

Discrete Optimal Transport by Parallel Network Simplex

Stefano Gualandi ^(a)

Joint works with:

G. Auricchio, A. Codegioni, M. Veneroni, E. Vercesi, G. Toscani ^(a)

F. Bassetti^(a), P.Y. Bouchet, L.M. Rousseau ^(c)

^(a) Università di Pavia, Dipartimento di Matematica, Italy

^(b) Politecnico di Milano, ^(c) École Polytechnique de Montréal

**Institute for Pure & Applied Mathematics
Deep Learning and Combinatorial Optimization**

Feb. 25, 2021

email: stefano.gualandi@unipv.it
twitter: [@famo2spaghi](https://twitter.com/famo2spaghi)
blog: <http://stegua.github.com>

Outline

- 1 Introduction to OT
- 2 Applications to ML
- 3 Parallel Network Simplex
- 4 Results

Section 1

1 Introduction to OT

2 Applications to ML

3 Parallel Network Simplex

4 Results

Optimal Transport: From Monge to ...

Optimal Transport originates from the work of Monge on finding the **minimum cost plan** for transporting a **distribution of *molécules*** from its origins to a given **target distribution** (1781).

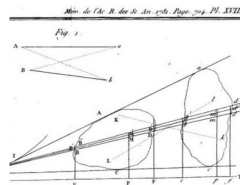


M É M O I R E
S U R L A
T H É O R I E D E S D É B L A I S
E T D E S R E M B L A I S.

Par M. M O N G E.

Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutume de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.

Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'en suit que le déblai & le remblai étant donnés de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits fera la moindre possible, & le prix du transport total fera un minimum.



Optimal Transport: From Monge to ...

Optimal Transport originates from the work of Monge on finding the **minimum cost plan** for transporting a **distribution of *molécules*** from its origins to a given **target distribution** (1781).

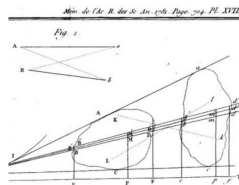


M É M O I R E
S U R L A
T H É O R I E D E S D É B L A I S
E T D E S R E M B L A I S.

Par M. M O N G E.

Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutume de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.

Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'en suit que le déblai & le remblai étant donnés de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits sera la moindre possible, & le prix du transport total fera un *minimum*.



In Combinatorial Optimization terms, the original Monge's problem is an **Assignment Problem**, which can be solved in strongly polynomial time with flow algorithms: $O(nm + n^2 \log(n))$ on general graphs and in $O(n^3)$ on bipartite complete graphs.

Ahuja, Magnanti, Orlin. Network flows: Theory, Algorithms, and Applications, 1988

Optimal Transport: From Monge to Kantorovich

An important contribution to Optimal Transport is due to Kantorovich, *Mathematical methods in the organization and planning of production*, (1939).

Apart from the Duality theory of Linear Programming, Kantorovich is the father of the **Transport Metric**, a.k.a. the *Wasserstein Metric*

Long History of the Monge-Kantorovich Transportation Problem

(Marking the centennial of L.V. Kantorovich's birth!)

A. M. Vershik



Figure 1. L.V. Kantorovich in his youth.

Leonid Vital'evich Kantorovich (1912–1986) was one of the great mathematicians and economists of the twentieth century.

In 2012, the centenary of his birth was marked in St. Petersburg. Short histories were presented describing some of the main parts of his legacy, which continue in importance today: *duality in linear programming*, the so-called “*Monge-Kantorovich transportation problem*,” and the “*Kantorovich metric*.” It is true that 2012 was also the 70th anniversary of the publication of his historic paper on *the transport metric*. The present article offers a somewhat expanded version of my talk on that occasion.

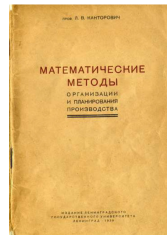
L. V. Kantorovich the Person

We remember Leonid Vital'evich Kantorovich for his massive contributions to foundations of mathematics, computational mathematics, and other areas, and in particular as one of the founders of mathematical economics.

He began as a child prodigy, entering Leningrad University *at the age of 14*. His first paper on descriptive set theory, which

reception was quite different. The ideas of Kantorovich, and Kantorovich himself, were not appreciated in Soviet Russia. For a long time—until the end of the 1990s—his ideas on mathematical economics were considered in official circles as anti-Marxist; consequently it was prohibited and even dangerous to study and develop them. Such a story, carried much further, is well-known in Soviet biology (“*perestroika*”). So, between 1947 and the late 1950s Kantorovich never mentioned his ideas about mathematical economics and linear programming. My colleagues and I heard his lectures on functional analysis (later published as a book with G. P. Akhiezer) but knew nothing of his economically related work. He lectured openly on the subject only after the beginning of Khrushchev’s “*otvet*” (“*thaw*”), the liberalization of 1955–1956.

L. V. Kantorovich (LV hereafter) has left a rich record of the defense of scientific truth. A majority of the Soviet economists of the generation of the 1940s and 1970s were pupils of LV, and many mathematicians (including the author) considered themselves his pupils.



Optimal Transport: From Monge to Kantorovich

An important contribution to Optimal Transport is due to Kantorovich, *Mathematical methods in the organization and planning of production*, (1939). Apart from the Duality theory of Linear Programming, Kantorovich is the father of the **Transport Metric**, a.k.a. the **Wasserstein Metric**

Long History of the Monge-Kantorovich Transportation Problem

(Marking the centennial of L.V. Kantorovich's birth!)

A. M. Vershik



Figure 1. L.V. Kantorovich in his youth.

Leonid Vital'evich Kantorovich (1912–1986) was one of the great mathematicians and economists of the twentieth century.

