Preconditioning for the Geometric Transportation Problem

Andrey Boris Khesin
University of Toronto, Toronto, Ontario, Canada

Aleksandar Nikolov
University of Toronto, Toronto, Ontario, Canada
http://www.cs.toronto.edu/~anikolov/
anikolov@cs.toronto.edu

Dmitry Paramonov
University of Toronto, Toronto, Ontario, Canada

Abstract

In the geometric transportation problem, we are given a collection of points P in d-dimensional Euclidean space, and each point is given a supply of $\mu(p)$ units of mass, where $\mu(p)$ could be a positive or a negative integer, and the total sum of the supplies is 0. The goal is to find a flow (called a transportation map) that transports $\mu(p)$ units from any point p with $\mu(p) > 0$, and transports $-\mu(p)$ units into any point p with $\mu(p) < 0$. Moreover, the flow should minimize the total distance traveled by the transported mass. The optimal value is known as the transportation cost, or the Earth Mover's Distance (from the points with positive supply to those with negative supply). This problem has been widely studied in many fields of computer science: from theoretical work in computational geometry, to applications in computer vision, graphics, and machine learning.

In this work we study approximation algorithms for the geometric transportation problem. We give an algorithm which, for any fixed dimension d, finds a $(1+\varepsilon)$-approximate transportation map in time nearly-linear in n, and polynomial in ε^{-1} and in the logarithm of the total supply. This is the first approximation scheme for the problem whose running time depends on n as $n \cdot \text{polylog}(n)$. Our techniques combine the generalized preconditioning framework of Sherman, which is grounded in continuous optimization, with simple geometric arguments to first reduce the problem to a minimum cost flow problem on a sparse graph, and then to design a good preconditioner for this latter problem.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Network flows

Keywords and phrases Earth Mover Distance, Transportation Problem, Minimum Cost Flow

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.15

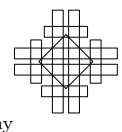
Related Version A full version is available at https://arxiv.org/abs/1902.08384.

Funding *Aleksandar Nikolov*: Supported by an NSERC Discovery Grant (RGPIN-2016-06333).

1 Introduction

In the *Geometric Transportation problem*, we are given a set P of n points in d-dimensional Euclidean space, and a function $\mu : P \to \mathbb{Z}$ that assigns a (positive or negative integer) weight to each point, so that $\sum_{p \in P} \mu(p) = 0$. We call μ the supply function. We can think of each point $p \in P$ as either a pile of earth, or a hole, and $\mu(p)$ gives, respectively, the amount of earth (if positive) or the size of the hole (if negative). The constraint on μ means that the total space in the holes equals the total amount of earth in the piles. Our goal is to find the most efficient way to transport the earth to the holes, where the cost of transporting a unit of mass from p to q equals the distance $\|p-q\|_2$ it must travel, measured in the Euclidean norm.

35th International Symposium on Computational Geometry (SoCG 2019).
Editors: Gill Barequet and Yusu Wang; Article No. 15; pp. 15:1–15:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


More formally, let $P_+ = \{p \in P : \mu(p) \geq 0\}$ be the "piles" and $P_- = \{q \in P : \mu(q) < 0\}$ be the "holes". We want to solve the minimum cost flow problem given by the following linear program

$$\begin{aligned}
&\text{Minimize} \sum_{p \in P_+, q \in P_-} f_{pq} \|p - q\|_2 \\
&\text{subject to} \\
&\forall p \in P_+ : \sum_{q \in P_-} f_{pq} = \mu(p), \\
&\forall q \in P_- : \sum_{p \in P_+} f_{pq} = \mu(q), \\
\forall p \in P_+, q \in P_- :& f_{pq} \geq 0.
\end{aligned}$$

Above, the constraints enforce that all demands are satisfied, i.e. all the earth goes into holes, and no hole is overfilled. A vector $f \in \mathbb{R}^{P_+ \times P_-}$ that satisfies these constraints is called a *transportation map* for (P, μ). The cost of an optimal transportation map is called the *transportation cost* of (P, μ), or also the Earth Mover's Distance, or 1-Wasserstein distance between the measures $\phi = \mu|_{P_+}$ and $\psi = -\mu|_{P_-}$, and is denoted $\textsc{cost}(P, \mu)$.

The geometric transportation problem is a discretized version of the continuous optimal mass transportation problem of Monge and Kantorovich. This is a classical problem in mathematics, with many beautiful connections to partial differential equations, geometry, and probability theory, among others [19]. The discrete problem above has also found a large number of applications, for example to shape matching [9, 10] and image retrieval [15, 13] in computer vision, and interpolation between distributions in graphics [6]. It is also of intrinsic interest as a natural problem in computational geometry.

The geometric transportation problem can be modeled as a minimum cost flow problem in the complete bipartite graph with bi-partition $P_+ \cup P_-$. This graph is dense, and solving the minimum cost flow problem exactly using the algorithm of Lee and Sidford would take time $O(n^{2.5} \operatorname{polylog}(U))$,[1] where $U = \sum_{p \in P_+} \mu(p)$. If instead we settle for an $(1+\varepsilon)$-approximation, then the recent algorithm of Sherman [18] gives running time $O(n^{2+o(1)}\varepsilon^{-2})$. However, there exist faster (in certain parameter regimes) approximation algorithms which exploit the geometric structure of the problem: after a long line of work, Sharathkumar and Agarwal [17] gave an algorithm that computes a $(1+\varepsilon)$-approximation in time $O(n\sqrt{U} \operatorname{polylog}(U, \varepsilon, n))$, and, recently, Agarwal et al. [1] gave several algorithms with different trade-offs, among them an $(1+\varepsilon)$-approximation algorithm running in time $O(n^{3/2}\varepsilon^{-d} \operatorname{polylog}(U, n))$. A nearly linear time $(1+\varepsilon)$-approximation algorithm was given by Sharathkumar and Agarwal [16] in the special case of unit supplies; their algorithm runs in time $O(n\varepsilon^{-O(d)} \operatorname{polylog}(n))$. Until now, no such algorithm was known for general supply functions.

