of $\mathcal{T}$, we construct a balanced instance of the min-cost flow problem on $\mathcal{S}_\square$ and solve the instance using Orlin's algorithm. Since each greedy edge in $\mathcal{G}$ belongs to the spanner $\mathcal{S}_\square$ of a unique cell $\square$, the combination of all flows computed for all cells of $\mathcal{T}$ routes the demand η. We describe the details next.

We compute the primal-dual pair (σ, y) in a bottom-up manner. For each cell $\square$, we construct a balanced demand function $\eta_\square$ on the point set $\mathcal{I}_\square$. In our construction, we treat the center point of $\square$ as a sink, which absorbs all unrouted mass among its subcells and children (or points, if $\square$ is a leaf). This mass is then routed in the instance computed at the parent of $\square$. More precisely, we define $\eta_\square$ as follows. For any subcell $\xi \in \mathsf{S}[\square]$, we simply set $\eta_\square(c_\xi) := \eta(c_\xi)$. If $\square$ is a leaf cell, then for any point $v \in (A \cup B) \cap \square$, we set $\eta_\square(v) := \eta(v)$. Otherwise, $\square$ is a non-leaf cell and for each child $\square' \in \mathsf{C}[\square]$, we set $\eta_\square(c_{\square'}) = \eta(c_{\square'}) - \eta_{\square'}(c_{\square'})$. Finally, for the center point $c_\square$, we set $\eta_\square(c_\square) = -\sum_{x \in \mathcal{I}_\square \setminus \{c_\square\}} \eta_\square(x)$. This completes the construction of $\eta_\square$. The pair $(\mathcal{S}_\square, \eta_\square)$ is a balanced instance for the min-cost flow.

For every cell $\square$, we compute a primal-dual flow $(\sigma_\square, y_\square)$ using Orlin's min-cost flow algorithm on the instance $(\mathcal{S}_\square, \eta_\square)$ [40]. We construct a set of dual weights $y(\cdot)$ for the points in V in a top-down manner. For a cell $\square$ and any point $u \in \mathcal{I}_\square$, we define the dual weight of u as $y(u) \leftarrow y_\square(u) - y_\square(c_\square) + y(c_\square)$. The definition of $y(\cdot)$ adjusts dual weights based on the intersection $\mathcal{I}_\square \cap \mathcal{I}_{\square'} = \{c_{\square'}\}$ for every non-leaf cell $\square$ and every $\square' \in \mathsf{C}[\square]$. Additionally, observe that each greedy edge (u, v) of $\mathcal{G}$ belongs to a unique min-cost flow instance $(\mathcal{S}_\square, \eta_\square)$. This is because $|\mathcal{I}_{\square_1} \cap \mathcal{I}_{\square_2}| \leq 1$ for all pairs of cells $\square_1, \square_2 \in \mathcal{T}$. For any greedy edge $(u, v) \in E$, define $\square_{uv}$ to be the unique cell of $\mathcal{T}$ containing the edge (u, v). We set $\sigma(u, v) = \sigma_{\square_{uv}}(u, v)$ for every greedy edge $(u, v) \in E$, and $\sigma(u, v) = 0$ for every shortcut edge $(u, v) \in E$. This completes the construction of our greedy primal-dual algorithm.

4.4 Multiplicative Weights Update (MWU) Framework. Using Indyk and Thaper's greedy algorithm [28], we first compute an estimate of the OT cost within a $O(\log n)$ factor in $O(n \log n)$ time, i.e. we compute a value $\tilde{g}$ such that $¢(\tau^*) \le \tilde{g} \le O(\log n) \cdot ¢(\tau^*)$. Using this estimate, we perform an exponential search in the range $\left[\frac{\tilde{g}}{O(\log n)}, \tilde{g}\right]$ with increments of factor $(1+\varepsilon)$. For any guess value g, the MWU algorithm either returns a flow $\sigma\colon E \to \mathbb{R}$ with $¢(\sigma) \le (1+\varepsilon)g$ or returns dual weights as a certificate that $g < ¢(\tau^*)$. We now describe the MWU algorithm for a fixed value of g.