In 2012, the centenary of his birth was marked in St. Petersburg. Short histories were presented describing some of the main parts of his legacy, which continue in importance today: *duality in linear programming*, the so-called “**Monge-Kantorovich transportation problem**,” and the “**Kantorovich metric**.” It is that 2012 was also the 70th anniversary of the publication of his historic paper on **the transport metric**. The present article offers a somewhat expanded version of my talk on that occasion.

L. V. Kantorovich the Person

We remember Leonid Vital'evich Kantorovich for his massive contributions to foundations of mathematics, computational mathematics, and other areas, and in particular as one of the founders of mathematical economics.

He began as a child prodigy, entering Leningrad University **at the age of 14**. His first paper on descriptive set theory, which

reception was quite different. The ideas of Kantorovich, and Kantorovich himself, were not appreciated in Soviet Russia. For a long time—until the end of the 1990s—his ideas on mathematical economics were considered in official circles as anti-Marxist, consequently it was prohibited and even dangerous to study and develop them. Such a story, carried much further, is well-known in Soviet biology (“*cyberkoshchistina*”). So, between 1947 and the late 1950s Kantorovich never mentioned his ideas about mathematical economics and linear programming. My colleagues and I heard his lectures on functional analysis (later published as a book with G. P. Akilov) but knew nothing of his economically related work. He lectured openly on the subject only after the beginning of Khrushchev's “*otvet*” (“*thaw*”), the liberalization of 1953–1956.

L. V. Kantorovich (LV hereafter) has left a rich record of the defence of scientific truth. A majority of the Soviet economists of the generation of the 1940s and 1950s were pupils of LV, and many mathematicians (including the author) considered themselves his pupils.



The Kantorovich reformulation of Monge's problem gives a **Transportation Problem**: the complexity raises to $O(n \log(n)(m + n \log(n)))$ on general graphs, and to $O(n^3 \log(n))$ for bipartite complete graphs.

Orlin. A faster strongly polynomial minimum cost flow algorithm.
Operations research 41(2) 338-350 (1993)

Optimal Transport: Probabilistic Interpretation

Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$ and

- μ and ν are two probability measures on X and Y
- $c : X \times Y \rightarrow \mathbb{R}_+$ is a cost function

Optimal Transport: Probabilistic Interpretation

Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$ and

- μ and ν are two probability measures on X and Y
- $c : X \times Y \rightarrow \mathbb{R}_+$ is a cost function

Definition 1 (Kantorovich-Rubinstein Functional)

The *Kantorovich-Rubinstein functional (I)* between μ and ν is defined as

$$\mathcal{W}_c(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{X \times Y} c(x, y) \pi(dx dy)$$

where $\Pi(\mu, \nu)$ is the set of all probability measures on $X \times Y$ that have marginals μ and ν .

Discrete Probability measures

We define the Dirac δ -measure centered at $x \in X \subseteq \mathbb{R}^q$ via

$$\delta_x(A) = \begin{cases} 1 & \text{if } x \in A \subseteq X \\ 0 & \text{if } x \notin A \subseteq X. \end{cases}$$

Discrete Probability measures

We define the Dirac δ -measure centered at $x \in X \subseteq \mathbb{R}^q$ via

$$\delta_x(A) = \begin{cases} 1 & \text{if } x \in A \subseteq X \\ 0 & \text{if } x \notin A \subseteq X. \end{cases}$$

Let μ and ν be two discrete probability measures defined on a spaces $X, Y \subseteq \mathbb{R}^q$

$$\mu(A) = \sum_{i=1}^m \mu_i \delta_{x_i}(A) \quad \text{and} \quad \nu(A) = \sum_{j=1}^n \nu_j \delta_{y_j}(A)$$

where $\sum_{i=1}^m \mu_i = 1$ and $\sum_{j=1}^n \nu_j = 1$.

Discrete Probability measures

We define the Dirac δ -measure centered at $x \in X \subseteq \mathbb{R}^q$ via

$$\delta_x(A) = \begin{cases} 1 & \text{if } x \in A \subseteq X \\ 0 & \text{if } x \notin A \subseteq X. \end{cases}$$

Let μ and ν be two discrete probability measures defined on a spaces $X, Y \subseteq \mathbb{R}^q$

$$\mu(A) = \sum_{i=1}^m \mu_i \delta_{x_i}(A) \quad \text{and} \quad \nu(A) = \sum_{j=1}^n \nu_j \delta_{y_j}(A)$$

where $\sum_{i=1}^m \mu_i = 1$ and $\sum_{j=1}^n \nu_j = 1$.

Example: Grey scale images are discrete measures: $x_i = (x_{i1}, x_{i2})$ is the pixel position, and μ_i is the pixel intensity at x_i .

Computational Optimal Transport

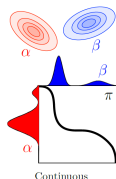
Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$

Computationally, we distinguish three cases:

Computational Optimal Transport

Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$

Computationally, we distinguish three cases:



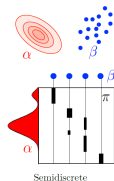
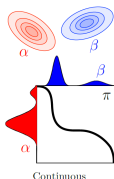
Continuous $\rightarrow$ **Continuous:**

both μ and ν are continuous functions

Computational Optimal Transport

Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$

Computationally, we distinguish three cases:



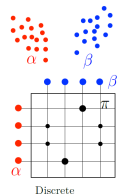
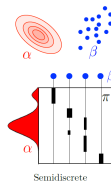
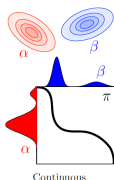
Continuous $\rightarrow$ Continuous:
both μ and ν are continuous functions

Continuous $\rightarrow$ Discrete:
 μ is continuous, but ν is a discrete empirical measure (sum of Dirac)

Computational Optimal Transport

Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$

Computationally, we distinguish three cases:



Continuous $\rightarrow$ Continuous:
both μ and ν are continuous functions

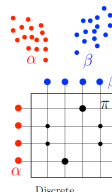
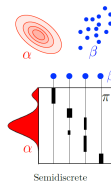
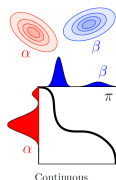
Continuous $\rightarrow$ Discrete:
 μ is continuous, but ν is a discrete empirical measure (sum of Dirac)

Discrete $\rightarrow$ Discrete:
 μ and ν are both discrete empirical measures

Computational Optimal Transport

Problem: Find the **Optimal (Mass) Transport** from μ to ν , where moving a unit mass from x to y costs $c(x, y)$

Computationally, we distinguish three cases:



Continuous $\rightarrow$ Continuous:
both μ and ν are continuous functions

Continuous $\rightarrow$ Discrete:
 μ is continuous, but ν is a discrete empirical measure (sum of Dirac)

Discrete $\rightarrow$ Discrete:
 μ and ν are both discrete empirical measures

Peyré, G. and Cuturi, M., 2019. Computational optimal transport: With applications to data science. Foundations and Trends in Machine Learning, 11(5-6), pp.355-607.

Preprint: <https://arxiv.org/abs/1803.00567>

Discrete Kantorovich-Rubinstein is a Linear Program

Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ be two discrete spaces.

Definition 2 (Kantorovich-Rubinstein Functional (II))

The **Kantorovich-Rubinstein functional** in the discrete setting is the following Linear Programming problem

$$\begin{aligned}
 (OT) \quad \mathcal{W}_c(\mu, \nu) = \min \quad & \sum_{x \in X} \sum_{y \in Y} c(x, y) \pi(x, y) \\
 \text{s.t.} \quad & \sum_{y \in Y} \pi(x, y) = \mu(x) & \forall x \in X \\
 & \sum_{x \in X} \pi(x, y) = \nu(y) & \forall y \in Y \\
 & \pi(x, y) \geq 0.
 \end{aligned}$$

Discrete Kantorovich-Rubinstein is a Linear Program

Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ be two discrete spaces.

Definition 2 (Kantorovich-Rubinstein Functional (II))

The **Kantorovich-Rubinstein functional** in the discrete setting is the following Linear Programming problem

$$\begin{aligned}
 (OT) \quad \mathcal{W}_c(\mu, \nu) = \min \quad & \sum_{x \in X} \sum_{y \in Y} c(x, y) \pi(x, y) \\
 \text{s.t.} \quad & \sum_{y \in Y} \pi(x, y) = \mu(x) & \forall x \in X \\
 & \sum_{x \in X} \pi(x, y) = \nu(y) & \forall y \in Y \\
 & \pi(x, y) \geq 0.
 \end{aligned}$$

Definition 3 (Kantorovich-Wasserstein metric)

When $X = Y$ and $c(x, y) = d^p(x, y)$, where d is a ground distance on X , the **Kantorovich-Wasserstein metric of order $p > 0$** is

$$W_p(\mu, \nu) := \mathcal{W}_{d^p}(\mu, \nu)^{\min(\frac{1}{p}, 1)}.$$

Optimal Transport: From Kantorovich to ...

- Modern Computational Optimal Transport for Machine Learning has focused on **entropic regularized costs**, which add a regularization parameter depending on a parameter ϵ . For $\epsilon = 0$, we recover the standard Optimal Transport, otherwise, we do not have a true metric.

Optimal Transport: From Kantorovich to ...

- Modern Computational Optimal Transport for Machine Learning has focused on **entropic regularized costs**, which add a regularization parameter depending on a parameter ϵ . For $\epsilon = 0$, we recover the standard Optimal Transport, otherwise, we do not have a true metric.
- Using the **Sinkhorn algorithm**, the regularized entropic version can compute the optimal distance in time $O(\frac{n^2}{\epsilon^2})$ [AWR17],

Optimal Transport: From Kantorovich to ...

- Modern Computational Optimal Transport for Machine Learning has focused on **entropic regularized costs**, which add a regularization parameter depending on a parameter ϵ . For $\epsilon = 0$, we recover the standard Optimal Transport, otherwise, we do not have a true metric.
- Using the **Sinkhorn algorithm**, the regularized entropic version can compute the optimal distance in time $O(\frac{n^2}{\epsilon^2})$ [AWR17], but we do not get the optimal transport plan (!)

Optimal Transport: From Kantorovich to ...

- Modern Computational Optimal Transport for Machine Learning has focused on **entropic regularized costs**, which add a regularization parameter depending on a parameter ϵ . For $\epsilon = 0$, we recover the standard Optimal Transport, otherwise, we do not have a true metric.
- Using the **Sinkhorn algorithm**, the regularized entropic version can compute the optimal distance in time $O(\frac{n^2}{\epsilon^2})$ [AWR17], but we do not get the optimal transport plan (!)
- The **Sinkhorn algorithm** is highly parallelizable,

Optimal Transport: From Kantorovich to ...

- Modern Computational Optimal Transport for Machine Learning has focused on **entropic regularized costs**, which add a regularization parameter depending on a parameter ϵ . For $\epsilon = 0$, we recover the standard Optimal Transport, otherwise, we do not have a true metric.
- Using the **Sinkhorn algorithm**, the regularized entropic version can compute the optimal distance in time $O(\frac{n^2}{\epsilon^2})$ [AWR17], but we do not get the optimal transport plan (!)
- The **Sinkhorn algorithm** is highly parallelizable, but suffers from stability numerical issues.

Optimal Transport: From Kantorovich to ... Fourier

- Modern Computational Optimal Transport for Machine Learning has focused on **entropic regularized costs**, which add a regularization parameter depending on a parameter ϵ . For $\epsilon = 0$, we recover the standard Optimal Transport, otherwise, we do not have a true metric.
- Using the **Sinkhorn algorithm**, the regularized entropic version can compute the optimal distance in time $O(\frac{n^2}{\epsilon^2})$ [AWR17], but we do not get the optimal transport plan (!)
- The **Sinkhorn algorithm** is highly parallelizable, but suffers from stability numerical issues.