A related line of work focuses on estimating the transportation cost, without necessarily computing a transportation map. An influential paper in this direction was that of Indyk [11], whose algorithm gives a constant factor approximation to the transportation cost in the case of unit supplies, in time $O(n \operatorname{polylog}(n))$. This result was extended to arbitrary supplies and to approximation factor $1+\varepsilon$ by Andoni et al. [4], whose algorithm runs in time $O(n^{1+o(1)})$, with the $o(1)$ factor in the exponent hiding dependence on ε. The result of Sherman [18], mentioned above, together with the existence of sparse Euclidean spanners, implies an

[1] We allow the implicit constants in the asymptotic notation to depend on the dimension d, except when the asymptotic notation is used in an exponent.

$O(n^{1+o(1)}\varepsilon^{-O(d)})$ time algorithm to estimate the transportation cost.[2] It is not immediately clear, however, how to use the flow in G to construct a transportation map of comparable cost in nearly linear time.

There is a related line of work [14, 2, 3] that studies the transportation problem when the cost of transporting mass from p to q is $\|p-q\|_2^r$, giving the r-Wasserstein distance. This appears to be a more challenging problem, and we do not address it further.

1.1 Our results and methods

Let us recall that the aspect ratio (or spread) of a pointset P is defined as the ratio of its diameter to the smallest distance between two distinct points. Our main result is a nearly linear time $(1+\varepsilon)$-approximation algorithm for the geometric transportation problem, as captured by the following theorem.

▶ **Theorem 1.** *There exists a randomized algorithm that on input an n-point set $P \subset \mathbb{Q}^d$ with aspect ratio Δ, and supply function $\mu : P \to \mathbb{Z}$, runs in time $O(n\varepsilon^{-O(d)} \log(\Delta)^{O(d)} \log n)$, and with probability at least $1/2$ finds a transportation map with cost at most $(1+\varepsilon) \cdot \textsc{cost}(P,\mu)$.*

There also exists a randomized algorithm that on input an n-point set $P \subset \mathbb{Q}^d$ and supply function $\mu : P \to \mathbb{Z}$ such that $U = \sum_{p \in P_+} \mu(p)$, runs in time $O(n\varepsilon^{-O(d)} \log(U)^{O(d)} \log(n)^2)$, and with probability at least $1/2$ finds a transportation map with cost at most $(1+\varepsilon) \cdot \textsc{cost}(P,\mu)$.

In constant dimension, the dependence of the running time on the aspect ratio Δ or total supply U is polylogarithmic, and the dependence on the approximation ε is polynomial. The dependence on n is just $O(n \log n)$ (respectively $O(n \log(n)^2)$). This is in contrast with prior work which either had a much larger dependence on n, or a polynomial dependence on U.

In the proof of this result, we employ a combination of geometric tools, and tools from continuous optimization developed for the general minimum cost flow problem. As a first step, we reduce the transportation problem to an (uncapacitated) minimum cost flow problem on a random sparse graph. This construction is closely related to the prior work of Sharathkumar and Agarwal in the case of unit capacities [16], and also to the estimation algorithm for the transportation cost in [4]. In particular, the sparse graph and minimum cost flow instance we construct are a simplification of the minimum cost flow instance in [4]. Together with the recent work of Sherman [18], this reduction is enough to get a $O(n^{1+o(1)}\varepsilon^{-O(d)})$ time algorithm to estimate the transportation cost. As mentioned above, this running time can also be achieved using a Euclidean spanner, and the random sparse graph we use can, in fact, be seen as a randomized spanner. Our graph, however, has a nice hierarchical structure not shared by other spanner constructions. In particular, there is a quadtree of the input point set such that any edge leaving a cell of the quadtree goes to either a sibling cell or a parent cell. This property is useful both for improving the running time further, and for computing a transportation map.

To further improve the running time, we open up Sherman's elegant framework, and combine it with classical geometric constructions. Sherman showed that repeatedly finding a flow which is approximately optimal, and approximately feasible gives a fast algorithm, as long as the problem being solved is sufficiently well conditioned. Unfortunately, most minimum cost flow problems are not well-conditioned in their natural formulations, and we need to construct a *preconditioner*. We exploit ideas from the known embeddings of

[2] We are indebted to an anonymous reviewer for this observation.

transportation cost into ℓ_1 [7, 12] to construct a preconditioner with condition number that depends polynomially on the approximation factor and the logarithm of the aspect ratio. The insight that these simple and well-known techniques can be repurposed to give high-quality preconditioners is one of our main conceptual contributions. Finally, we also give a simple method to extract a transportation map from a low-cost flow in our random sparse graph.

2 Notation and basic notions

We use the notation $\|x\|_p$ for the ℓ_p norm of a vector $x \in \mathbb{R}^d$: $\|x\|_p = \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$. We will assume that the input $P \subset \mathbb{Q}^d$ to our problem lies in $[0, \Delta]^d$, and that the smallest Euclidean distance between any two points is at least 1. Since any shifts, rotations, or dilations of the points do not change the problem, this assumption is equivalent to assuming that the aspect ratio of P, i.e. the ratio between the diameter of P and the smallest distance between two distinct points, is bounded between Δ and $\sqrt{d}\Delta$.

In fact, at the cost of a small increase in the approximation factor, we can reduce to this case of bounded aspect ratio. The reduction produces a point set P for which Δ is bounded by a factor that depends on $U = \sum_{p \in P_+} \mu(p)$.

▶ **Lemma 2** ([11, 4]). *Suppose that there exists a function T, increasing in all its parameters, and an algorithm which, for any $\varepsilon, \delta < 1$, on input of size n in $[0, \Delta]^d \cap \mathbb{Z}^d$, runs in time $O(nT(\Delta, \varepsilon, \delta))$, and with probability $1 - \delta$ computes a $1 + \varepsilon$ approximation to the geometric transportation problem. Then there exists an algorithm that takes any input $P \subset \mathbb{R}^d$ of size n and $\mu : P \to \mathbb{Z}$ such that $U = \sum_{p \in P_+} \mu(p)$, runs in time $O(nT(cdU/\varepsilon^2, \varepsilon, \delta n))$ for an absolute constant c, and, with probability $1 - \delta$, achieves an approximation of $1 + O(\varepsilon)$ for the geometric transportation problem on P and μ.*

With this lemma, the second algorithm in Theorem 1 follows from the first one. For this reason, we can focus our presentation on the case of bounded aspect ratio.

