# Computing Wasserstein-$p$ Distance Between Images with Linear Cost

Yidong Chen[1,2] Chen Li[1] Zhonghua Lu[1,*]
[1]Computer Network Information Center, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
chenyidong@cnic.cn, lichen@sccas.cn, zhlu@cnic.cn

## Abstract

*When the images are formulated as discrete measures, computing Wasserstein-p distance between them is challenging due to the complexity of solving the corresponding Kantorovich's problem. In this paper, we propose a novel algorithm to compute the Wasserstein-p distance between discrete measures by restricting the optimal transport (OT) problem on a subset. First, we define the restricted OT problem and prove the solution of the restricted problem converges to Kantorovich's OT solution. Second, we propose the SparseSinkhorn algorithm for the restricted problem and provide a multi-scale algorithm to estimate the subset. Finally, we implement the proposed algorithm on CUDA and illustrate the linear computational cost in terms of time and memory requirements. We compute Wasserstein-p distance, estimate the transport mapping, and transfer color between color images with size ranges from* $64 \times 64$ *to* $1920 \times 1200$. *(Our code is available at https://github.com/ucascnic/CudaOT)*

## 1. Introduction

The long history of optimal transport (OT) problems can be traced back to the pioneering work of Monge (1781), Toloston (1930), Kantorovich (1942) and Brenier (1991) [6]. To this day, this theory has provided a fertile ground for research with deep connections to convexity [18], partial differential equations [8], economic problems [13], machine learning [3] and computer vision [28].

The theory of OT defines a distance on probability measures, called the Wasserstein distance. The Wasserstein distance can not be calculated analytically in most cases and it is computationally more costly than $L^p$-distance. The computational effort to calculate Wasserstein distance and solve optimal coupling remains the bottleneck in many applications as the problem size grows. Consequently, efficient algorithms for computing Wasserstein distance are needed.

The straightforward way to solve the discrete OT problems is to use linear programming based algorithms such as the Hungarian method [26], the auction algorithm [4] and the network simplex [31], which are typically numerically robust. The limitation of them is the algorithmic complexity, especially the memory requirements for solving large problems. The approximation based methods, however, add an entropy entry to the original formulation and make the objective be a strongly convex function, which helps develop iteration based algorithms such as the Sinkhorn algorithm [41], the Greenkhorn algorithm [2] and Screening Sinkhorn [1]. The Sinkhorn algorithm provides an efficient and scalable approximation to the original OT problem and it is easy to be parallelized. However, for large problems, it is costly to store the dense matrix in memory and numerical issues appear as some of the elements of the kernel become too negligible to be stored as positive numbers and become instead null. The Greenkhorn, has a better performance of convergence compared to the Sinkhorn and the computational cost is much lower since it only requires updating a specific row or column in each iteration, but it remains a numerical issue for large discrete OT problems. An effective way to remove the numerical issues of the Sinkhorn algorithm is conducting the iteration on a log domain [25, 40], which guarantees a bounded kernel during every iteration. The drawback is that it requires many additional computations of exponential operation in each iteration. A stabilized sparse algorithm, which brings the idea of generating a stabilized kernel during every iteration, was studied in [37] to not only remove numerical issues of the Sinkhorn algorithm but also solve the discrete OT problem between large point clouds. A related scaling algorithm was studied in [10] for computing the unbalanced OT problems. The limitation of the sparse algorithm is that it requires considerable time to generate the sparse kernel during iterations. The multi-scale scheme for measures approximation was studied in [30] for solving the semi-discrete OT problems and showed a significant improvement in terms of solution time and memory requirements for solving the discrete OT problems such as the Shielding method [36], which is efficient for computing Wasserstein-2 distance but converges very slow for computing Wasserstein-1 distance. There are also gradient-based algorithms [14, 24] for solving large OT problems with less complexity than the Sinkhorn algorithm. These algorithms work sufficiently for the semi-discrete OT problems [7, 27] but remain numerical issues for large dense discrete OT problems since the gradient will become nearly zeros during iterations.

This paper focuses on developing the Sinkhorn based algorithm on a restricted subset to reduce the time and memory requirements during iterations. Our work complements both lines of work, theoretical and practical. By providing the proof to guarantee the convergence of the OT problem restricted on a subspace for general nonnegative cost functions, we develop a multi-scale sparse Sinkhorn (M3S) algorithm and implement it on CUDA for solving large dense discrete OT problems with linear cost in terms of solution time and memory requirements. In practice, we can compute solutions with problem sizes going up to two million variables on a modern GPU, much faster than other methods and without loss of accuracy.

The main contributions of this paper are as follows. Firstly, we introduce the restricted regularized OT problem and prove its convergence to the original unregularized OT problem. Secondly, based on the guarantee of the convergence, we propose the M3S algorithm for computing large dense discrete optimal transport problem. Thirdly, we provide a CUDA implementation of the proposed algorithm for computing Wasserstein-$p$ distance and transferring color between images up to 1920 × 1200 in size.

The remainder of the paper is organized as follows. In Section 2, we review the unregularized and the regularized discrete OT. Section 3 and Section 4 contain our main contribution, that is, the convergence of the restricted regularized discrete OT and the M3S algorithm. In Section 5, we present numerical results for the proposed algorithm. Section 6 concludes the paper.

# 2. Discrete Optimal Transport and Entropy Regularization

**Notations**. We denote non-negative real numbers by $\mathbb{R}_+$, the set of integers $\{1,\dots,n\}$ by $[n]$. The standard Euclidean inner product is denoted by $\langle\cdot,\cdot\rangle$. The probability simplex is denoted by $\Sigma_n := \{a_i \in \mathbb{R}_+ : \sum_{i=1}^n a_i = 1\}$. For a discrete, finite space $Z$ (typically $X$, $Y$ and $X \times Y$) we write $|Z|$ for its cardinality. For a matrix $A \in \mathbb{R}^{n\times m}$ its support is defined by $\mathrm{spt}A = \{(i,j)|A_{ij} > 0\}$. The coordinates $r_i(A)$ and $c_j(A)$ denote the $i$th row sum and $j$th column sum of $A$. For $a, b \in \mathbb{R}^n$ the operators $\odot$ and $\oslash$ denote pointwise multiplication and division, e.g. $a \odot b \in \mathbb{R}^n, (a \odot b)_i := a_i \cdot b_i$ for $i \in [n]$. The functions exp and log are extended to $\mathbb{R}^n$ by pointwise application to all components: $\exp(a)_i := \exp(a_i)$. We use $\mathbf{1}_n$ and $\mathbf{0}_n$ to denote the all-ones and all-zeroes vectors in $\mathbb{R}^n$. $\mathrm{diag}(a)$ represents the diagonal matrix with the vector $a$ on the diagonal.

