A Numerical Algorithm with Linear Complexity for Multi-Marginal Optimal Transport with L^1 Cost

Chunhui Chen[1], Jing Chen[2], Baojia Luo[3], Shi Jin[4] and Hao Wu[1,*]

[1] *Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China.*
[2] *School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798.*
[3] *Theory Lab, Central Research Institute, 2012 Labs, Huawei Technologies Co. Ltd., Hong Kong, SAR, China.*
[4] *School of Mathematical Sciences, Institute of Natural Sciences, and MOE-LSC Shanghai Jiao Tong University, Shanghai 200240, China.*

Abstract. Numerically solving multi-marginal optimal transport (MMOT) problems is computationally prohibitive, even for moderate-scale instances involving $l \geq 4$ marginals with support sizes of $N \geq 1000$. The cost in MMOT is represented as a tensor with N^l elements. Even accessing each element once incurs a significant computational burden. In fact, many algorithms require direct computation of tensor-vector products, leading to a computational complexity of $\mathcal{O}(N^l)$ or beyond. In this paper, inspired by our previous work [Comm. Math. Sci., 20, 2022], we observe that the costly tensor-vector products in the Sinkhorn Algorithm can be computed with a recursive process by separating summations and dynamic programming. Based on this idea, we propose a fast tensor-vector product algorithm to solve the MMOT problem with L^1 cost, achieving a miraculous reduction in the computational cost of the entropy regularized solution to $\mathcal{O}(N)$. Numerical experiment results confirm such high performance of this novel method which can be several orders of magnitude faster than the original Sinkhorn algorithm.

AMS subject classifications: 49M25, 65K10

Key words: Multi-marginal optimal transport, Sinkhorn algorithm, linear complexity, entropy regularization.

---

*Corresponding author. *Email addresses:* `hwu@tsinghua.edu.cn` (H. Wu), `cch21@mails.tsinghua.edu.cn` (C. Chen), `jing.chen@ntu.edu.sg` (J. Chen), `luobaojia2@huawei.com` (B. Luo), `shijin-m@sjtu.edu.cn` (S. Jin)

1 Introduction

Multi-marginal optimal transport, first proposed by Gangbo and Święch [18], is an extension of the classical optimal transport problem [7,36]. It aims to find an optimal transport plan that minimizes total cost while fitting multiple marginal distributions. MMOT problems naturally arise in various fields, such as machine learning [2, 5, 11, 20, 32], incompressible fluid dynamics [4,8,42], density function theory [10,14,22,23], Schrödinger bridge [12,19], and tomographic reconstruction [1], and thus have attracted wide attention in recent years.

However, the heavy computational burden of solving general MMOT problems limits its broad application. For l-marginal distributions with support sizes of N, the l-th order cost tensor in MMOT problems contains N^l elements. To fully obtain the information of the cost tensor, it is inevitable to repeatedly access all elements in the tensor, leading to a significant computational cost. For example, the generalized Sinkhorn algorithms [6, 7, 30, 34, 40] require repeated computation of tensor-vector products, resulting in a computational complexity of $\mathcal{O}(N^l)$. More severely, directly solving linear programming problems has a computational complexity of $N^{\mathcal{O}(l)}$. Therefore, these methods remain computationally prohibitive even for moderate-scale MMOT problems. Some modified algorithms with lower computational complexity have been proposed for specific MMOT problems, such as the MMOT problem with a tree structure [19,41] and the Wasserstein barycenter [3,6,39].

In this work, we propose a novel implementation of the Sinkhorn algorithm for solving the entropy regularized MMOT problem with L_1 cost applications in image processing [38], computer vision [35] and seismic tomography [16,33,43], which has linear computational complexity relative to support size N. This work is a follow-up work of the fast Sinkhorn algorithms [28, 29], which observe the special structure of the kernel matrix with Wasserstein-1 metric and utilize dynamic programming techniques [21,24–26], achieving linear computational complexity for solving up to 2-marginal optimal transport problem. Unlike the previous situation, the kernel matrix evolves into an l-th order tensor in the l-marginal optimal transport problem. The computational burden of the Sinkhorn algorithm becomes prohibitive due to the $\mathcal{O}(N^l)$ operations required by the tensor-vector products. To address this problem, we observe a similar special structure of the kernel tensor and accelerate the tensor-vector products using the series rearrangement and dynamic programming techniques [25], which results in a fast Sinkhorn algorithm with $\mathcal{O}(N)$ computational complexity.

The rest of the paper is organized as follows. In Section 2, we review the MMOT problem and the generalized Sinkhorn algorithm for the 1-dimensional (1D) and 3-marginal case. Then, we introduce the key fast tensor-vector product technique and provide a detailed implementation of our algorithm in Section 3. This algorithm can be conducted in more general scenarios, such as high-dimensional and l-marginal optimal transport problems, which are presented in Section 4. In Section 5, numerical experiments are carried

out to demonstrate the overwhelming advantage of our algorithm in terms of computational efficiency. Finally, we conclude this paper in Section 6.

2   Multi-marginal optimal transport and the Sinkhorn algorithm

We first review the MMOT problem and the Sinkhorn algorithm [6]. To streamline our discussion, we exclusively showcase the 3-marginal optimal transport problem and its corresponding Sinkhorn algorithm in 1D space. However, it is important to note that this algorithm can be seamlessly extrapolated to address more marginal cases in higher dimensional space. For a more comprehensive understanding, we refer the readers to [6].

We consider the discrete MMOT problem, which is a discretization of the continuous MMOT problem [34]. It solves the following minimization problem:

$$W(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) = \inf_{\mathcal{T}\in\Pi} \langle \mathcal{C},\mathcal{T} \rangle = \inf_{\mathcal{T}\in\Pi} \sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{k=1}^{N_3} c_{ijk}t_{ijk}, \tag{2.1}$$

in which $\boldsymbol{u}\in\mathbb{R}^{N_1}, \boldsymbol{v}\in\mathbb{R}^{N_2}, \boldsymbol{w}\in\mathbb{R}^{N_3}$ are three discrete probabilistic distributions

$$\boldsymbol{u}=(u_1,u_2,\cdots,u_{N_1}), \quad \boldsymbol{v}=(v_1,v_2,\cdots,v_{N_2}), \quad \boldsymbol{w}=(w_1,w_2,\cdots,w_{N_3}).$$

Here $\mathcal{C}=(c_{ijk})\in\mathbb{R}^{N_1\times N_2\times N_3}$ is the cost tensor, and $\mathcal{T}=(t_{ijk})\in\mathbb{R}_{0+}^{N_1\times N_2\times N_3}$ is the multi-marginal transport plan, satisfying the linear constraints

$$\mathcal{T}\in\Pi=\left\{\mathcal{T}=(t_{ijk})\,\Big|\,\sum_{j=1}^{N_2}\sum_{k=1}^{N_3}t_{ijk}=u_i,\,\sum_{i=1}^{N_1}\sum_{k=1}^{N_3}t_{ijk}=v_j,\,\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}t_{ijk}=w_k,\,t_{ijk}\geq 0\right\}. \tag{2.2}$$

In this work, our discussion is suitable for any N_1,N_2 and N_3. For the sake of simplicity, we assume $N_1=N_2=N_3=N$ in the rest of the paper.

The generalized Sinkhorn algorithm [6] was proposed to solve the regularized MMOT problem by introducing an entropy regularization term

$$W_\varepsilon(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) = \inf_{\mathcal{T}\in\Pi} \sum_{i,j,k=1}^{N} c_{ijk}t_{ijk}+\varepsilon t_{ijk}\ln(t_{ijk}).$$

The above optimization problem can be solved using the method of Lagrange multipliers, with the Lagrangian given by

$$\begin{aligned}
\mathcal{L}(\mathcal{T},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) =&\; \sum_{i,j,k=1}^{N}\varepsilon\ln\left(\frac{t_{ijk}}{K_{ijk}}\right)t_{ijk}+\sum_{i=1}^{N}\alpha_i\left(\sum_{j,k=1}^{N}t_{ijk}-u_i\right)\\
&+\sum_{j=1}^{N}\beta_j\left(\sum_{i,k=1}^{N}t_{ijk}-v_j\right)+\sum_{k=1}^{N}\gamma_k\left(\sum_{i,j=1}^{N}t_{ijk}-w_k\right),
\end{aligned} \tag{2.3}$$

where $K_{ijk} = e^{-c_{ijk}/\varepsilon}$. Define $\boldsymbol{\phi} = (\phi_1, \cdots, \phi_N), \boldsymbol{\psi} = (\psi_1, \cdots, \psi_N), \boldsymbol{\varphi} = (\varphi_1, \cdots, \varphi_N)$ as

$$\phi_i = e^{-\frac{1}{3} - \frac{\alpha_i}{\varepsilon}}, \quad \psi_j = e^{-\frac{1}{3} - \frac{\beta_j}{\varepsilon}}, \quad \varphi_k = e^{-\frac{1}{3} - \frac{\gamma_k}{\varepsilon}}.$$

Taking the derivative of the Lagrangian with respect to t_{ijk} and setting it equal to zero yields

$$t_{ijk} = \phi_i \psi_j \varphi_k K_{ijk}, \quad 1 \le i, j, k \le N. \tag{2.4}$$

Combining Eq. (2.4) and the constraints in Eq. (2.2), we can obtain that when the regularized MMOT problem attains its optimal solution, $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi}$ satisfy

$$\phi_i \sum_{j,k=1}^{N} K_{ijk} \psi_j \varphi_k = u_i, \quad \psi_j \sum_{i,k=1}^{N} K_{ijk} \phi_i \varphi_k = v_j, \quad \varphi_k \sum_{i,j=1}^{N} K_{ijk} \phi_i \psi_j = w_k. \tag{2.5}$$

Since the elements in $\mathcal{K} = (K_{ijk})$ are strictly positive, the generalized Sinkhorn algorithm [6] can be applied to find scaling variables $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi}$ satisfying the above equations by iteratively updating

$$\boldsymbol{\phi}^{(t+1)} = \boldsymbol{u} \oslash \left(\mathcal{K} \times_j \boldsymbol{\psi}^{(t)} \times_k \boldsymbol{\varphi}^{(t)} \right), \tag{2.6}$$

$$\boldsymbol{\psi}^{(t+1)} = \boldsymbol{v} \oslash \left(\mathcal{K} \times_i \boldsymbol{\phi}^{(t+1)} \times_k \boldsymbol{\varphi}^{(t)} \right), \tag{2.7}$$

$$\boldsymbol{\varphi}^{(t+1)} = \boldsymbol{w} \oslash \left(\mathcal{K} \times_i \boldsymbol{\phi}^{(t+1)} \times_j \boldsymbol{\psi}^{(t+1)} \right), \tag{2.8}$$

in which $\oslash$ represents pointwise division and t denotes the iterative step. Here the tensor-vector products follow the notation in [15], having the forms of

$$\left(\mathcal{K} \times_j \boldsymbol{\psi}^{(t)} \times_k \boldsymbol{\varphi}^{(t)} \right)_i = \sum_{j=1}^{N} \sum_{k=1}^{N} K_{ijk} \psi_j^{(t)} \varphi_k^{(t)}, \tag{2.9}$$

$$\left(\mathcal{K} \times_i \boldsymbol{\phi}^{(t+1)} \times_k \boldsymbol{\varphi}^{(t)} \right)_j = \sum_{i=1}^{N} \sum_{k=1}^{N} K_{ijk} \phi_i^{(t+1)} \varphi_k^{(t)}, \tag{2.10}$$

$$\left(\mathcal{K} \times_i \boldsymbol{\phi}^{(t+1)} \times_j \boldsymbol{\psi}^{(t+1)} \right)_k = \sum_{i=1}^{N} \sum_{j=1}^{N} K_{ijk} \phi_i^{(t+1)} \psi_j^{(t+1)}. \tag{2.11}$$

The final MMOT distance is given by

$$W_\epsilon(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) = \sum_{i,j,k=1}^{N} \phi_i \psi_j \varphi_k c_{ijk} K_{ijk} = \langle \boldsymbol{\varphi}, (\mathcal{C} \odot \mathcal{K}) \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi} \rangle, \tag{2.12}$$

in which $\odot$ represents the Hadamard product of tensors. The generalized Sinkhorn algorithm is presented in Algorithm 1.

Algorithm 1. Generalized Sinkhorn Algorithm.

    Input: u,v,w of size $(N,1),\varepsilon$, tol, itr_max.
    Output: $W_\epsilon(u,v,w)$.
 1: Initialize $\boldsymbol{\phi},\boldsymbol{\psi},\boldsymbol{\varphi}=\mathbf{1}_N/N,t=0,\mathrm{Res}=\mathrm{Inf}$.
 2: **while** $(t<\mathrm{itr\_max})$ & $(\mathrm{Res}>\mathrm{tol})$ **do**
 3:     $t\leftarrow t+1$.
 4:     $\boldsymbol{\phi}\leftarrow u\oslash(\mathcal{K}\times_j\boldsymbol{\psi}\times_k\boldsymbol{\varphi})$.
 5:     $\boldsymbol{\psi}\leftarrow v\oslash(\mathcal{K}\times_i\boldsymbol{\phi}\times_k\boldsymbol{\varphi})$.
 6:     $\boldsymbol{\varphi}\leftarrow w\oslash(\mathcal{K}\times_i\boldsymbol{\phi}\times_j\boldsymbol{\psi})$.
 7:     $\mathrm{Res}\leftarrow\mathrm{sum}\big(|\boldsymbol{\phi}\odot(\mathcal{K}\times_j\boldsymbol{\psi}\times_k\boldsymbol{\varphi})-u|+|\boldsymbol{\psi}\odot(\mathcal{K}\times_i\boldsymbol{\phi}\times_k\boldsymbol{\varphi})-v|\big)$.
 8: **return** $W_\epsilon(u,v,w)=\langle\boldsymbol{\varphi},(C\odot\mathcal{K})\times_i\boldsymbol{\phi}\times_j\boldsymbol{\psi}\rangle$.