For a set V, we call $f \in \mathbb{R}^{V \times V}$ a *flow* if it is anti-symmetric, i.e. $f_{uv} = -f_{vu}$ for every $u, v \in V$. Intuitively, we think of $f_{uv} > 0$ as flow going in the direction from u to v. For a graph $G = (V, E)$, we define a flow on G to be a flow f supported on E, i.e. one such that $f_{uv} = 0$ for any $(u, v) \notin E$. The *divergence* of a flow f at u is the quantity $\sum_{v \in V} f_{uv}$, i.e. the excess of the flow leaving u over the flow entering u. When the divergence at u is 0, we say that *flow conservation* is satisfied at u.

In general, we assume that our graphs are stored in the adjacency list representation, and that flow values are stored with the adjacency list.

3 Reduction to minimum cost flow on a sparse graph

The first step in our algorithm is to reduce the geometric transportation problem, which is naturally modeled as a minimum cost flow problem on a complete bipartite graph, to another minimum cost flow on a sparse random graph. The following construction is a simplified version of one used in [4].

3.1 Graph construction

Recall that $P \subset [0, \Delta^d]$. We start by constructing a grid containing all the points in P as follows. We sample a uniformly random point $x \in [0, \Delta]^d$, and define the cell $C_0 = [-\Delta, \Delta]^d + x$. Note that all points in P are in C_0. We say that C_0 is on *level 0*. The set of

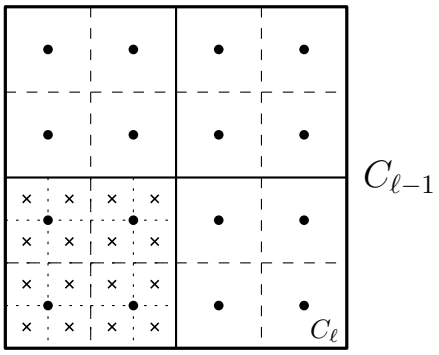


Figure 1 A cell $C_{\ell-1}$ on level $\ell-1$ and the four cells on level ℓ contained in it. The subcells of $C_{\ell-1}$ are shown with dashed lines, and the corresponding net points in $N_{\ell-1}$ with black dots. The subcells of one of the cells on level ℓ are shown with dotted lines, and the corresponding net point in N_ℓ with crosses.

cells on level ℓ is labelled $\mathcal{C}_\ell$ and is constructed by taking each cell C in $\mathcal{C}_{\ell-1}$ and dividing it into 2^d equally-sized cubes of side length half the side length of C, and retaining those that contain points in P. We say that C is the *parent* of the cells into which it was divided. Conversely, those 2^d cells are called the *children* of C. This division is continued until level L, where all points in P lie in different cells in $\mathcal{C}_L$. With probability 1, no point of P lands on the boundary of any cell, so we will ignore this event.

Since the side length of any cell on level ℓ is half of the side length of a cell on level $\ell-1$, any cell in $\mathcal{C}_\ell$ has side length $2^{1-\ell}\Delta$, and we refer to this length as Δ_l. Also note that, since the closest two points in P are at least distance 1 apart, we have $L \le \log_2(2\sqrt{d}\Delta)$, so $L = O(\log \Delta)$. Moreover, any point $p \in P$ lies in at most $L+1$ cells, one per level, and, since in $\mathcal{C}_\ell$ we only retain cells that contain points in P, we have that $|\mathcal{C}_0 \cup \ldots \cup \mathcal{C}_L| \le n(L+1) = O(n \log \Delta)$.

Let ε_0 be a small number such that ε_0^{-1} is an *even* integer. This ε_0 is not the same as the value used when we speak of a $(1+\varepsilon)$-approximation, although the two are related, as we will see. Next, for each $\ell \in \{0, \ldots, L$, we take each cell $C \in \mathcal{C}_\ell$, and we divide it into ε_0^{-d} *subcells*, each of side length $\varepsilon_0 \Delta_l$. The set of all such subcells of all $C \in \mathcal{C}_\ell$ is denoted $\widetilde{\mathcal{C}}_\ell$. To each subcell we associate a *net point* at its centre and we denote the set of all net points on level ℓ as N_ℓ. See Figure 1 for an illustration. Note that $|N_\ell| = |\mathcal{C}_\ell|\varepsilon_0^{-d}$, so $|N_0 \cup \ldots \cup N_L| \le n(L+1)\varepsilon_0^{-d} = O(n\varepsilon_0^{-d} \log \Delta)$.

If p is a point in C_0, let $C_\ell(p) \in \widetilde{\mathcal{C}}_\ell$ be the unique subcell on level ℓ that contains p. Similarly, let $N_\ell(p)$ be the net point on level ℓ closest to p. Equivalently, $N_\ell(p)$ is the center of $C_\ell(p)$. We say that $N_\ell(p)$ and $N_{\ell-1}(p)$ are each other's *child* and *parent*, respectively.

We are now ready to construct our graph $G = (V, E)$. We set $V = P \cup N_0 \cup N_1 \cup \cdots \cup N_L$, the set of all of the points in the original problem and all of the netpoints we have constructed. By the previous discussion, $|V| = O(n\varepsilon_0^{-d} \log \Delta)$. We build our graph using three types of edges. The first set, E_1, connects points p in P to their closest net point on level L, i.e. $E_1 = \{(p, N_L(p)) : p \in P\}$. The size of this set is $|E_1| = n$. The second set of edges, E_2, connects all net points of subcells of a given cell pairwise. Thus,

$$E_2 = \{(u, v) : \ell \in \{0, 1, \ldots, L\}, C \in \mathcal{C}_\ell, u, v \in N_\ell \cap C \text{ s.t. } u \ne v\}.$$

Since $|\mathcal{C}_\ell| \le n$, and, for any $C \in \mathcal{C}_\ell$, $|N_\ell \cap C| = \varepsilon_0^{-d}$, we have that $|E_2| \le (L+1)n\varepsilon_0^{-2d} = O(n\varepsilon_0^{-2d} \log \Delta)$. The last set of edges, E_3, connects net points to their parent net points, i.e.

$$E_3 = \{(N_{\ell-1}(u), u) : \ell \in \{1, 2, \ldots, L\}, u \in N_\ell\}.$$

The size of E_3 is less than the total number of net points, so $|E_3| = O(n\varepsilon_0^{-d} \log \Delta)$. Then, the set of edges is defined as $E = E_1 \cup E_2 \cup E_3$, and this completes the description of $G = (V, E)$. The number of edges is dominated by the size of E_2, and is in $O(n\varepsilon_0^{-2d} \log \Delta)$. We will be calculating distances between points along paths in this graph, and we define the distance/cost of any edge to equal the Euclidean distance between the two endpoints of the edge.