## 2.1. Discrete Optimal Transport

Given discrete probability measures $\mu$ and $\nu$ such that

$$\mu = \sum_{i=1}^n \mu_i \delta_{x_i}, \quad \nu = \sum_{j=1}^m \nu_j \delta_{y_i}, \tag{1}$$

$$\sum_{i=1}^n \mu_i = \sum_{j=1}^m \nu_j = 1, \quad \mu_i, \nu_j \geq 0, \tag{2}$$

the set $\Pi(\mu,\nu) := \{\pi \in \mathbb{R}_+^{n\times m} | \pi \mathbf{1}_m = \mu, \pi^T \mathbf{1}_n = \nu\}$ is called the couplings between $\mu$ and $\nu$. Let $c \in \mathbb{R}_+^{n\times m}$ be a cost matrix, such that the cost of taking one unit of mass from $x_i \in X$ to $y_j \in Y$ is given by $c_{ij}$. The discrete optimal transport problem between $\mu$ and $\nu$ is given by

$$L_c(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} E(\pi) := \langle c, \pi\rangle. \tag{3}$$

We focus on the general case with $c$ being a non-negative dense matrix. In this case, we call (3) the dense or full problem. The problem is linear programming with the best theoretical complexity $O(n^{\frac{5}{2}})$ when $m = n$ [29].

## 2.2. Entropy Regularization

**Definition 1 (Entropy Function)** *For discrete measure $\pi \in \mathbb{R}_+^{n\times m}$ , we define the entropy $H(\pi)$ of $\pi$ by*

$$H(\pi) := -\sum_{i,j} (\pi_{ij} \log \pi_{ij} - \pi_{ij}),$$

**Remark 1** *If $\pi$ is not strictly positive for some $i$, $j$ such that $\pi_{ij} = 0$, the value $\pi_{ij} \log \pi_{ij}$ is defined as 0. Since $\lim_{z\to 0} z \log z = 0$, the entropy function $H(\pi)$ is a continuous function defined on $\mathbb{R}_+^{n\times m}$.*

The entropy regularization of the discrete optimal OT problem is given by

$$\min_{\pi\in\Pi(\mu,\nu)} E_\varepsilon(\pi) := \langle c, \pi\rangle - \varepsilon H(\pi), \tag{4}$$

where $\varepsilon > 0$ is a regularization parameter. By applying the Fenchel-Rockafellar duality theory ( [34] Section 6), the corresponding dual problem is given by

$$\max_{(\alpha,\beta)\in(\mathbb{R}^n,\mathbb{R}^m)} J_\varepsilon(\alpha,\beta) := \langle \alpha, \mu\rangle + \langle \beta, \nu\rangle - \varepsilon\langle e^{\alpha/\varepsilon}, K e^{\beta/\varepsilon}\rangle,$$

where $K \in \mathbb{R}_+^{n\times m}$ is the Gibbs kernel with entries $K_{ij} := \exp(-c_{ij}/\varepsilon)$. The optimal coupling $\pi_\varepsilon^*$ of the problem (4) can be described by

$$\pi_\varepsilon^* = \mathrm{diag}(\exp(\alpha/\varepsilon)) K \mathrm{diag}(\exp(\beta/\varepsilon)). \tag{5}$$

The combination of (5) and the marginal conditions induces a natural and famous Sinkhorn algorithm by updating variables $u^{(l)}$ and $v^{(l)}$ with

$$u^{(l+1)} = \mu \oslash (Kv^{(l)}), \quad v^{(l+1)} = \nu \oslash (K^T u^{(l+1)}),$$

where the initial value $v^{(0)}$ is given by $v^{(0)} = \mathbf{1}_m$.

As $\varepsilon \to 0$, the solution of the regularized OT problem (4) converges to the unregularized problem (3) [11], and the primal-dual gap between $E_\varepsilon(\pi)$ and $J_\varepsilon(\alpha, \beta)$ has been well estimated in [37]. However, in practice, the memory required to store the kernel $K$ becomes too large to be ignored. For examples, if we compute the Wasserstein-$p$ distance between two $256 \times 256$ gray images by Sinkhorn algorithm, the size of kernel is $2^{16} \times 2^{16}$. It requires 32 GB memory for the kernel to be stored in double precision, which is very demanding and low efficiency.

## 3. Optimal Transport Restricted on Subset

We propose the regularized optimal transport restricted on a subset and prove that the solution of the restricted optimal transport converges to the unregularized problem (3) as $\varepsilon \to 0$ under certain conditions, which inspires us to develop our algorithm to compute the Wasserstein distance and the optimal coupling fast and precisely by using less memory.

**Definition 2 (Restricted OT Problem)** *Let $c \in \mathbb{R}_+^{n\times m}$ be a cost matrix and $\mathcal{N}_0$ be the basis set, the indices of the entries of c, i.e $\mathcal{N}_0 = [n] \times [m]$. If $\mathcal{N} \subset \mathcal{N}_0$, the unregularized optimal transport problem restricted on $\mathcal{N}$ is given by*

$$\min_{\pi \in \Pi^{\mathcal{N}}(\mu,\nu)} E_\varepsilon^{\mathcal{N}}(\pi) := \sum_{(i,j)\in\mathcal{N}} \pi_{ij} c_{ij}, \tag{6}$$

*where the restricted coupling $\Pi^{\mathcal{N}}(\mu, \nu)$ is given by*

$$\begin{aligned}\Pi^{\mathcal{N}}(\mu,\nu) := \{\pi \in \mathbb{R}_+^{n\times m} | \sum_{\{j|(i,j)\in\mathcal{N}\}} \pi_{ij} = \mu_i, i \in [n], \\ \sum_{\{i|(i,j)\in\mathcal{N}\}} \pi_{ij} = \nu_j, j \in [m], \\ \pi_{ij} = 0, (i,j) \in \mathcal{N}_0 \setminus \mathcal{N}\}.\end{aligned}$$

*For $\varepsilon > 0$, the regularized OT problem restricted on $\mathcal{N}$ is defined as*

$$\min_{\pi \in \Pi^{\mathcal{N}}(\mu,\nu)} E_\varepsilon^{\mathcal{N}}(\pi) := \sum_{(i,j)\in\mathcal{N}} \left( \pi_{ij} c_{ij} - \varepsilon(\pi_{ij} \log \tfrac{1}{\pi_{ij}} + \pi_{ij}) \right). \tag{7}$$

The problem (6) and (7) are feasible if the set $\Pi^{\mathcal{N}}(\mu, \nu)$ is nonempty. Clearly, if we could estimate a suitable subset such that the set $\mathcal{N}$ has a smaller cardinality $|\mathcal{N}|$ and guarantee the feasibility of the problem, then the problem can be solved faster and with less memory. The key point is, under what conditions the problem is feasible and whether the restricted regularized problem converges to the unregularized problem. We state the following theorem and prove it.