Remark 2.1. Theoretically, obtaining approximate solutions with high accuracy requires an adequately small entropic parameter ε, which may cause over- or underflow in numerical calculations during the Sinkhorn iteration. The log-domain stabilization technique [13] can be employed to address this issue. When the infinite norm of ϕ,ψ or ϕ exceeds a given threshold τ, the excessive parts of ϕ,ψ,ϕ are absorbed into $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}$ to avoid the over- and underflow

$$\boldsymbol{\alpha}\ \leftarrow\ \boldsymbol{\alpha}+\varepsilon\ln(\boldsymbol{\phi}),\quad \boldsymbol{\beta}\ \leftarrow\ \boldsymbol{\beta}+\varepsilon\ln(\boldsymbol{\psi}),\quad \boldsymbol{\gamma}\ \leftarrow\ \boldsymbol{\gamma}+\varepsilon\ln(\boldsymbol{\varphi}),$$
$$\boldsymbol{\phi}\ \leftarrow\ \mathbf{1}_N,\qquad\qquad \boldsymbol{\psi}\ \leftarrow\ \mathbf{1}_N,\qquad\qquad \boldsymbol{\varphi}\ \leftarrow\ \mathbf{1}_N.$$

Correspondingly, the tensor $\mathcal{K}$ should also be rescaled as $K_{ijk}\leftarrow e^{(\alpha_i+\beta_j+\gamma_k)/\varepsilon}K_{ijk}$.

3 An algorithm for MMOT with linear complexity

The computational complexity of Algorithm 1 is $\mathcal{O}(N^3)$ for 3-marginal MMOT problems, and it scales to $\mathcal{O}(N^l)$ for the l-marginal case. The bottleneck arises from the repeated tensor-vector products related to $\mathcal{K}$ and $C\odot\mathcal{K}$ in lines 4-8.

Inspired by the fast Sinkhorn algorithms for Wasserstein-1 distance [28, 29], we aim to reduce the computational complexity from $\mathcal{O}(N^l)$ to $\mathcal{O}(N)$. For the sake of simplicity, we discuss the acceleration of Algorithm 1 for 3-marginal case in this section. The implementation can be naturally extended to the l-marginal case for any $l\geq 3$, which will be discussed in Section 4.2. On a uniform mesh with a grid spacing of $h^\dagger$, the cost tensor $\mathcal{C}=(c_{ijk})$ based on the L^1 norm writes

$$c_{ijk}=(|i-j|+|i-k|+|j-k|)h.$$

---

†The discussion can be naturally extended to the case of non-uniform mesh.

Thus, the corresponding kernel tensor $\mathcal{K} = (K_{ijk})$ satisfies

$$K_{ijk} = \lambda^{|i-j|+|i-k|+|j-k|},$$

in which $\lambda = e^{-h/\varepsilon}$.

To eliminate the absolute value operator in $\mathcal{K}$, we categorize the subscripts (i,j,k) according to their orders into 6 sets[‡]

$$
\begin{aligned}
D_1 &= \{(i,j,k) \mid 1 \leq i \leq j \leq k \leq N\}, \quad D_2 = \{(i,j,k) \mid 1 \leq j < i \leq k \leq N\}, \\
D_3 &= \{(i,j,k) \mid 1 \leq i \leq k < j \leq N\}, \quad D_4 = \{(i,j,k) \mid 1 \leq j \leq k < i \leq N\}, \\
D_5 &= \{(i,j,k) \mid 1 \leq k < i \leq j \leq N\}, \quad D_6 = \{(i,j,k) \mid 1 \leq k < j < i \leq N\}.
\end{aligned}
\tag{3.1}
$$

Then, the element of tensor $\mathcal{K}$ has a simplified form of

$$K_{ijk} = \lambda^{a_p i + b_p j + c_p k}, \quad \forall (i,j,k) \in D_p, \quad p = 1,\ldots,6, \tag{3.2}$$

where a_p, b_p, c_p have specific values

$$
\begin{aligned}
a_1 &= -2, & a_2 &= 0, & a_3 &= -2, & a_4 &= 2, & a_5 &= 0, & a_6 &= 2, \\
b_1 &= 0, & b_2 &= -2, & b_3 &= 2, & b_4 &= -2, & b_5 &= 2, & b_6 &= 0, \\
c_1 &= 2, & c_2 &= 2, & c_3 &= 0, & c_4 &= 0, & c_5 &= -2, & c_6 &= -2.
\end{aligned}
\tag{3.3}
$$

Similarly, $\mathcal{C} \odot \mathcal{K}$ satisfies

$$(\mathcal{C} \odot \mathcal{K})_{ijk} = (a_p i + b_p j + c_p k) h \lambda^{a_p i + b_p j + c_p k}, \quad \forall (i,j,k) \in D_p, \quad p = 1,\ldots,6.$$

In this section, we first propose two fast implementations to accelerate the tensor-vector products $\mathcal{K} \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$ in Section 3.1 and $(\mathcal{C} \odot \mathcal{K}) \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$ in Section 3.2, achieving a computation complexity of $\mathcal{O}(N)$. Subsequently, we embed the fast tensor-vector product into the Sinkhorn algorithm, proposing the fast Sinkhorn algorithm with linear complexity for the MMOT problem in Section 3.3.

3.1 Fast tensor-vector product of $\mathcal{K} \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$

The tensor-vector product writes

$$(\mathcal{K} \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi})_k = \sum_{i=1}^{N} \sum_{j=1}^{N} K_{ijk} \phi_i \psi_j = \sum_{p=1}^{6} J_{k,p}, \quad k = 1, 2, \ldots, N, \tag{3.4}$$

in which $J_{k,p}$ are given by

$$J_{k,1} = \sum_{i=1}^{k} \sum_{j=i}^{k} \phi_i \psi_j \lambda^{a_1 i + b_1 j + c_1 k}, \qquad J_{k,2} = \sum_{j=1}^{k-1} \sum_{i=j+1}^{k} \phi_i \psi_j \lambda^{a_2 i + b_2 j + c_2 k}, \tag{3.5a}$$

[‡]For the l-marginal case, we need to categorize subscripts $(i_1, i_2, \cdots, i_l)$. Therefore, $l!$ sets are required.

$$J_{k,3} = \sum_{i=1}^{k} \sum_{j=k+1}^{N} \phi_i \psi_j \lambda^{a_3 i + b_3 j + c_3 k}, \quad J_{k,4} = \sum_{j=1}^{k} \sum_{i=k+1}^{N} \phi_i \psi_j \lambda^{a_4 i + b_4 j + c_4 k}, \tag{3.5b}$$

$$J_{k,5} = \sum_{i=k+1}^{N} \sum_{j=i}^{N} \phi_i \psi_j \lambda^{a_5 i + b_5 j + c_5 k}, \quad J_{k,6} = \sum_{j=k+1}^{N-1} \sum_{i=j+1}^{N} \phi_i \psi_j \lambda^{a_6 i + b_6 j + c_6 k}. \tag{3.5c}$$

Directly computing Eq. (3.4) for all $k = 1, \ldots, N$ takes $\mathcal{O}(N^3)$ operations. In fact, $J_{k,p}$ satisfy the recurrence relations

$$
\begin{cases}
J_{k,1} = \lambda^{c_1} J_{k-1,1} + \psi_k P_{k,1}, \\
J_{1,1} = \phi_1 \psi_1 \lambda^{a_1 + b_1 + c_1},
\end{cases}
\qquad
\begin{cases}
J_{k,2} = \lambda^{c_2} J_{k-1,2} + \phi_k P_{k-1,2}, \\
J_{1,2} = 0,
\end{cases}
$$

$$J_{k,3} = P_{k,3} Q_{k+1,3}, \qquad\qquad J_{k,4} = P_{k,4} Q_{k+1,4}, \tag{3.6}$$

$$
\begin{cases}
J_{k,5} = \lambda^{-c_5} J_{k+1,5} + \phi_{k+1} P_{k+1,5}, \\
J_{N,5} = 0,
\end{cases}
\qquad
\begin{cases}
J_{k,6} = \lambda^{-c_6} J_{k+1,6} + \psi_{k+1} P_{k+2,6}, \\
J_{N,6} = 0, \quad J_{N-1,6} = 0.
\end{cases}
$$

Here $P_{k,p}$ and $Q_{k,p}$ have the forms of

$$P_{k,1} = \lambda^{(b_1+c_1)k} \sum_{i=1}^{k} \phi_i \lambda^{a_1 i}, \qquad P_{k,2} = \lambda^{a_2(k+1)+c_2(k+1)} \sum_{j=1}^{k} \psi_j \lambda^{b_2 j},$$

$$P_{k,3} = \lambda^{(b_3+c_3)k} \sum_{i=1}^{k} \phi_i \lambda^{a_3 i}, \qquad Q_{k+1,3} = \lambda^{-b_3 k} \sum_{j=k+1}^{N} \psi_j \lambda^{b_3 j},$$

$$P_{k,4} = \lambda^{(a_4+c_4)k} \sum_{j=1}^{k} \psi_j \lambda^{b_4 j}, \qquad Q_{k+1,4} = \lambda^{-a_4 k} \sum_{i=k+1}^{N} \phi_i \lambda^{a_4 i}, \tag{3.7}$$

$$P_{k,5} = \lambda^{a_5 k + c_5(k-1)} \sum_{j=k}^{N} \psi_j \lambda^{b_5 j}, \quad P_{k,6} = \lambda^{b_6(k-1)+c_6(k-2)} \sum_{i=k}^{N} \phi_i \lambda^{a_6 i},$$

which can be also computed in a recursive manner

$$
\begin{cases}
P_{k,1} = \lambda^{b_1+c_1} P_{k-1,1} + \phi_k \lambda^{(a_1+b_1+c_1)k}, \\
P_{1,1} = \phi_1 \lambda^{a_1+b_1+c_1},
\end{cases}
\qquad
\begin{cases}
P_{k,2} = \lambda^{a_2+c_2} P_{k-1,2} + \psi_k \lambda^{a_2(k+1)+b_2 k + c_2(k+1)}, \\
P_{1,2} = \psi_1 \lambda^{2a_2+b_2+2c_2},
\end{cases}
$$

$$
\begin{cases}
P_{k,3} = \lambda^{b_3+c_3} P_{k-1,3} + \phi_k \lambda^{(a_3+b_3+c_3)k}, \\
P_{1,3} = \phi_1 \lambda^{a_3+b_3+c_3},
\end{cases}
\qquad
\begin{cases}
Q_{k,3} = \lambda^{b_3} Q_{k+1,3} + \psi_k \lambda^{b_3}, \\
Q_{N,3} = \psi_N \lambda^{b_3},
\end{cases}
$$

$$
\begin{cases}
P_{k,4} = \lambda^{a_4+c_4} P_{k-1,4} + \psi_k \lambda^{(a_4+b_4+c_4)k}, \\
P_{1,4} = \psi_1 \lambda^{a_4+b_4+c_4},
\end{cases}
\qquad
\begin{cases}
Q_{k,4} = \lambda^{a_4} Q_{k+1,4} + \phi_k \lambda^{a_4}, \\
Q_{N,4} = \phi_N \lambda^{a_4},
\end{cases}
\tag{3.8}
$$

$$
\begin{cases}
P_{k,5} = \lambda^{-a_5-c_5} P_{k+1,5} + \psi_k \lambda^{a_5 k + b_5 k + c_5(k-1)}, \\
P_{N,5} = \psi_N \lambda^{a_5 N + b_5 N + c_5(N-1)},
\end{cases}
\qquad
\begin{cases}
P_{k,6} = \lambda^{-b_6-c_6} P_{k+1,6} + \phi_k \lambda^{a_6 k + b_6(k-1) + c_6(k-2)}, \\
P_{N,6} = \phi_N \lambda^{a_6 N + b_6(N-1) + c_6(N-2)}.
\end{cases}
$$

The recurrence formulas (3.6) and (3.8) induce the fast tensor-vector product algorithm (FTVP-1) to compute Eq. (3.4). The pseudo-code is presented in Algorithm 2, with $\mathcal{O}(N)$ computation complexity specified in Table 1.

Algorithm 2. Fast Tensor-Vector Product-1 (FTVP-1).

　　Input: $\boldsymbol{\phi},\boldsymbol{\psi}$ of size $(N,1),\lambda$.

　　Output: $\mathcal{K}\times_i\boldsymbol{\phi}\times_j\boldsymbol{\psi}$.

1: **procedure** FTVP-1$(\boldsymbol{\phi},\boldsymbol{\psi},\lambda)$

2:　　$J_1,J_2,J_3,J_4,J_5,J_6=\mathbf{0}_N,J_{1,1}=\phi_1\psi_1$.

3:　　$P_1,P_2,P_3,P_4,P_5,P_6,Q_3,Q_4=\mathbf{0}_N$.

4:　　$P_{1,1}=\phi_1,P_{1,2}=\lambda^2\psi_1,P_{1,3}=\phi_1,Q_{N,3}=\lambda^2\psi_N$.

