Faster Algorithms for the Geometric Transportation Problem*

Pankaj K. Agarwal[1], Kyle Fox[1], Debmalya Panigrahi[1], Kasturi R. Varadarajan[2], and Allen Xiao[1]

1 **Duke University, Durham, NC**
2 **University of Iowa, Iowa City, IA**

Abstract

Let $R, B \subset \mathbb{R}^d$, for constant d, be two point sets with $|R| + |B| = n$, and let $\lambda : R \cup B \to \mathbb{N}$ such that $\sum_{r\in R} \lambda(r) = \sum_{b\in B} \lambda(b)$ be demand functions over R and B. Let $\mathrm{d}(\cdot,\cdot)$ be a suitable distance function such as the L_p distance. The transportation problem asks to find a map $\tau : R \times B \to \mathbb{N}$ such that $\sum_{b\in B} \tau(r,b) = \lambda(r)$, $\sum_{r\in R} \tau(r,b) = \lambda(b)$, and $\sum_{r\in R, b\in B} \tau(r,b)\mathrm{d}(r,b)$ is minimized. We present three new results for the transportation problem when $\mathrm{d}(\cdot,\cdot)$ is any L_p metric:

- For any constant $\epsilon > 0$, an $O(n^{1+\epsilon})$ expected time randomized algorithm that returns a transportation map with expected cost $O(\log^2(1/\epsilon))$ times the optimal cost.
- For any $\epsilon > 0$, a $(1+\epsilon)$-approximation in $O(n^{3/2}\epsilon^{-d}\,\mathrm{polylog}(U)\,\mathrm{polylog}(n))$ time, where $U = \max_{p\in R\cup B} \lambda(p)$.
- An exact strongly polynomial $O(n^2 \,\mathrm{polylog}\, n)$ time algorithm, for $d = 2$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Transportation map, earth mover's distance, shape matching, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.7

1 Introduction

Let R and B be two point sets in $\mathbb{R}^d$ with $|R| + |B| = n$, where d is a constant, and let $\lambda : R \cup B \to \mathbb{N}$ be a function satisfying $\sum_{r\in R} \lambda(r) = \sum_{b\in B} \lambda(b)$. We denote $U := \max_{p\in R\cup B} \lambda(p)$. We call a function $\tau : R \times B \to \mathbb{N}$, a *transportation map* between R and B if $\sum_{b\in B} \tau(r,b) = \lambda(r)$ for all $r \in R$ and $\sum_{r\in R} \tau(r,b) = \lambda(b)$ for all $b \in B$. Informally, for a point $r \in R$, the value of $\lambda(r)$ represents the *supply* at r, while for a point $b \in B$, the value of $\lambda(b)$ represents the *demand* at b. A transportation map represents a plan for moving the supplies at points in R to meet the demands at points in B.

The cost of a transportation τ is defined as $\mu(\tau) = \sum_{(r,b)\in R\times B} \tau(r,b)\mathrm{d}(r,b)$, where $\mathrm{d}(\cdot,\cdot)$ is a suitable distance function such as the L_p distance. The *Hitchcock-Koopmans transportation problem* (or simply *transportation problem*) on $\Sigma = (R, B, \lambda)$ is to find the minimum-cost transportation map for Σ, denoted $\tau^* := \tau^*(\Sigma)$. The cost $\mu(\tau^*)$ is often referred to as the *transportation distance* or *earth mover's distance*.

* Work by Agarwal, Fox, and Xiao is supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by grant 2012/229 from the U.S.-Israel Binational Science Foundation. Work by Fox and Panigrahi is supported in part by NSF grants CCF-1527084 and CCF-1535972. Work by Varadarajan is supported by NSF awards CCF-1318996 and CCF-1615845

The transportation problem is a discrete version of the so-called *optimal transport*, or *Monge-Kantarovich*, problem, originally proposed by the French mathematician Gaspard Monge in 1781. This latter problem has been extensively studied in mathematics since the early 20th century. See the book by Villani [27]. In addition to this connection, the (discrete) transportation problem has a wide range of applications, including similarity computation between a pair of images, shapes, and distributions, computing the barycenter of a family of distributions, finding common structures in a set of shapes, fluid mechanics, and partial differential equations. Motivated by these applications, this problem has been studied extensively in many fields including computer vision, computer graphics, machine learning, optimization, and mathematics. See e.g. [12–14, 20, 23] and references therein for a few examples.

The transportation problem can be formulated as an instance of the uncapacitated min-cost flow problem in a complete bipartite graph, in which edges have no capacity constraints. The min-cost flow problem has been widely studied; see [18] for a detailed review of known results. The uncapacitated min-cost flow problem in a graph with n vertices and m edges can be solved in $O((m + n \log n)n \log n)$ time using Orlin's algorithm [19] or $\tilde{O}(m\sqrt{n} \operatorname{polylog} U)$ time[1] using the algorithm by Lee and Sidford [18].

For transportation in geometric settings, Atkinson and Vaidya [8] adapted the Edmonds-Karp algorithm to exploit geometric properties, and obtained an $\tilde{O}(n^{2.5} \log U)$ time algorithm for any L_p-metric, and $\tilde{O}(n^2)$ for L_1, L_∞-metrics. The Atkinson-Vaidya algorithm was improved using faster data structures for dynamic nearest-neighbor searching, first by [1] and most recently by [17], for a running time of $\tilde{O}(n^2 \log U)$. Sharathkumar and Agarwal [21] designed a $(1 + \epsilon)$-approximation algorithm with a $\tilde{O}((n\sqrt{nU} + U \log U) \log(n/\epsilon))$ running time.

More efficient algorithms are known for estimating the the optimal cost (earth mover's distance) without actually computing the transportation, provided that $U = n^{O(1)}$. Indyk [16] gave an algorithm to find an $O(1)$-approximate estimate in $\tilde{O}(n)$ time with probability at least $1/2$. Cabello *et al.* [9] reduced the problem to min-cost flow using a geometric spanner, obtaining an $(1 + \epsilon)$-approximate estimate in $\tilde{O}(n^2)$ time. Andoni *et al.* [4] give a streaming algorithm that finds a $(1 + \epsilon)$-approximate estimate in $O(n^{1+o_\epsilon(1)})$ time. However, in many applications, one is interested in computing the map itself and not just the transportation distance [12, 13]. This is the problem that we address in this paper.

The special case of the transportation problem where every point has unit demand/supply is called the *geometric bipartite matching* problem. After a sequence of papers [3, 21, 25, 26], a near-linear $\tilde{O}(n)$ time $(1 + \epsilon)$-approximation was found by Agarwal and Sharathkumar [22] for this problem. On the other hand, before our work, no constant-factor approximation in subquadratic time was known for the transportation problem with arbitrary demands and supplies, even for the special case of $U = O(n^2)$.

Our results. We present three new results for the geometric transportation problem, for any L_p-metric.

Our first result (Section 2) is a randomized algorithm that for any $\epsilon > 0$, computes in $O(n^{1+\epsilon})$ expected time a transportation map whose expected cost is $O(\log^2(1/\epsilon))\mu(\tau^*)$. The expected cost improves to $O(\log(1/\epsilon))\mu(\tau^*)$ if the spread[2] of $R \cup B$ is $n^{O(1)}$. The overall structure of our algorithm is a simpler version of the matching algorithm by Agarwal and

[1] We use $\tilde{O}(f(n))$ to denote $O(f(n) \operatorname{polylog}(n))$.

[2] The *spread* of a point set S is the ratio of the maximum and the minimum distance between a pair of points in S.

Varadarajan [3], but several new ideas are needed to handle arbitrary demands and supplies.

We note that our algorithm can be extended to spaces with bounded doubling dimension. For example, suppose R, B lie in a subspace of $\mathbb{R}^d$ such that the doubling dimension with respect to $\mathrm{d}(\cdot,\cdot)$ is a constant, $\mathrm{d}(\cdot,\cdot)$ is computable in $O(1)$ time, and the spread of $R \cup B$ is $n^{O(1)}$. Then our algorithm can be adapted so that it computes in $O(n^{1+\epsilon})$ expected time a transportation map whose expected cost is $O(\log(1/\epsilon))\mu(\tau^*)$. Recall that Indyk's algorithm [16] estimates the cost of τ^* within an $O(1)$ factor assuming that $U = n^{O(1)}$. Using our ideas, his algorithm can be extended to arbitrary values of U. In particular, $\mu(\tau^*)$ can be estimated within an $O(1)$ factor in $\tilde{O}(n)$ time.