For imaging problems, and whenever we do not need the optimal plan, we can replace the Wasserstein Metric with the mathematically equivalent **Fourier metric**, which can be computed using the **Fast Fourier Transform** algorithm.

Auricchio et al. The equivalence of Fourier-based and Wasserstein metrics on imaging problems, (2020).

<https://arxiv.org/abs/2005.06530>

Combinatorial version of Optimal Transport

In this talk, we focus on the Combinatorial Optimization version of Optimal Transport, solved via the Network Simplex algorithm, since:

- 1 We want to use a true Metric
- 2 We might need the optimal transport plan
- 3 We are trying to exploit the GPU power for Combinatorial optimization problems (hard challenge!)

Section 2

1 Introduction to OT

2 Applications to ML

3 Parallel Network Simplex

4 Results

Gene Mover Distance

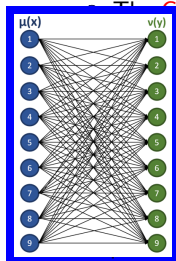
- The **Gene Mover's Distance (GMD)** is a similarity measure between a pair of cells based on their gene expression profiles obtained via single-cell RNA sequencing [BCG⁺21]: gene expressions gives *marginals* μ and ν .

Gene Mover Distance

- The **Gene Mover's Distance (GMD)** is a similarity measure between a pair of cells based on their gene expression profiles obtained via single-cell RNA sequencing [BCG⁺21]: gene expressions gives *marginals* μ and ν .
- Using the embedding `gene2vec`: $\mathcal{G} \rightarrow \mathbb{R}^{200}$ (see [DJD⁺19]), which maps any gene $i \in \mathcal{G}$ to a vector $\mathbf{x}_i \in \mathbb{R}^{200}$, we can define the ground distance:

$$C^{\text{g2v}}(i, j) := \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (1)$$

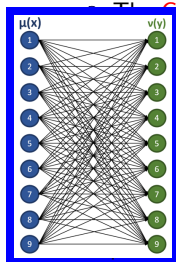
Gene Mover Distance



The **Gene Mover's Distance (GMD)** is a similarity measure between a pair of gene expression profiles based on their gene expression profiles obtained via single-cell sequencing [BCG⁺21]: gene expressions gives *marginals* μ and ν . Using the embedding $\text{gene2vec}: \mathcal{G} \rightarrow \mathbb{R}^{200}$ (see [DJD⁺19]), which maps gene $i \in \mathcal{G}$ to a vector $\mathbf{x}_i \in \mathbb{R}^{200}$, we can define the ground distance:

$$C^{\text{g}2\text{v}}(i, j) := \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (1)$$

Gene Mover Distance



The **Gene Mover's Distance (GMD)** is a similarity measure between a pair of genes based on their gene expression profiles obtained via single-cell sequencing [BCG⁺21]: gene expressions gives *marginals* μ and ν . Using the embedding $\text{gene2vec}: \mathcal{G} \rightarrow \mathbb{R}^{200}$ (see [DJD⁺19]), which maps the gene $i \in \mathcal{G}$ to a vector $\mathbf{x}_i \in \mathbb{R}^{200}$, we can define the ground distance:

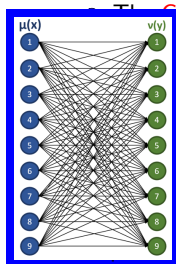
$$C^{\text{g}2\text{v}}(i, j) := \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (1)$$

- Otherwise, as ground distance in the OT, we can use the **Pearson distance**:

$$C^P(i, j) := \sqrt{1 - \rho_{i,j}} \quad (2)$$

(when two genes are positively correlated this is small)

Gene Mover Distance



The **Gene Mover's Distance (GMD)** is a similarity measure between a pair of cells based on their gene expression profiles obtained via single-cell sequencing [BCG⁺21]: gene expressions gives *marginals* μ and ν . Using the embedding `gene2vec`: $\mathcal{G} \rightarrow \mathbb{R}^{200}$ (see [DJD⁺19]), which maps the gene $i \in \mathcal{G}$ to a vector $\mathbf{x}_i \in \mathbb{R}^{200}$, we can define the ground distance:

$$C^{\text{g2v}}(i, j) := \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (1)$$

- Otherwise, as ground distance in the OT, we can use the **Pearson distance**:

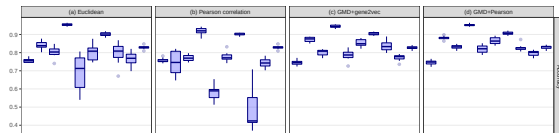
$$C^P(i, j) := \sqrt{1 - \rho_{i,j}} \quad (2)$$

(when two genes are positively correlated this is small)

- Solving OT using the gene expression profiles as “*marginals*” and one of this cost function, yields a distance between a pair of cells.

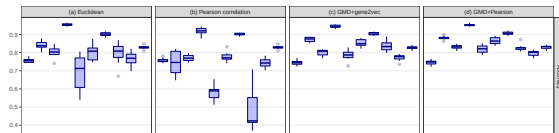
Gene Mover Distance within a k -NN algorithm

- Acute Myeloid Leukemia: Binary Classification of single cells (normal vs. malignant), for 10 patients.