5:　　$P_{1,4}=\psi_1,Q_{N,4}=\lambda^2\phi_N,P_{N,5}=\lambda^2\psi_N,P_{N,6}=\lambda^4\phi_N$.

6:　　**for** $k=1:N-1$ **do**

7:　　　　$P_{k+1,1}=\lambda^2P_{k,1}+\phi_{k+1}$.

8:　　　　$P_{k+1,2}=\lambda^2P_{k,2}+\lambda^2\psi_{k+1}$.

9:　　　　$P_{k+1,3}=\lambda^2P_{k,3}+\phi_{k+1}$.

10:　　　　$Q_{N-k,3}=\lambda^2Q_{N-k+1,3}+\lambda^2\psi_{N-k}$.

11:　　　　$P_{k+1,4}=\lambda^2P_{k,4}+\psi_{k+1}$.

12:　　　　$Q_{N-k,4}=\lambda^2Q_{N-k+1,4}+\lambda^2\phi_{N-k}$.

13:　　　　$P_{N-k,5}=\lambda^2P_{N-k+1,5}+\lambda^2\psi_{N-k}$.

14:　　　　$P_{N-k,6}=\lambda^2P_{N-k+1,6}+\lambda^4\phi_{N-k}$.

15:　　**for** $k=1:N-1$ **do**

16:　　　　$J_{k+1,1}=\lambda^2J_{k,1}+\psi_{k+1}P_{k+1,1}$.

17:　　　　$J_{k+1,2}=\lambda^2J_{k,2}+\phi_{k+1}P_{k,2}$.

18:　　　　$J_{k,3}=P_{k,3}Q_{k+1,3}$.

19:　　　　$J_{k,4}=P_{k,4}Q_{k+1,4}$.

20:　　**for** $k=N:-1:2$ **do**

21:　　　　$J_{k-1,5}=\lambda^2J_{k,5}+\phi_kP_{k,5}$.

22:　　**for** $k=N-1:-1:2$ **do**

23:　　　　$J_{k-1,6}=\lambda^2J_{k,6}+\psi_kP_{k+1,6}$.

24: **return** $J_1+J_2+J_3+J_4+J_5+J_6$.

Table 1: The number of multiplicative and additive operations in FTVP-1.

	$P_{k,p}$	$Q_{k,p}$	$J_{k,p}$	J
Number of operations	18N	6N	14N	5N
Total	43N			

3.2   Fast tensor-vector product of $(\mathcal{C} \odot \mathcal{K}) \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$

The tensor-vector product writes

$$((\mathcal{C} \odot \mathcal{K}) \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi})_k = \sum_{i=1}^N \sum_{j=1}^N c_{ijk} K_{ijk} \phi_i \psi_j = h \sum_{p=1}^6 \hat{J}_{k,p}, \tag{3.9}$$

in which $\hat{J}_{k,p}$ for $p=1,2,\ldots,6$ are given by

$$\hat{J}_{k,1} = \sum_{i=1}^k \sum_{j=i}^k (a_1 i + b_1 j + c_1 k) \phi_i \psi_j \lambda^{a_1 i + b_1 j + c_1 k},$$

$$\hat{J}_{k,2} = \sum_{j=1}^{k-1} \sum_{i=j+1}^k (a_2 i + b_2 j + c_2 k) \phi_i \psi_j \lambda^{a_2 i + b_2 j + c_2 k},$$

$$\hat{J}_{k,3} = \sum_{i=1}^k \sum_{j=k+1}^N (a_3 i + b_3 j + c_3 k) \phi_i \psi_j \lambda^{a_3 i + b_3 j + c_3 k},$$

$$\hat{J}_{k,4} = \sum_{j=1}^k \sum_{i=k+1}^N (a_4 i + b_4 j + c_4 k) \phi_i \psi_j \lambda^{a_4 i + b_4 j + c_4 k}, \tag{3.10}$$

$$\hat{J}_{k,5} = \sum_{i=k+1}^N \sum_{j=i}^N (a_5 i + b_5 j + c_5 k) \phi_i \psi_j \lambda^{a_5 i + b_5 j + c_5 k},$$

$$\hat{J}_{k,6} = \sum_{j=k+1}^N \sum_{i=j+1}^N (a_6 i + b_6 j + c_6 k) \phi_i \psi_j \lambda^{a_6 i + b_6 j + c_6 k}.$$

Directly computing Eq. (3.9) for all $k=1,\ldots,N$ takes $\mathcal{O}(N^3)$ operations. In fact, $\hat{J}_{k,p}$ satisfy the recurrence relations

$$
\begin{cases}
\hat{J}_{k,1} = \lambda^{c_1} (\hat{J}_{k-1,1} + c_1 J_{k-1,1}) + \psi_k (R_{k,1} + (b_1 + c_1) k P_{k,1}), \\
\hat{J}_{1,1} = (a_1 + b_1 + c_1) \phi_1 \psi_1 \lambda^{a_1 + b_1 + c_1},
\end{cases}
$$

$$
\begin{cases}
\hat{J}_{k,2} = \lambda^{c_2} (\hat{J}_{k-1,2} + c_2 J_{k-1,2}) + \phi_k (R_{k-1,2} + (a_2 + c_2) k P_{k-1,2}), \\
\hat{J}_{1,2} = 0,
\end{cases}
$$

$$\hat{J}_{k,3} = R_{k,3} Q_{k+1,3} + P_{k,3} S_{k+1,3} + c_3 k J_{k,3}, \tag{3.11}$$

$$\hat{J}_{k,4} = R_{k,4} Q_{k+1,4} + P_{k,4} S_{k+1,4} + c_4 k J_{k,4},$$

$$
\begin{cases}
\hat{J}_{k,5} = \lambda^{-c_5} (\hat{J}_{k+1,5} - c_5 J_{k+1,5}) + \phi_{k+1} (R_{k+1,5} + (a_5 (k+1) + c_5 k) P_{k+1,5}), \\
\hat{J}_{N,5} = 0,
\end{cases}
$$

$$
\begin{cases}
\hat{J}_{k,6} = \lambda^{-c_6} (\hat{J}_{k+1,6} - c_6 J_{k+1,6}) + \psi_{k+1} (R_{k+2,6} + (b_6 (k+1) + c_6 k) P_{k+2,6}), \\
\hat{J}_{N,6} = 0, \quad \hat{J}_{N-1,6} = 0.
\end{cases}
$$

Here $J_{k,p}, P_{k,p}, Q_{k,p}$ are the same as those in Eqs. (3.5) and (3.7). Additionally, $R_{k,p}$ and $S_{k,p}$ have the forms of

$$R_{k,1} = \lambda^{(b_1+c_1)k} \sum_{i=1}^{k} a_1 i \phi_i \lambda^{a_1 i}, \qquad R_{k,2} = \lambda^{(a_2+c_2)(k+1)} \sum_{j=1}^{k} b_2 j \psi_j \lambda^{b_2 j},$$

$$R_{k,3} = \lambda^{(b_3+c_3)k} \sum_{i=1}^{k} a_3 i \phi_i \lambda^{a_3 i}, \qquad S_{k+1,3} = \lambda^{-b_3 k} \sum_{j=k+1}^{N} b_3 j \psi_j \lambda^{b_3 j},$$

$$R_{k,4} = \lambda^{(a_4+c_4)k} \sum_{j=1}^{k} b_4 j \psi_j \lambda^{b_4 i}, \qquad S_{k+1,4} = \lambda^{-a_4 k} \sum_{i=k+1}^{N} a_4 i \phi_i \lambda^{a_4 i}, \qquad (3.12)$$

$$R_{k,5} = \lambda^{a_5 k + c_5(k-1)} \sum_{j=k}^{N} b_5 j \psi_j \lambda^{b_5 j}, \quad R_{k,6} = \lambda^{b_6(k-1)+c_6(k-2)} \sum_{i=k}^{N} a_6 i \phi_i \lambda^{a_6 i},$$

which can also be computed in a recursive manner

$$\begin{cases} R_{k,1} = \lambda^{b_1+c_1} R_{k-1,1} + a_1 k \phi_k \lambda^{(a_1+b_1+c_1)k}, \\ R_{1,1} = a_1 \phi_1 \lambda^{a_1+b_1+c_1}, \end{cases} \qquad \begin{cases} R_{k,2} = \lambda^{a_2+c_2} R_{k-1,2} \\ \qquad + b_2 k \psi_k \lambda^{a_2(k+1)+b_2 k+c_2(k+1)}, \\ R_{1,2} = b_2 \psi_1 \lambda^{2a_2+b_2+2c_2}, \end{cases}$$

$$\begin{cases} R_{k,3} = \lambda^{b_3+c_3} R_{k-1,3} + a_3 k \phi_k \lambda^{(a_3+b_3+c_3)k}, \\ R_{1,3} = a_3 \phi_1 \lambda^{a_3+b_3+c_3}, \end{cases} \qquad \begin{cases} S_{k,3} = \lambda^{b_3} S_{k+1,3} + b_3 k \psi_k \lambda^{b_3}, \\ S_{N,3} = b_3 N \psi_N \lambda^{b_3}, \end{cases} \qquad (3.13)$$

$$\begin{cases} R_{k,4} = \lambda^{a_4+c_4} R_{k-1,4} + b_4 k \psi_k \lambda^{(a_4+b_4+c_4)k}, \\ R_{1,4} = b_4 \psi_1 \lambda^{a_4+b_4+c_4}, \end{cases} \qquad \begin{cases} S_{k,4} = \lambda^{a_4} S_{k+1,4} + a_4 k \phi_k \lambda^{a_4}, \\ S_{N,4} = a_4 N \phi_N \lambda^{a_4}, \end{cases}$$

$$\begin{cases} R_{k,5} = \lambda^{-a_5-c_5} R_{k+1,5} \\ \qquad + b_5 k \psi_k \lambda^{a_5 k+b_5 k+c_5(k-1)}, \\ R_{N,5} = b_5 \psi_N \lambda^{a_5 N+b_5 N+c_5(N-1)}, \end{cases} \qquad \begin{cases} R_{k,6} = \lambda^{-b_6-c_6} R_{k+1,6} \\ \qquad + a_6 k \phi_k \lambda^{a_6 k+b_6(k-1)+c_6(k-2)}, \\ R_{N,6} = a_6 \phi_N \lambda^{a_6 N+b_6(N-1)+c_6(N-2)}. \end{cases}$$

The recurrence formulas (3.11) and (3.13) induce the fast tensor-vector product algorithm (FTVP-2) to compute Eq. (3.9). The pseudo-code is presented in Algorithm 3, with $\mathcal{O}(N)$ computation complexity specified in Table 2.

Table 2: The number of multiplicative and additive operations in FTVP-2.

	$P_{k,p}$	$Q_{k,p}$	$J_{k,p}$	$R_{k,p}$	$S_{k,p}$	$\hat{J}_{k,p}$	$\hat{J}$
Number of operations	$18N$	$6N$	$14N$	$24N$	$8N$	$38N$	$5N$
Total	$113N$						

Algorithm 3. Fast Tensor-Vector Product-2 (FTVP-2).

    Input: $\boldsymbol{\phi},\boldsymbol{\psi}$ of size $(N,1),\lambda,h$.

    Output: $(\mathcal{C}\odot\mathcal{K})\times_i\boldsymbol{\phi}\times_j\boldsymbol{\psi}$.

1: **procedure** FTVP-2$(\boldsymbol{\phi},\boldsymbol{\psi},\lambda)$

2:     $\hat{\boldsymbol{J}}_1,\hat{\boldsymbol{J}}_2,\hat{\boldsymbol{J}}_3,\hat{\boldsymbol{J}}_4,\hat{\boldsymbol{J}}_5,\hat{\boldsymbol{J}}_6=\mathbf{0}_N$.

3:     $\boldsymbol{R}_1,\boldsymbol{R}_2,\boldsymbol{R}_3,\boldsymbol{R}_4,\boldsymbol{R}_5,\boldsymbol{R}_6,\boldsymbol{S}_3,\boldsymbol{S}_4=\mathbf{0}_N$.

4:     $R_{1,1}=-2\phi_1,R_{1,2}=-2\lambda^2\psi_1,R_{1,3}=-2\phi_1,S_{N,3}=2N\lambda^2\psi_N$.

5:     $R_{1,4}=-2\psi_1,S_{N,4}=2N\lambda^2\phi_N,R_{N,5}=2\lambda^2\psi_N,R_{N,6}=2\lambda^4\phi_N$.

6:     **for** $k=1:N-1$ **do**

7:         $R_{k+1,1}=\lambda^2R_{k,1}-2(k+1)\phi_{k+1}$.

8:         $R_{k+1,2}=\lambda^2R_{k,2}-2(k+1)\lambda^2\psi_{k+1}$.

9:         $R_{k+1,3}=\lambda^2R_{k,3}-2(k+1)\phi_{k+1}$.

10:        $S_{N-k,3}=\lambda^2S_{N-k+1,3}+2(N-k)\lambda^2\psi_{N-k}$.

11:        $R_{k+1,4}=\lambda^2R_{k,4}-2(k+1)\psi_{k+1}$.

12:        $S_{N-k,4}=\lambda^2S_{N-k+1,4}+2(N-k)\lambda^2\phi_{N-k}$.

13:        $R_{N-k,5}=\lambda^2R_{N-k+1,5}+2(N-k)\lambda^2\psi_{N-k}$.