For the rest of the paper we will use G to refer to the random graph constructed above. Having stored the random shift of C_0, we can enumerate the vertices in time $O(n\varepsilon_0^{-d} \log \Delta)$ by going through each point in P and checking which cells it lies in. We can then store the vertices in a static dictionary [8]. Similarly, we can construct the adjacency lists of the vertices in time $O(n\varepsilon_0^{-2d} \log \Delta)$.

3.2 Preserving Euclidean distance

Let $\mathrm{dist}_G(u, v)$ be the shortest path distance between two nodes u and v of G, i.e.

$$\mathrm{dist}_G(u, v) = \min \sum_{i=1}^{k} \|u_i - u_{i-1}\|_2,$$

with the minimum taken over paths $u = u_0, \ldots, u_k = v$ connecting u and v in G. Our goal for this section is to show that, for any two p and q in P, the expected value of $\mathrm{dist}_G(u, v)$ is close to the Euclidean distance between them. In other words, we want to show that our random graph construction provides a randomized embedding of Euclidean distance into the shortest path metric of a sparse graph. This is similar to the embeddings used in [4] and [17, 16], and can be seen as a randomized spanner construction, in which the graph G only needs to have low stretch in expectation.

For two points p and q in P, we define $\ell(p, q)$ to be the level of the smallest cell containing both p and q, which can depend on the random shift of the grid. This means that $C_{\ell(p,q)}(p) = C_{\ell(p,q)}(q)$, but $C_{\ell(p,q)+1}(p) \neq C_{\ell(p,q)+1}(q)$. The following definition will be useful.

▶ **Definition 3.** *Given two points $p, q \in P$, the canonical path between p and q in the random graph G consists of the edges*

$$\{(N_\ell(p), N_{\ell-1}(p) : \ell(p, q) + 1 \leq \ell \leq L\} \cup \{(N_{\ell(p,q)}(p), N_{\ell(p,q)}(q))\} \\ \cup \{(N_\ell(q), N_{\ell-1}(q) : \ell(p, q) + 1 \leq \ell \leq L\} \cup \{(N_L(p), p), (N_L(q), q)\}.$$

The canonical path could be much longer than $\|p - q\|_2$ because, if p and q are two nearby points that are separated by a grid line, the canonical path will be much larger than the Euclidean distance. However, when p and q are very close together, the likelihood that such a grid line will fall between them is low. To formalize this intuition, we need the following well-known bound on the probability that p and q will be split at a certain level of the tree, which is elementary, and goes back at least to Arora's work on Euclidean TSP [5].

▶ **Lemma 4.** *Over the random shift of C_0, $\forall p, q \in P$,*

$$\mathbb{P}(C_\ell(p) \neq C_\ell(q)) \leq \frac{\|p - q\|_1}{\Delta_\ell} \leq \frac{\sqrt{d}\|p - q\|_2}{\Delta_\ell}.$$

Using this standard lemma, we can prove the bound on the distortion between Euclidean distance and the expected shortest path distance in G. The proof, which is standard, is presented in the full version of the paper.

▶ **Lemma 5.** *Let $p, q \in P$. Then, for any shift of C_0, $\|p-q\|_2 \leq \mathrm{dist}_G(p,q)$. Moreover, over the random shift of C_0, $\mathbb{E}[\mathrm{dist}_G(p,q)] \leq (1+O(\varepsilon_0 L))\|p-q\|_2$.*

3.3 Approximating the transportation cost with minimum cost flow

We are now ready to translate the transportation problem to a minimum cost flow problem in G, captured by the following linear program.

$$\text{Minimize} \sum_{(u,v)\in E} f_{uv}\|u-v\|_2 \tag{1}$$

subject to

$$\forall u,v \in V : f_{uv} = -f_{vu} \tag{2}$$

$$\forall p \in P : \sum_{u\in V} f_{pu} = \mu(p), \tag{3}$$

$$\forall u \in V \setminus P : \sum_{v\in v} f_{uv} = 0. \tag{4}$$

Let us denote by $\textsc{cost}(G,\mu)$ the value of the optimal solution to the problem (1)–(4). Note that, since G is a random graph, $\textsc{cost}(G,\mu)$ is also a random variable.

The next theorem shows that that $\textsc{cost}(G,\mu)$ approximates the transportation cost $\textsc{cost}(P,\mu)$. The proof is given in the full version of the paper.

▶ **Theorem 6.** *For any shift of C_0, $\textsc{cost}(P,\mu) \leq \textsc{cost}(G,\mu)$. Moreover, over the random shift of C_0,*

$$\mathbb{E}[\textsc{cost}(G,\mu)] \leq (1+O(\varepsilon_0 L))\textsc{cost}(P,\mu).$$

4 Solving the minimum cost flow problem

In this section we describe the generalized preconditioning framework of Sherman [18], and describe how to use it to solve the minimum cost flow problem (1)–(4) in nearly linear time.

4.1 Generalized preconditioning framework

We take a short detour to describe the generalized preconditioning framework of Sherman. Suppose that $\|\cdot\|$ is a norm on $\mathbb{R}^m$, and we are given inputs $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^n$. The framework is concerned with solving the following optimization problem over the variables $f \in \mathbb{R}^m$:

$$\text{Minimize } \|f\| \tag{5}$$

$$\text{subject to } \quad Af = b,. \tag{6}$$

This formulation can capture many important problems, including, crucially for us, the uncapacitated minimum cost flow problem,[3] as we explain later.

A central definition in the framework is that of the nonlinear condition number. Before we state it, let us recall that the norm of a linear map $T : X \to Y$, where $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ are two finite dimensional normed vector spaces, is defined as

$$\|T\| = \sup_{x\neq 0} \frac{\|T(x)\|_Y}{\|x\|_X}.$$

[3] Uncapacitated minimum cost flow is commonly known as the transportation problem. We do not use this terminology, in order to avoid confusion with the geometric transportation problem.

We adopt the same definition for non-linear maps, as well. For an $n \times m$ matrix A, and norms $\|\cdot\|_X$ on $X = \mathbb{R}^n$ and $\|\cdot\|_Y$ on the column span Y of A, we use the notation $\|A\|_{\|\cdot\|_X \to \|\cdot\|_Y}$ to refer to the norm of the linear map from X to Y represented by A.