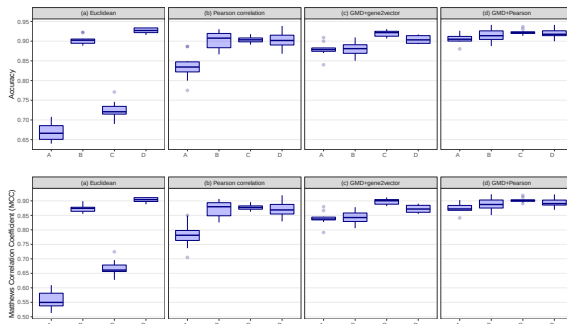


Gene Mover Distance within a k -NN algorithm

- Acute Myeloid Leukemia: Binary Classification of single cells (normal vs. malignant), for 10 patients.



- Human Pancreas: Multi-Class Classification (14 classes), 4 patients



Interpolation between distribution for Barycenters

Lemma 4 (Interpolation between two measures (Chap. 7, in [PC⁺19]))

Given (i) two weights $(\lambda_1, \lambda_2) \in \mathbb{R}_+$ satisfying $\lambda_1 + \lambda_2 = 1$, (ii) two discrete measures μ and ν defined on X , and (iii) an optimal transportation plan π^* minimizing the functional $W_2(\mu, \nu)$, that is, problem (OT) with $c(x_i, y_j) = \|x_i - y_j\|_2^2$, the interpolated average measure ρ between μ and ν is

$$\rho = f(\mu, \nu, \lambda_1, \lambda_2) := \sum_{i=1}^n \sum_{j=1}^n \pi_{ij}^* \delta_{(\lambda_1 x_i + \lambda_2 y_j)}. \quad (3)$$

Measure 1



Measure 2



Fig. 3. Weighted Wasserstein Barycenter of two measures with $(\lambda_1, \lambda_2) = (1 - t, t)$.

Interpolation between distribution for Barycenters

Lemma 4 (Interpolation between two measures (Chap. 7, in [PC⁺19]))

Given (i) two weights $(\lambda_1, \lambda_2) \in \mathbb{R}_+$ satisfying $\lambda_1 + \lambda_2 = 1$, (ii) two discrete measures μ and ν defined on X , and (iii) an optimal transportation plan π^* minimizing the functional $W_2(\mu, \nu)$, that is, problem (OT) with $c(x_i, y_j) = \|x_i - y_j\|_2^2$, the interpolated average measure ρ between μ and ν is

$$\rho = f(\mu, \nu, \lambda_1, \lambda_2) := \sum_{i=1}^n \sum_{j=1}^n \pi_{ij}^* \delta_{(\lambda_1 x_i + \lambda_2 y_j)}. \quad (3)$$

Measure 1



t = 0.3



Measure 2



Fig. 3. Weighted Wasserstein Barycenter of two measures with $(\lambda_1, \lambda_2) = (1 - t, t)$.

Interpolation between distribution for Barycenters

Lemma 4 (Interpolation between two measures (Chap. 7, in [PC⁺19]))

Given (i) two weights $(\lambda_1, \lambda_2) \in \mathbb{R}_+$ satisfying $\lambda_1 + \lambda_2 = 1$, (ii) two discrete measures μ and ν defined on X , and (iii) an optimal transportation plan π^* minimizing the functional $W_2(\mu, \nu)$, that is, problem (OT) with $c(x_i, y_j) = \|x_i - y_j\|_2^2$, the interpolated average measure ρ between μ and ν is

$$\rho = f(\mu, \nu, \lambda_1, \lambda_2) := \sum_{i=1}^n \sum_{j=1}^n \pi_{ij}^* \delta_{(\lambda_1 x_i + \lambda_2 y_j)}. \quad (3)$$

Measure 1



t = 0.6



Measure 2



Fig. 3. Weighted Wasserstein Barycenter of two measures with $(\lambda_1, \lambda_2) = (1 - t, t)$.

Interpolation between distribution for Barycenters

Lemma 4 (Interpolation between two measures (Chap. 7, in [PC⁺19]))

Given (i) two weights $(\lambda_1, \lambda_2) \in \mathbb{R}_+$ satisfying $\lambda_1 + \lambda_2 = 1$, (ii) two discrete measures μ and ν defined on X , and (iii) an optimal transportation plan π^* minimizing the functional $W_2(\mu, \nu)$, that is, problem (OT) with $c(x_i, y_j) = \|x_i - y_j\|_2^2$, the interpolated average measure ρ between μ and ν is

$$\rho = f(\mu, \nu, \lambda_1, \lambda_2) := \sum_{i=1}^n \sum_{j=1}^n \pi_{ij}^* \delta_{(\lambda_1 x_i + \lambda_2 y_j)}. \quad (3)$$



Fig. 3. Weighted Wasserstein Barycenter of two measures with $(\lambda_1, \lambda_2) = (1 - t, t)$.

Primal Heuristic for Wasserstein Barycenters

Definition 5 (Wasserstein Barycenter (Chap. 5 in [San15]))

The problem of computing the Weighted Wasserstein Barycenter of m discrete measures $\mu_1, \dots, \mu_m$, with weights $\lambda_1, \dots, \lambda_m$, such that $\lambda_i \geq 0$ and $\sum_{i=1}^m \lambda_i = 1$, is defined as

$$\rho^*(\mu, \lambda) := \arg \min_{\rho \in \mathcal{S}^n} \sum_{k=1}^m \lambda_k (W_2(\mu_k, \rho))^2. \quad (4)$$

Primal Heuristic for Wasserstein Barycenters

Definition 5 (Wasserstein Barycenter (Chap. 5 in [San15]))

The problem of computing the Weighted Wasserstein Barycenter of m discrete measures $\mu_1, \dots, \mu_m$, with weights $\lambda_1, \dots, \lambda_m$, such that $\lambda_i \geq 0$ and $\sum_{i=1}^m \lambda_i = 1$, is defined as

$$\rho^*(\mu, \lambda) := \arg \min_{\rho \in \mathcal{S}^n} \sum_{k=1}^m \lambda_k (W_2(\mu_k, \rho))^2. \quad (4)$$



Fig. 8. Random sample of images from Translated MNIST.