14:        $R_{N-k,6}=\lambda^2R_{N-k+1,6}+2(N-k)\lambda^4\phi_{N-k}$.

15:     **for** $k=1:N-1$ **do**

16:        $\hat{J}_{k+1,1}=\lambda^2\hat{J}_{k,1}+2\lambda^2J_{k,1}+\psi_{k+1}R_{k+1,1}+2(k+1)\psi_{k+1}P_{k+1,1}$.

17:        $\hat{J}_{k+1,2}=\lambda^2\hat{J}_{k,2}+2\lambda^2J_{k,2}+\phi_{k+1}R_{k,2}+2(k+1)\phi_{k+1}P_{k,2}$.

18:        $\hat{J}_{k,3}=R_{k,3}Q_{k+1,3}+P_{k,3}S_{k+1,3}$.

19:        $\hat{J}_{k,4}=R_{k,4}Q_{k+1,4}+P_{k,4}S_{k+1,4}$.

20:     **for** $k=N:-1:2$ **do**

21:        $\hat{J}_{k-1,5}=\lambda^2\hat{J}_{k,5}+2\lambda^2J_{k,5}+\phi_kR_{k,5}-2(k-1)\phi_kP_{k,5}$.

22:     **for** $k=N-1:-1:2$ **do**

23:        $\hat{J}_{k-1,6}=\lambda^2\hat{J}_{k,6}+2\lambda^2J_{k,6}+\psi_kR_{k+1,6}-2(k-1)\psi_kP_{k+1,6}$.

24: **return** $h(\hat{\boldsymbol{J}}_1+\hat{\boldsymbol{J}}_2+\hat{\boldsymbol{J}}_3+\hat{\boldsymbol{J}}_4+\hat{\boldsymbol{J}}_5+\hat{\boldsymbol{J}}_6)$.

3.3 Algorithm implementation

The major computational burden of the generalized Sinkhorn algorithm (Algorithm 1) lies in the tensor-vector products related to $\mathcal{K}$ in lines 4-7 and that related to $\mathcal{C}\odot\mathcal{K}$ in line 8. Here we propose a step-by-step approach to reduce the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(N)$:

- $\mathcal{K}\times_j\boldsymbol{\psi}\times_k\boldsymbol{\varphi}$ in lines 4 and 7: We define $\tilde{\mathcal{K}}=(\tilde{K}_{ijk})$ satisfying $\tilde{K}_{ijk}=K_{kij}=\lambda^{b_pi+c_pj+a_pk}$, $\forall(i,j,k)\in D_p$. Then it can be verified that

$$\mathcal{K}\times_j\boldsymbol{\psi}\times_k\boldsymbol{\varphi}=\tilde{\mathcal{K}}\times_i\boldsymbol{\psi}\times_j\boldsymbol{\varphi},$$

which can be computed with $\mathcal{O}(N)$ computational cost according to the discussion in Section 3.1, because $\tilde{\mathcal{K}}$ has the similar form of Eq. (3.2);

- $\mathcal{K} \times_i \boldsymbol{\phi} \times_k \boldsymbol{\varphi}$ in lines 5 and 7: We can apply the same technique as used above to reduce the complexity to $\mathcal{O}(N)$;

- $\mathcal{K} \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$ in line 6: According to Algorithm 2, the complexity can be reduced to $\mathcal{O}(N)$;

- $(\boldsymbol{C} \odot \mathcal{K}) \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$ in line 8: According to Algorithm 3, the complexity can be reduced to $\mathcal{O}(N)$.

Based on the above discussions, we successfully accelerate the generalized Sinkhorn algorithm and propose a fast Sinkhorn algorithm with $\mathcal{O}(N)$ computational complexity. The pseudo-code is presented in Algorithm 4.

Algorithm 4. The Algorithm for MMOT with Linear Complexity.

    Input: $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ of size $(N,1), \varepsilon$, tol, itr_max.
    Output: $W_\epsilon(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$.
1: Initialize $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi} = \mathbf{1}_N / N, t = 0, \text{Res} = \text{Inf}$.
2: **while** $(t < \text{itr\_max})$ & $(\text{Res} > \text{tol})$ **do**
3:     $t \leftarrow t+1$.
4:     $\boldsymbol{\phi} \leftarrow \boldsymbol{u} \oslash \text{FTVP-1}(\boldsymbol{\psi}, \boldsymbol{\varphi}, \lambda)$.
5:     $\boldsymbol{\psi} \leftarrow \boldsymbol{v} \oslash \text{FTVP-1}(\boldsymbol{\phi}, \boldsymbol{\varphi}, \lambda)$.
6:     $\boldsymbol{\varphi} \leftarrow \boldsymbol{w} \oslash \text{FTVP-1}(\boldsymbol{\phi}, \boldsymbol{\psi}, \lambda)$.
7:     $\text{Res} \leftarrow \text{sum}(|\boldsymbol{\phi} \odot \text{FTVP-1}(\boldsymbol{\psi}, \boldsymbol{\varphi}, \lambda) - \boldsymbol{u}| + |\boldsymbol{\psi} \odot \text{FTVP-1}(\boldsymbol{\phi}, \boldsymbol{\varphi}, \lambda) - \boldsymbol{v}|)$.
8: **return** $W_\epsilon(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) = h\langle \boldsymbol{\varphi}, \text{FTVP-2}(\boldsymbol{\phi}, \boldsymbol{\psi}, \lambda)\rangle$.

The log-domain stabilization can also be aggregated into the fast tensor-vector product algorithm. The key point is to accelerate the tensor-vector products related to the rescaled tensor $\hat{\mathcal{K}} = (e^{(\alpha_i + \beta_j + \gamma_k)/\varepsilon} K_{ijk})$, which can be achieved by modifying the update formulas of $P_{k,p}, Q_{k,p}, J_{k,p}$ in FTVP-1. The pseudo-code is presented in Algorithm 5.

4 Various generalization

In this section, we aim to extend the capabilities of our algorithm to higher dimensions and more margins.

4.1 High dimension generalization

In this subsection, we consider the two-dimensional case of our algorithm as an example, noting that its generalization to three or higher dimensions follows a similar ap-

---

Algorithm 5. Fast Tensor-Vector Product with log-Domain Stabilization (FTVP-LOG).

   Input: $\boldsymbol{\phi}, \boldsymbol{\psi}$ of size $(N,1), \lambda$.

   Output: $\hat{\mathcal{K}} \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$.

1: **procedure** FTVP-1$(\boldsymbol{\phi}, \boldsymbol{\psi}, \lambda)$

2:     $J_1, J_2, J_3, J_4, J_5, J_6 = \mathbf{0}_N, J_{1,1} = \phi_1 \psi_1$.

3:     $P_1, P_2, P_3, P_4, P_5, P_6, Q_3, Q_4 = \mathbf{0}_N$.

4:     $P_{1,1} = \phi_1, P_{1,2} = \lambda^2 \psi_1, P_{1,3} = \phi_1, Q_{N,3} = \lambda^2 \psi_N$.

5:     $P_{1,4} = \psi_1, Q_{N,4} = \lambda^2 \phi_N, P_{N,5} = \lambda^2 \psi_N, P_{N,6} = \lambda^4 \phi_N$.

6:     **for** $k = 1 : N-1$ **do**

7:         $P_{k+1,1} = \lambda^2 e^{(\alpha_k - \alpha_{k+1})/\varepsilon} P_{k,1} + \phi_{k+1}$.

8:         $P_{k+1,2} = \lambda^2 e^{(\beta_k - \beta_{k+1})/\varepsilon} P_{k,2} + \lambda^2 \psi_{k+1}$.

9:         $P_{k+1,3} = \lambda^2 e^{(\alpha_k - \alpha_{k+1})/\varepsilon} P_{k,3} + \phi_{k+1}$.

10:        $Q_{N-k,3} = \lambda^2 e^{(\beta_{N-k+1} - \beta_{N-k})/\varepsilon} Q_{N-k+1,4} + \lambda^2 \psi_{N-k}$.

11:        $P_{k+1,4} = \lambda^2 e^{(\beta_k - \beta_{k+1})/\varepsilon} P_{k,2} + \psi_{k+1}$.

12:        $Q_{N-k,4} = \lambda^2 e^{(\alpha_{N-k+1} - \alpha_{N-k})/\varepsilon} Q_{N-k+1,4} + \lambda^2 \phi_{N-k}$.

13:        $P_{N-k,5} = \lambda^2 e^{(\alpha_{N-k+1} - \alpha_{N-k})/\varepsilon} P_{N-k+1,5} + \lambda^2 \psi_{N-k}$.

14:        $P_{N-k,6} = \lambda^2 e^{(\beta_{N-k+1} - \beta_{N-k})/\varepsilon} P_{N-k+1,6} + \lambda^4 \phi_{N-k}$.

15:    **for** $k = 1 : N-1$ **do**

16:        $J_{k+1,1} = \lambda^2 e^{(\gamma_{k+1} - \gamma_k)/\varepsilon} J_{k,1} + \psi_{k+1} e^{(\alpha_{k+1} + \beta_{k+1} + \gamma_{k+1})/\varepsilon} P_{k+1,1}$.

17:        $J_{k+1,2} = \lambda^2 e^{(\gamma_{k+1} - \gamma_k)\varepsilon} J_{k,2} + \phi_{k+1} e^{(\alpha_{k+1} + \beta_k + \gamma_{k+1})\varepsilon} P_{k,2}$.

18:        $J_{k,3} = e^{(\alpha_k + \beta_{k+1} + \gamma_k)/\varepsilon} P_{k,3} Q_{k+1,3}$.

19:        $J_{k,4} = e^{(\alpha_{k+1} + \beta_k + \gamma_k)\varepsilon} P_{k,4} Q_{k+1,4}$.

20:    **for** $k = N : -1 : 2$ **do**

21:        $J_{k-1,5} = \lambda^2 e^{(\gamma_{k-1} - \gamma_k)/\varepsilon} J_{k,5} + \phi_k e^{(\alpha_k + \beta_k + \gamma_{k-1})/\varepsilon} P_{k,5}$.

22:    **for** $k = N-1 : -1 : 2$ **do**

23:        $J_{k-1,6} = \lambda^2 e^{(\gamma_{k-1} - \gamma_k)/\varepsilon} J_{k,6} + \psi_k e^{(\alpha_{k+1} + \beta_k + \gamma_{k-1})/\varepsilon} P_{k+1,6}$.

24: **return** $J_1 + J_2 + J_3 + J_4 + J_5 + J_6$.

---

proach with no fundamental distinction. Similar to Section 2, we consider the discrete two-dimensional MMOT problem, which solves the following minimization problem:

$$W(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) = \inf_{\mathcal{T} \in \Pi} \langle \mathcal{C}, \mathcal{T} \rangle = \inf_{\mathcal{T} \in \Pi} \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{M_1} \sum_{j_1=1}^{N_2} \sum_{j_2=1}^{M_2} \sum_{k_1=1}^{N_3} \sum_{k_2=1}^{M_3} c_{i_1 i_2 j_1 j_2 k_1 k_2} t_{i_1 i_2 j_1 j_2 k_1 k_2}. \qquad (4.1)$$

Here

$$\mathcal{C} = (c_{i_1 i_2 j_1 j_2 k_1 k_2}) \in \mathbb{R}^{N_1 M_1 \times N_2 M_2 \times N_3 M_3}$$

is the cost tensor, and

$$\mathcal{T} = (t_{i_1 i_2 j_1 j_2 k_1 k_2}) \in \mathbb{R}_{0+}^{N_1 M_1 \times N_2 M_2 \times N_3 M_3}$$

is the multi-marginal transport plan, satisfying the linear constraints

$$\mathcal{T} \in \Pi = \left\{ \mathcal{T} = (t_{i_1 i_2 j_1 j_2 k_1 k_2}) \,\middle|\, t_{i_1 i_2 j_1 j_2 k_1 k_2} \geq 0, \sum_{j_1=1}^{N_2} \sum_{j_2=1}^{M_2} \sum_{k_1=1}^{N_3} \sum_{k_2=1}^{M_3} t_{i_1 i_2 j_1 j_2 k_1 k_2} = u_{i_1 i_2}, \right.$$
$$\left. \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{M_1} \sum_{k_1=1}^{N_3} \sum_{k_2=1}^{M_3} t_{i_1 i_2 j_1 j_2 k_1 k_2} = v_{j_1 j_2}, \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{M_1} \sum_{j_1=1}^{N_2} \sum_{j_2=1}^{M_2} t_{i_1 i_2 j_1 j_2 k_1 k_2} = w_{k_1 k_2} \right\}. \quad (4.2)$$

Similar to the discussion in Section 2, our discussion is suitable for any $N_1, M_1, N_2,$ M_2, N_3 and M_3. For the sake of simplicity, we assume $N_1 = N_2 = N_3 = N$ and $M_1 = M_2 = M_3 = M$ in the rest of the paper. Similar to the entropic regularization technique introduced in Section 2, we consider the following regularized MMOT problem:

$$W_\epsilon(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) = \inf_{\mathcal{T} \in \Pi} \sum_{i_1,j_1,k_1=1}^{N} \sum_{i_2,j_2,k_2=1}^{M} c_{i_1 i_2 j_1 j_2 k_1 k_2} t_{i_1 i_2 j_1 j_2 k_1 k_2} + \varepsilon t_{i_1 i_2 j_1 j_2 k_1 k_2} \ln(t_{i_1 i_2 j_1 j_2 k_1 k_2}).$$