Our second result (Section 3) is a $(1+\epsilon)$-approximation algorithm to the transportation problem that runs in $\tilde{O}(n^{3/2}\epsilon^{-d}\operatorname{polylog}(U))$ time. Using a quad-tree based well-separated pair decomposition (WSPD) [11] of a point set, we construct a graph G with $O(n)$ vertices and $O(n/\epsilon^d)$ edges, and reduce the problem of computing a $(1+\epsilon)$-approximate transportation map to computing the min-cost flow in G. Next, we compute a min-cost flow f^* in G using the Lee-Sidford [18] algorithm. Finally, we recover in $O(n/\epsilon^d)$ time a transportation map $\tau : R \times B \to \mathbb{N}$ from f^* such that $\mu(\tau) \leq (1+\epsilon)\mu(\tau^*)$. Our algorithm can be extended to spaces with bounded doubling dimension by using the appropriate WSPD construction [24] for such spaces. In particular, if the doubling dimension is D and the spread of $R \cup B$ is $n^{O(1)}$, then a $(1+\epsilon)$-approximate transportation map can be computed in time $\tilde{O}(n^{3/2}\epsilon^{-O(D)}\operatorname{polylog}(U))$.

Our third result is an exact, strongly polynomial $\tilde{O}(n^2)$ time algorithm for $d = 2$, thereby matching (up to poly-logarithmic factors) the best exact algorithm for geometric matching [17]. Our algorithm is an implementation of Orlin's strongly polynomial min-cost flow algorithm [19], an augmenting-paths algorithm with edge contractions. A naive application of Orlin's algorithm has a running time of $\tilde{O}(n^3)$. By exploiting the geometry of the underlying graph, we improve this running time to $\tilde{O}(n^2)$ in the plane.

Proofs of many lemmas, extensions of our approximation algorithms, and the details of our exact algorithm are omitted from this extended abstract due to space constraints. They will appear in the full version of the paper.

2 A Near-Linear Approximation

Let $\Sigma = (R, B, \lambda)$ be an instance of the transportation problem in $\mathbb{R}^d$. We say that Σ has *bounded spread* if the spread of $R \cup B$ is bounded by n^a for some constant $a > 0$. We present a randomized recursive algorithm that, given Σ and a parameter $\epsilon > 0$, returns a transportation map in $O(n^{1+\epsilon})$ expected time whose expected cost is $O(\log(1/\epsilon))\mu(\tau^*)$ if Σ has bounded spread, and $O(\log^2(1/\epsilon))\mu(\tau^*)$ otherwise (recall that τ^* is the optimal map). We assume that n is sufficiently large so that n^ϵ is at least a suitably large constant.

We first give a high-level description of the algorithm without describing how each step is implemented efficiently. Next, we analyze the cost of the transportation map computed by the algorithm. We then discuss an efficient implementation of the algorithm. For simplicity, we describe the algorithm and its analysis for dimension $d = 2$; the algorithm extends to $d > 2$ in a straightforward manner.

We need the notion of *randomly shifted grids*, as in [3, 5]. Formally, let $\square = [a-\ell, a] \times [b-\ell, b]$ be a square of side length ℓ with (a, b) as its top right corner. For a parameter $\Delta > 0$ (grid cell length), set $l = \lceil \log_2 \left(1 + \frac{\ell}{\Delta}\right) \rceil$, and $L = 2^{l+1}\Delta$. Let $\square_L = [a-L, a] \times [b-L, b]$ be the square of side length L with (a, b) as its top-right corner. We choose uniformly at random a point $\xi \in [0, \Delta)^2$ and set $\square_{shifted} := \square_L + \xi$. Note that $\square \subseteq \square_{shifted}$. Let $\mathbb{G}(\square, \Delta)$ be the partition of $\square_{shifted}$ into the uniform grid of side length Δ; $\mathbb{G}(\square, \Delta)$ has $2^{l+1} \times 2^{l+1}$

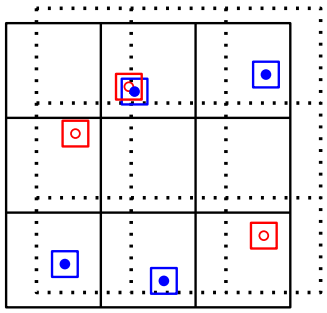


Figure 1. Moats, a safe grid (solid), an unsafe grid (dotted).

grid cells. $\mathbb{G}(\square, \Delta)$ is called the randomly shifted grid on $\square$.

2.1 A high-level description

A recursive subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ consists of point sets R and B, and a demand function $\lambda : \mathsf{R} \cup \mathsf{B} \to \mathbb{N}$ such that $\lambda(\mathsf{R}) = \lambda(\mathsf{B})$. We denote $|\mathsf{R} \cup \mathsf{B}| = m$. If $m \leq n^{\epsilon/4}$, we call Σ a *base* subproblem and compute an optimal transportation using Orlin's algorithm. Thus, assume that $m > n^{\epsilon/4}$.

Let $\delta = 1/6$. Also, let $\square$ be the smallest axis-aligned square containing $\mathsf{R} \cup \mathsf{B}$, with its sidelength denoted ℓ. Set $\Delta = \ell/m^{\delta}$. The first step of the algorithm is to choose a randomly shifted grid $\mathbb{G} = \mathbb{G}(\square, \Delta)$ that has the following additional property: any two points in $\mathsf{R} \cup \mathsf{B}$ that are within a distance of ℓ/m^3 lie in the same grid cell. We call a grid satisfying this property *safe*. Algorithmically, we place an axis-parallel square of side length $2\ell/m^3$ around every $p \in \mathsf{R} \cup \mathsf{B}$, called the *moat* of p; $\mathbb{G}$ is safe if none of its grid lines cross any moat (see Figure 1). If $\mathbb{G}$ is not safe, we sample a new random shift. It can be verified that $\mathbb{G}$ is safe with probability at least $1 - 1/m$.

Let $\Pi \subseteq \mathbb{G}$ be the set of nonempty grid cells, i.e., ones that contain at least one point of $\mathsf{R} \cup \mathsf{B}$. For each cell $\pi \in \Pi$, we create a recursive instance Σ_π, which we refer to as an *internal subproblem*. Each Σ_π aims to transport as much as possible within π. Whatever we are unable to transport locally within cells of Π, we transport globally with a single *external subproblem* $\Sigma_\square$. We now describe these subproblems in more detail.

For each cell $\pi \in \Pi$, we define the *excess* χ_π of π to be the absolute difference between the red and blue demand in π. Without loss of generality, assume $\lambda(\mathsf{R} \cap \pi) \geq \lambda(\mathsf{B} \cap \pi)$, i.e. that the excess of π is red. Roughly speaking, the entirety of $\mathsf{B} \cap \pi$ is used for the internal subproblem, while $\mathsf{R} \cap \pi$ is arbitrarily partitioned such that $\lambda(\mathsf{B} \cap \pi)$ red demand is used for the internal subproblem, and the remainder (of total demand $= \chi_\pi$) is used for the external subproblem. We pick an arbitrary maximal subset of points $(\mathsf{R}_{ex})_\pi \subseteq \mathsf{R} \cap \pi$ such that $\lambda((\mathsf{R}_{ex})_\pi) \leq \chi_\pi$. Let $\mathsf{R}_\pi = (\mathsf{R} \cap \pi) \setminus (\mathsf{R}_{ex})_\pi$, $\mathsf{B}_\pi = \mathsf{B} \cap \pi$, and $(\mathsf{B}_{ex})_\pi = \emptyset$. If $\lambda((\mathsf{R}_{ex})_\pi) < \chi_\pi$, we arbitrarily pick a point p in R_π, and split p into two copies, say p' and p'', with $\lambda(p') = \chi_\pi - \lambda((\mathsf{R}_{ex})_\pi)$ and $\lambda(p'') = \lambda(p) - \lambda(p')$. We then add p' to $(\mathsf{R}_{ex})_\pi$ and replace p with p'' in R_π. This step ensures that $\lambda((\mathsf{R}_{ex})_\pi) = \chi_\pi$. Let λ_π be the restriction of λ to $\mathsf{R}_\pi \cup \mathsf{B}_\pi$; by construction, $\lambda_\pi(\mathsf{R}_\pi) = \lambda_\pi(\mathsf{B}_\pi)$. The internal subproblem for π is $\Sigma_\pi = (\mathsf{R}_\pi, \mathsf{B}_\pi, \lambda_\pi)$.