Primal Heuristic for Wasserstein Barycenters

Definition 5 (Wasserstein Barycenter (Chap. 5 in [San15]))

The problem of computing the Weighted Wasserstein Barycenter of m discrete measures $\mu_1, \dots, \mu_m$, with weights $\lambda_1, \dots, \lambda_m$, such that $\lambda_i \geq 0$ and $\sum_{i=1}^m \lambda_i = 1$, is defined as

$$\rho^*(\mu, \lambda) := \arg \min_{\rho \in S^n} \sum_{k=1}^m \lambda_k (W_2(\mu_k, \rho))^2. \quad (4)$$

Optimal		<u>opt gap</u> : 00.000 % <u>runtime</u> : 859 s
Euclidean		<u>opt gap</u> : 32.243 % <u>runtime</u> : 000 s
Convo		<u>opt gap</u> : 01.121 % <u>runtime</u> : 050 s
Iterative		<u>opt gap</u> : 00.487 % <u>runtime</u> : 009 s
Farthest		<u>opt gap</u> : 18.887 % <u>runtime</u> : 007 s
PairRnd		<u>opt gap</u> : 00.298 % <u>runtime</u> : 007 s
PairFar		<u>opt gap</u> : 02.227 % <u>runtime</u> : 007 s

Key component

In both applications, the Network Simplex is the key component to get:
the **optimal value** and the **optimal transportation plan**.

Section 3

- 1 Introduction to OT
- 2 Applications to ML
- 3 Parallel Network Simplex**
- 4 Results

Parallel Network Simplex

(a) LP Simplex Algorithm

- 1 Generate Initial BFS
- 2 Choose Entering Variable
- 3 Determine Leaving Variable
- 4 Move to New Basic Solution
- 5 Update dual variables

(b) Parallel Network Simplex

- 1 Generate Initial Basis Tree
- 2 Choose Entering Arc (in parallel)
- 3 Determine Leaving Arc (2 threads)
- 4 Move to New Basic Tree
- 5 Update dual variables (parallel?)

Parallel Network Simplex

(a) LP Simplex Algorithm

- ① Generate Initial BFS
- ② Choose Entering Variable
- ③ Determine Leaving Variable
- ④ Move to New Basic Solution
- ⑤ Update dual variables

(b) Parallel Network Simplex

- ① Generate Initial Basis Tree
- ② Choose Entering Arc (in parallel)
- ③ Determine Leaving Arc (2 threads)
- ④ Move to New Basic Tree
- ⑤ Update dual variables (parallel?)

For a review of parallel implementation: *Towards a practical parallelisation of the simplex method*, by J.A.J Hall [Hal10].

For a parallel Network Simplex algorithm: *Parallel simplex for large pure network problems: Computational testing and sources of speedup* [BH94].

To avoid cycling: *Strong Feasible Basis* [Cun76]

Parallel Network Simplex

(a) LP Simplex Algorithm

- 1 Generate Initial BFS
- 2 Choose Entering Variable
- 3 Determine Leaving Variable
- 4 Move to New Basic Solution
- 5 Update dual variables

(b) Parallel Network Simplex

- 1 Generate Initial Basis Tree
- 2 Choose Entering Arc (in parallel)
- 3 Determine Leaving Arc (2 threads)
- 4 Move to New Basic Tree
- 5 Update dual variables (parallel?)

For a review of parallel implementation: *Towards a practical parallelisation of the simplex method*, by J.A.J Hall [Hal10].

For a parallel Network Simplex algorithm: *Parallel simplex for large pure network problems: Computational testing and sources of speedup* [BH94].

To avoid cycling: *Strong Feasible Basis* [Cun76]

We are making a major effort to implement the Network Simplex exploiting a **modern GPU**, and
Ongoing Work: to replace the **Extended AT** data structure with the **Extended API**, to parallelize the dual updates [FL97]

Column (or cut) generation perspective

Considering a subset of the arc variables $\bar{E} \subset E$:

(a) Restricted Master Problem

$$\min \sum_{\{i,j\} \in \bar{E}} c_{ij} \pi_{ij} \quad (5)$$

$$\text{s.t.} \quad \sum_{\{i,j\} \in \bar{E}} \pi_{ij} \geq \mu_i, \forall i \in I \quad (6)$$

$$\sum_{\{i,j\} \in \bar{E}} \pi_{ij} \leq \nu_j, \forall j \in J \quad (7)$$

$$\pi_{ij} \geq 0, \forall \{i,j\} \in \bar{E}. \quad (8)$$

(b) Dual Restricted Master Problem

$$\max \sum_{i \in I} \mu_i u_i - \sum_{j \in J} \nu_j v_j \quad (9)$$

$$\text{s.t.} \quad u_i - v_j \leq c_{ij}, \forall \{i,j\} \in \bar{E} \quad (10)$$

$$u_i \geq 0, \forall i \in I \quad (11)$$

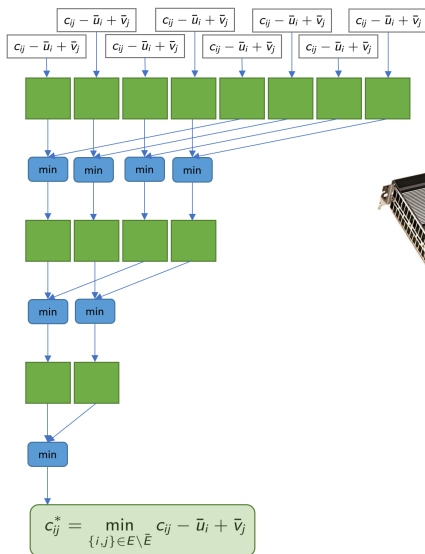
$$v_j \geq 0, \forall j \in J. \quad (12)$$