Based on analogous derivation, the above problem achieves optimum when scaling variables $\boldsymbol{\phi} = (\phi_{i_1 i_2}), \boldsymbol{\psi} = (\psi_{j_1 j_2}), \boldsymbol{\varphi} = (\varphi_{k_1 k_2}) \in \mathbb{R}^{NM}$ satisfy

$$\boldsymbol{\phi} \odot (\mathcal{K} \times_{j_1 j_2} \boldsymbol{\psi} \times_{k_1 k_2} \boldsymbol{\varphi}) = \boldsymbol{u}, \quad (4.3)$$

$$\boldsymbol{\psi} \odot (\mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi} \times_{k_1 k_2} \boldsymbol{\varphi}) = \boldsymbol{v}, \quad (4.4)$$

$$\boldsymbol{\varphi} \odot (\mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi} \times_{j_1 j_2} \boldsymbol{\psi}) = \boldsymbol{w}, \quad (4.5)$$

in which the tensor-vector products write

$$(\mathcal{K} \times_{j_1 j_2} \boldsymbol{\psi} \times_{k_1 k_2} \boldsymbol{\varphi})_{i_1 i_2} = \sum_{j_1,k_1=1}^{N} \sum_{j_2,k_2=1}^{M} K_{i_1 i_2 j_1 j_2 k_1 k_2} \psi_{j_1 j_2} \varphi_{k_1 k_2}, \quad (4.6)$$

$$(\mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi} \times_{k_1 k_2} \boldsymbol{\varphi})_{j_1 j_2} = \sum_{i_1,k_1=1}^{N} \sum_{i_2,k_2=1}^{M} K_{i_1 i_2 j_1 j_2 k_1 k_2} \phi_{i_1 i_2} \varphi_{k_1 k_2}, \quad (4.7)$$

$$(\mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi} \times_{j_1 j_2} \boldsymbol{\psi})_{k_1 k_2} = \sum_{i_1,j_1=1}^{N} \sum_{i_2,j_2=1}^{M} K_{i_1 i_2 j_1 j_2 k_1 k_2} \phi_{i_1 i_2} \psi_{j_1 j_2}. \quad (4.8)$$

The optimal transport plan is given by

$$t_{i_1 i_2 j_1 j_2 k_1 k_2} = \phi_{i_1 i_2} \psi_{j_1 j_2} \varphi_{k_1 k_2} K_{i_1 i_2 j_1 j_2 k_1 k_2},$$

and the optimal transport distance is

$$W_\epsilon(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) = \sum_{i_1,j_1,k_1=1}^{N} \sum_{i_2,j_2,k_2=1}^{M} c_{i_1 i_2 j_1 j_2 k_1 k_2} \phi_{i_1 i_2} \psi_{j_1 j_2} \varphi_{k_1 k_2} K_{i_1 i_2 j_1 j_2 k_1 k_2}$$
$$= \langle \boldsymbol{\varphi}, (\mathcal{C} \odot \mathcal{K}) \times_{i_1 i_2} \boldsymbol{\phi} \times_{j_1 j_2} \boldsymbol{\psi} \rangle. \quad (4.9)$$

Here, the cost tensor $C = (c_{i_1 i_2 j_1 j_2 k_1 k_2}) \in \mathbb{R}^{NM \times NM \times NM}$ based on the L^1 norm of a uniform 2D mesh with a vertical spacing of h_1 and a horizontal spacing of h_2 writes

$$c_{i_1 i_2 j_1 j_2 k_1 k_2} = (|i_1 - j_1| + |i_1 - k_1| + |j_1 - k_1|)h_1 + (|i_2 - j_2| + |i_2 - k_2| + |j_2 - k_2|)h_2,$$

and the corresponding kernel tensor $\mathcal{K} = (K_{i_1 i_2 j_1 j_2 k_1 k_2}) \in \mathbb{R}^{NM \times NM \times NM}$ writes

$$K_{i_1 i_2 j_1 j_2 k_1 k_2} = \lambda_1^{|i_1 - j_1| + |i_1 - k_1| + |j_1 - k_1|} \lambda_2^{|i_2 - j_2| + |i_2 - k_2| + |j_2 - k_2|},$$

where $\lambda_1 = e^{-h_1/\varepsilon}$ and $\lambda_2 = e^{-h_2/\varepsilon}$. The generalized Sinkhorn algorithm can be applied to calculate $(\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi})$ by iteratively updating

$$\boldsymbol{\phi}^{(t+1)} = \boldsymbol{u} \oslash \left(\mathcal{K} \times_{j_1 j_2} \boldsymbol{\psi}^{(t)} \times_{k_1 k_2} \boldsymbol{\varphi}^{(t)} \right), \tag{4.10}$$

$$\boldsymbol{\psi}^{(t+1)} = \boldsymbol{v} \oslash \left(\mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi}^{(t+1)} \times_{k_1 k_2} \boldsymbol{\varphi}^{(t)} \right), \tag{4.11}$$

$$\boldsymbol{\varphi}^{(t+1)} = \boldsymbol{w} \oslash \left(\mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi}^{(t+1)} \times_{j_1 j_2} \boldsymbol{\psi}^{(t+1)} \right). \tag{4.12}$$

Similar to the discussion of our algorithm for the 1D problem in Section 3, the bottleneck of computational efficiency arises from the tensor-vector products, such as $\boldsymbol{J} = \mathcal{K} \times_{i_1 i_2} \boldsymbol{\phi} \times_{j_1 j_2} \boldsymbol{\psi}$, whose element satisfies

$$J_{k_1 k_2} = \sum_{i_1, j_1 = 1}^{N} \sum_{i_2, j_2 = 1}^{M} \lambda_1^{|i_1 - j_1| + |i_1 - k_1| + |j_1 - k_1|} \lambda_2^{|i_2 - j_2| + |i_2 - k_2| + |j_2 - k_2|} \phi_{i_1 i_2} \psi_{j_1 j_2}. \tag{4.13}$$

Obviously, directly calculating this tensor-vector product requires $\mathcal{O}(N^3 M^3)$ operations, which is an unacceptable computational cost. Now, we delve into a detailed discussion on reducing the computational cost to $\mathcal{O}(NM)$.

First, $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{J} \in \mathbb{R}^{NM}$ are flattened into 1D vectors in column-major order

$$\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_M), \quad \boldsymbol{\psi} = (\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \cdots, \boldsymbol{\psi}_M), \quad \boldsymbol{J} = (\boldsymbol{J}_1, \boldsymbol{J}_2, \cdots, \boldsymbol{J}_M),$$

in which the sub-vectors $\boldsymbol{\phi}_{i_2}, \boldsymbol{\psi}_{j_2}, \boldsymbol{J}_{k_2} \in \mathbb{R}^N$ are

$$\boldsymbol{\phi}_{i_2} = (\phi_{1 i_2}, \phi_{2 i_2}, \cdots, \phi_{N i_2}), \quad \boldsymbol{\psi}_{j_2} = (\psi_{1 j_2}, \psi_{2 j_2}, \cdots, \psi_{N j_2}), \quad \boldsymbol{J}_{k_2} = (J_{1 k_2}, J_{2 k_2}, \cdots, J_{N k_2}).$$

By defining a tensor $\mathcal{K}_N = (K_{i_1 j_1 k_1, N}) \in \mathbb{R}^{N \times N \times N}$ with the element

$$K_{i_1 j_1 k_1, N} = \lambda_1^{|i_1 - j_1| + |i_1 - k_1| + |j_1 - k_1|},$$

the Eq. (4.13) can be rewritten in vector form as

$$J_{k_2} = \sum_{i_2 = 1}^{M} \sum_{j_2 = 1}^{M} \lambda_2^{|i_2 - j_2| + |i_2 - k_2| + |j_2 - k_2|} \left(\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2} \right).$$

We use the same technique in Section 3 to eliminate the absolute value operation above, separating the summations into 6 components according to the order of subscripts

$$J_{k_2} = \sum_{p=1}^{6} J_{k_2,p},$$

each of which has a simple form of

$$J_{k_2,1} = \sum_{i_2=1}^{k_2} \sum_{j_2=i_2}^{k_2} \lambda_2^{2(k_2-i_2)} (\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2}),$$

$$J_{k_2,2} = \sum_{j_2=1}^{k_2-1} \sum_{i_2=j_2+1}^{k_2} \lambda_2^{2(k_2-j_2)} (\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2}),$$

$$J_{k_2,3} = \sum_{i_2=1}^{k_2} \sum_{j_2=k_2+1}^{M} \lambda_2^{2(j_2-i_2)} (\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2}),$$

$$J_{k_2,4} = \sum_{j_2=1}^{k_2} \sum_{i_2=k_2+1}^{M} \lambda_2^{2(i_2-j_2)} (\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2}),$$ 　(4.14)

$$J_{k_2,5} = \sum_{i_2=k_2+1}^{M} \sum_{j_2=i_2}^{M} \lambda_2^{2(j_2-k_2)} (\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2}),$$

$$J_{k_2,6} = \sum_{j_2=k_2+1}^{M-1} \sum_{i_2=j_2+1}^{M} \lambda_2^{2(i_2-k_2)} (\mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{i_2} \times_{j_1} \boldsymbol{\psi}_{j_2}).$$

In fact, $J_{k_2,p}$ satisfy the recurrence relations

$$\begin{cases} J_{k_2,1} = \lambda_2^2 J_{k_2-1,1} + \mathcal{K}_N \times_{i_1} \boldsymbol{P}_{k_2,1} \times_{j_1} \boldsymbol{\psi}_{k_2}, \\ J_{1,1} = \mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_1 \times_{j_1} \boldsymbol{\psi}_1, \end{cases} \quad \begin{cases} J_{k_2,2} = \lambda_2^2 J_{k_2-1,2} + \mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{k_2} \times_{j_1} \boldsymbol{P}_{k_2-1,2}, \\ J_{1,2} = \boldsymbol{0}, \end{cases}$$

$$J_{k_2,3} = \mathcal{K}_N \times_{i_1} \boldsymbol{P}_{k_2,3} \times_{j_1} \boldsymbol{Q}_{k_2+1,3}, \qquad J_{k_2,4} = \mathcal{K}_N \times_{i_1} \boldsymbol{Q}_{k_2+1,4} \times_{j_1} \boldsymbol{P}_{k_2,4},$$

$$\begin{cases} J_{k_2-1,5} = \lambda_2^2 J_{k_2,5} + \mathcal{K}_N \times_{i_1} \boldsymbol{\phi}_{k_2} \times_{j_1} \boldsymbol{P}_{k_2,5}, \\ J_{M,5} = \boldsymbol{0}, \end{cases} \quad \begin{cases} J_{k_2-1,6} = \lambda_2^2 J_{k_2,6} + \mathcal{K}_N \times_{i_1} \boldsymbol{P}_{k_2+1,6} \times_{j_1} \boldsymbol{\psi}_{k_2}, \\ J_{M,6} = \boldsymbol{0}, J_{M-1,6} = \boldsymbol{0}. \end{cases}$$

Here $\boldsymbol{P}_{k_2,p}, \boldsymbol{Q}_{k_2,p} \in \mathbb{R}^N$ have the forms of

$$\boldsymbol{P}_{k_2,1} = \sum_{i_2=1}^{k_2} \lambda_2^{2(k_2-i_2)} \boldsymbol{\phi}_{i_2}, \qquad \boldsymbol{P}_{k_2,2} = \sum_{j_2=1}^{k_2} \lambda_2^{2(k_2-j_2+1)} \boldsymbol{\psi}_{j_2},$$

$$\boldsymbol{P}_{k_2,3} = \sum_{i_2=1}^{k_2} \lambda_2^{2(k_2-i_2)} \boldsymbol{\phi}_{i_2}, \qquad \boldsymbol{Q}_{k_2,3} = \sum_{j_2=k_2}^{M} \lambda_2^{2(j_2-k_2+1)} \boldsymbol{\psi}_{j_2},$$

$$P_{k_2,4} = \sum_{j_2=1}^{k_2} \lambda_2^{2(k_2-j_2)} \psi_{j_2}, \qquad Q_{k_2,4} = \sum_{i_2=k_2}^{M} \lambda_2^{2(i_2-k_2+1)} \phi_{i_2}$$

$$P_{k_2,5} = \sum_{j_2=k_2}^{M} \lambda_2^{2(j_2-k_2+1)} \psi_{j_2}, \quad P_{k_2,6} = \sum_{i_2=k_2}^{M} \lambda_2^{2(i_2-k_2+2)} \phi_{i_2},$$

which can also be computed in a recursive manner

$$\begin{cases} P_{k_2,1} = \lambda_2^2 P_{k_2-1,1} + \phi_{k_2}, \\ P_{1,1} = \phi_1, \end{cases} \quad \begin{cases} P_{k_2,2} = \lambda_2^2 P_{k_2-1,2} + \lambda_2^2 \psi_{k_2}, \\ P_{1,2} = \lambda_2^2 \psi_1, \end{cases}$$

$$\begin{cases} P_{k_2,3} = \lambda_2^2 P_{k_2-1,3} + \phi_{k_2}, \\ P_{1,3} = \phi_1, \end{cases} \quad \begin{cases} Q_{k_2,3} = \lambda_2^2 Q_{k_2+1,3} + \lambda_2^2 \psi_{k_2}, \\ Q_{M,3} = \lambda_2^2 \psi_M, \end{cases}$$

$$\begin{cases} P_{k_2,4} = \lambda_2^2 P_{k_2-1,4} + \psi_{k_2}, \\ P_{1,4} = \psi_1, \end{cases} \quad \begin{cases} Q_{k_2,4} = \lambda_2^2 Q_{k_2+1,4} + \lambda_2^2 \phi_{k_2}, \\ Q_{M,4} = \lambda_2^2 \phi_M, \end{cases}$$

$$\begin{cases} P_{k_2,5} = \lambda_2^2 P_{k_2+1,5} + \lambda_2^2 \psi_{k_2}, \\ P_{M,5} = \lambda_2^2 \psi_M, \end{cases} \quad \begin{cases} P_{k_2,6} = \lambda_2^2 P_{k_2+1,6} + \lambda_2^4 \phi_{k_2}, \\ P_{M,6} = \lambda_2^4 \phi_M. \end{cases}$$

The pseudo-code of the implementation above is presented in Algorithm 6, with $\mathcal{O}(MN)$ computational complexity. Similar idea can be naturally applied to accelerate the tensor-vector product $(\mathcal{C} \odot \mathcal{K}) \times_{i_1 i_2} \phi \times_{j_1 j_2} \psi$ in Eq. (4.9), and so are omitted.