Set $T = \lceil 4\rho^2 \varepsilon^{-2} \log |E| \rceil$. The algorithm runs in at most T iterations, where in each iteration, it maintains a *pre-flow* vector σ^t, i.e., a vector over the edges where inflow and outflow may not sum to the demand of a vertex, satisfying $¢(\sigma^t) \le g$. Initially, set $\sigma^0(u,v) = \frac{g}{\|u-v\| \cdot |E|}$ for each edge $(u,v) \in E$. For each iteration t, define the *residual demand* $\eta^t_{\text{res}}(\cdot)$ as

$$\eta^t_{\text{res}}(u) = \eta(u) - \sum_{v:(u,v)\in E} (\sigma^{t-1}(u,v) - \sigma^{t-1}(v,u)).$$

Let $(\sigma^t_{\text{res}}, y^t)$ be the primal-dual flow computed by our greedy algorithm for the residual demands η^t_{res}. Recall that $(\sigma^t_{\text{res}}, y^t)$ satisfies (C1) and (C2). If $\langle \eta^t_{\text{res}}, y^t \rangle \le \varepsilon g$, then (C2) implies that $¢(\sigma^t_{\text{res}}) \le \varepsilon g$. Since σ^t_{res} routes the residual demands, the flow function $\widetilde{\tau} = \sigma^{t-1} + \sigma^t_{\text{res}}$ routes the original demand η with a cost $¢(\sigma^t) \le (1+\varepsilon)g$. In this case, the algorithm returns $\widetilde{\tau}$ as the desired flow and terminates.

Otherwise, $\langle \eta^t_{\text{res}}, y^t \rangle > \varepsilon g$ and we update the flow along each edge $e = (u,v)$ of G based on the slack $s^t(u,v) = \frac{y^t(u) - y^t(v)}{\|u-v\|}$ of e with respect to dual weights y^t:

$$\sigma^t(u,v) \leftarrow \exp\left(\frac{\varepsilon}{2\rho^2} s^t(u,v)\right) \cdot \sigma^{t-1}(u,v).$$

We emphasize that flow along an edge is increasing if the slack is large. Then, one needs to rescale σ^t so that its cost is bounded above by g. If the algorithm does not terminate within T rounds, we conclude that the value of g is an under-estimate of the cost of the min-cost flow; we increase g by a factor of $(1+\varepsilon)$ and repeat the MWU algorithm. This completes the description of the MWU framework.

4.5 Analysis. The following two lemmas prove that our algorithm satisfies conditions (C1) and (C2) for a sufficiently small approximation factor.

LEMMA 4.2. *For any edge $(u,v) \in E$, $|y(u) - y(v)| \le O(d^{3/2} h \varepsilon^{-1}) \|u - v\|$.*

LEMMA 4.3. $\sum_{(u,v)\in E} \sigma(u,v)\|u-v\| \le \sum_{u \in V} y(u)\eta(u)$.

The following lemma, which just combines the inequalities of Lemmas 4.2 and 4.3, guarantees that our greedy primal-dual algorithm is a $O(\log \log n)$ approximation.

LEMMA 4.4. *Suppose σ^* is an optimal discrete transport plan between μ and ν, and (σ, y) is a primal-dual pair computed by our greedy algorithm. Then,*

$$\mathbb{E}\left[\sum_{(u,v)\in E} \sigma(u,v)\|u-v\|\right] \le O(d^{3/2}\varepsilon^{-1} \log\log n)\, ¢(\sigma^*).$$

It is shown in [50] that a ε-OT plan can be obtained from our greedy algorithm using the MWU method and shortcuting the resulting approximate min-cost flow. By Lemma 4.1, the cost of the approximate min-cost flow on $\mathcal{G}$ is a $(1+O(\varepsilon))$-approximation of the OT cost in expectation.

Next, we bound the running time of our greedy algorithm. For any cell $\square$, the algorithm computes an exact primal-dual solution to min-cost flow on $\mathcal{I}_\square$ with demands $\eta_\square(\cdot)$ using Orlin's algorithm [40] in $O(|\mathcal{I}_\square|^3)$ time. Each leaf cell $\square$ satisfies $|\mathcal{I}_\square| \le (\varepsilon^{-1} \log\log n)^d = O((h/\varepsilon)^d)$ since $h = O(\log\log n)$, and each non-leaf cell $\square$ satisfies $|\mathcal{I}_\square| = |\mathsf{C}[\square]| + |\mathsf{S}[\square]| + 1 = O\left(n_\square^{1/3} + (h/\varepsilon)^d\right)$. The total number of points inside the cells of level i is

n; i.e, $\sum_{\square\in\mathcal{L}[i]} n_\square = n$. Furthermore, the total number of points of $A\cup B$ in leaf cells and non-empty subcells of the cells at level i is at most n. Therefore,

$$\sum_{\square\in\mathcal{L}[i]} |\mathcal{I}_\square|^3 = \sum_{\square\in\mathcal{L}[i]} O\left(n_\square + \left(h\varepsilon^{-1}\right)^{3d}\right) = O\left(n\left(h\varepsilon^{-1}\right)^{2d}\right).$$

Summing over all levels of $\mathcal{T}$, the total running time of the greedy algorithm is $O\left(n\left(h/\varepsilon\right)^{2d+1}\right)$. Putting everything together, the overall running time of the algorithm is $O(n\varepsilon^{2d+5}\log(n)\log^{2d+5}(\log n))$.