The pricing (separation) problem is:

$$(P_1) \quad c_{ij}^* = \min_{\{i,j\} \in E \setminus \bar{E}} c_{ij} - \bar{u}_i + \bar{v}_j. \quad (13)$$

Separation of constraint (10) is “embarrassingly simple”,
hence, well suited for **GPU computation**

Pricing: Massively Parallel **Min** Reduction on GPU



Parallel GPU threads
(up to 1024 concurrent)



Sifting and Network Simplex

Algorithm 1: Parallel Network Simplex with Sifting

- 1 Solve restricted master problem (5) with given $\bar{E}$;
 - 2 **while** $c_{ij}^* < 0$: *Negative Reduced Cost variables exists*: **do**
 - 3 Copy dual prices $\bar{u}_i, \bar{v}_j$ to GPU (8 Gb/s);
 - 4 Massively Parallel pricing on GPU to get $\tilde{E}$ (CUDA kernel);
 - 5 Copy $\tilde{E}$ to host computer (8 Gb/s);
 - 6 Augment variables: $\bar{E} \leftarrow \bar{E} \cup \tilde{E}$;
 - 7 Purge variables with large positive reduced costs;
 - 8 Re-optimize restricted master (5) (warm start, CPU thread parallelism);
-

Sifting and Network Simplex

Algorithm 2: Parallel Network Simplex with Sifting

- 1 Solve restricted master problem (5) with given $\bar{E}$;
 - 2 **while** $c_{ij}^* < 0$: *Negative Reduced Cost variables exists*: **do**
 - 3 Copy dual prices $\bar{u}_i, \bar{v}_j$ to GPU (8 Gb/s);
 - 4 Massively Parallel pricing on GPU to get $\tilde{E}$ (CUDA kernel);
 - 5 Copy $\tilde{E}$ to host computer (8 Gb/s);
 - 6 Augment variables: $\bar{E} \leftarrow \bar{E} \cup \tilde{E}$;
 - 7 Purge variables with large positive reduced costs;
 - 8 Re-optimize restricted master (5) (warm start, CPU thread parallelism);
-

We solve a sequence of very sparse network problems.
We keep in memory only $O(|I| + |J|)$ arc variables.

Sifting and Network Simplex

Algorithm 3: Parallel Network Simplex with Sifting

- 1 Solve restricted master problem (5) with given $\bar{E}$;
 - 2 **while** $c_{ij}^* < 0$: *Negative Reduced Cost variables exists*: **do**
 - 3 Copy dual prices $\bar{u}_i, \bar{v}_j$ to GPU (8 Gb/s);
 - 4 Massively Parallel pricing on GPU to get $\tilde{E}$ (CUDA kernel);
 - 5 Copy $\bar{E}$ to host computer (8 Gb/s);
 - 6 Augment variables: $\bar{E} \leftarrow \bar{E} \cup \tilde{E}$;
 - 7 Purge variables with large positive reduced costs;
 - 8 Re-optimize restricted master (5) (warm start, CPU thread parallelism);
-

We solve a sequence of very sparse network problems.
We keep in memory only $O(|I| + |J|)$ arc variables.

Bixby, Gregory, Lustig, Marsten, and Shanno. *Very large-scale linear programming: A case study in combining interior point and simplex methods*. Operations Research, 40(5) 885-897 (1992).

Section 4

- 1 Introduction to OT
- 2 Applications to ML
- 3 Parallel Network Simplex
- 4 Results**

Implementation details: Code and Dataset

- All the algorithms coded in standard ANSI C++11
- GPU programming using the NVIDIA CUDA toolkit v11.2: development of CUDA kernels.
- Multicore CPU parallel algorithms use OpenMP 4.5
- As benchmarks, with locations $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^2$, we use:
 - ① **DOTmark** grey scale images [SSG17], a standard benchmark for computing Wasserstein distances.
- All results refer to a Dell workstation with an Intel Xeon CPU, 10 physical cores at 3.3GHz, 32GB of RAM, equipped with an NVIDIA Quadro P6000 GPU.

Tonight Results with Parallel Network Simplex

Size	Meth.	Num. Vars	Total	Average runtime in seconds			
				(speedup)	Master	Pricing	(speedup)
32×32	Full	1048572	0.29	(×3.8)			
	CPU	3044	0.33	(×4.4)	0.02	0.31	(×4.9)
	GPU	3036	0.08		0.01	0.06	
	Sinkhorn		0.67				
64×64	Full	16777212	13.7	(×11.2)			
	CPU	12240	4.4	(×3.6)	0.8	3.6	(×6.6)
	GPU	12266	1.2		0.7	0.5	
	Sinkhorn		3.5				
128×128	Full (*)	268435452	1542.3	(×56)			
	CPU	49361	64.2	(×2.3)	23.2	41.0	(×6.4)
	GPU	49277	27.6		21.1	6.4	
	Sinkhorn		21.7				

- **Full**: Complete bipartite model with CPU Parallel Simplex
- **CPU**: Column generation with CPU Parallel Simplex
- **GPU**: Column generation with GPU Parallel Simplex (CUDA kernel)
- **Sinkhorn**: Stabilized Sinkhorn algorithm, CPU Parallel (Schmitzer, 2016)

NOTE: Speedup refers only to GPU vs. Full and CPU. Averages over 90 instances per row.