   The above technique can be naturally extended to the d-dimensional case. Next, we discuss the number of addition and multiplication operations for d dimensional MMOT problem. Let the number of additions required in d-dimensional space be denoted as NA_d, and the number of multiplications as NM_d. Without loss of generality, assuming that the grid size N is the same across all dimensions, we obtain the following recurrence relation:

$$\text{NA}_d = (6\text{NA}_{d-1} + 17)N, \quad \text{NA}_1 = 17N,$$
$$\text{NM}_d = (6\text{NM}_{d-1} + 20)N, \quad \text{NM}_1 = 26N,$$

   From this, we can further derive the expression for the number of operations in an arbitrary d-dimensional space, as summarized in Table 3. Meanwhile, the above discussion applies analogously to the tensor product $(\mathcal{C} \odot \mathcal{K}) \times_{i_1 i_2} \phi \times_{j_1 j_2} \psi$ summarized in Table 4 as well.

4.2 l-marginal generalization

In this subsection, we extend our algorithm to solve general l-marginal optimal transport problems. For ease of exposition, we focus on one-dimensional probabilistic distribution on a uniform mesh with a grid spacing of h. Its generalization to high-dimensional distributions follows an analogous approach as discussed in Section 4.1.

Algorithm 6. Fast Tensor-Vector Product for 2D MMOT.

> **Input:** $\boldsymbol{\phi}, \boldsymbol{\psi}$ of size $(N \times M), \lambda_1, \lambda_2$.
> **Output:** $\mathcal{K} \times_i \boldsymbol{\phi} \times_j \boldsymbol{\psi}$.

1: **procedure** FTVP_2D$(\boldsymbol{\phi}, \boldsymbol{\psi}, \lambda_1, \lambda_2)$
2:     $J_{1,1} = \text{FTVP-1}(\boldsymbol{\phi}_1, \boldsymbol{\psi}_1, \lambda_1), J_{1,2}, J_{1,3}, J_{1,4}, J_{1,5}, J_{1,6} = \mathbf{0}_N$.
3:     $P_{1,1} = \boldsymbol{\phi}_1, P_{1,2} = \lambda_2^2 \boldsymbol{\psi}_1, P_{1,3} = \boldsymbol{\phi}_1, Q_{M,3} = \lambda_2^2 \boldsymbol{\psi}_M$.
4:     $P_{1,4} = \boldsymbol{\psi}_1, Q_{M,4} = \lambda_2^2 \boldsymbol{\phi}_M, P_{M,5} = \lambda_2^2 \boldsymbol{\psi}_M, P_{M,6} = \lambda_2^4 \boldsymbol{\phi}_M$.
5:     **for** $k = 1 : M-1$ **do**
6:         $P_{k+1,1} = \lambda_2^2 P_{k,1} + \boldsymbol{\phi}_{k+1}$.
7:         $P_{k+1,2} = \lambda_2^2 P_{k,2} + \lambda_2^2 \boldsymbol{\psi}_{k+1}$.
8:         $P_{k+1,3} = \lambda_2^2 P_{k,3} + \boldsymbol{\phi}_{k+1}$.
9:         $Q_{M-k,3} = \lambda_2^2 Q_{M-k+1,3} + \lambda_2^2 \boldsymbol{\psi}_{M-k}$.
10:        $P_{k+1,4} = \lambda_2^2 P_{k,4} + \boldsymbol{\psi}_{k+1}$.
11:        $Q_{M-k,4} = \lambda_2^2 Q_{M-k+1,4} + \lambda_2^2 \boldsymbol{\phi}_{M-k}$.
12:        $P_{M-k,5} = \lambda_2^2 P_{M-k+1,5} + \lambda_2^2 \boldsymbol{\psi}_{M-k}$.
13:        $P_{M-k,6} = \lambda_2^2 P_{M-k+1,6} + \lambda_2^4 \boldsymbol{\phi}_{M-k}$.
14:     **for** $k = 1 : M-1$ **do**
15:        $J_{k+1,1} = \lambda_2^2 J_{k,1} + \text{FTVP-1}(P_{k+1,1}, \boldsymbol{\psi}_{k+1}, \lambda_1)$.
16:        $J_{k+1,2} = \lambda_2^2 J_{k,2} + \text{FTVP-1}(\boldsymbol{\phi}_{k+1}, P_{k,2}, \lambda_1)$.
17:        $J_{k,3} = \text{FTVP-1}(P_{k,3}, Q_{k+1,3}, \lambda_1)$.
18:        $J_{k,4} = \text{FTVP-1}(Q_{k+1,4}, P_{k,4}, \lambda_1)$.
19:     **for** $k = M : -1 : 2$ **do**
20:        $J_{k-1,5} = \lambda_2^2 J_{k,5} + \text{FTVP-1}(\boldsymbol{\phi}_k, P_{k,5}, \lambda_1)$.
21:     **for** $k = M-1 : -1 : 2$ **do**
22:        $J_{k-1,6} = \lambda_2^2 J_{k,6} + \text{FTVP-1}(P_{k+1,6}, \boldsymbol{\psi}_k, \lambda_1)$.
23:     **for** $k = 1 : M$ **do**
24:        $J_k = J_{k,1} + J_{k,2} + J_{k,3} + J_{k,4} + J_{k,5} + J_{k,6}$.
25: **return** $(J_1, J_2, \cdots, J_M)$

Table 3: The number of additive (NA) and multiplicative (NM) operations required for $\mathcal{K} \times_{i_1 i_2 \cdots i_d} \boldsymbol{\phi} \times_{j_1 j_2 \cdots j_d} \boldsymbol{\psi}$.

Dimension d	NA	NM	Total
$d = 1$	$17N$	$26N$	$43N$
$d = 2$	$119N^2$	$176N^2$	$295N^2$
d	$\left(\dfrac{17}{5} \times 6^d - \dfrac{17}{5}\right) N^d$	$(5 \times 6^d - 4) N^d$	$\left(\dfrac{42}{5} \times 6^d - \dfrac{37}{5}\right) N^d$

Table 4: The number of additive and multiplicative operations required for $(\mathcal{C} \odot \mathcal{K}) \times_{i_1 i_2 \cdots i_d} \boldsymbol{\phi} \times_{j_1 j_2 \cdots j_d} \boldsymbol{\psi}$.

Dimension d	NA	NM	Total
$d=1$	$41N$	$72N$	$113N$
$d=2$	$615N^2$	$1066N^2$	$1681N^2$
d	$\left(\dfrac{41}{13} \times 14^d - \dfrac{41}{13}\right) N^d$	$\left(\dfrac{71}{13} \times 14^d - \dfrac{58}{13}\right) N^d$	$\left(\dfrac{112}{13} \times 14^d - \dfrac{99}{13}\right) N^d$

Given l discrete distributions $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_l$, where $\boldsymbol{u}_j = (u_{j,1}, u_{j,2}, \cdots, u_{j,N_j}) \in \mathbb{R}^{N_j}$, the discrete l-marginal optimal transport problem [6] is formulated as

$$W(\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_l) = \inf_{\mathcal{T} \in \Pi} \langle \mathcal{C}, \mathcal{T} \rangle = \inf_{\mathcal{T} \in \Pi} \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_l=1}^{N_l} c_{i_1 i_2 \cdots i_l} t_{i_1 i_2 \cdots i_l}. \tag{4.15}$$

Here, $\mathcal{C} = (c_{i_1 i_2 \cdots i_l}) \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_l}$ is the l-th order cost tensor, and the L_1 norm based pairwise-interaction cost is

$$c_{i_1 i_2 \cdots i_l} = \sum_{1 \leq p < q \leq l} |i_p - i_q| h.$$

The collection of all admissible multi-marginal transport plan $\mathcal{T} = (t_{i_1 i_2 \cdots i_l}) \in \mathbb{R}_{0+}^{N_1 \times N_2 \times \cdots \times N_l}$ is given by

$$\Pi = \left\{ \mathcal{T} = (t_{i_1 i_2 \cdots i_l}) \,|\, t_{i_1 i_2 \cdots i_l} \geq 0, \sum_{i_1=1}^{N_1} \cdots \sum_{i_{j-1}=1}^{N_{j-1}} \sum_{i_{j+1}=1}^{N_{j+1}} \cdots \sum_{i_l=1}^{N_l} t_{i_1 i_2 \cdots i_l} = u_{j,i_j}, \forall j = 1, 2, \ldots, l \right\}. \tag{4.16}$$

Without loss of generality, we assume $N_j = N$ for all $j = 1, 2, \ldots, l$. By introducing the entropic regularization technique, the optimal transport plan $\mathcal{T} = (t_{i_1 i_2 \cdots i_l})$ of the regularized l-marginal optimal transport problem

$$W_\varepsilon(\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_l) = \inf_{\mathcal{T} \in \Pi} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_l=1}^{N} c_{i_1 i_2 \cdots i_l} t_{i_1 i_2 \cdots i_l} + \varepsilon t_{i_1 i_2 \cdots i_l} \ln(t_{i_1 i_2 \cdots i_l}),$$

writes

$$t_{i_1 i_2 \cdots i_l} = \phi_{1,i_1} \phi_{2,i_2} \cdots \phi_{l,i_l} K_{i_1 i_2 \cdots i_l}, \quad \forall 1 \leq i_1, i_2, \cdots, i_l \leq N. \tag{4.17}$$

Here, $\boldsymbol{\phi}_l = (\phi_{l,i_l}) \in \mathbb{R}_{0+}^N$ denotes the l-th scaling variables, and the corresponding kernel tensor $\mathcal{K} = (K_{i_1 i_2 \cdots i_l}) \in \mathbb{R}^{N^l}$ is defined as

$$K_{i_1 i_2 \cdots i_l} = \lambda^{\sum_{1 \leq p < q \leq l} |i_p - i_q|}, \quad \lambda = e^{-\frac{h}{\varepsilon}}.$$

By substituting (4.17) into the marginal constraint (4.16), the optimality condition (2.5) for the 3-marginal optimal transport problem can be analogously extended to general l-marginal cases, which is

$$\boldsymbol{\phi}_j \odot (\mathcal{K} \times_{i_1} \boldsymbol{\phi}_1 \cdots \times_{i_{j-1}} \boldsymbol{\phi}_{j-1} \times_{i_{j+1}} \boldsymbol{\phi}_{j+1} \cdots \times_{i_l} \boldsymbol{\phi}_l) = \boldsymbol{u}_j, \quad j = 1, 2, \ldots, l. \tag{4.18}$$

The element of the above tensor-vector product has the form of

$$(\mathcal{K} \times_{i_1} \boldsymbol{\phi}_1 \cdots \times_{i_{j-1}} \boldsymbol{\phi}_{j-1} \times_{i_{j+1}} \boldsymbol{\phi}_{j+1} \cdots \times_{i_l} \boldsymbol{\phi}_l)_{i_j}$$

$$= \sum_{i_1=1}^{N} \cdots \sum_{i_{j-1}=1}^{N} \sum_{i_{j+1}=1}^{N} \cdots \sum_{i_l=1}^{N} K_{i_1 i_2 \cdots i_l} \phi_{1,i_1} \cdots \phi_{j-1,i_{j-1}} \phi_{j+1,i_{j+1}} \cdots \phi_{l,i_l}. \tag{4.19}$$

Hence, the corresponding Sinkhorn iteration scheme to find $\boldsymbol{\phi}_j$ writes

$$\boldsymbol{\phi}_1^{(t+1)} = \boldsymbol{u}_1 \oslash \left(\mathcal{K} \times_{i_2} \boldsymbol{\phi}_2^{(t)} \times_{i_3} \boldsymbol{\phi}_3^{(t)} \cdots \times_{i_l} \boldsymbol{\phi}_l^{(t)}\right), \tag{4.20}$$

$$\boldsymbol{\phi}_2^{(t+1)} = \boldsymbol{u}_2 \oslash \left(\mathcal{K} \times_{i_1} \boldsymbol{\phi}_1^{(t+1)} \times_{i_3} \boldsymbol{\phi}_3^{(t)} \cdots \times_{i_l} \boldsymbol{\phi}_l^{(t)}\right), \tag{4.21}$$

$$\vdots$$

$$\boldsymbol{\phi}_l^{(t+1)} = \boldsymbol{u}_l \oslash \left(\mathcal{K} \times_{i_1} \boldsymbol{\phi}_1^{(t+1)} \times_{i_2} \boldsymbol{\phi}_2^{(t+1)} \cdots \times_{i_{l-1}} \boldsymbol{\phi}_{l-1}^{(t+1)}\right), \tag{4.22}$$

and the l-marginal optimal transport distance is given by

$$W_\epsilon(\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_l) = \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_l=1}^{N} c_{i_1 i_2 \cdots i_l} \phi_{1,i_1} \phi_{2,i_2} \cdots \phi_{l,i_l} K_{i_1 i_2 \cdots i_l}$$

$$= \langle \boldsymbol{\phi}_l, (\mathcal{C} \odot \mathcal{K}) \times_{i_1} \boldsymbol{\phi}_1 \times_{i_2} \boldsymbol{\phi}_2 \cdots \times_{i_{l-1}} \boldsymbol{\phi}_{l-1} \rangle. \tag{4.23}$$