**Theorem 1** *Let $\pi^*$ be the optimal solution with maximal entropy within the set of all optimal solutions of the unregularized problem as in (3), namely*

$$\pi^* = \arg\min_\pi \{-H(\pi) : \pi \in \Pi(\mu,\nu), \langle c, \pi\rangle = L_c(\mu,\nu)\}$$

*If $\mathrm{spt}\pi^* \subset \mathcal{N}$, then*

*$(i)$ $\min_{\pi\in\Pi(\mu,\nu)} E_\varepsilon^{\mathcal{N}}(\pi)$ is feasible.*

*$(ii)$ The unique solution $\hat{\pi}^*(\varepsilon)$ of (7) converge to $\pi^*$, namely*

$$\lim_{\varepsilon\to 0} \hat{\pi}^*(\varepsilon) = \pi^*. \tag{8}$$

*The limitation is taken with every entry of $\hat{\pi}^*(\varepsilon)$.*

*$(iii)$ The $\min_{\pi\in\Pi(\mu,\nu)} E_\varepsilon^{\mathcal{N}}(\pi)$ converge to the unregularized Kantorovich problem as in Definition (3), namely*

$$\lim_{\varepsilon\to 0} \min_{\pi\in\Pi(\mu,\nu)} E_\varepsilon^{\mathcal{N}}(\pi) = L_c(\mu,\nu). \tag{9}$$

If we restrict the original regularized problem (4) on a sparse subset $\mathcal{N}$ such that $\mathrm{spt}\pi^* \subset \mathcal{N}$, it would be sufficient to solve the problem and get an approximate solution. For such a subset $\mathcal{N}$ and a matrix $A \in R^{n\times m}$, we denote its rows and columns' sum on the subset by $r^{\mathcal{N}}(A) := \sum_{\{j|(\cdot,j)\in\mathcal{N}\}} A(\cdot, j) \in \mathbb{R}^n$ and $c^{\mathcal{N}}(A) := \sum_{\{i|(i,\cdot)\in\mathcal{N}\}} A(i, \cdot) \in \mathbb{R}^m$, respectively. The Sinkhorn iteration restricted on subset $\mathcal{N}$ is given in Algorithm 1.

**Algorithm 1** RestrictedSinkhorn $(\mathcal{N}, K, \mu, \nu, \varepsilon)$

```
k ← 0
v^(0) ← 1_m
repeat
    if k is even then
        u_i^(k+1) ← μ_i / r_i^N(K v^(k)) for all i ∈ [n]
        v^(k+1) ← v^(k)
    else
        v_j^(k+1) ← ν_j / c_j^N(K^T u^(k+1)) for all j ∈ [m]
        u^(k+1) ← u^(k)
    end if
    k ← k + 1
    π_ij^(k) ← u_i^(k) exp(−c_ij/ε) v_j^(k) for (i, j) ∈ N
until ||r^N(π^(k)) − μ||_2 + ||c^N(π^(k)) − ν||_2 < δ^2
α^(k) ← ε log u^(k), β^(k) ← ε log v^(k)
Output π^(k), α^(k), β^(k)
```

The only difference between the Sinkhorn algorithm and the RestrictedSinkhorn algorithm is that we conduct the iterations on a subset rather than the whole domain [23]. The RestrictedSinkhorn algorithm outputs two extra variables, $u^{(k)}$ and $v^{(k)}$, which play a key role in finding a new subset $\mathcal{N}$ such that $\text{spt}\pi^* \subset \mathcal{N}$. We'll discuss them later. The following theorem guarantees the convergence of the RestrictedSinkhorn algorithm. (Proof. See Appendix).

**Theorem 2** *Let $\mathcal{N} \subset \mathcal{N}_0$ be a subspace such that $\text{spt}\pi^* \subset \mathcal{N}$. Algorithm 1 outputs a matrix $\pi$ satisfying $||r^{\mathcal{N}}(\pi) - \mu||_2 + ||c^{\mathcal{N}}(\pi) - \nu||_2 < \delta^2$ after $O(\rho h(\delta)^{-2} \log(h/s))$ iterations, where $h = \sum_{(i,j)\in\mathcal{N}} \pi_{ij}$, ,$\rho = \max\{||r^{\mathcal{N}}(\pi)||_\infty, ||c^{\mathcal{N}}(\pi)||_\infty\}$ and $s = \min_{\{(i,j)\in\mathcal{N}, \pi_{ij}>0\}} \pi_{ij}$.*

**Corollary 1** *Let $\mathcal{N} \subset \mathcal{N}_0$ be a subspace such that $\text{spt}\pi^* \subset \mathcal{N}$. Algorithm 1 outputs a matrix $\pi$ satisfying $||r^{\mathcal{N}}(\pi) - \mu||_1 + ||c^{\mathcal{N}}(\pi) - \nu||_1 < \delta^2$ after $O(N\rho h(\delta)^{-2} \log(h/s))$ iterations, where $N = \max\{m, n\}$.*

The extra factor of $N$ in Corollary 1 is needed since we have to conduct $N$ more times iterations to convert the $l_2$ bound to the $l_1$ bound. The stoping criterion $||r^{\mathcal{N}}(\pi^{(k)}) - \mu||_2 + ||c^{\mathcal{N}}(\pi^{(k)}) - \nu||_2 < \delta^2$ in Algorithm 1 could be adjust to $||r^{\mathcal{N}}(\pi^{(k)}) - \mu||_1 + ||c^{\mathcal{N}}(\pi^{(k)}) - \nu||_1 < \delta$, which follows the idea from [2] and is a stronger $l_1$ approximation. This adjustment is justified by the following theorem.

**Theorem 3** *Let $\mathcal{N} \subset \mathcal{N}_0$ be a subspace such that $\text{spt}\pi^* \subset \mathcal{N}$. Algorithm 1 with stopping criterion $||r^{\mathcal{N}}(\pi^{(k)}) - \mu||_1 + ||c^{\mathcal{N}}(\pi^{(k)}) - \nu||_1 < \delta$ outputs a matrix $\pi$ satisfying $||r^{\mathcal{N}}(\pi) - \mu||_1 + ||c^{\mathcal{N}}(\pi) - \nu||_1 < \delta$ after $O((\delta)^{-2} \log(h/s))$ iterations.*

Algorithm 1 conducts the standard Sinkhorn iteration on a subset, which will cause numerical issues as $\varepsilon \to 0$. To avoid that, we define the following stabilized kernel to replace the original kernel. The replacement generates stabilized iterations, which are mathematically equivalent to the original Sinkhorn algorithm but make a significant improvement in practice.