Having recalled these notions, we proceed to define the condition number.

▶ **Definition 7.** *Let $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ be two finite dimensional normed vector spaces. The non-linear condition number κ of a linear map $T : X \to Y$ is defined by*

$$\kappa(T) = \inf_S \|T\|\|S\|,$$

where $S : Y \to X$ ranges over all (not necessarily linear) maps such that for all $x \in X$ we have $T(S(Tx)) = Tx$.

For an $n \times m$ matrix A, and norms $\|\cdot\|_X$ on $X = \mathbb{R}^m$ and $\|\cdot\|_Y$ on the column span Y of A, we define $\kappa_{\|\cdot\|_X \to \|\cdot\|_Y}(A)$ as $\kappa(T)$, where $T : X \to Y$ is the linear map represented by the matrix A.

This definition generalizes the standard condition number, which is the special case in which both norms are taken to be Euclidean.

The generalized preconditioning framework is based on composing rough approximation algorithms to get a high-quality approximation. The rough approximation algorithms are further allowed to violate the constraints of the problem slightly. Thus, they achieve a bi-criteria approximation, captured by the following definition.

▶ **Definition 8.** *Let $\|\cdot\|_Y$ be a norm defined on the range of the matrix A in (6). Let f^* be an optimal solution to the problem (5)–(6). Then $f \in \mathbb{R}^m$ is called an (α, β)-solution (with respect to $\|\cdot\|_Y$) to (5)–(6) if $\|f\| \le \alpha\|f^*\|$ and, moreover,*

$$\|Af - b\|_Y \le \beta\|A\|_{\|\cdot\| \to \|\cdot\|_Y}\|f^*\|.$$

An algorithm that, when given inputs A and b, outputs an (α, β)-solution f is called an (α, β)-solver (with respect to $\|\cdot\|_Y$) for (5)–(6).

We use the next result of Sherman, which gives a solver for (5)–(6) with running time controlled by condition number of the constraint matrix.

▶ **Theorem 9** ([18])**.** *Let $\varepsilon, \beta > 0$, and suppose that the norm in (5) is the ℓ_1^m norm. Let $\kappa = \kappa_{\|\cdot\|_1 \to \|\cdot\|_1}(A)$, and let M be an upper bound on the time necessary to compute matrix-vector products with A and $A^\top$. Then there exists a $(1+\varepsilon, \beta)$-solver with respect to $\|\cdot\|_1$ for the problem (5)–(6) with running time bounded by*

$$O\left(\kappa^2(m+n+M)\log(m)(\varepsilon^{-2} + \log(1/\beta))\right).$$

This algorithm is based on repeatedly applying an (α, β) solver with much worse dependence on β; in fact, at the cost of a slightly worse dependence on κ, for the latter one can use a simple solver based on the multiplicative weights update method.

Rescaling f coordinatewise gives us the following easy corollary, whose proof is omitted.

▶ **Corollary 10.** *Let $\varepsilon, \beta > 0$, and let $c \in \mathbb{R}^m_{>0}$. Suppose that the norm in (5) is given by $\|f\|_c = \sum_{i=1}^m c_i f_i$. Let $\kappa = \kappa_{\|\cdot\|_c \to \|\cdot\|_1}(A)$, and let M be the time necessary to compute matrix-vector products with A and $A^\top$. Then there exists a $(1+\varepsilon, \beta)$-solver with respect to $\|\cdot\|_1$ for the problem (5)–(6) with running time bounded by $O\left(\kappa^2(m+n+M)\log(m)(\varepsilon^{-2} + \log(1/\beta))\right)$.*

While Theorem 9 and Corollary 10 allow us to find solutions which are very close to being feasible, they do not give an exactly feasible solution. The final result we use from Sherman allows us to use a solver which exactly satisfies the constraints, but only achieves a large approximation ratio, to round an approximately feasible solution to an exactly feasible one.

To state the result, we need to define the composition of two solvers $\mathcal{F}$ and $\mathcal{F}'$. The composed algorithm $\mathcal{F}' \circ \mathcal{F}$ first calls $\mathcal{F}$ on A and b to get f; then it calls $\mathcal{F}'$ on A and the residual vector $b - Af$ to get a solution f'; finally it outputs $f + f'$.

▶ **Theorem 11** ([18]). *Let $C \geq 1$, and $\varepsilon, \beta > 0$. Let $\kappa = \kappa_{\|\cdot\| \to \|\cdot\|_Y}(A)$, where $\|\cdot\|$ is the norm in (5), and $\|\cdot\|_Y$ is some norm on the column span of A. Then, if $\mathcal{F}$ is a $(1+\varepsilon, \varepsilon\beta/\kappa)$-solver, and $\mathcal{F}'$ is a $(C, 0)$-solver, the composition $\mathcal{F}' \circ \mathcal{F}$ is a $(1 + \varepsilon + C\varepsilon\beta, 0)$-solver.*

4.2 Preconditioner construction

The formulation (5)–(6) captures the (uncapacitated) minimum cost flow problem with arbitrary demands. We will explain this for the graph $G = (V, E)$ defined in Section 3.2 and the minimum cost flow problem (1)–(4). Let us pick an arbitrary orientation on the edges E, and call the directed edges $\vec{E}$. Then take A to be the directed vertex by edge incidence matrix of G: it is indexed by $V \times \vec{E}$, and for any node u, and any directed edge $e = (v, w)$, set $A_{u,e} = 1$ if $u = v$, $A_{u,e} = -1$ if $u = w$, and $A_{u,e} = 0$ otherwise. We represent a flow f by its restriction to $\vec{E}$, seen as a vector in $\mathbb{R}^{\vec{E}}$. Slightly abusing notation, we will use the letter f both for this vector, and for the flow, which is defined both for $(u, v) \in \vec{E}$, and for (v, u), with $f_{vu} = -f_{uv}$. For a flow vector f, the product Af gives us the vector of divergences, i.e. for any $u \in V$

$$(Af)_u = \sum_{v:(u,v)\in\vec{E}} f_{uv} - \sum_{v:(v,u)\in\vec{E}} f_{vu} = \sum_{v:(u,v)\in E} f_{uv}.$$

We define the vector $b \in \mathbb{R}^V$ to encode the supplies, i.e. for $p \in P$ we set $b_p = \mu(p)$, and for $u \in V \setminus P$ we set $b_u = 0$. It follows that the constraint $Af = b$ encodes (3)–(4). Finally, let us denote the cost of an edge $e = (u, v) \in \vec{E}$ by $c(e) = \|u - v\|_2$. Then, for the norm in the objective (5), we choose $\|f\|_c = \sum_{e\in E} c(e)|f_e|$. With these choices of A, b, and $\|\cdot\| = \|\cdot\|_c$, an optimal solution to (5)–(6) gives an optimal solution to (1)–(4). For the rest of this section we will fix A, b, c, and $\|\cdot\|_c$ to be as just defined.