Similar to the discussion of $l = 3$ in Section 3, the bottleneck of computational efficiency for the l-marginal optimal transport problem lies in the tensor-vector products

$$\boldsymbol{J}^{(l)} = \mathcal{K} \times_{i_1} \boldsymbol{\phi}_1 \times_{i_2} \boldsymbol{\phi}_2 \cdots \times_{i_{l-1}} \boldsymbol{\phi}_{l-1}, \tag{4.24}$$

$$\hat{\boldsymbol{J}}^{(l)} = (\mathcal{C} \odot \mathcal{K}) \times_{i_1} \boldsymbol{\phi}_1 \times_{i_2} \boldsymbol{\phi}_2 \cdots \times_{i_{l-1}} \boldsymbol{\phi}_{l-1}, \tag{4.25}$$

where the superscript (l) indicates the l-marginals. The elements of $\boldsymbol{J}^{(l)} = (J_{i_l}^{(l)})$ and $\hat{\boldsymbol{J}}^{(l)} = (\hat{J}_{i_l}^{(l)})$ have the forms of

$$J_{i_l}^{(l)} = \sum_{i_1=1}^{N} \cdots \sum_{i_{l-1}=1}^{N} \lambda^{\sum_{1 \le p < q \le l} |i_p - i_q|} \prod_{q=1}^{l-1} \phi_{q,i_q}, \tag{4.26}$$

and

$$\hat{J}_{i_l}^{(l)} = h \sum_{i_1=1}^{N} \cdots \sum_{i_{l-1}=1}^{N} \left(\sum_{1 \le p < q \le l} |i_p - i_q| \right) \lambda^{\sum_{1 \le p < q \le l} |i_p - i_q|} \prod_{q=1}^{l-1} \phi_{q,i_q}. \tag{4.27}$$

Obviously, the direct calculations require $\mathcal{O}(N^l)$ operations, which are unacceptable computational costs. We show as follows how to accelerate the tensor-vector products to linear computational complexity relative to support size N.

To start, we eliminate the absolute value operation in Eqs. (4.26) and (4.27) by separating the summations into $l!$ components according to the order of subscripts

$$J^{(l)} = \sum_{p=1}^{l!} J_p^{(l)}, \quad \hat{J}^{(l)} = h \sum_{p=1}^{l!} \hat{J}_p^{(l)}, \tag{4.28}$$

in which the elements of $J_p^{(l)} = (J_{i_l,p}^{(l)})$ and $\hat{J}_p^{(l)} = (\hat{J}_{i_l,p}^{(l)})$ write

$$J_{i_l,p}^{(l)} = \sum_{(i_1,\cdots,i_l) \in D_p} \left(\lambda^{\alpha_{k,p} i_l} \prod_{q=1}^{l-1} \phi_{q,i_q} \lambda^{\alpha_{q,p} i_q} \right), \tag{4.29}$$

and

$$\hat{J}_{i_l,p}^{(l)} = \sum_{(i_1,\cdots,i_l) \in D_p} \left(a_{l,p} i_l + \sum_{q=1}^{l-1} a_{q,p} i_q \right) \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=1}^{l-1} \phi_{q,i_q} \lambda^{\alpha_{q,p} i_q} \right). \tag{4.30}$$

Here, D_p are the sets defined by the order of subscripts

$$\begin{aligned}
D_1 &= \{(i_1,\cdots,i_l) \mid 1 \le i_1 \le i_2 \le \cdots \le i_l \le N\}, \\
D_2 &= \{(i_1,\cdots,i_l) \mid 1 \le i_2 < i_1 \le \cdots \le i_l \le N\}, \\
&\vdots \\
D_{l!} &= \{(i_1,\cdots,i_l) \mid 1 \le i_l < i_{l-1} < \cdots < i_1 \le N\}.
\end{aligned} \tag{4.31}$$

We first consider the calculation of $J_p^{(l)} = (J_{i_l,p}^{(l)})$ and $\hat{J}_p^{(l)} = (\hat{J}_{i_l,p}^{(l)})$ for the case of $p=1$. It is straightforward to show that

$$J_{i_l,1}^{(l)} = \sum_{1 \le i_1 \le \cdots \le i_{l-1} \le i_l} \left(\lambda^{\alpha_{l,1} i_l} \prod_{q=1}^{l-1} \phi_{q,i_q} \lambda^{\alpha_{q,1} i_q} \right) = \lambda^{\alpha_{l,1}} J_{i_l-1,1}^{(l)} + \phi_{l-1,i_l} J_{i_l,1}^{(l-1)}, \tag{4.32}$$

in which

$$J_{i_l,1}^{(l-1)} = \sum_{1 \le i_1 \le \cdots \le i_{l-2} \le i_l} \left(\lambda^{(\alpha_{l-1,1}+\alpha_{l,1}) i_l} \prod_{q=1}^{l-2} \phi_{q,i_q} \lambda^{\alpha_{q,1} i_q} \right).$$

The above formula suggests that to compute $J_1^{(l)} = (J_{i_l,1}^{(l)})$ in the l-marginal case, only $2N$ multiplications and N additions are required, provided that $J_1^{(l-1)} = (J_{i_{l-1},1}^{(l-1)})$ has been obtained in the $(l-1)$-marginal case. This recursive process with respect to l can be iterated until reaching $J_1^{(3)} = (J_{i_3,1}^{(3)})$ in the 3-marginal case, enabling $J_1^{(l)}$ to be computed with $\mathcal{O}(N)$ operations.

Similarly, $\hat{J}^{(l)}_{i_l,1}$ satisfy

$$
\begin{aligned}
\hat{J}^{(l)}_{i_l,1} &= \sum_{1\le i_1\le\cdots\le i_{l-1}\le i_l}\left(\alpha_{l,1}i_l+\sum_{q=1}^{l-1}\alpha_{q,1}i_q\right)\left(\lambda^{\alpha_{l,1}i_l}\prod_{q=1}^{l-1}\phi_{q,i_q}\lambda^{\alpha_{q,1}i_q}\right)\\
&= \lambda^{\alpha_{l,1}}\hat{J}^{(l)}_{i_l-1,1}+\phi_{l-1,i_l}\hat{J}^{(l-1)}_{i_l,1}+\alpha_{l,1}\lambda^{\alpha_{l,1}}J^{(l)}_{i_l-1,1}+\alpha_{l,1}\phi_{l-1,i_l}J^{(l-1)}_{i_l,1}\\
&= \lambda^{\alpha_{l,1}}\left(\hat{J}^{(l)}_{i_l-1,1}+\alpha_{l,1}J^{(l)}_{i_l-1,1}\right)+\phi_{l-1,i_l}\left(\hat{J}^{(l-1)}_{i_l,1}+\alpha_{l,1}J^{(l-1)}_{i_l,1}\right),
\end{aligned}
\tag{4.33}
$$

in which

$$
\hat{J}^{(l-1)}_{i_l,1}=\sum_{1\le i_1\le\cdots\le i_{l-2}\le i_l}\left(\alpha_{l-1,1}i_l+\sum_{q=1}^{l-2}\alpha_{q,1}i_q\right)\left(\lambda^{(\alpha_{l-1,1}+\alpha_{l,1})i_l}\prod_{q=1}^{l-2}\phi_{q,i_q}\lambda^{\alpha_{q,1}i_q}\right).
$$

Note that, in (4.33), both of the computational complexity of $J^{(l)}_1$ and $J^{(l-1)}_1$ are $\mathcal{O}(N)$. Therefore, to compute $\hat{\boldsymbol{J}}^{(l)}_1=(\hat{J}^{(l)}_{i_l,1})$ in the l-marginal case, only $3N$ additions and $4N$ multiplications are required if $\hat{\boldsymbol{J}}^{(l-1)}_1=(\hat{J}^{(l-1)}_{i_{l-1},1})$ in the $(l-1)$-marginal case has been obtained. This recursive process with respect to l can be iterated until reaching $\hat{\boldsymbol{J}}^{(3)}_1=(\hat{J}^{(3)}_{i_3,1})$ in the 3-marginal case, thereby enabling the calculation of $\hat{\boldsymbol{J}}^{(l)}_1$ with $\mathcal{O}(N)$ operations.

Similar recursive processes of $\mathcal{O}(N)$ cost can be straightforwardly applied to the calculation of other $J^{(l)}_p$ and $\hat{J}^{(l)}_p$ and hence are not included here. So far, the tensor-vector products (4.24) and (4.25) can both be calculated with linear computational complexity relative to support size N.

4.2.1 Addition and multiplication counts

In this section, we count the addition and multiplication operations of the fast tensor-vector products for the l-marginal case. We consider the calculation of $\boldsymbol{J}^{(l)}$ and $\hat{\boldsymbol{J}}^{(l)}$, which involves the calculation of all $J^{(l)}_p$ (4.29), $\hat{J}^{(l)}_p$ (4.30), and the summations (4.28). Since $p=1,\dots,l!$, the summation requires $(l!-1)N$ additions. Depending on the value of i_l in D_p, there are three situations:

- **Situation 1**: D_p in which i_l is the maximum among $i_1,\cdots,i_l$, given by a general form of

$$
D_p=\{(i_1,\cdots,i_l)\,|\,1\le i_{\sigma_1}\le\cdots\le i_{\sigma_{l-1}}\le i_l\le N\},
$$

for instance D_1. Here $(\sigma_1,\sigma_2,\cdots,\sigma_{l-1})$ is a permutation of $(1,2,\dots,l-1)$, and $\le$ may be replaced by $<$ depending on the definition (4.31). Then $J^{(l)}_p$ has a general form of

$$
J^{(l)}_{i_l,p}=\sum_{1\le i_{\sigma_1}\le\cdots\le i_{\sigma_{l-1}}\le i_l}\left(\lambda^{\alpha_{l,p}i_l}\prod_{q=1}^{l-1}\phi_{\sigma_q,i_{\sigma_q}}\lambda^{\alpha_{\sigma_q,p}i_{\sigma_q}}\right).
\tag{4.34}
$$

By using D_1 as an example, Eq. (4.32) indicates that only N additions and $2N$ multiplications are required to compute $J_1^{(l)} = (J_{i_l,1}^{(l)})$ in the l-marginal case, provided that $J_1^{(l-1)} = (J_{i_{l-1},1}^{(l-1)})$ has been obtained in the $(l-1)$-marginal case. It means that, in this situation, $J_{i_l,p}^{(l)}$ can be computed with $(l-1)N$ additions and $(2l-2)N$ multiplications. Since the permutation $(\sigma_1, \sigma_2, \cdots, \sigma_{l-1})$ has $(l-1)!$ arrangements, totally it requires $(l-1)!(l-1)N$ additions and $(l-1)!(2l-2)N$ multiplications to calculate all $J_p^{(l)}$ in this situation.

Additionally, $\hat{J}_p$ has a general form of

$$\hat{J}_{i_l,p}^{(l)} = \sum_{1 \leq i_{\sigma_1} \leq \cdots \leq i_{\sigma_{l-1}} \leq i_l} \left(\alpha_{l,p} i_l + \sum_{q=1}^{l-1} \alpha_{\sigma_q,p} i_{\sigma_q} \right) \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=1}^{l-1} \phi_{\sigma_q, i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right). \tag{4.35}$$

By using D_1 as an example, Eq. (4.33) indicates that only $3N$ additions and $4N$ multiplications are required to compute $\hat{J}_1^{(l)} = (\hat{J}_{i_l,1}^{(l)})$ in the l-marginal case, provided that $\hat{J}_1^{(l-1)} = (\hat{J}_{i_{l-1},1}^{(l-1)})$ in the $(l-1)$-marginal case, $J_p^{(l)}$ and $J_p^{(l-1)}$ have been obtained. It requires $1N$ additions and $3N$ multiplications to compute $\hat{J}_1^{(2)}$. Thus, in this situation, $\hat{J}_{i_l,p}^{(l)}$ can be computed with $(3l-5)N$ additions and $(4l-5)N$ multiplications. Except for the calculation of $J_p^{(l)}$, it requires additional $(l-1)!(3l-5)N$ additions and $(l-1)!(4l-5)N$ multiplications to calculate all $\hat{J}_p^{(l)}$ in this situation.