**Definition 3 (Stabilized Kernel [37])** *For $\varepsilon > 0$, $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^m$, the stabilized kernel $K(\varepsilon, \alpha, \beta)$ associated with cost matrix c is given by*

$$K_{ij}(\varepsilon, \alpha, \beta) = \exp\left(\frac{1}{\varepsilon}(\alpha_i + \beta_j - c_{ij})\right), \quad (i,j) \in \mathcal{N}_0. \tag{10}$$

**Definition 4 (Restricted Kernel)** *For a space $\mathcal{N} \subset \mathcal{N}_0$, the restricted kernel $K^{\mathcal{N}}$ is defined as*

$$K_{ij}^{\mathcal{N}} := \begin{cases} K_{ij} & (i,j) \in \mathcal{N}, \\ 0 & (i,j) \in \mathcal{N}_0 \setminus \mathcal{N}. \end{cases} \tag{11}$$

To avoid numerical issues during iteration, we intended to start from choosing a suitable subspace $\mathcal{N}$ to conduct iterations and then refine the subset to a 'smaller' one, which is summarised in the Algorithm 2. We will show how to generate the subspace $\mathcal{N}$ by multi-scale scheme in the next section.

**Algorithm 2** SparseSinkhorn $(X, Y, \mathcal{N}, \alpha, \beta, \varepsilon, \theta)$

1: initialize $l \leftarrow 0$, $\varepsilon^{(0)}$
2: $\mathcal{N}^{(0)} \leftarrow \mathcal{N}$, $\alpha^{(0)} \leftarrow \alpha$, $\beta^{(0)} \leftarrow \beta$
3: **repeat**
4:     generate the kernel $K^{\mathcal{N}^{(l)}}$ by (11)
5:     $(\pi^{(l+1)}, \alpha^{(l+1)}, \beta^{(l)}) \leftarrow$ RestrictedSinkhorn($\mathcal{N}^{(l)}$, $K^{\mathcal{N}^{(l)}}$, $\mu$, $\nu$, $\varepsilon^{(l)}$)
6:     $\varepsilon^{(l+1)} \leftarrow \varepsilon^{(l)}/(1+\sigma)$
7:     $\mathcal{N}^{(l+1)} \leftarrow \emptyset$
8:     **for** $(i,j) \in \mathcal{N}^{(l)}$ **do**
9:         **if** $K_{ij}^{\mathcal{N}^{(l)}}(\varepsilon^{(l+1)}, \alpha^{(l+1)}, \beta^{(l+1)}) > \theta$ **then**
10:             $\mathcal{N}^{(l+1)} \leftarrow \mathcal{N}^{(l+1)} \cup (i,j)$
11:         **end if**
12:     **end for**
13:     $l \leftarrow l + 1$
14: **until** $\varepsilon^{(l)} < \varepsilon$
15: **Output** $\pi^{(l)}, \alpha^{(l)}, \beta^{(l)}$

The scaling steps $\varepsilon^{(l+1)} \leftarrow \varepsilon^{(l)}/(1+\sigma)$ (Algorithm 2 step 12) generate a decreasing sequence $\{\varepsilon^{(l)}\}$ that converges to zeros $(\sigma > 0)$. For positive parameter $\varepsilon_l$, Algorithm 2 step 5 performs the RestrictedSinkhorn algorithm on a subset $\mathcal{N}^{(l)}$, which outputs a coupling $\pi^{*(l+1)}$ and two dual variables $\alpha^{(l+1)}$ and $\beta^{(l+1)}$. Followed by the truncation steps (Algorithm 2 steps 7-12), the new subset is generated by truncating the kernel with the parameter $\theta$. The idea of truncating kernel to generate a new subset takes inspiration from [37]. However, we should point out that our algorithm create the new subset $\mathcal{N}^{(l+1)}$ by truncating the kernel on the old subset $\mathcal{N}^{(l)}$ rather than the whole space $X \times Y$, which reduces the computational cost significantly. Following the idea from [42], we provide a theoretical analysis to valid the proposed truncation steps when $|X| = |Y| = n$.

**Definition 5** *Let $V$ be the vertices of the set $\Pi(\mu, \nu)$, the suboptimal gap $\Delta$ of the restricted OT is defined as*

$$\Delta \triangleq \inf_{\{\pi\in V: \langle c,\pi\rangle > \langle c,\pi^*\rangle\}} \langle c, \pi\rangle - \langle c, \pi^*\rangle. \tag{12}$$

**Theorem 4** *Let $\varepsilon^{(0)} < \frac{\Delta}{n\log n - \log(s/2)}$ and $\theta < (\frac{s}{2} - \delta)^{(1+\sigma)}$, Algorithm 2 steps 7-12 output a subset $\mathcal{N}^{(l+1)}$ such that $\text{spt}\pi^* \subset \mathcal{N}^{(l+1)}$, where $s = \min_{\{(i,j)\in\mathcal{N}, \pi_{ij}>0\}} \pi_{ij}$.*

## 4. Estimate the Subset by Multi-Scale Scheme

Algorithm 2 approaches the unregularized OT problem by performing Sinkhorn iterations on a restricted subspace

$\mathcal{N}$. The remaining problem is, how to provide a suitable $\mathcal{N}$ for Algorithm 2. We employ the multi-scale scheme to identify the subset $\mathcal{N}$ by using the solution from the previous scale. Additionally, the solution at the previous scale provides a initialization for the current scale, which reduces the memory usage significantly.

**Definition 6 (Hierarchical Partition [38])** *A hierarchical partition for a discrete set $X$ is an ordered tuple $(X^{(0)}, \dots, X^{(K-1)})$ where $X^{(0)} = \{\{x\} : x \in X\}$ is the trivial partition of $X$ into singletons and the child cell is constructed by merging cells from the child cell's previous level. For $k \in \{1, \cdots, K-1\}$ and any $x_i \in X^{(k)}$ there exists some $\hat{X} \subset X^{(k-1)}$ such that $x_i = \bigcup_{\hat{x}_i \in \hat{X}} \hat{x}_i$, and we call $\hat{x}_i$ a child of $x_i$.*

For discrete measure $\mu = \sum_{i=1}^n \mu_i \delta_{x_i}$, its multi-scale measure scheme is the tuple $(\mu^{(0)}, \dots, \mu^{(K-1)})$ which can be decomposed from fine $(k = 0)$ to coarse $(k = K-1)$ scales by setting

$$\mu^{(k)} = \sum_{i \in J^{(k)}} \mu_i^{(k)} \delta_{x_i^{(k)}}, \quad X^{(k)} = \{x_i^{(k)}, i \in J^{(k)}\}. \quad (13)$$

Starting from $\mu^{(0)} = \mu$ (and $X^{(0)} = \{X\}$), we extract a clustering $X^{(k)} = \cup_{i \in J^{(k)}} C_i^{(k)}$ of the support of $X^k$ of $\mu^k$, and we denote by $X^{(k+1)} = \{x_i^{(k+1)}, i \in J^{(k+1)}\}$ the corresponding cluster centroids. Next, we compute the weights by gathering the mass in each cluster $\mu_i^{(k+1)} = \mu^k(C_i^{(k)})$.