We now describe the external subproblem. Let $\mathsf{R}_{ex} = \bigcup_{\pi \in \Pi} (\mathsf{R}_{ex})_\pi$, $\mathsf{B}_{ex} = \bigcup_{\pi \in \Pi} (\mathsf{B}_{ex})_\pi$, and set λ_{ex} as the restriction of λ to $\mathsf{R}_{ex} \cup \mathsf{B}_{ex}$. To solve the excess demand instance $\Sigma_{ex} = (\mathsf{R}_{ex}, \mathsf{B}_{ex}, \lambda_{ex})$, we merge the excess in each cell into a single artificial point at the center of the cell. The resulting transportation instance has relatively few points ($O(m^{2\delta})$)

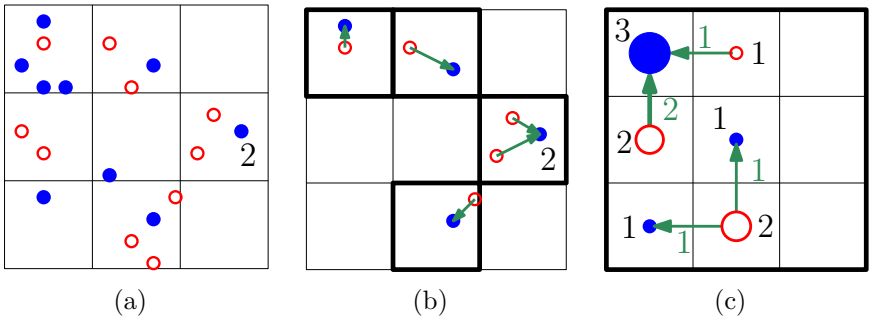


Figure 2. A subproblem (a) with its internal subproblems (b) and external subproblem (c).

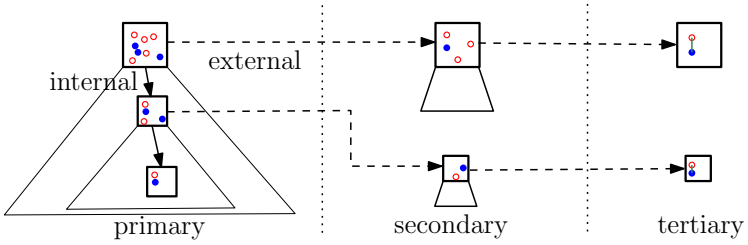


Figure 3. Primary-secondary classification of recursive problems.

and distorts the "real" distances by an amount proportional to the side length of the cell. If $\lambda(\mathsf{R} \cap \pi) > \lambda(\mathsf{B} \cap \pi)$, we create a red point r_π at the center of π and define the demand of r_π, denoted $\lambda_\square(r_\pi)$, to be χ_π. Similarly, if $\lambda(\mathsf{B} \cap \pi) > \lambda(\mathsf{R} \cap \pi)$, we create a blue point b_π at the center of π with $\lambda_\square(b_\pi) = \chi_\pi$. Let $\mathsf{R}_\square$ (resp., $\mathsf{B}_\square$) be the set of red (resp., blue) points that were created at the centers of cells in Π. We create the external subproblem $\Sigma_\square = (\mathsf{R}_\square, \mathsf{B}_\square, \lambda_\square)$; $\Sigma_\square$ acts as an approximate view of the actual excess instance Σ_{ex}. See Figure 2.

For each cell $\pi \in \Pi$, we recursively compute a transportation map τ_π on the internal subproblem Σ_π. If the root instance – the original input to our transportation problem – has bounded spread, we compute an optimal solution $\tau_\square$ for the external subproblem $\Sigma_\square$ using Orlin's algorithm. If the root instance does not have bounded spread, then we recursively compute an approximately optimal solution $\tau_\square$ for the external subproblem $\Sigma_\square$. Note that irrespective of the spread of the original instance, every external subproblem $\Sigma_\square$ has spread bounded by $O(n^\delta)$, i.e., has bounded spread. We categorize subproblems by the number of external subproblems in the recursive chain leading to them: Σ is *primary* if there are none; *secondary* if there is exactly one; and *tertiary* if there are two. All tertiary problems are solved exactly using Orlin's algorithm, as are base subproblems in the primary and secondary recursion. See Figure 3 for a visualization of the recursion tree of the algorithm.

Finally, we construct a transportation map τ for Σ by combining the solutions to the internal and external subproblems. For a pair $(r, b) \in \mathsf{R}_\pi \times \mathsf{B}_\pi$, we simply set $\tau(r, b) = \tau_\pi(r, b)$. For the external subproblem, we first convert the transportation map $\tau_\square$ on $\Sigma_\square$ into a map for Σ_{ex}, as follows: For each red point $r_\pi \in \mathsf{R}_\square$ (resp., blue point in $\mathsf{B}_\square$), at the center of a cell $\pi \in \Pi$, we "redistribute" the transport from r_π (resp., b_π) to the points of $(\mathsf{R}_{ex})_\pi$ (resp., $(\mathsf{B}_{ex})_\pi$) to compute a transportation map τ_{ex} of Σ_{ex}. That is, for any $r_\pi, b_\pi \in \mathsf{R}_\square \times \mathsf{B}_\square$, we assign the units of $\tau_\square(r_\pi, b_\pi)$ among the pairs in $(\mathsf{R}_{ex})_\pi \times (\mathsf{B}_{ex})_\pi$ in an arbitrary manner, while respecting the demands. We then set $\tau(r, b) = \tau_{ex}(r, b)$ for $(r, b) \in \mathsf{R}_{ex} \times \mathsf{B}_{ex}$. This

completes the description of the algorithm.

2.2 Cost analysis

There are two sources of error in our algorithm: the distortion between $\Sigma_\square$ and Σ_{ex}, and the error from restricting the solution to the internal/external partitioning of demand.

δ-closeness. We first formalize the way that $\Sigma_\square$ approximates Σ_{ex} when it shifts demand to cell centers. We introduce a notion called δ-closeness between transportation instances: informally, two instances are δ-close if we can shift the demands of one to form the other, without moving any demand more than δ. We give a formal definition next.

Let $\Sigma = (R, B, \lambda)$ be an instance of the transportation problem and τ a transportation map for Σ. We define $\mu_\infty(\tau) = \max_{(r,b):\tau(r,b)>0} \mathrm{d}(r, b)$ as the maximum distance used in τ. Let $\Sigma' = (R', B', \lambda')$ be another instance of the transportation problem with $\lambda(R) = \lambda'(R')$. Consider the transportation instances $\Sigma_R = (R, R', \lambda_R)$ and $\Sigma_B = (B, B', \lambda_B)$ where λ_R (resp., λ_B) is the demand of points in R and R' (resp., B and B') in Σ and Σ' respectively. We call Σ and Σ' *δ-close* if there exist transportation maps τ_R and τ_B of Σ_R and Σ_B such that $\mu_\infty(\tau_R), \mu_\infty(\tau_B) \le \delta$, i.e., demands of R (resp., B) (and therefore the units of τ) points of R' (resp., B') within distance δ. We then say that the resulting transportation τ' in Σ' is a map *derived* from τ in Σ. The next lemma follows immediately from definition of δ-closeness.

▶ **Lemma 1.** *Let Σ and Σ' be two δ-close instances of the transportation problem with U being the total demand of each. Let τ be a transportation map of Σ and let τ' be a transportation map of Σ' derived from τ. Then, $|\mu(\tau') - \mu(\tau)| \le 2\delta U$.*

▶ **Observation 1.** *Any point in a grid cell of side length Δ is within $(\Delta/\sqrt{2})$ of the center; so $\Sigma_\square$ and Σ_{ex} are $(\Delta/\sqrt{2})$-close.*

Thus, the lemma relates the transportation map for Σ_{ex} to the solution for $\Sigma_\square$ produced by the external subproblem.