Unfortunately, we cannot directly use Corollary 10 to get the running time we are aiming for, since the condition number of the matrix A could be large. We address this by designing a preconditioner: another matrix B, of full column rank, which can be applied quickly, and has the property that BA has small condition number. This allows us to apply Corollary 10 to the problem (5)–(6) with the modified, but equivalent, constraint $BAf = Bb$, and get a fast algorithm for the minimum cost flow problem in G.

In our construction of the preconditioner B we will use the following lemma, which was implicit in [18]. We omit the proof, which follows from the arguments in [18].

▶ **Lemma 12.** *Let $\|\cdot\|_X$ be a norm on $\mathbb{R}^n$, and let H be an $n \times m$ matrix. Suppose that there exists a $\gamma > 0$ such that for any h in the column span Y of H, the norm $\|\cdot\|_Y$ satisfies*

$$\|h\|_Y \leq \min\{\|f\|_X : f \in \mathbb{R}^n, Hf = h\} \leq \gamma\|h\|_Y.$$

Then $\kappa_{\|\cdot\|_X \to \|\cdot\|_Y}(H) \leq \gamma$.

We will design a preconditioner matrix B such that $\|\tilde{b}\|_1$ approximates the cost of the minimum cost flow with supply vector $\tilde{b}$. Then the fact that BA has small condition number will follow from Lemma 12. The following lemma captures the construction of B, which is inspired by embedding of the Earth Mover Distance in ℓ_1 [7, 12].

▶ **Lemma 13.** *There exists a matrix $B \in \mathbb{R}^{V\times V}$ of full column rank with at most $O(|V|\log\Delta)$ nonzero entries, such that the following holds. For any $\tilde{b} \in \mathbb{R}^V$ such that $\sum_{u\in V}\tilde{b}_u = 0$, we have*

$$\|B\tilde{b}\|_1 \le \min\{\|f\|_c : f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}\} \le \gamma\|B\tilde{b}\|_1,$$

for $\gamma = O(\log(\Delta)/\varepsilon_0)$. Moreover, a flow f satisfying $Af = \tilde{b}$ of cost at most $\gamma\|B\tilde{b}\|_1$ can be constructed in time $O(n\varepsilon_0^{-d}\log\Delta)$ given $B\tilde{b}$.

Proof. The matrix B has one row associated with each vertex $u \in V$. For $p \in P$, we let $B_{p,p} = \|p - N_L(p)\|_2$ and $B_{p,u} = 0$ for any $u \ne p$. Every other vertex $u \in V \setminus P$ is a net point. Suppose that u is in N_ℓ; then for any $v \in V$ such that $N_\ell(v) = u$, $B_{u,v} = \frac{\varepsilon_0\Delta_\ell}{4(L+1)}$, and for all other v, $B_{u,v} = 0$. This defines the matrix B. Notice that each vertex $u \in V$ contributes to at most $L+2$ nonzero entries of B: one for each $\ell \in \{0,\dots,L\}$ corresponding to $N_\ell(u)$, and one more when $u \in P$. It is easy to verify directly that B has full column rank.

The definition of B guarantees that

$$\|B\tilde{b}\|_1 = \sum_{p\in P}|\tilde{b}_p|\|p - N_L(p)\|_2 + \frac{1}{L+1}\sum_{\ell=0}^{L}\frac{\varepsilon_0\Delta_\ell}{4}\sum_{\widetilde{C}\in\widetilde{\mathcal{C}}_\ell}\left|\sum_{u\in\widetilde{C}\cap V}\tilde{b}_u\right|. \tag{7}$$

The remainder of the proof is broken down into several claims. For the first claim, let us extend the definition of $\textsc{cost}(G,\cdot)$ to supplies which can be non-zero on net points as well by

$$\textsc{cost}(G,\tilde{b}) = \min\{\|f\|_c : f \in \mathbb{R}^{\vec{E}}, Af = \tilde{b}\}.$$

▷ Claim 14. $\|B\tilde{b}\|_1 \le \textsc{cost}(G,\tilde{b})$.

Proof. In any feasible flow f, the total cost of the flow on edges incident to points $p \in P$ must equal the first term in (7). Then, we just need to show that the remaining terms are a lower bound on the total cost of the flow on the remaining edges. In fact we will show that, for each ℓ,

$$\frac{\varepsilon_0\Delta_\ell}{4}\sum_{\widetilde{C}\in\widetilde{\mathcal{C}}_\ell}\left|\sum_{u\in\widetilde{C}\cap V}\tilde{b}_u\right| \le \textsc{cost}(G,\tilde{b}) - \sum_{p\in P}|\tilde{b}_p|\|p - N_L(p)\|_2. \tag{8}$$

It follows that the average of the term on the left over all ℓ is also a lower bound on the right hand side, which proves the claim. To establish inequality (8) notice that, in any feasible f, flow of value at least $\left|\sum_{u\in\widetilde{C}\cap V}\tilde{b}_u\right|$ must enter or leave each $\widetilde{C} \in \widetilde{\mathcal{C}}_\ell$, and every edge e from a vertex in $\widetilde{C}$ to a vertex outside of $\widetilde{C}$ has cost at least $c(e) \ge \varepsilon_0\Delta_\ell/2$ by the definition of G. Therefore, the total cost of flow leaving or entering $\widetilde{C}$ is at least $\frac{\varepsilon_0\Delta_\ell}{2}\left|\sum_{u\in\widetilde{C}\cap V}\tilde{b}_u\right|$. Adding up these terms over all $\widetilde{C} \in \widetilde{\mathcal{C}}_\ell$ accounts for the cost of any edge of G at most twice, and (8) follows. ◁

▷ Claim 15. $\textsc{cost}(G,\tilde{b}) \le \gamma\|B\tilde{b}\|_1$ for $\gamma = O(L/\varepsilon_0) = O(\log(\Delta)/\varepsilon_0)$.