Let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ be vectors with hierarchical partitions $(X^{(0)}, \dots, X^{(K-1)})$ and $(Y^{(0)}, \dots, Y^{(K-1)})$, where $X^{(k)} = \{x_i^{(k)}, i \in J_X^{(k)}\}$ and $Y^{(k)} = \{y_j^{(k)}, j \in J_Y^{(k)}\}$. For cost matrix $c \in \mathbb{R}^{n \times m}$, the cost map $\mathcal{C}$ is given by $\mathcal{C} : X \times Y \to \mathbb{R}^{n \times m}$ with $\mathcal{C}(x_i, y_j) = c_{ij}$. The hierarchical partition of cost matrix $(c^{(0)}, \dots, c^{(K-1)})$ is

$$\begin{aligned} c_{ij}^{(k)} &= \min_{(x,y)} \mathcal{C}(x,y), (i,j) \in J_X^{(k)} \times J_Y^{(k)}, \\ (x,y) &\in \left(\text{children}(x_i^{(k)}), \text{children}(y_j^{(k)})\right). \end{aligned} \quad (14)$$

for $k \in \{0, \cdots, K-1\}$. Fig. 1 (left) shows a three-level multi-scale structure of two discrete measures $X$ and $Y$ (Each measure has eight elements and the measure $(X, Y)$ has 64 elements). The hierarchical partition is created by combining the adjoined measure from the previous level. The first layer $(k = 0)$ denotes the original measure and $(x_i^{(0)}, y_i^{(0)})$ is a child of $(x_i^{(1)}, y_i^{(1)})$ in the second layer $(k = 0)$. The third layer has only four elements. Each of the element is the combination of the measures $(x_i^{(1)}, y_i^{(1)})$ from the second layer. We further illustrate how the multi-scale scheme is combined with the SparseSinkhorn algorithm to approach the restricted OT problem (4). We start computing the optimal coupling from the coarse layer by calling the SparseSinkhorn algorithm. After that a new coupling is generated and the subset is estimated for the next layer. Then the SparseSinkhorn algorithm is called again, and a new coupling is computed. The loops continue until reaching the first layer with $k = 0$. Finally, we generate the subset $\mathcal{N}^{(0)}$ and compute the optimal coupling $\pi^{(0)}$. The procedure is summarised in Algorithm 3. Please refer to Fig. 1 (right) for a specific example.

**Algorithm 3** Multi-Scale SparseSinkhorn

Construct multi-scale structures $\{(X^k, \mu^{(k)})\}_{k=0}^{K-1}$, $\{(Y^{(k)}, \nu^{(k)})\}_{k=0}^{K-1}$ and $\{c^{(k)}\}_{k=0}^{K-1}$
$\alpha^{(K-1)}, \beta^{(K-1)} \leftarrow \mathbf{0}$
$\mathcal{N}^{(K-1)} \leftarrow J_X^{(K-1)} \times J_Y^{(K-1)}$
**for** $k = K-1, \dots, 0$ **do**
  $(\pi^{(k)}, \alpha^{(k)}, \beta^{(k)}) \leftarrow$ SparseSinkhorn($X^{(k)}$, $Y^{(k)}$, $\mathcal{N}^{(k)}$, $\alpha^{(k)}$, $\beta^{(k)}$, $\varepsilon^{(k)}, \theta$)
  **if** $k > 0$ **then**
    $\mathcal{N}^{(k-1)} \leftarrow \emptyset$
    **for** $(\hat{i}, \hat{j}) \in \text{spt}\pi^{(k)}$ **do**
      **for** $(x_i^{(k-1)}, y_j^{(k-1)}) \in$ (children($x_{\hat{i}}^{(k)}$), children($y_{\hat{j}}^{(k)}$)) **do**
        $\mathcal{N}^{(k-1)} \leftarrow \mathcal{N}^{(k-1)} \cup (i, j)$
        $(\alpha_i^{(k-1)}, \beta_j^{(k-1)}) \leftarrow (\alpha_{\hat{i}}^{(k)}, \beta_{\hat{j}}^{(k)})$
      **end for**
    **end for**
  **end if**
**end for**
Output $\pi^{(0)}$

**Remark 2** *Using the multi-scale scheme to compute the OT problem was first proposed in [30] and studied in [21, 27, 37]. The difference between our algorithm and the existing methods is that the methods [27, 30] only use the solution from the coarse as a warm start but do not provide a new subset for the next layer. Moreover, we compute the subset $\mathcal{N}^{(l-1)}$ according to the set $\text{spt}\pi^{(k)}$ while the multi-scale algorithm [37] computes the subset in the set $X \times Y$, which is much more time-consuming to estimate the subset compared with ours.*

## 5. Numerical Experiments

Numerical experiments are presented in this section to demonstrate the performance of the M3S algorithm in terms of computing time and memory requirements. Moreover, we compute the Wasserstein-$p$ distance between color images and transfer color between $1920 \times 1200$ images. All reported run-times are obtained on a computer with 64G-B memory, a 2.7GHz Intel Xeon E5-2697 processor, and a GPU of RTX (2080Ti).

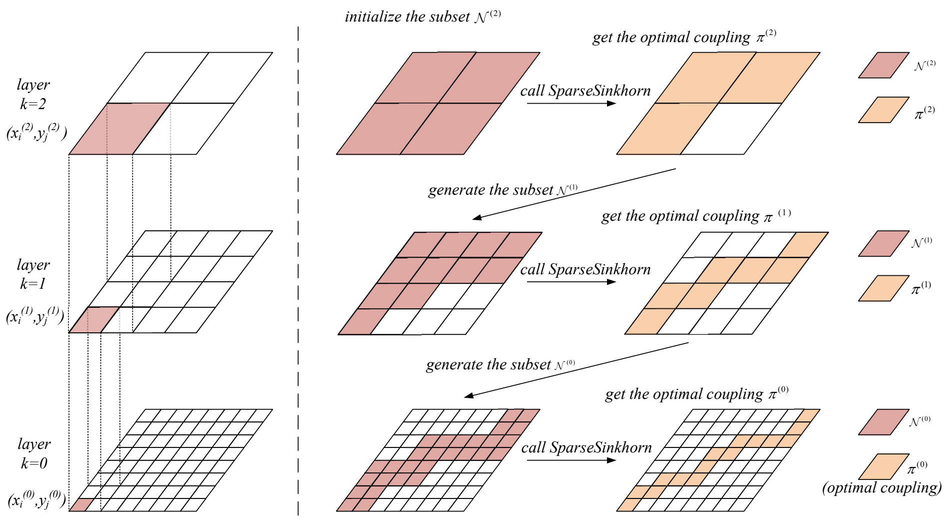


Figure 1. Left: A three level hierarchical partition of discrete measure $(X, Y)$. Right: How the multi-scale scheme is combined with the SparseSinkhorn algorithm. The pink area denotes the subset $\{\mathcal{N}^{(l)}\}$. The algorithm starts from the third layer for $k = 3$. The subset $N^{(2)}$ is initialized in the entire space and the SparseSinkhorn algorithm is called to get the optimal coupling $\pi^{(2)}$. The support of the coupling is used to generate the subset $\mathcal{N}^{(1)}$, after which the SparseSinkhorn algorithm is called to get the optimal coupling $\pi^{(1)}$. Finally, we generate $\mathcal{N}^{(0)}$ and get the optimal coupling $\pi^{(0)}$ in the first layer. The adjoined measure of $(x_i^{(3)}, y_i^{(3)})$ is its the children measure $(x_i^{(2)}, y_i^{(2)})$.