Partitioning of demand. Now that we have quantified the error between $\Sigma_\square$ and Σ_{ex}, we begin our analysis of the second source of error. First, we bound the error in a single subproblem (i.e. one subdividing grid), and then the error for a single pair $(r, b) \in R \times B$ across all subproblems. Eventually, we combine these two arguments to bound the expected error due to the partitioning across all subproblems and all pairs of points.

Fix a recursive problem $\Sigma = (R, B, \lambda)$, with cell side length Δ. Let χ_π, R_π, B_π, $(R_{ex})_\pi$, $(B_{ex})_\pi$ for $\pi \in \Pi$ and $R_\square$, $B_\square$, R_{ex}, B_{ex}, λ_{ex}, be as defined in Section 2.1 for the instance $\Sigma = (R, B, \lambda)$. Let $\mathcal{I} = \bigcup_{\pi\in\Pi} R_\pi \times B_\pi$ be the set of "local" point pairs, solved by the algorithm within internal subproblems. We refer to a pair $(r, b) \in (R \times B) \setminus \mathcal{I}$ as "non-local".

The next lemma outlines a method for deforming an arbitrary transportation map to one that respects the local/non-local partitioning used by the algorithm.

▶ **Lemma 2.** *Let $\Sigma = (R, B, \lambda)$ be a recursive subproblem with cell side length Δ, and let $\hat{\tau}$ be an arbitrary transportation map for Σ. Let $\hat{\mathsf{X}} = \sum_{(r,b)\notin\mathcal{I}} \hat{\tau}(r, b)$ be the total non-local transport in $\hat{\tau}$, and $\mathsf{X} = \sum_{\pi\in\Pi} \chi_\pi$ be the total excess of Σ. Then, there exists a transportation map $\tilde{\tau}$ comprising local solutions $\tilde{\tau}_\pi$ for each Σ_π and a non-local solution $\tilde{\tau}_{ex}$ for Σ_{ex}, such that the following properties hold:*

(A) The cost of the transportation map $\tilde{\tau}$ is given by

$$\mu(\tilde{\tau}) = \sum_{\pi\in\Pi} \mu(\tilde{\tau}_\pi) + \mu(\tilde{\tau}_{ex}) \le \mu(\hat{\tau}) + 8\sqrt{2}\Delta\hat{\mathsf{X}}.$$

(B) The local transport in $\tilde{\tau}$ satisfies $\tilde{\tau}(r,b) \geq \hat{\tau}(r,b)$ for all $(r,b) \in \mathcal{I}$, and the difference $\sum_{(r,b)\in\mathcal{I}} (\tilde{\tau}(r,b) - \hat{\tau}(r,b)) \leq 3\hat{\mathsf{X}}$.

(C) The non-local transport in $\tilde{\tau}$ satisfies $\hat{\mathsf{X}} \geq \mathsf{X} = \tilde{\mathsf{X}} := \sum_{(r,b)\notin\mathcal{I}} \tilde{\tau}(r,b)$.

Error parameter η. In the previous lemma, we bounded the error due to a single subproblem. We now bound the error due to a single pair of points $(r,b) \in R \times B$, using a random variable $\eta(r,b)$, defined as the cell side length of the first recursive grid to split (r,b) into different cells.

Formally, recall that a recursive subproblem may split a point $p \in R \cup B$ into two copies p' and p'' with $\lambda(p') + \lambda(p'') = \lambda(p)$; one of them passed to the external subproblem, and the other passed down to an internal subproblem. Abusing notation slightly, we use R and B to denote the multisets that contain all copies of points that are split along with the updated demands. Hence, for any base subproblem (R_i, B_i, λ_i) (i.e., the recursive base case) every point $p \in R_i \cup B_i$ can be identified with a point $p \in R \cup B$ such that $\lambda_i(p) = \lambda(p)$. With this interpretation, we define a function $\eta : R \times B \to \mathbb{R}_{\geq 0}$ as follows: If there is a base subproblem (R_i, B_i, λ_i) such that $(r,b) \in R_i \times B_i$, we set $\eta(r,b) = 0$. Otherwise, there is a recursive subproblem such that $(r,b) \in \mathsf{R} \times \mathsf{B}$, and r and b are split into different cells of the randomly shifted grid. In this case, $\eta(r,b)$ denotes the side length of the grid cells, i.e. $\eta(r,b) = \ell/m^\delta$ where ℓ is the length of the smallest square containing $\mathsf{R} \cup \mathsf{B}$, and $m = |\mathsf{R} \cup \mathsf{B}|$.

The next lemma bounds the expected value of the error parameter $\eta(r,b)$ in terms of the distance $\mathrm{d}(r,b)$ for any pair of points $(r,b) \in R \times B$. Its proof uses our choice of safe grids to argue that, though the recursion depth can be large, the number of recursive subproblems that can potentially split (r,b) is small.

▶ **Lemma 3.** *There exists a constant $c_1 > 0$ such that for any $(r,b) \in R \times B$, the expectation $\mathsf{E}[\eta(r,b)] \leq c_1 \log_2(1/\epsilon)\mathrm{d}(r,b)$.*

Proof. Let n be the number of points in the input instance, and m be the number of points in the subproblem which splits (r,b), and ℓ the side length of the smallest orthogonal bounding square of that subproblem (i.e. $\eta(r,b) = \ell/m^\delta$). We can assume that $m > n^{\epsilon/4}$, since otherwise the subproblem is a base case and $\eta(r,b) = 0$.

Define $\underline{\ell} := \mathrm{d}(r,b)/\sqrt{2}$; clearly, $\ell \geq \underline{\ell}$. We partition the interval $[n^{\epsilon/4}, n]$ into $u = \lceil \log_2(4/\epsilon) \rceil$ intervals of the form $[n_j, n_j^2]$, where $n_j = n^{2^{-j}}$, for $1 \leq j \leq u$. There exists an index j^* where $m \in [n_{j^*}, n_{j^*}^2]$, and $\ell \leq \overline{\ell} := (n_{j^*}^2)^3 \mathrm{d}(r,b)$ because the grid is safe. Thus, (r,b) must be split by a grid with ℓ in the interval $[\underline{\ell}, \overline{\ell}]$, where $\overline{\ell}/\underline{\ell} = \sqrt{2} n_{j^*}^6$.

The interval $[\underline{\ell}, \overline{\ell}]$ can be covered by $7/\delta > (1/\delta)(6 + \log_2(\sqrt{2})/\log_2(n_{j^*}))$ intervals of the form $J_i = [n_{j^*}^{i\delta}\underline{\ell}, n_{j^*}^{(i+1)\delta}\underline{\ell}]$ for $0 \leq i \leq 7/\delta = O(1)$. For each value of i, the algorithm produces at most one subproblem — whose bounding square contains both r, b — with side length in J_i. The total set of intervals, by enumeration, bounds the number of subproblems whose grids could possibly split (r,b) as $O(\log(1/\epsilon))$.

For our subproblem with m points and side length ℓ, the probability that a (safe) randomly shifted grid splits (r,b) is no more than $3\mathrm{d}(r,b)/(\ell/m^\delta)$. Recall that the value of $\eta(r,b)$, in this case, is ℓ/m^δ. Summing over the $O(\log(1/\epsilon))$ subproblems where (r,b) can be split, the expected value of $\eta(r,b)$ is $O(\log(1/\epsilon))\mathrm{d}(r,b)$. ◀

Expected cost of algorithm. We are now ready to analyze the expected cost of τ, the algorithm's transportation map. First, we analyze the cost if Σ has bounded spread. Recall that, in this case, our algorithm computes an optimal solution for each external subproblem (using Orlin's algorithm).

▶ **Lemma 4.** *If Σ is a transportation instance with bounded spread, then there exists a constant $c_2 > 0$ such that for any transportation map $\hat{\tau}$ of Σ,*

$$\mu(\tau) \leq \mu(\hat{\tau}) + c_2 \sum_{(r,b)\in R\times B} \hat{\tau}(r,b)\eta(r,b).$$

An immediate corollary of Lemmas 3 and 4 is:

▶ **Corollary 5.** *If Σ has bounded spread, then $\mathsf{E}[\mu(\tau)] = O(\log(1/\epsilon))\mu(\tau^*)$.*

Proof of Lemma 4. We prove the lemma by induction on the number of points in the subproblem. If Σ is a base problem, then τ is an optimal transport of Σ and the lemma holds. Otherwise $\square$, the smallest square containing $R \cup B$, is split into a set of grid cells. Following the notation in Section 2.1, let Π be the set of non-empty cells and Δ the side length of each grid cell.