Proof. We prove the claim by constructing an appropriate feasible flow f. We proceed to construct f by levels, from the bottom up. The construction will be efficient, also proving the claim after "moreover" in the statement of the lemma. On level L, for any $p \in P$, we set

$f_{pu} = \tilde{b}_p$, where $u = N_L(p)$. Then, for levels $\ell = L-1, L-2, \ldots, 1$, we execute the following procedure in every subcell $\widetilde{C} \in \widetilde{\mathcal{C}}_{\ell-1}$. Let us define, for any node u and the current flow f, the surplus

$$\delta(u,f) = \sum_{v:(u,v)\in\vec{E}} f_{uv} - \sum_{v:(v,u)\in\vec{E}} f_{vu} - \tilde{b}_u = \sum_{v:(u,v)\in E} f_{uv} - \tilde{b}_u.$$

I.e. this is how much more flow leaves u than should, as prescribed by $\tilde{b}$. Pick any two u and v in $N_\ell \cap \widetilde{C}$ such that $\delta(u,f) > 0 > \delta(v,f)$, and add $\min\{|\delta(u,f)|, |\delta(v,f)|\}$ units of flow to f_{vu}. Continue until we have that for all $u \in N_\ell \cap \widetilde{C}$ the surpluses $\delta(u,f)$ have the same sign. Since at every step of this procedure we make the surplus of at least one node 0, and we always decrease the absolute value of the surplus at any node, we will stop after at most $|N_\ell \cap \widetilde{C}|$ steps. Finally, for the node $u \in N_{\ell-1}$ which is the center of $\widetilde{C}$, and for all nodes $v \in N_\ell \cap \widetilde{C}$ for which $\delta(v,f) \neq 0$, we set $f_{uv} = \delta(v,f)$. After this final step, every node $u \in N_\ell \cap \widetilde{C}$ has surplus 0.

For $\ell = 0$, we perform essentially the same procedure, but in the entire cell C_0. I.e. we pick any two u and v in $N_\ell \cap C_0$ such that $\delta(u,f) > 0 > \delta(v,f)$, and add $\min\{|\delta(u,f)|, |\delta(v,f)|\}$ units of flow to f_{vu}. Once again, after at most $|N_0|$ steps all surpluses will have the same sign, and, as we show below, will in fact be 0, so f will be feasible.

An easy induction argument shows that, once we have processed levels $L, L-1, \ldots, \ell$, for any $\ell > 0$, the surplus at any node $u \in N_L \cup \ldots \cup N_\ell$ is 0, and the same is true for the surplus at any $p \in P$. To show that the flow f is feasible, it is enough to show that after processing the cell C_0 on level 0, all nodes have surplus 0. Notice that, while processing any subcell $\widetilde{C}$, we do not change the total surplus $\sum_{u\in V\cap\widetilde{C}} \delta(u,f)$. This means that, for any $\ell > 0$, after having processed a subcell $\widetilde{C} \in \widetilde{\mathcal{C}}_{\ell-1}$ with center u, we have

$$\delta(u,f) = -\sum_{v\in\widetilde{C}\cap V} \tilde{b}_v = -\sum_{v:N_{\ell-1}(v)=u} \tilde{b}_v. \tag{9}$$

In particular, we have that before we start processing C_0, $\sum_{u\in N_0} \delta(u,f) = -\sum_{u\in V} \tilde{b}_u = 0$. Since, once again, every step we make during the processing of C_0 preserves the total surplus, and we stop when all surpluses have the same sign, then at the time we are done it must be the case that every node has surplus 0, and, therefore, f is a feasible flow.

It remains to bound the cost of the flow f constructed above. We can charge the cost of the subcells processed on each level while construting f to one of the terms of the right hand side of (7), and this completes the proof of the claim. The details of this calculation appear in the full version of the paper. ◁

The two claims finish the proof of the lemma, together with the observation that the construction of the flow f can be implemented recursively in time $O(n\varepsilon_0^{-d} \log \Delta)$. ◀

Our main result for solving the minimum cost flow problem in G follows. The proof, which at this point is straightforward, is given in the full version of the paper.

▶ **Theorem 16.** *A flow f feasible for the problem (1)–(4), with cost at most $1 + O(\varepsilon)$ factor larger than the optimal cost, can be computed in time*

$$O\left(\frac{\log(\Delta)^2}{\varepsilon_0^2}(|E| + |V|\log(\Delta))\log(|E|)\left(\frac{1}{\varepsilon^2} + \log\Big(\frac{\log(\Delta)}{\varepsilon_0}\Big)\right)\right).$$

5 Generating a transportation map

Theorem 16 guarantees we can obtain an approximately optimal flow in our graph G in nearly linear time. We wish to turn this flow into a transportation map on P. To accomplish this, we will repeatedly transform our flow into other flows, without increasing the cost, and ending at a flow which is also a transportation map. We will no longer keep the flow supported on the edges of E, and will instead allow positive flow between arbitrary pairs of points. We will gradually make sure that there is positive flow only between points in P (as opposed to net points).

We first begin by defining a notion we call uniform flow parity.

▶ **Definition 17.** *Given a flow $f \in \mathbb{R}^{V\times V}$, and a vertex $u \in V$, f is said to satisfy, or have uniform flow parity at u if for all remaining $v \in V$, f_{uv} has the same sign. In other words, either there is flow going out of v or there is flow going into v, but there is no flow passing through v. The flow f is said to satisfy uniform flow parity if f has uniform flow parity on every $v \in V$.*

The next easy lemma shows that a flow that satisfies uniform flow parity is supported on edges between vertices with non-zero divergence.

▶ **Lemma 18.** *Suppose that a flow $f \in \mathbb{R}^{V\times V}$ satisfies uniform flow parity at a vertex $u \in V$, and that $\sum_{v\in V} f_{uv} = 0$. Then $f_{uv} = 0$ for all $v \in V$.*

Proof. Assume, towards contradiction, that f has 0 divergence at u but $f_{uv} \neq 0$ for some $v \in V$. Then there must be some $w \in V$ such that f_{uw} has the opposite sign to f_{uv}, or $\sum_{v\in V} f_{uv} = 0$ would not hold. This contradicts uniform flow parity. ◀

We apply Lemma 18 through the following corollary of it.