### 5.1. Performance Analysis

We first consider the OT between pairs of $n \times n$ grayscale images. The cost matrix $c \in \mathbb{R}^{n^2 \times n^2}$ is computed by $c(x, y) = |x - y|^p$, which is the $l_p$ distances between pixel locations. The experiment is tested on the DOT benchmark [39] with image size ranges from $32 \times 32$ to $512 \times 512$, which means that the cardinalities of $X$ and $Y$ ranged from $2^{10}$ to $2^{18}$ and the dimension of the full coupling spaces between $2^{20}$ and $2^{36}$. We consider transports between two images with equal size, i.e. $|X| = |Y|$. We first compute the OT problem with the $l_p$ distances between pixel locations for $p \in [1, 2.5]$. For $p = 1$, we compute the earth mover's distance. We set $\theta = 10^{-8}$ for refining the new subset and $\delta = 10^{-5}$ for estimating the error bound.

The performance of the M3S algorithm is compared with our GPU implementation of the Sinkhorn Algorithm, and compared with CPU implementation of the CPLEX network simplex [12], the FastEMD [32], the Greenkhorn algorithm [2] implemented by the POT library [19]. Moreover, we compare the M3S with the multi-scale Sinkhorn (M2S) implemented by [17]. We also equip both algorithms with the *keops* backend [9] and set the same parameters for comparison. Fig. 2 (right) compares the memory usage of different algorithms for computing the Wasserstein-1 distance, namely, we compute the Earth Mover Distance (EMD). The M3S algorithm shows an $\mathcal{O}(n)$ increase in memory requirements. The Sinkhorn algorithm shows an $\mathcal{O}(n^2)$ increase in memory requirements, and it runs out of memory on our device (RTX 2080ti) after the problem size becomes larger than $128 \times 128 (2^{14})$. Both the FastEMD and the transport network simplex show an $O(n^3)$ increase in memory requirements. Among all of our tests, the FastEMD method uses the smallest memory compared with the Sinkhorn algorithm and the Network simplex, but it still cost about 30.2GB memory to calculate the EMD between two images with size $256 \times 256$. The proposed algorithm shows a linear memory requirement for computing the EMD as $N$ increases to more than $2^{36}$.

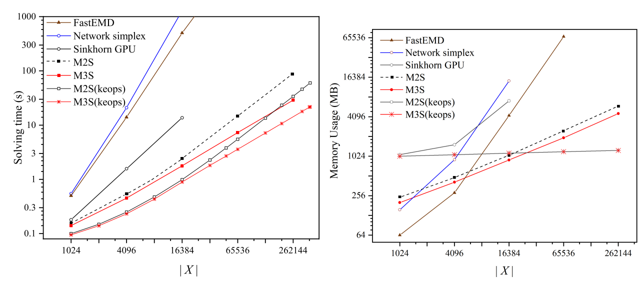


Figure 2. Running time and memory requirement of different algorithms. Right: The solving time of the FastEMD grows with the order of $O(n^3)$. Sinkhorn algorithm shows an $O(n^2)$ time complexity. The M2S algorithm grows with the $O(n\log(n))$ complexity while the M3S grows with linear complexity. Left: Both the M2S and M3S algorithm shows linear memory requirement and the M3S algorithm requires less memory. When equipped with the *keops* backend, M3S and M2S allocated the same buffer on GPU and M3S runs faster as the problem size grows.

Fig. 2 (left) shows the solving time for computing the EMD by different algorithms. Among all of the tests, the

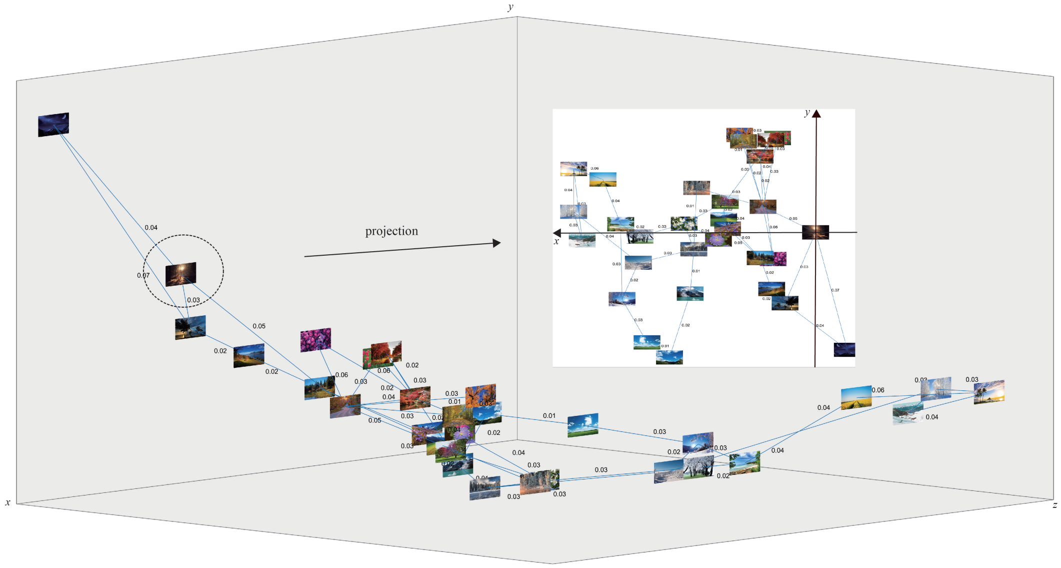


Figure 4. Computing Wasserstein-2 distance between 30 color images (1920 × 1200) by the M3S algorithm. We compute the Wasserstein distance between images and construct the distance matrix. We use the multidimensional scale to reconstruct the distance matrix and get the three-dimensional coordinates of the center point of each image. The images are presented in $\mathbf{R}^3$ according to their coordinates. We take one image as a reference, and the other images are projected onto the plane of this image. Our work is an extension of measuring the color images distance in 3D space. (see [35], Fig. 6)

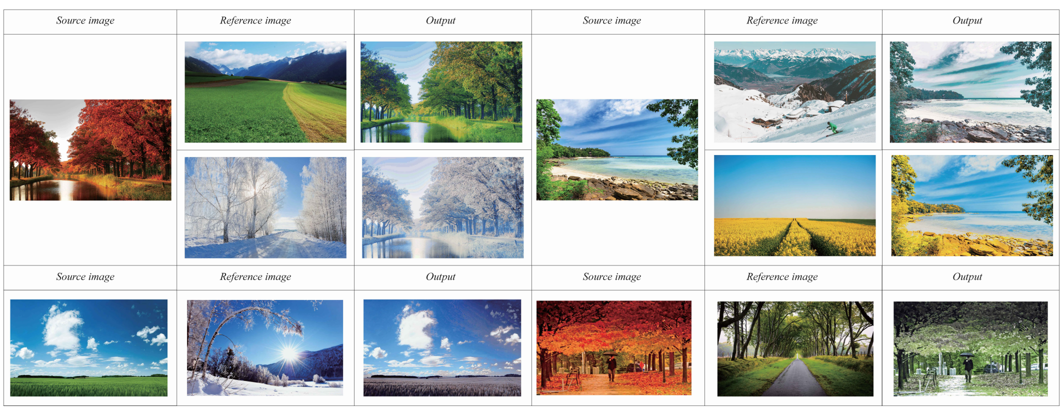


Figure 5. Color transfer between 1920 × 1200 images. We compute the optimal transport mapping of each pixel from the source image to the reference image in RGB space by the proposed algorithm.