Recall that τ is the combination of solutions τ_π for the internal subproblems $\Sigma_\pi = (R_\pi, B_\pi, \lambda_\pi)$ of $\pi \in \Pi$, and the map τ_{ex} for $\Sigma_{ex} = (R_{ex}, B_{ex}, \lambda_{ex})$ derived from the solution $\tau_\square$ to the external subproblem $\Sigma_\square = (R_\square, B_\square, \lambda_\square)$. From Observation 1, $\Sigma_\square$ and Σ_{ex} are $(\Delta/\sqrt{2})$-close; thus Lemma 1 implies $\mu(\tau_\square) \leq \mu(\tau_{ex}) + \sqrt{2}\Delta\mathsf{X}$. We have

$$\mu(\tau) = \mu(\tau_{ex}) + \sum_{\pi\in\Pi} \mu(\tau_\pi) \leq \mu(\tau_\square) + \sqrt{2}\Delta\mathsf{X} + \sum_{\pi\in\Pi} \mu(\tau_\pi). \tag{1}$$

Thus, using (1), we can bound τ by bounding the local (τ_π) and non-local ($\tau_\square$) solutions individually.

Let $\tilde{\tau}$ be the transportation map created by deforming $\hat{\tau}$ in Lemma 2, with $\tilde{\tau}_\pi$ and $\tilde{\tau}_{ex}$ its restrictions to local and non-local pairs of points respectively. To bound $\tau_\square$, notice that that $\tilde{\tau}_{ex}$ solves $\Sigma_{ex} = (R_{ex}, B_{ex}, \lambda_{ex})$, and Σ_{ex} is $(\Delta/\sqrt{2})$-close to $\Sigma_\square$. We apply Lemma 1 and optimality of $\tau_\square$ to conclude,

$$\mu(\tau_\square) \leq \mu(\tilde{\tau}_{ex}) + \sqrt{2}\Delta\mathsf{X}. \tag{2}$$

We now bound the local solutions. Since $\tilde{\tau}_\pi$ is a transportation map of the internal subproblem Σ_π, by the induction hypothesis,

$$\mu(\tau_\pi) \leq \mu(\tilde{\tau}_\pi) + c_2 \sum_{(r,b)\in R_\pi\times B_\pi} \tilde{\tau}_\pi(r,b)\eta(r,b). \tag{3}$$

We can now combine (2) and (3) to bound τ in (1).

$$\begin{aligned}
\mu(\tau) &\leq \mu(\tau_\square) + \sum_{\pi\in\Pi} \mu(\tau_\pi) + \sqrt{2}\Delta\mathsf{X} \\
&\leq \mu(\tilde{\tau}_{ex}) + 2\sqrt{2}\Delta\mathsf{X} + \sum_{\pi\in\Pi} \left[\mu(\tilde{\tau}_\pi) + c_2 \sum_{(r,b)\in R_\pi\times B_\pi} \tilde{\tau}_\pi(r,b)\eta(r,b)\right] && \text{(by (2), (3))} \\
&= \mu(\tilde{\tau}) + c_2 \sum_{(r,b)\in\mathcal{J}} \tilde{\tau}(r,b)\eta(r,b) + 2\sqrt{2}\Delta\mathsf{X} && \text{(Lem. 2(A))} \\
&\leq \mu(\hat{\tau}) + c_2 \sum_{(r,b)\in\mathcal{J}} \tilde{\tau}(r,b)\eta(r,b) + 10\sqrt{2}\Delta\hat{\mathsf{X}} && \text{(Lem. 2(A), 2(C))} \\
&= \mu(\hat{\tau}) + c_2 \sum_{(r,b)\in R\times B} \hat{\tau}(r,b)\eta(r,b) + \Gamma,
\end{aligned}$$

$$\text{where } \Gamma = c_2 \sum_{(r,b)\in\mathcal{I}} \left(\tilde{\tau}(r,b) - \hat{\tau}(r,b)\right)\eta(r,b) + 10\sqrt{2}\Delta\hat{\mathsf{X}} - c_2 \sum_{(r,b)\notin\mathcal{I}} \hat{\tau}(r,b)\eta(r,b).$$

By definition, $\eta(r,b) = \Delta$ for $(r,b) \notin \mathcal{I}$, $\eta(r,b) \leq \Delta/n^{\epsilon/4} \leq \Delta/4$ for $(r,b) \in \mathcal{I}$, and $\sum_{(r,b)\notin\mathcal{I}} \hat{\tau}(r,b) = \hat{\mathsf{X}}$. Therefore, using Lemma 2(B),

$$\Gamma \leq c_2\frac{\Delta}{4}\cdot 3\hat{\mathsf{X}} + 10\sqrt{2}\Delta\hat{\mathsf{X}} - c_2\Delta\hat{\mathsf{X}} = \left(10\sqrt{2} - \frac{c_2}{4}\right)\Delta\hat{\mathsf{X}} \leq 0.$$

provided that $c_2 \geq 40\sqrt{2}$. Hence, $\mu(\tau) \leq \mu(\hat{\tau}) + c_2\sum_{(r,b)}\hat{\tau}(r,b)\eta(r,b)$. This completes the proof of the lemma. ◀

The general case. We now analyze the cost of the transportation map for the general case, when the spread of $R \cup B$ is arbitrary.

Recall Figure 3 and the categorization of recursive subproblems as primary, secondary, or tertiary based on the number of external subproblem invocations on its path in the recursion tree of the algorithm. We now introduce two functions $\eta_1, \eta_2 : R \times B \to \mathbb{R}_{\geq 0}$ corresponding to the errors introduced in the primary and secondary recursions. (Note that tertiary subproblems are solved exactly; hence, there is no error introduced in solving a tertiary subproblem.)

The function η_1 corresponds to the primary recursion and is the same as the η defined above, i.e., $\eta_1(r,b) = 0$ if (r,b) belongs to a primary base subproblem, otherwise it is the length of the grid cell at the subproblem which splits r and b. The function $\eta_2(r,b)$ corresponds to the secondary recursion for (r,b). If (r,b) belongs to a primary base problem (i.e. does not appear in any secondary recursion), then we set $\eta_2(r,b) = 0$. Otherwise, let $\bar{r} \in R_\square$ and $\bar{b} \in B_\square$ be the centers of the grid cells of the primary subproblem where r and b were split. Then, $\eta_2(r,b)$ is defined to be $\eta(\bar{r},\bar{b})$ for the secondary recursion on $(R_\square, B_\square)$.

We now state two lemmas that are counterparts of Lemmas 3 and 4 for the general case.

▶ **Lemma 6.** *For any pair $(r,b) \in R \times B$, $\mathsf{E}\left[\eta_2(r,b)\right] = O(\log^2(1/\epsilon))\mathrm{d}(r,b)$.*

▶ **Lemma 7.** *There exists a constant $c_3 > 0$ such that for any transportation map $\hat{\tau}$ of Σ,*

$$\mu(\tau) \leq \mu(\hat{\tau}) + c_3 \sum_{(r,b)\in R\times B} \hat{\tau}(r,b)\left[\eta_1(r,b) + \eta_2(r,b)\right].$$

The bound on the expected cost follows directly from the above two lemmas.

▶ **Corollary 8.** $\mathsf{E}[\mu(\tau)] = O(\log^2(1/\epsilon))\mu(\tau^*)$.

2.3 An efficient implementation

We now explain how the various steps of the algorithm are implemented to run in $O(n^{1+\epsilon})$ time. There are three main steps in the algorithm: (i) partitioning a recursive subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ into internal subproblems and an external subproblem; (ii) solving subproblems recursively; (iii) recovering the transportation map τ of Σ from the internal and external solutions τ_π ($\pi \in \Pi$) and $\tau_\square$. A recursive subproblem partitions its points into internal subproblems, but generates an additional set of points (at cell centers) for its external subproblem; let us call these *external points*. We bound the number of external points generated in the next lemma.