5 Conclusion

In this paper, we presented algorithms for the discrete and semi-discrete OT problems. Our techniques exploit the geometric structure of OT plans that led to simple and efficient algorithms for constructing near-optimal transport plans in low dimensions. There are a few natural open questions: First, it is unknown whether there exists a strongly polynomial time algorithm for semi-discrete OT problem when the continuous distribution also has nice geometric structure, such as a collection of geometric objects. Second, our algorithms are near-optimal in n, but incur exponential dependence in d. This raises the question whether there exists an algorithm for constructing an ε-OT plan whose running time is near-linear in n and polynomial in ε and d. Finally, is there an efficient algorithm for computing an ε-OT plan between two continuous distributions?

Acknowledgement

Work by P.A. and K.Y. has been partially supported by NSF grants IIS-18-14493, CCF-20-07556, and CCF-22-23870. Work by S.R. and P.S. has been partially supported by NSF CCF-1909171 and NSF CCF-2223871. We thank the anonymous reviewers for their useful comments.

References

[1] P. K. Agarwal, H.-C. Chang, S. Raghvendra, and A. Xiao. Deterministic, near-linear ε-approximation algorithm for geometric bipartite matching. In *Proc. 54th Annual ACM Sympos. Theory of Comput.*, pages 1052–1065, 2022.

[2] P. K. Agarwal, S. Raghvendra, P. Shirzadian, and R. Sowle. An improved ε-approximation algorithm for geometric bipartite matching. In *Proc. 18th Scandinavian Sympos. and Workshops Algorithm Theory*, pages 6:1–6:20, 2022.

[3] P. K. Agarwal, S. Raghvendra, P. Shirzadian, and K. Yao. Fast and accurate approximations of the optimal transport in semi-discrete and discrete settings. *arXiv preprint arXiv:2311.02172*, 2023.

[4] P. K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. *In Proc. 46th Annual ACM Sympos. Theory of Comput.*, page 555–564, 2014.

[5] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surveys*, 30(4):412–458, 1998.

[6] P. K. Agarwal and K. R. Varadarajan. A near-linear constant-factor approximation for Euclidean bipartite matching? In *20th Annual Sympos. Comput. Geom.*, pages 247–252, 2004.

[7] L. Ambrogioni, U. Guclu, and M. van Gerven. Wasserstein variational gradient descent: From semi-discrete optimal transport to ensemble variational inference. *arXiv preprint arXiv:1811.02827*, 2018.

[8] A. Andoni, P. Indyk, and R. Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proc. 19th Annual ACM-SIAM Sympos. Discrete Algorithms*, pages 343–352, 2008.

[9] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Comput.*, 8(1):121–164, 2012.

[10] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th Annual ACM-SIAM Sympos. Discrete Algorithms*, pages 147–155, 2002.

[11] F. Aurenhammer, F. Hoffmann, and B. Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1):61–76, 1998.

[12] A. Backurs, Y. Dong, P. Indyk, I. Razenshteyn, and T. Wagner. Scalable nearest neighbor search for optimal transport. *In Internat. Conf. on Machine Learning*, pages 497–506, 2020.

[13] J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[14] E. Bernton, P. E. Jacob, M. Gerber, and C. P. Robert. On parameter estimation with the Wasserstein distance. *Information and Inference: A J. of IMA*, 8(4):657–676, 2019.