FastEMD shows an $O(n^3)$ increase of time requirement and the time required by the proposed algorithm grows linearly. It takes an average of 28.90 seconds for computing the EMD between two images with size $512 \times 512$. Other methods fail to converge within the observation time (1200 seconds). We make a further study by comparing the average solving time for cost $c(x, y) = |x - y|^p$ for different $p$. It can be seen that the solving time varies with different $p$ in Fig. 3 (left). We observe an average of 7 seconds for solving the OT between two images with size $256 \times 256$ while $p$ ranges from $1$ to $1.4$, and as $p$ becomes close to $2.0$, the solving time decreases. Fig. 3 (right) shows the memory usage for storing the sparse matrix $K(x, y)$, which takes up the main memory resources on GPU. For different $p$, the memory we allocated grows linearly with the problem size. We allocate 64MB memory on GPU for the OT problem when the images' size is $64 \times 64$ and allocate four times memory $256$MB as the images' size goes to $128 \times 128$. The relative error $|W_p - W_p^*|/W_p^*$ is reported in Tab.1 for different image size $n$. The cost function is given by $c(x, y) = |x - y|^p$. The exact solution $W_p^*$ is obtained by the FastEMD, which computes the solution of the unregularized problem (3).

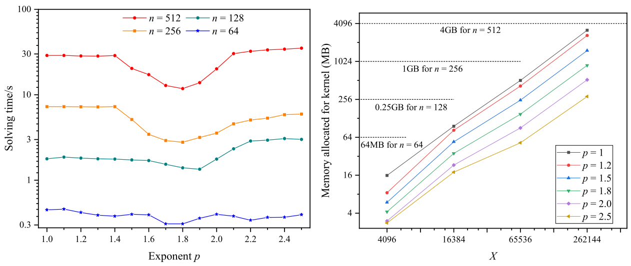


Figure 3. Solving time and memory usage for cost $c(x, y) = |x - y|^p$ with different $p$.

Table 1. Relative error between $W_p$ and $W_p^*$ computed by the M3S and FastEMD, respectively. '-' means that the FastEMD runs out of memory so that the relative error is not compared.

| Relative error ($10^{-4}$) \ Image size | | 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|---|
| p | 1 | 0.134 | 0.492 | 3.56 | 1.27 | — |
| | 1.2 | 0.157 | 0.0822 | 6.23 | 2.34 | — |
| | 1.5 | 0.104 | 0.132 | 0.0978 | 0.0522 | — |
| | 1.7 | 0.113 | 0.180 | 0.863 | 0.411 | — |
| | 2 | 0.109 | 0.803 | 0.0675 | 0.923 | 0.460 |

### 5.2. Wasserstein Distance Between Color Images

The Wasserstein-$p$ distance is a useful metric for images [35], because it quantifies the intuitive notion of image similarity, especially to quantity the distance between images with different sizes. In this section we navigate through a database of color images by computing the Wasserstein-2 distance for color images up to $1920 \times 1200$ in size. We consider the color images as points in the image's RGB space. The base images are mapped to RGB space and then the distances between all of the images are computed. To visualize the relative location of images in $\mathbf{R}^3$ space, we employ the multi-dimensional scaling (MDS) techniques [5], which embeds a set of images as points in a Euclidean space. The visualization of the 3-D MDS embedding can be used to organize and refine the results of the nearest-neighbor query in a perceptually intuitive way. Users can quickly navigate to the portion of the image space of interest by computing the Wasserstein-2 distance with the proposed algorithm.

Fig. 4 shows the Wasserstein-2 distance between 30 different color images, the distance matrix $\mathbf{W}_{30\times30}$ is constructed with $\mathbf{W}(i, i) = 0$ and each entry $\mathbf{W}(i, j)$ is the Wasserstein-2 distance between the images $i$ and $j$. The three-dimensional coordinates of each image are obtained by embedding the matrix in $\mathbf{R}^3$ space. The smaller the distance is, the more similar the two images are. The Wasserstein distance is the navigation through a database of large size color images.

### 5.3. Color Transfer Between Large Size Images

Color transfer has been receiving considerable attention in the computer graphics and computer vision communities. The purpose of color transfer is to recolor a given image or video by deriving a mapping between that image and another image as a reference [15]. The user can modify the original image by choosing a reference image such that the original image acquires the palette of that reference. Computing optimal transport mapping for color transfer has been studied in [16, 20, 22, 33]. However, the computational cost increase heavily as the image size goes larger. In this section, we transfer color between images by computing the OT map with the proposed algorithm. We denote $u : \Omega_u \subset \mathbb{Z}^2 \mapsto \Sigma \subset \mathbb{R}$, where $\Omega_u$ is the pixel grid of $u$ and $\Sigma$ is the quantized RGB color space. We denote $X_i = (U_i) \in \mathbb{R}^3$ to specify the spatial component $(x_i \in \Omega_u)$ and the color component $(U_i \in \Sigma)$. The source measure $\mu_X$ is constructed with $\mu_X : X \mapsto \sum_{i \in I_X} \mu_i \delta_{X_i}(X)$ and similarly for the target measure $\nu_Y$. $N_x$ is the number of all pixels in the image. The regularized OT problem between two measures $X$ and $Y$ is given by

$$\pi_\varepsilon^* = \arg \min_{\pi \in \Pi(\mu, \nu)} \langle C, \pi \rangle - \varepsilon H(\pi). \tag{15}$$

The transport cost is taken as $C_{ij} = ||U_i - V_j||_2^2$, the square of the $l_2$ norm of the RGB space. To keep the sparsity of the kernel, the point clouds are normalized to the $[0, 1]^3$. We define a one-to-one optimal coupling mapping $T$ from a optimal coupling $\pi_\varepsilon^*$:

$$T(X_i) = Y_j, j \in \arg\min(\pi_\varepsilon)_{ij}. \tag{16}$$

By computing the optimal coupling and the OT mapping, we present six different examples of color transfer between seasons in Fig. 5. The OT mapping (16) defines a pixel-to-pixel color transfer without smoothing transport maps and avoid false colors artifacts as well as a loss of color contrast.