▶ **Lemma 9.** *The total number of external points over all recursive subproblems is $O(n)$.*

A base subproblem of size $n_i \leq n^{\epsilon/4}$ is solved in $O(n_i^3 \log n_i)$ time using Orlin's algorithm. We distribute this on the n_i points by charging $O(n_i^2 \log n_i) = O(n^{\epsilon/2} \log n)$ to each point. Note every point in $R \cup B$, as well as every external point, belongs to at most one base subproblem. Since, by Lemma 9, the number of external points is $O(n)$, it follows that the total time spent solving base subproblems is $O(n) \cdot O(n^{\epsilon/2} \log n) = O(n^{1+\epsilon})$.

The time spent in recovering the transportation map τ for Σ from its internal and external subproblems is proportional to the number of external points in Σ. Hence, by Lemma 9, step (iii) takes $O(n)$ time.

Finally, implementation of step (i) depends on whether the instance $\Sigma = (R, B, \lambda)$ has bounded spread.

The bounded spread case. In this case, step (i) is implemented naively. We choose a random shift, distribute the points of $\mathsf{R} \cup \mathsf{B}$ among the grid cells in $O(m \log m)$ time, where $|\mathsf{R} \cup \mathsf{B}| = m$, and check in additional $O(m)$ time whether the shift is safe. So step (i) can be implemented in $O(m \log m)$ expected time. We charge $O(\log m)$ time to each point of $\mathsf{R} \cup \mathsf{B}$. Since the spread is $n^{O(1)}$, the depth of recursion is $O(1/\epsilon)$. Therefore, each input point is charged $O(\frac{1}{\epsilon} \log n)$ units of time, implying that steps (i) over all subproblems take $O(n \log n)$ expected time (note that ϵ is a constant). As the size of the external subproblem at $\mathsf{R} \cup \mathsf{B}$ is $O(m^{2\delta})$, and $\delta = 1/6$, the time for solving it exactly is $O(m)$. Putting everything together and applying Corollary 5, we obtain the following.

▶ **Theorem 10.** *Let Σ be an instance of the transportation problem in $\mathbb{R}^d$, where d is a constant. Let Σ have size n and bounded spread, and let $\epsilon > 0$ be a constant. A transportation map of Σ can be computed in $O(n^{1+\epsilon})$ expected time whose expected cost is $O(\log(1/\epsilon))\mu(\tau^*)$, where τ^* is an optimal transport of Σ.*

The general case. Let $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ be a recursive subproblem, and $\square$ the smallest square containing $\mathsf{R} \cup \mathsf{B}$. As defined earlier, let $\Sigma_\square = (\mathsf{R}_\square, \mathsf{B}_\square, \lambda_\square)$ be the external subproblem generated by Σ using the points of $\mathsf{R}_{ex}, \mathsf{B}_{ex}$. Since $\Sigma_\square$ has bounded spread and is solved recursively, the running time to solve $\Sigma_\square$ is $O(|\mathsf{R}_\square \cup \mathsf{B}_\square|^{1+\epsilon}) = O(|\mathsf{R}_{ex} \cup \mathsf{B}_{ex}|^{1+\epsilon})$. Summing over all recursive problems, by Lemma 9, the total time spent in solving all external subproblems is $O(n^{1+\epsilon})$.

Step (i) is more challenging in this case because the depth of recursion can be as large as $\Omega(n)$. We can neither spend linear time at a recursive subproblem, nor can we afford to visit each cell of the randomly shifted grid $\mathbb{G}$ explicitly to compute the set Π of non-empty cells. To avoid checking all cells of $\mathbb{G}$ explicitly, we (implicitly) construct a quad tree $\mathbb{T}$ on $\mathbb{G}$ — i.e. leaves of $\mathbb{T}$ are cells of $\mathbb{G}$ and the root of $\mathbb{T}$ is $\square$. The depth of $\mathbb{T}$ is $O(\log m)$. The role of $\mathbb{T}$ will be to guide the search for non-empty cells of $\mathbb{G}$. Again, we will not construct $\mathbb{T}$ explicitly.

To avoid spending $\Omega(m)$ time in step (i), we do not maintain the set $\mathsf{R} \cup \mathsf{B}$ explicitly. We build a 2D dynamic orthogonal range searching data structure that maintains a set $X \subset \mathbb{R}^2$ of weighted points, and supports the following operations:

- $\textsc{Wt}(\rho)$: Given a rectangle ρ, return $w(X \cap \rho)$.
- $\textsc{Report}(\rho, \Delta)$: Report a maximal subset Y of $X \cap \rho$ such that $w(Y) \leq \Delta$.
- $\textsc{Empty}(\rho)$: Return Yes if $X \cap \rho = \emptyset$ and No otherwise.
- $\textsc{Delete}(p)$: Delete p from X.
- $\textsc{ReduceWt}(p, \Delta)$: Update $w(p) := w(p) - \Delta$, assuming $w(p) \geq \Delta$.

Using a range-tree based data structure, each operation except for Report can be performed in $O(\log^2 m)$ time [2]. Report requires $O(\log^2 n + k)$ time, where k is the number of reported points.

We maintain two copies $\mathbb{D}_R, \mathbb{D}_B$ of this data structure. The first one is initialized with R and the supplies, and the second with B and the demands. We use $\mathcal{R}$ and $\mathcal{B}$ to denote the current sets in these data structures.

With each recursive subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ we associate a bounding rectangle ρ that contains $\mathsf{R} \cup \mathsf{B}$. For the root problem, ρ is the smallest square containing $R \cup B$; for others it is defined recursively. We maintain the invariant that when the subproblem $\Sigma = (\mathsf{R}, \mathsf{B}, \lambda)$ is being processed,

(i) $\mathcal{R} \cap \rho = \mathsf{R}$ and for any $r \in \mathcal{R} \cap \rho, w(r) = \lambda(r)$,
(ii) $\mathcal{B} \cap \rho = \mathsf{B}$ and for any $b \in \mathcal{B} \cap \rho, w(b) = \lambda(b)$.

We first compute Π, the set of non-empty cells of $\mathbb{G}$, using $\mathbb{T}$ and the data structures $\mathbb{D}_R, \mathbb{D}_B$. We visit $\mathbb{T}$ in a top-down manner. Suppose we are at a node $v \in \mathbb{T}$, and let $\square_v$ be the square associated with v. We call $\textsc{Empty}(\rho \cap \square_v)$ on both $\mathbb{D}_R$ and $\mathbb{D}_B$ to check whether $(\mathsf{R} \cup \mathsf{B}) \cap \square_v = \emptyset$. If Yes, we ignore the subtree rooted at v. If No and v is a leaf of $\mathbb{T}$, i.e., $\square_v$ is a nonempty cell of $\mathbb{G}$, we add $\square_v$ to Π. If v is an internal node and $(\mathsf{R} \cup \mathsf{B}) \cap \square_v \neq \emptyset$, we recursively search the children of v in $\mathbb{T}$. Since the depth of $\mathbb{T}$ is $O(\log n)$, the above procedure visits $O(|\Pi| \log n)$ nodes of $\mathbb{T}$. The total time spent in computing Π is thus $O(|\Pi| \log^3 n)$.

For each cell $\pi \in \Pi$, we can compute the total demands of $\lambda(\mathsf{R} \cap \pi)$ and $\lambda(\mathsf{B} \cap \pi)$ — and thus χ_π — using the $\textsc{Wt}(\rho \cap \pi)$ operations on $\mathbb{D}_R, \mathbb{D}_B$. Without loss of generality, assume $\lambda(\mathsf{R} \cap \pi) > \lambda(\mathsf{B} \cap \pi)$. Using $\textsc{Report}(\rho \cap \pi, \chi)$, we report a maximal subset of points of $\mathsf{R} \cap \pi$ whose total weight is at most χ. We then delete each of these points (by Delete) and reduce the weight (by Reduce) of one additional point in $\mathsf{R} \cap \pi$ or $\mathsf{B} \cap \pi$ if needed. Let $(R_\pi, B_\pi, \lambda_\pi)$ be the recursive (internal) subproblem generated for π with $\rho_\pi = \rho \cap \pi$ as the associated rectangle. Then the above update operation ensures that $\mathcal{R} \cap \rho_\pi = \mathsf{R}_\pi$, $\mathcal{B} \cap \rho_\pi = \mathsf{B}_\pi$, and their weights are consistent with λ_π.