▶ **Corollary 19.** *Let f be a flow on $V \times V$. Suppose that for any $u \in V \setminus P$, $\sum_{v\in V} f_{uv} = 0$ and for any $p \in P$, $\sum_{v\in V} f_{pv} = \mu(p)$. Then, if f satisfies uniform flow parity, it is a transportation map for P and μ.*

Proof. By Lemma 18, f is supported on $P \times P$. Moreover, since for any $p \in P_+$ we have $\sum_{q\in P} f_{pq} = \mu(p) > 0$, and f satisfies uniform flow parity, it must be the case that $f_{pq} \geq 0$ for all $q \in P$. Similarly, for any $p \in P_-$ it must be the case that $f_{pq} \leq 0$ for all $q \in P$, or, equivalently, $f_{qp} \geq 0$ for all $q \in P$. It follows that f is in fact supported on $P_+ \times P_-$, and is, therefore, a transportation map. ◀

Let us take a flow f which is an approximately optimal solution to (1)–(4). We can extend f to $V \times V$ by setting $f_{uv} = 0$ for $(u, v) \notin E$. If we can transform f into another flow f' without increasing its cost, so that f' satisfies uniform flow parity, then we can use f' as an approximately optimal transportation map. Towards this goal, we define the Cancellation Procedure Cancel-Vertex(f, u), given in Algorithm 1.

Algorithm 1 The Cancellation Procedure.

```
procedure Cancel-Vertex(f, u)
    while ∃v, w : f_vu > 0 > f_wu do
        x = min{f_vu, f_uw}
        f_vu ← f_vu − x
        f_uw ← f_uw − x
        f_vw ← f_vw + x
    end while
end procedure
```

The essential properties of CANCEL-VERTEX(f, u) are collected in the following lemma, whose proof is omitted from this extended abstract.

▶ **Lemma 20.** *Let f be a flow on $V \times V$, and $u \in V$. The while loop in* CANCEL-VERTEX*(f, u) makes at most $|\{v : f_{uv} \neq 0\}|$ many iterations, and, letting f' be the flow after the procedure is called, we have the following properties:*

1. *All divergences are preserved, i.e. $\sum_{w \in V} f'_{vw} = \sum_{w \in V} f_{vw}$ for all $v \in V$.*
2. *The flow f' satisfies uniform flow parity at u.*
3. *If f satisfies uniform flow parity at some vertex $v \in V$, then so does f'.*
4. *The size of the support of f is at most that of f'.*
5. *The cost of f' with respect to the cost function $c(u, v) = \|u - v\|_2$ is at most the cost of f.*

Suppose that we maintain f in a sparse representation; namely, we keep an adjacency list of the edges on which f is not zero, with the corresponding flow values. Then, after preprocessing the adjacency list of u in linear time to find the edges with positive and negative flow, every iteration of the while loop in the Cancellation Procedure can be executed in constant time, and the total running time, by Lemma 20, is bounded by $O(\deg_f(u))$, where $\deg_f(u) = |\{v : f_{uv} \neq 0\}|$.

Corollary 19 and Lemma 20 imply that if we apply the Cancellation Procedure to the flow f and every vertex of the graph G, then the resulting flow has cost no greater than that of f, is supported on a set of edges of size bounded by $|E|$, and is a transportation map. In the next theorem, which is our main result for constructing a transportation map from a flow f in G, we show that if we apply the procedure first to netpoints in N_L, then to N_{L-1}, etc., then the total running time is nearly linear. The proof, which is deferred to the full version of the paper, simply accounts for the current degree of any node to which we apply the Cancellation Procedure.

▶ **Theorem 21.** *There exists an algorithm running in time $O\left(n\varepsilon^{-d}\log(\Delta) + \varepsilon^{-2d}\log(\Delta)\right)$ that, given as input a flow f which is feasible for the minimum cost flow problem defined in Section 3.3, outputs a transportation map for P and μ of cost no larger than that of f.*

Combining Theorem 6, used with ε_0 set to a sufficiently small multiple of ε/L, and Theorems 16 and 21 gives the first claim of Theorem 1, but with the approximation holding in expectation. At the cost of increasing ε by a factor of 2, Markov's inequality shows that the approximation also holds with probability at least $1/2$. Then the second statement in Theorem 1 follows from the first one, and Lemma 2.

References

1. Pankaj K. Agarwal, Kyle Fox, Debmalya Panigrahi, Kasturi R. Varadarajan, and Allen Xiao. Faster Algorithms for the Geometric Transportation Problem. In *Symposium on Computational Geometry*, volume 77 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
2. Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *STOC*, pages 555–564. ACM, 2014.
3. Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Weed. Approximating the Quadratic Transportation Metric in Near-Linear Time. *CoRR*, abs/1810.10046, 2018.
4. Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *STOC*, pages 574–583. ACM, 2014.
5. Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *J. ACM*, 45(5):753–782, 1998.

6 Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement Interpolation Using Lagrangian Mass Transport. *ACM Trans. Graph.*, 30(6):158:1–158:12, December 2011. doi:10.1145/2070781.2024192.

7 Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388. ACM, 2002.

8 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. Assoc. Comput. Mach.*, 31(3):538–544, 1984. doi:10.1145/828.1884.

9 Panos Giannopoulos and Remco C Veltkamp. A pseudo-metric for weighted point sets. In *European Conference on Computer Vision*, pages 715–730. Springer, 2002.

10 Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

11 Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *SODA*, pages 39–42. SIAM, 2007.

12 Piotr Indyk and Nitin Thaper. Fast Image Retrieval via Embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision*, 2003.

13 Qin Lv, Moses Charikar, and Kai Li. Image Similarity Search with Compact Data Structures. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 208–217, New York, NY, USA, 2004. ACM. doi:10.1145/1031171.1031213.

14 Jeff M. Phillips and Pankaj K. Agarwal. On Bipartite Matching under the RMS Distance. In *CCCG*, 2006.

15 Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

16 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. In *STOC*, pages 385–394. ACM, 2012.

17 R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *SODA*, pages 306–317. SIAM, 2012.

18 Jonah Sherman. Generalized Preconditioning and Undirected Minimum-Cost Flow. In *SODA*, pages 772–780. SIAM, 2017.

19 Cédric Villani. *Topics in optimal transportation*, volume 58 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2003. doi:10.1007/b12016.