## 6. Conclusion

We proposed the restricted regularized OT problem and proved the convergence of the restricted optimal solution. Based on the theoretical guarantee, we introduced the M3S algorithm for solving the regularized OT in a subset. The proposed algorithm was implemented on CUDA with linear cost for computing OT problem in terms of time and memory requirement. The proposed algorithm is accurate enough to compute the Wasserstein-$p$ distance and compare features between color images' RGB clouds. The color transfer was realized between images with the size as large as $1920 \times 1200$ by computing the optimal transport map between RGB color space.

**Acknowledgments**: This research was supported by the National Natural Science Foundation of China (Grant No. 61873254) and Science and Technology Major Project of Guangxi (Guike AA18118054).

## References


[1] Mokhtar Z. Alaya, Maxime Berar, Gilles Gasso, and Alain Rakotomamonjy. Screening sinkhorn algorithm for regularized optimal transport. In *Advances in Neural Information Processing Systems*, pages 12169–12179, 2019. 1

[2] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pages 1961–1971, 2017. 1, 4, 6

[3] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862*, 2017. 1

[4] D. P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14(1):105–123, 1988. 1

[5] Ingwer Borg and J. Lingoes. *Modern Multidimensional Scaling*. The University of Chicago Press, 1987. 8

[6] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, 1991. 1

[7] Bruno and Lvy. A numerical algorithm for $l_2$ semi-discrete optimal transport in 3d. *Esaim Mathematical Modelling and Numerical Analysis*, 49(6):1693–1715, 2015. 2

[8] L. Caffarelli, L. Nirenberg, and J. Spruck. Correction to: The dirichlet problem for nonlinear second-order elliptic equations i. monge-ampere equation. *Communications on Pure and Applied Mathematics*, 40(5):659–662, 1987. 1

[9] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunes, Francois-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021. 6

[10] Lenaic Chizat, Gabriel Peyre, Bernhard Schmitzer, and Francois-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018. 1

[11] R. Cominetti and J. San Martn. Asymptotic analysis of the exponential penalty trajectory in linear programming. *Mathematical Programming*, 67(1):169–187, 1994. 3

[12] IBM Corporation. *IBM ILOG CPLEX Optimizer*. IBM, 2013. 6

[13] D. Dizdar. Optimal transport methods in economics. *Journal of Economics*, 125(3):309–312, 2018. 1

[14] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *Proceedings of the 35th International Conference on Machine Learning(ICML) in PMLR*, pages 1367–1376, 2018. 2

[15] H. Sheikh Faridul, T. Pouli, C. Chamaret, J. Stauder, E. Reinhard, D. Kuzovkin, and A. Tremeau. Colour mapping: A review of recent methods, extensions and applications. *Computer Graphics Forum*, 35(1):59–88, 2016. 8

[16] S. Ferradans, N. Papadakis, G Peyre, and Jean-Francois Aujol. Regularized discrete optimal transport. *Siam Journal on Imaging Sciences*, 7(3):428–439, 2013. 8

[17] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690, 2019. 6

[18] Alessio Figalli and Ludovic Rifford. Continuity of optimal transport maps and convexity of injectivity domains on small deformations of s2. *Communications on Pure and Applied Mathematics*, 62(12):1670–1706, 2010. 1

[19] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. 6

[20] Qian Fu, Ying He, Fei Hou, Juyong Zhang, Anxiang Zeng, and Yong-Jin Liu. Vectorization based color transfer for portrait images. *Computer-Aided Design*, 115(1):111–121, 2019. 8

[21] Samuel Gerber and Mauro Maggioni. Multiscale strategies for computing optimal transport. *Journal of Machine Learning Research*, 18(72):1–32, 2017. 5

[22] M. Grogan and R. Dahyot. L-2 divergence for robust colour transfer. *Computer Vision and Image Understanding*, 181(1):39–49, 2019. 8

[23] B. Kalantari, I. Lari, F. Ricca, and B. Simeone. On the complexity of general matrix scaling and entropy minimization via the ras algorithm. *mathematical programming*, 112(2):371–401, 2008. 4

[24] Jun Kitagawa, Quentin Merigot, and Boris Thibert. A newton algorithm for semi-discrete optimal transport. *Journal of the European Mathematical Society*, 21:2603–2651, 03 2016. 2

[25] JJ Kosowsky and AL Yuille. Solving the assignment problem with statistical physics. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, pages 159–164, 1991. 1

[26] HW Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics*, 52(1):7–21, 2005. 1

[27] Arthur Leclaire and Julien Rabin. A fast multi-layer approximation to semi-discrete optimal transport. In *Scale Space and Variational Methods in Computer Vision*, pages 341–353, 2019. 2, 5

[28] John Lee, Nicholas P. Bertrand, and Christopher J. Rozell. Unbalanced optimal transport regularization for imaging problems. *IEEE Transactions on Computational Imaging*, 6(1):1219–1232, 2020. 1

[29] Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in o(vrank) iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 424–433, 2014. 2

[30] Quentin Mrigot. A multiscale approach to optimal transport. *Computer Graphics Forum*, 30(5):1583–1592, 2011. 1, 5

[31] James B. Orlin. Network flows - theory, algorithms and applications. *Journal of the Operational Research Society*, 45(11):791–796, 1993. 1

[32] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*, pages 460–467, 2009. 6

[33] Julien Rabin, Sira Ferradans, and Nicolas Papadakis. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4852–4856, 2014. 8

[34] R. Tyrrel Rockafellar. *Convex Analysis*. Princeton University Press, 1970. 2

[35] Yossi Rubner, Leonidas Guibas, and Carlo Tomasi. The earth movers distance, multi-dimensional scaling, and color-based image retrieval. In *in Proceedings of the ARPA Image Understanding Workshop*, pages 661–668, 1997. 7, 8

[36] Bernhard Schmitzer. A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision*, 56(2):238–259, 2016. 1

[37] Bernhard Schmitzer. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing*, 41(3):1443–1481, 2019. 1, 3, 4, 5

[38] Bernhard Schmitzer and Christoph Schnorr. A hierarchical approach to optimal transport. In *Scale Space and Variational Methods in Computer Vision*, volume 7893, pages 452–464, 2013. 5

[39] Joern Schrieber, Dominic Schuhmacher, and Carsten Gottschlich. Dotmark - a benchmark for discrete optimal transport. *IEEE Access*, 5(3):271–282, 2017. 6

[40] Meisam Sharify, Stphane Gaubert, and Laura Grigori. Solution of the optimal assignment problem by diagonal scaling algorithms. *arXiv:1104.3830*, 2013. 1

[41] Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *American Mathematical Society*, 74(4):195–198, 1974. 1

[42] J Weed. An explicit analysis of the entropic penalty in linear programming. *arXiv:1806.01879*, 2018. 4