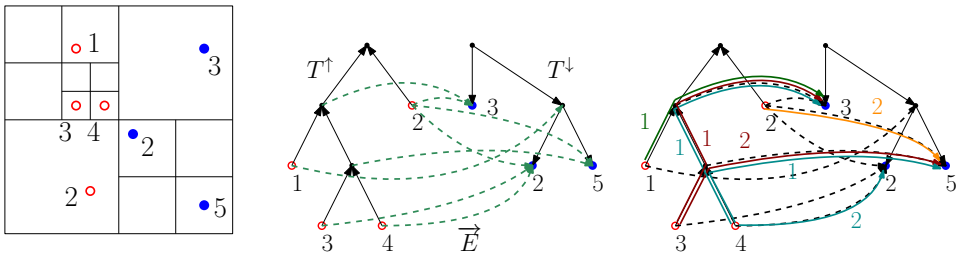
$\mathbb{D}_R$ and $\mathbb{D}_B$ can also be used to test whether the random shift is safe: For each $\pi \in \Pi$, we check whether the moat of any point in $(\mathsf{R} \cup \mathsf{B}) \cap \pi$ intersects an edge of π. This is equivalent to checking whether $\square_\pi \cap (\mathsf{R} \cup \mathsf{B}) = \emptyset$, where $\square_\pi$ is the set of points that are within distance ℓ/m^3 from the boundary of π. This test can be done in $O(\log^2 n)$ time using the Empty procedure. The total expected time spent in generating internal subproblems Σ_π and external subproblems $\Sigma_\square$ of Σ is $O(|\Pi| \log^3 n + m_\square \log^2 n)$, where $m_\square$ is the total number of external points.

By Lemma 9, the total number of nonempty cells over all subproblems is $O(n)$, and the number of external points is $O(n)$. Thus, the expected time spent in step (i) overall is $O(n \log^3 n)$. Putting everything together, we obtain the main result of this section.

▶ **Theorem 11.** *Let Σ be an instance of the transportation problem in $\mathbb{R}^d$, where d is a constant. Let Σ have size n, and let $\epsilon > 0$ be a constant. A transportation map of Σ can be computed in $O(n^{1+\epsilon})$ expected time whose expected cost is $O(\log^2(1/\epsilon))\mu(\tau^*)$, where τ^* is an optimal transport of Σ.*

3 A $(1+\epsilon)$-Approximate Algorithm

In this section, we describe a $(1+\epsilon)$-approximation algorithm for the transportation problem, based on a reduction to min-cost flow. As in Section 2, we hierarchically cluster points, but this time for the purpose of approximately representing all $\Theta(n^2)$ pairwise distances between R and B compactly. At a high level, our algorithm is as follows:



(a) Quadtree on $R \cup B$. (b) Composition of G. (c) Decomposition of flow.

Figure 4. Construction of G and recovering the transportation.

(i) Compute a hierarchical clustering of $R \cup B$, using a quadtree (input points are leaves).
(ii) Construct a sparse directed acyclic graph $G = (V, E)$ over the clusters with R as the set of source nodes, and B as the set of sink nodes with the following property: for every pair $(r, b) \in R \times B$, there is a unique path in G from r to b, with cost roughly $\mathrm{d}(r, b)$. Then, a minimum-cost flow from sources to sinks in G approximates the optimal transportation map on (R, B, λ).
(iii) Compute an optimal flow f^* in G using the algorithm by Lee and Sidford [18].
(iv) Recover a transportation map τ from f^*.

The hierarchical structure is important to us for two reasons: it is the foundation for the compact representation (well-separated pair decomposition), and enables a near-linear time procedure for recovering the transportation map (step (iv)). This fast recovery step is what distinguishes this algorithm from the similar reduction in Cabello *et al.* [9], and why *geometric spanners* [6, 7, 10], as black boxes, seem to be insufficient.

Construction of the graph. For simplicity, we describe the algorithm in $\mathbb{R}^2$ under Euclidean distance. Let $\square$ be the smallest orthogonal square containing $R \cup B$. We construct a *compressed* quad tree T on $R \cup B$ with $\square$ as the square associated with the root of T. A compressed quadtree prunes certain interior nodes of a standard quadtree, guaranteeing that T has $O(n)$ nodes. We can construct a compressed quadtree in $O(n \log n)$ time, see e.g. [15]. Each node v of T is associated with a square $\square_v$. For each node $v \in T$, let $R_v = R \cap \square_v$ and $B_v = B \cap \square_v$. The sets R_v, B_v form a hierarchical clustering of $R \cup B$.

To construct $G = (V, E)$, we make two copies of T: the *up-tree* $T^\uparrow = (V^\uparrow, E^\uparrow)$ and *down-tree* $T^\downarrow = (V^\downarrow, E^\downarrow)$. We orient the edges of $E^\uparrow$ upward — from a node to its parent, and orient the edges of $E^\downarrow$ downward — from a node to its child. We delete blue points from $T^\uparrow$ and red points from $T^\downarrow$, thus $T^\uparrow$ contains only R, and $T^\downarrow$ contains only B. We set $V = V^\uparrow \cup V^\downarrow$ and $E = E^\uparrow \cup E^\downarrow \cup \overrightarrow{E}$ where $\overrightarrow{E} \subseteq V^\uparrow \times V^\downarrow$ is a set of *cross edges* connecting $T^\uparrow$ to $T^\downarrow$ that we define below. See Figure 4.

Originally proposed by Callahan and Kosaraju [11], the notion of a *well-separated pair decomposition* (WSPD) of a point set S is widely used to represent the pairwise distances of S approximately in a compact manner. A simple WSPD construction using compressed quadtrees is described in Chapter 3 of [15]. Using this algorithm, we construct $\overrightarrow{E}$ as follows: Set $\delta = \frac{\epsilon}{4}$. For a pair of nodes $u, v \in T^\uparrow \times T^\downarrow$, we define $c(u, v) = \min\{\mathrm{d}(x, y) \mid x \in \square_u, y \in \square_v\}$ to be the minimum distance between the squares $\square_u$ and $\square_v$. Using the algorithm in [15], we compute a pair decomposition $D \subseteq V^\uparrow \times V^\downarrow$ with the following properties:

(W1) For every $(r, b) \in R \times B$, there is a unique pair $(u, v) \in D$ such that $r \in R_u, b \in B_v$.
(W2) For every $(u, v) \in D$, $\max\{\mathrm{diam}(\square_u), \mathrm{diam}(\square_v)\} \le \delta \cdot c(u, v)$.

(W3) $|D| = O(n/\epsilon^2)$.

After having constructed $T^\uparrow$ and $T^\downarrow$, that algorithm constructs D in $O(n/\epsilon^2)$ time. We set $\overrightarrow{E} = D$ with each edge oriented from $T^\uparrow$ to $T^\downarrow$. The cost of each edge in $E^\uparrow \cup E^\downarrow$ is set to 0, and the cost of each $(u, v) \in \overrightarrow{E}$ is $c(u, v)$. Finally, we define the *excess* $\chi(v)$ at each node v of G: we set $\chi(v) = 0$ for all internal nodes v of $T^\uparrow \cup T^\downarrow$, $\chi(r) = \lambda(r)$ for $r \in R$, and $\chi(b) = -\lambda(b)$ for $b \in B$. The capacity of all edges in G is set to ∞. Let (G, c, χ) be the resulting min-cost flow instance. The total time spent in constructing G is $O(n \log n + n/\epsilon^2)$.

Cost analysis. Flow moves up from the leaves of $T^\uparrow$ through its interior, crosses along the cross edges into $T^\downarrow$, and finally descends to the sinks at leaves of $T^\downarrow$. By construction and (W1), any pair $(r, b) \in R \times B$ has a unique path $p(r, b)$ from r to b in G, using a single cross edge. We can map any transport τ (injectively) to a feasible flow f_τ on G, by placing a flow of $\tau(r, b)$ on $p(r, b)$. Similarly, any flow f can be mapped to a feasible transportation τ_f by *decomposing* f into flow on source-sink paths: by the classical flow decomposition theorem, f can be decomposed into a set $\{f(p(r, b)\}$ of flows on the $p(r, b)$ (since G is a directed acyclic graph, any decomposition has no flow cycles, only paths). Then, setting $\tau_f(r, b) = f(p(r, b))$ for all $(r, b) \in R \times B$ is a feasible transportation.