[15] J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry*, volume 36. Springer, 2013.
[16] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. of ACM*, 42(1):67–90, 1995.
[17] R. Chartrand, B. Wohlberg, K. Vixie, and E. Bollt. A gradient descent solution to the Monge-Kantorovich problem. *Applied Math. Sci.*, 3(22):1071–1080, 2009.
[18] M. J. Cullen and R. J. Purser. An extended lagrangian theory of semi-geostrophic frontogenesis. *J. of Atmospheric Sci.*, 41(9):1477–1497, 1984.
[19] F. De Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun. Blue noise through optimal transport. *ACM Trans. on Graphics*, 31(6):1–11, 2012.
[20] I. Deshpande, Z. Zhang, and A. G. Schwing. Generative modeling using the sliced Wasserstein distance. In *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, pages 3483–3491, 2018.
[21] P. M. Esfahani and D. Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Math. Prog.*, 171(1):115–166, 2018.
[22] S. Fortune. Voronoi diagrams and delaunay triangulations. *Comput. in Euclidean Geom.*, pages 225–265, 1995.
[23] K. Fox and J. Lu. A near-linear time approximation scheme for geometric transportation with arbitrary supplies and spread. In *Proc. 36th Annual Sympos. Comput. Geom.*, pages 45:1–45:18, 2020.
[24] K. Fox and J. Lu. A deterministic near-linear time approximation scheme for geometric transportation. *arXiv preprint arXiv:2211.03891*, 2022.
[25] A. Genevay, M. Cuturi, G. Peyré, and F. Bach. Stochastic optimization for large-scale optimal transport. *Advances in Neural Information Processing Systems*, 29, 2016.
[26] A. Genevay, G. Peyre, and M. Cuturi. Learning generative models with Sinkhorn divergences. *In Internat. Conf. on Artificial Intelligence and Stat.*, page 1608–1617, 2018.
[27] R. Gupta, P. Indyk, and E. Price. Sparse recovery for earth mover distance. In *Proc. 48th Annual Allerton Conf. on Communication, Control, and Comput.*, pages 1742–1744. IEEE, 2010.
[28] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *3rd Internat. Workshop on Statistical and Comput. Theories of Vision*, volume 2, page 5, 2003.
[29] H. Janati, M. Cuturi, and A. Gramfort. Wasserstein regularization for sparse multi-task regression. In *The 22nd Internat. Conf. on Artificial Intelligence and Stat.*, pages 1407–1416. PMLR, 2019.
[30] A. B. Khesin, A. Nikolov, and D. Paramonov. Preconditioning for the geometric transportation problem. *arXiv preprint arXiv:1902.08384*, 2019.
[31] J. Kitagawa. An iterative scheme for solving the optimal transportation problem. *Calculus of Variations and Partial Differential Equations*, 51(1):243–263, 2014.
[32] J. Kitagawa, Q. Mérigot, and B. Thibert. Convergence of a Newton algorithm for semi-discrete optimal transport. *J. of European Math. Society*, 21(9):2603–2651, 2019.
[33] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quart.*, 2(1-2):83–97, 1955.
[34] B. Lévy and E. L. Schwindt. Notions of optimal transport theory and how to implement them on a computer. *Comput. & Graphics*, 72:135–148, 2018.
[35] H. Liu, G. U. Xianfeng, and D. Samaras. A two-step computation of the exact GAN Wasserstein distance. *In Internat. Conf. on Machine Learning*, pages 3159–3168, 2018.
[36] G. Luise, A. Rudi, M. Pontil, and C. Ciliberto. Differential properties of Sinkhorn approximation for learning with Wasserstein distance. *Advances in Neural Information Processing Systems*, 31, 2018.
[37] Q. Merigot and B. Thibert. Optimal transport: discretization and algorithms. In *Handbook of Numerical Analysis*, volume 22, pages 133–212. Elsevier, 2021.
[38] J.-M. Mirebeau. Discretization of the 3d monge- ampere operator, between wide stencils and power diagrams. *ESAIM: Math. Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 49(5):1511–1523, 2015.
[39] V. I. Oliker and L. D. Prussner. On the numerical solution of the equation $\frac{\partial^2 z}{\partial x^2}\frac{\partial^2 z}{\partial y^2} - \left(\frac{\partial^2 z}{\partial x \partial y}\right) = f$ and its discretizations, i. *Numerische Mathematik*, 54(3):271–293, 1989.
[40] J. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proc. Twentieth Annual ACM Sympos. Theory of Comput.*, pages 377–387, 1988.
[41] G. Peyré, M. Cuturi, et al. Computational optimal transport: With applications to data science. *Found. and Trends® in Machine Learning*, 11(5-6):355–607, 2019.
[42] H. Qin, Y. Chen, J. He, and B. Chen. Wasserstein blue noise sampling. *ACM Trans. on Graphics (TOG)*, 36(5):1–13, 2017.
[43] S. Raghvendra and P. K. Agarwal. A near-linear time ε-approximation algorithm for geometric bipartite matching. *J. of ACM*, 67(3):1–19, 2020.
[44] T. Salimans, H. Zhang, A. Radford, and D. Metaxas. Improving GANs using optimal transport. *In Internat. Conf.*

on Learning Representations, 2018.
[45] R. Seshadri and K. K. Srinivasan. Algorithm for determining path of maximum reliability on a network subject to random arc connectivity failures. *Transport. Research Rec.*, 2467(1):80–90, 2014.
[46] R. Sharathkumar and P. K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proc. 23rd Annual ACM-SIAM Sympos. Discrete Algo.*, pages 306–317. SIAM, 2012.
[47] J. Sherman. Generalized preconditioning and undirected minimum-cost flow. In *Proc. Twenty-Eighth Annual ACM-SIAM Sympos. Discrete Algo.*, pages 772–780, 2017.
[48] M. van Kreveld, F. Staals, A. Vaxman, and J. Vermeulen. Approximating the earth mover's distance between sets of geometric objects. *arXiv preprint arXiv:2104.08136*, 2021.
[49] C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.
[50] G. Zuzic. A simple boosting framework for transshipment. *arXiv preprint arXiv:2110.11723*, 2021.