Conclusions

- The “old fashioned” Network Simplex can still play an important role in modern Machine Learning applications

Conclusions

- The “old fashioned” Network Simplex can still play an important role in modern Machine Learning applications
- Column Generation + Sifting is crucial to scale to very large instances

Conclusions

- The “old fashioned” Network Simplex can still play an important role in modern Machine Learning applications
- Column Generation + Sifting is crucial to scale to very large instances
- To exploit a GPU is challenging due to the data transfer bottleneck, but it can offer at least a $\times 2.3$ overall speedup

Conclusions

- The “old fashioned” Network Simplex can still play an important role in modern Machine Learning applications
- Column Generation + Sifting is crucial to scale to very large instances
- To exploit a GPU is challenging due to the data transfer bottleneck, but it can offer at least a $\times 2.3$ overall speedup

A version of the CPU Parallel Simplex algorithm, customized for Spatial Statistics applications [RC20, RLWS20], a pre-release is available at:

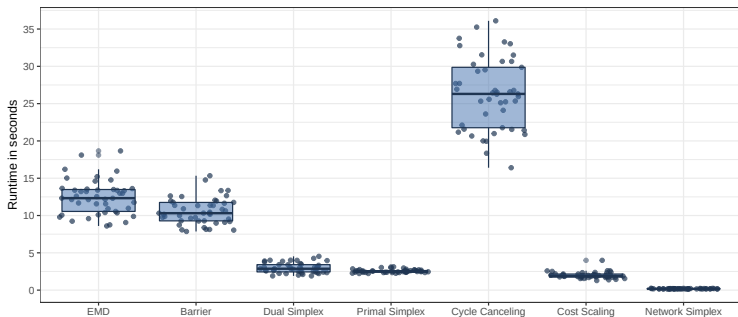


<https://github.com/eurostat/Spatial-KWD>

Thanks

Questions?

Network Simplex vs. Other Exact Methods

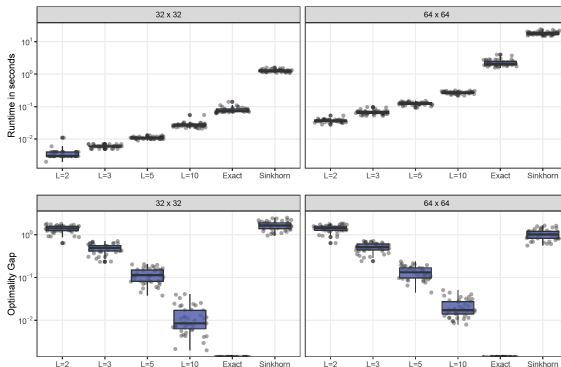


Barrier, Primal, and Dual Simplex refer to Gurobi v8.0.

Cycle Canceling, Cost Scaling, and Network Simplex to COIN-OR Lemon.

Bassetti, G., Veneroni. On the Computation of Kantorovich–Wasserstein Distances Between Two-Dimensional Histograms by Uncapacitated Minimum Cost Flows. *SIAM Journal on Optimization* 30 (3), 2441–2469.
Preprint: <https://arxiv.org/abs/1804.00445>

Network Simplex vs. Stabilized Sinkhorn



Comparison with Sinkhorn as implemented by:

Schmitzer, *Stabilized sparse scaling algorithms for entropy regularized transport problems*, (2016). Preprint: <https://arxiv.org/abs/arXiv:1610.06519>.

Bassetti, G., Veneroni. On the Computation of Kantorovich–Wasserstein Distances Between Two-Dimensional Histograms by Uncapacitated Minimum Cost Flows. *SIAM Journal on Optimization* 30 (3), 2441–2469.
Preprint: <https://arxiv.org/abs/1804.00445>



Jason Altschuler, Jonathan Weed, and Philippe Rigollet.

Near-linear time approximation algorithms for optimal transport via sinkhorn iteration.

arXiv preprint arXiv:1705.09634, 2017.



Riccardo Bellazzi, Andrea Codegioni, Stefano Gualandi, Giovanna Nicora, and Eleonora Vercesi.

The gene mover's distance: Single-cell similarity via optimal transport.

arXiv preprint arXiv:2102.01218, 2021.



Richard Barr, Fred Glover, and Darwin Klingman.

Enhancements of spanning tree labelling procedures for network optimization.

INFOR: Information Systems and Operational Research, 17(1):16–34, 1979.



Richard S Barr and Betty L Hickman.

Parallel simplex for large pure network problems: Computational testing and sources of speedup.

Operations Research, 42(1):65–80, 1994.



William H Cunningham.

A network simplex method.

Mathematical Programming, 11(1):105–116, 1976.



J. Du, P. Jia, Y. Dai, C. Tao, Z. Zhao, and D. Zhi.

Gene2vec: distributed representation of genes based on co-expression.

BMC genomics, 20(1):82, 2019.



Michael Florian and Denis Lebeuf.

An efficient implementation of the network simplex method.

In *Network Optimization*, pages 265–291. Springer, 1997.



JAJ Hall.

Towards a practical parallelisation of the simplex method.

Computational Management Science, 7(2):139–170, 2010.



Péter Kovács.

Minimum-cost flow algorithms: an experimental evaluation.

Optim. Methods Softw., 30(1):94–127, 2015.



Gabriel Peyré, Marco Cuturi, et al.

Computational optimal transport.

Foundations and Trends® in Machine Learning, 11(5-6):355–607, 2019.



Fabio Ricciato and Angelo Coluccia.

On the estimation of spatial density from mobile network operator data.

arXiv preprint arXiv:2009.05410, 2020.



Fabio Ricciato, Giampaolo Lanzieri, Albrecht Wirthmann, and Gerdy Seynaeve.

Towards a methodological framework for estimating present population density from mobile network operator data.

Pervasive and Mobile Computing, 68:101263, 2020.



Filippo Santambrogio.

Optimal Transport for applied mathematicians.

Birkäuser, NY, pages 99–102, 2015.



Jörn Schrieber, Dominic Schuhmacher, and Carsten Gottschlich.

Dotmark—a benchmark for discrete optimal transport.

IEEE Access, 5:271–282, 2017.