By (W2) and the triangle inequality, $\mu(f_\tau) \le \mu(\tau)$ and $\mu(\tau_f) \le (1+\epsilon)\mu(f)$. We can apply these transformations to bound the approximation quality of a transportation recovered by decomposing the *optimal* flow f^* of G. We have

$$\mu(\tau^*) \le \mu(\tau_{f^*}) \le (1 + \epsilon)\mu(f^*) \le (1 + \epsilon)\mu(f_{\tau^*}) \le (1 + \epsilon)\mu(\tau^*).$$

If we compute the optimal f^* on G and construct some τ_{f^*} by a flow decomposition of f^*, then τ_{f^*} meets the claimed approximation quality — this is precisely what we do. Note that this cost analysis applies regardless of the specific flow decomposition of f^*. Our recovery procedure in (iv) is a greedy decomposition.

Recovering a transportation map. Let f^* be the optimal flow from R to B in G. We use a two-part greedy algorithm to decompose f^*: assigning the flow from R to the cross edges through the up-tree, then claiming the assigned flow using the down-tree. Both steps amount to performing a flow decomposition on the portion of the flow lying in each tree, treating the cross edge endpoints as sinks (resp., sources) with demand equal to the sum of outgoing (resp., incoming) flow. Both are arborescences, so flow decomposition can be done with a postorder traversal. After solving both trees, we combine the paths assigned into and out of each cross edge to find end-to-end path flows.

We only describe the decomposition for $T^\uparrow$ in detail; it is nearly identical for $T^\downarrow$. For each cross edge $(u, v) \in \overrightarrow{E}$, this produces lists $A_R(u, v)$ and $A_B(u, v)$ which hold pairs (p, F) indicating *point $p \in R \cup B$ contributes F units of the flow through (u, v)*. The tree is processed in postorder: a node is visited only after all its children. Denote the children of node u by children(u). Each node $u \in T^\uparrow$ maintains a list $L(u)$ of the positive-demand red points in its subtree, and a list $N(u)$ of the positive-flow cross edges leaving u. $L(u)$ is initialized by joining lists $L(w)$ from each $w \in$ children(u) (at the leaves, we initialize $L(r) = \{r\}$ for $r \in R$). While $N(u)$ is not empty, let $(u, v) \in N(u)$ with flow $f^*(u, v) > 0$. Take any point $r \in L(u)$ and add to $A_R(u, v)$ a pair (r, F), with $F = \min\{\lambda(r), f^*(u, v)\}$, also updating $\lambda(r) \leftarrow \lambda(r) - F$ and $f^*(u, v) \leftarrow f^*(u, v) - F$. This has the effect of removing either r from $L(u)$, (u, v) from $N(u)$, or both. Once $N(u) = \emptyset$, all cross edges leaving u have their flow assigned.

Finally, we complete the decomposition using $A_R(u, v)$ and $A_B(u, v)$, for each cross edge (u, v). While both $A_R(u, v)$ and $A_B(u, v)$ are nonempty, let $(r, F_r) \in A_R(u, v)$ and $(b, F_b) \in$

$A_B(u, v)$. Output $\tau(r, b) := \min\{F_r, F_b\}$, update $F_r \leftarrow F_r - \tau(r, b)$ and $F_b \leftarrow F_b - \tau(r, b)$, and remove from the lists any pair (p, F) for which $F = 0$.

We charge the list union which constructs $L(u)$ to the children of u. Each iteration performs a constant number of list operations and either removes a node from $L(u)$, or a cross edge from $N(u)$. Each removal occurs exactly once for every $r \in R$ and $(u, v) \in \overrightarrow{E}$, so we charge iterations to the $r \in R$ or $(u, v) \in \overrightarrow{E}$ removed that iteration. The processing of $A_R(u, v)$ and $A_B(u, v)$ can also be charged per iteration to the pair (p, F) removed in that iteration, which is then charged back to the $p \in R \cup B$ or $(u, v) \in \overrightarrow{E}$ whose removal introduced (p, F). Thus, the total running time is $O(|V| + |\overrightarrow{E}|) = O(n/\epsilon^2)$.

We computed an optimal flow $f^* : E \to \mathbb{N}$ using the algorithm by Lee and Sidford [18]. Since $|E| = O(n/\epsilon^2)$, their algorithm takes $\tilde{O}(n^{3/2}\epsilon^{-2} \operatorname{polylog} U)$ time; recall, $U = \max_{p \in R \cup B} \lambda(p)$ is the maximum demand. This step dominates the running time compared to the construction of G and the recovery algorithm. We state the main theorem in terms of an arbitrary d.

▶ **Theorem 12.** *Let Σ be an instance of the transportation problem in $\mathbb{R}^d$ where d is a constant. Let Σ have size n, and let $\epsilon > 0$ be a constant. A transportation map τ for Σ can be computed in $\tilde{O}(n^{3/2}\epsilon^{-d} \operatorname{polylog} U)$ time with cost $\mu(\tau) \leq (1 + \epsilon)\mu(\tau^*)$.*

References

1 P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29(3):912–953, 1999.

2 P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.

3 P. K. Agarwal and K. R. Varadarajan. A near-linear constant-factor approximation for Euclidean bipartite matching? In *Proc. of the 20th ACM Symp. on Comp. Geometry*, pages 247–252, 2004.

4 A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proc. of the 46th Ann. ACM Symp. on Theory of Comp.*, pages 574–583, 2014.

5 S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. of the 37th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 2–11, 1996.

6 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. H. M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. of the 27th Ann. ACM Symp. on Theory of Comp.*, pages 489–498, 1995.

7 S. Arya, D. M. Mount, and M. H. M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. of the 35th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 703–712, 1994.

8 D. S. Atkinson and P. M. Vaidya. Using geometry to solve the transportation problem in the plane. *Algorithmica*, 13(5):442–461, 1995.

9 S. Cabello, P. Giannopoulos, C. Knauer, and G. Rote. Matching point sets with respect to the Earth Mover's Distance. *Comput. Geom.*, 39(2):118–133, 2008.

10 P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. of the 4th Ann. ACM/SIGACT-SIAM Symp. on Discrete Algo.*, pages 291–300, 1993.

11 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. ACM*, 42(1):67–90, 1995.

12 M. Cuturi and A. Doucet. Fast computation of Wasserstein barycenters. In *Proc. of the 31th Internat. Conf. on Machine Learning*, pages 685–693, 2014.

13 A. Gramfort, G. Peyré, and M. Cuturi. Fast optimal transport averaging of neuroimaging data. In *Proc. of the 24th Internat. Conf. on Infor. Processing in Medical Imaging*, pages 261–272. Springer, 2015.

14 K. Grauman and T. Darrell. Fast contour matching using approximate earth mover's distance. In *Proc. of the 24th Ann. IEEE Conf. on Comp. Vision and Pattern Recog.*, volume 1, pages I–220. IEEE, 2004.

15 S. Har-Peled. *Geometric Approximation Algorithms*, volume 173. American Mathematical Society Providence, 2011.

16 P. Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *Proc. of the 18th Ann. ACM-SIAM Symp. on Discrete Algo.*, pages 39–42, 2007.

17 H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *CoRR*, abs/1604.03654, 2016.

18 Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}$(vrank) iterations and faster algorithms for maximum flow. In *Proc. of the 55th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 424–433, 2014.

19 J. B. Orlin. A faster strongly polynominal minimum cost flow algorithm. In *Proc. of the 20th Annual ACM Symp. on Theory of Comp., May 2-4, 1988, Chicago, Illinois, USA*, pages 377–387, 1988.

20 Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *6th Internat. Conf. on Comp. Vision*, pages 59–66, 1998.

21 R. Sharathkumar and P. K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proc. of the 23rd Ann. ACM-SIAM Symp. on Discrete Algo.*, pages 306–317, 2012.

22 R. Sharathkumar and P. K. Agarwal. A near-linear time ε-approximation algorithm for geometric bipartite matching. In *Proc. of the 44th Ann. ACM Symp. on Theory of Comp.*, pages 385–394, 2012.

23 J. Solomon, R. M. Rustamov, L. J. Guibas, and A. Butscher. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics*, 33(4):67:1–67:12, 2014.

24 K. Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proc. of the 36th Ann. ACM Symp. on Theory of Comp.*, pages 281–290, 2004.

25 P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18(6):1201–1225, 1989.

26 K. R. Varadarajan and P. K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *Proc. of the 10th Ann. ACM-SIAM Symp. on Discrete Algo.*, pages 805–814, 1999.

27 C. Villani. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.