- **Situation 2**: D_p in which i_l is the minimum among $i_1, \cdots, i_l$, given by a general form of
$$D_p = \{ (i_1, \cdots, i_l) \mid 1 \leq i_l \leq i_{\sigma_1} \leq \cdots \leq i_{\sigma_{l-1}} \leq N \},$$
for instance $D_{l!}$. Then $J_p^{(l)}$ has a general form of

$$J_{i_l,p}^{(l)} = \sum_{i_l \leq i_{\sigma_1} \leq \cdots \leq i_{\sigma_{l-1}} \leq N} \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=1}^{l-1} \phi_{\sigma_q, i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right). \tag{4.36}$$

We can obtain the similar conclusion with the first situation that $J_{i_l,p}^{(l)}$ can be computed with $(l-1)N$ additions and $(2l-2)N$ multiplications. Since the permutation $(\sigma_1, \sigma_2, \cdots, \sigma_{l-1})$ has $(l-1)!$ arrangements, totally it requires $(l-1)!(l-1)N$ additions and $(l-1)!(2l-2)N$ multiplications to calculate all $J_p^{(l)}$ in this situation.

A similar conclusion can be also drawn for

$$\hat{J}_{i_l,p}^{(l)} = \sum_{i_l \leq i_{\sigma_1} \leq \cdots \leq i_{\sigma_{l-1}} \leq N} \left(\alpha_{l,p} i_l + \sum_{q=1}^{l-1} \alpha_{\sigma_q,p} i_{\sigma_q} \right) \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=1}^{l-1} \phi_{\sigma_q, i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right). \tag{4.37}$$

Except for the calculation of $J_p^{(l)}$, it requires additional $(l-1)!(3l-5)N$ additions and $(l-1)!(4l-5)N$ multiplications to calculate all $J_p^{(l)}$ in this situation.

- **Situation 3**: D_p in which i_l is neither the minimum nor the maximum among $i_1,\cdots,i_l$, given by a general form of

$$D_p = \{(i_1,\cdots,i_l)\,|\,1\leq i_{\sigma_1}\leq i_{\sigma_2}\leq\cdots i_{\sigma_r}\leq i_l\leq i_{\sigma_{r+1}}\leq\cdots\leq i_{\sigma_{l-1}}\leq N\}.$$

Then $J_p^{(l)}$ has a general form of

$$
\begin{aligned}
J_{i_l,p}^{(l)} &= \sum_{(i_1,\cdots,i_l)\in D_p}\left(\lambda^{\alpha_{l,p}i_l}\prod_{q=1}^{l-1}\phi_{q,i_q}\lambda^{\alpha_{q,p}i_q}\right)\\
&= \left(\sum_{1\leq i_{\sigma_1}\leq\cdots\leq i_{\sigma_r}\leq i_l}\lambda^{\alpha_{l,p}i_l}\prod_{q=1}^{r}\phi_{\sigma_q,i_{\sigma_q}}\lambda^{\alpha_{\sigma_q,p}i_{\sigma_q}}\right)\left(\sum_{i_l\leq i_{\sigma_{r+1}}\leq\cdots\leq i_{\sigma_{l-1}}\leq N}\prod_{q=r+1}^{l-1}\phi_{\sigma_q,i_{\sigma_q}}\lambda^{\alpha_{\sigma_q,p}i_{\sigma_q}}\right)\\
&= P_{i_l,p}Q_{i_l,p}.
\end{aligned}
$$

Here $P_{i_l,p}$ and $Q_{i_l,p}$ have the following forms:

$$P_{i_l,p} = \sum_{1\leq i_{\sigma_1}<\cdots<i_{\sigma_r}<i_l}\lambda^{\alpha_{l,p}i_l}\prod_{q=1}^{r}\phi_{\sigma_q,i_{\sigma_q}}\lambda^{\alpha_{\sigma_q,p}i_{\sigma_q}},$$

$$Q_{i_l,p} = \sum_{i_l\leq i_{\sigma_{r+1}}\leq\cdots\leq i_{\sigma_{l-1}}\leq N}\prod_{q=r+1}^{l-1}\phi_{\sigma_q,i_{\sigma_q}}\lambda^{\alpha_{\sigma_q,p}i_{\sigma_q}}.$$

It is observed that $P_{i_l,p}$ has a similar form of $J_{i_l,1}^{(r)}$ in the $(r+1)$-marginal case, thus can be computed with rN additions and $2rN$ multiplications. A similar conclusion can be drawn for $Q_{i_l,p}$ in the $(l-r)$-marginal case, requiring $(l-r-1)N$ additions and $2(l-r-1)N$ multiplications. $J_{i_l,p}^{(l)} = P_{i_l,p}Q_{i_l,p}$ takes additional N multiplications. As a result, $J_{i_l,p}^{(l)}$ can be computed with $(l-1)N$ additions and $(2l-1)N$ multiplications. Since there are $l!-2(l-1)!$ cases in this situation, totally it requires $(l!-2(l-1)!)\times(l-1)N$ additions and $(l!-2(l-1)!)(2l-1)N$ multiplications to calculate all $J_p^{(l)}$ in this situation.

Additionally, $\hat{J}_p$ has a general form of

$$
\begin{aligned}
\hat{J}_{i_l,p}^{(l)} &= \sum_{(i_1,\cdots,i_l)\in D_p}\left(\alpha_{l,p}i_l+\sum_{q=1}^{l-1}\alpha_{q,p}i_q\right)\left(\lambda^{\alpha_{l,p}i_l}\prod_{q=1}^{l-1}\phi_{q,i_q}\lambda^{\alpha_{q,p}i_q}\right)\\
&= \sum_{(i_1,\cdots,i_l)\in D_p}\left(\sum_{q=1}^{r}\alpha_{\sigma_q,p}i_{\sigma_q}\right)\left(\lambda^{\alpha_{l,p}i_l}\prod_{q=1}^{l-1}\phi_{q,i_q}\lambda^{\alpha_{\sigma_q,1}i_{\sigma_q}}\right)
\end{aligned}
$$

$$+ \sum_{(i_1,\cdots,i_l)\in D_p} \left(\sum_{q=r+1}^{l-1} \alpha_{\sigma_q,p} i_{\sigma_q} \right) \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=1}^{l-1} \phi_{\sigma_q,i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right)$$

$$+ \sum_{(i_1,\cdots,i_l)\in D_p} (\alpha_{l,p} i_l) \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=1}^{l-1} \phi_{\sigma_q,i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right)$$

$$= R_{i_l,p} P_{i_l,p} + S_{i_l,p} Q_{i_l,p} + \alpha_{l,p} i_l J_{i_l,p},$$

in which

$$R_{i_l,p} = \sum_{1 \le i_{\sigma_1} \le \cdots \le i_{\sigma_r} \le i_l} \left(\sum_{q=1}^{r} \alpha_{\sigma_q,p} i_{\sigma_q} \right) \left(\prod_{q=1}^{r} \phi_{\sigma_q,i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right),$$

$$S_{i_l,p} = \sum_{i_l \le i_{\sigma_{r+1}} \le \cdots \le i_{\sigma_{l-1}} \le N} \left(\sum_{q=r+1}^{l-1} \alpha_{\sigma_q,p} i_{\sigma_q} \right) \left(\lambda^{\alpha_{l,p} i_l} \prod_{q=r+1}^{l-1} \phi_{\sigma_q,i_{\sigma_q}} \lambda^{\alpha_{\sigma_q,p} i_{\sigma_q}} \right).$$

Similarly, $R_{i_l,p}$ has a similar form of $\hat{J}_{i_l,1}^{(r)}$ in the $(r+1)$-marginal case, thus can be computed with $(3r-2)N$ additions and $(4r-1)N$ multiplications. A similar conclusion can be drawn for $S_{i_l,p}$ in the $(l-r)$-marginal case, requiring $(3(l-r)-5)N$ additions and $4(l-r)-5N$ multiplications. $\hat{J}_{i_l,p}^{(l)} = R_{i_l,p} P_{i_l,p} + S_{i_l,p} Q_{i_l,p} + \alpha_{l,p} i_l J_{i_l,p}$ takes additional $2N$ additions and $3N$ multiplications. As a result, except for the calculation of $J_p^{(l)}$, it requires additional $(l!-2(l-1)!)(3l-5)N$ additions and $(l!-2(l-1)!)(4l-3)N$ multiplications to compute all $\hat{J}_{i_l,p}^{(l)}$ in this situation.

By summing up the operation counts above, Tables 5 and 6 present the number of additive and multiplicative operations required for the fast tensor-vector products in the l-marginal optimal transport problem.

Table 5: The number of additive and multiplicative operations required for the fast tensor-vector product $\mathcal{K} \times_{i_1} \boldsymbol{\phi}_1 \times_{i_2} \boldsymbol{\phi}_2 \cdots \times_{i_{l-1}} \boldsymbol{\phi}_{l-1}$.

Margin l	NA	NM	Total
$l=3$	$17N$	$26N$	$43N$
$l=4$	$95N$	$156N$	$251N$
$l=5$	$599N$	$1032N$	$1631N$
l	$(l(l!)-1)N$	$(2l^2-l-2)(l-1)!N$	$((3l^2-l-2)(l-1)!-1)N$

5 Numerical experiments

In this section, we conduct numerical experiments to demonstrate the accuracy and efficiency of our algorithm in both 1D and 2D cases. The generalized Sinkhorn algorithm is

Table 6: The number of additive and multiplicative operations required for the fast tensor-vector product $(\mathcal{C} \odot \mathcal{K}) \times_{i_1} \boldsymbol{\phi}_1 \times_{i_2} \boldsymbol{\phi}_2 \cdots \times_{i_{l-1}} \boldsymbol{\phi}_{l-1}$.

Margin l	NA	NM	Total
$l=3$	$41N$	$72N$	$113N$
$l=4$	$263N$	$444N$	$707N$
$l=5$	$1799N$	$2976N$	$4775N$
l	$((4l-5)(l!)-1)N$	$(6l^2-4l-6)(l-1)!N$	$((10l^2-9l-6)(l-1)!-1)N$

regarded as the ground truth of the entropy regularized MMOT problem. In the following experiments, we illustrate the computational costs of the two algorithms, as well as the disparities in their results.

In the numerical experiments, the regularization parameter ε is set to 10^{-1} for both our algorithm and the Sinkhorn algorithm unless otherwise specified. For the sake of efficiency comparison, we conduct 100 Sinkhorn iterations uniformly across all experiments. For each scenario, the computational time and relative error results are averaged over 100 experiments. All the experiments are conducted in Matlab R2020a on a PC with 16G RAM, 11-th Gen Intel (R) Core (TM) i5-1135G7 CPU @2.40GHz.

5.1 1D random distribution

In the first experiment, we consider three discrete 1D random distributions on the interval $[0,1]$, sampled by N grid points uniformly distributed. Three discrete vectors $\boldsymbol{u}^1, \boldsymbol{u}^2, \boldsymbol{u}^3$ are randomly generated on the grid points, where each element follows a uniform distribution over $[0,1]$. We compare the performances of the original Sinkhorn algorithm and our algorithm in computing $W_\varepsilon(u^1/\|u^1\|_1, u^2/\|u^2\|_1, u^3/\|u^3\|_1)$.

In Table 7, we output the computational time of the two algorithms and the difference between the transport plans computed by both algorithms. Notably, as the number of grid points increases, we observe a significant speed-up ratio while the optimal transport plans computed by both algorithms remain nearly identical. This shows the sub-

Table 7: The 1D random distribution problem. The comparison between the Sinkhorn algorithm and our algorithm with different numbers of grid points N. P_O and P_S are the transport plans of our algorithm and the Sinkhorn algorithm, respectively.

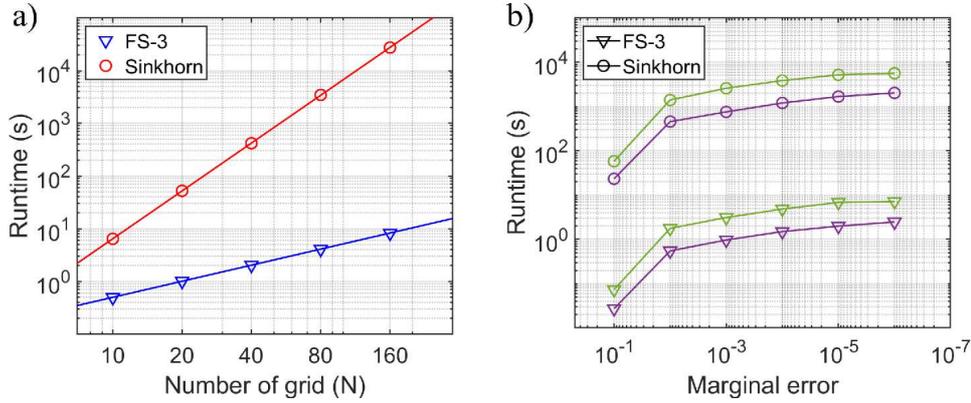| N	Computational time (s)		Speed-up ratio	$\|P_O - P_S\|_F$
	Ours	Sinkhorn		
10	4.99×10^{-1}	6.48×10^0	1.30×10^1	1.13×10^{-16}
20	1.01×10^0	5.26×10^1	5.18×10^1	5.53×10^{-17}
40	2.05×10^0	4.20×10^2	2.05×10^2	5.68×10^{-17}
80	4.12×10^0	3.47×10^3	8.43×10^2	4.12×10^{-17}
160	8.27×10^0	2.74×10^4	3.32×10^3	3.58×10^{-17}

Figure 1: The 1D random distribution problem. (a): The computational time of our algorithm and the Sinkhorn algorithm with different numbers of grid points N. (b): The computational time required to reach the corresponding marginal error for $N = 80$ under $\varepsilon = 0.1$ (purple) and $\varepsilon = 0.05$ (green).

stantial advantage of our algorithm in computation efficiency. In Fig. 1(a), we compare the runtime of the Sinkhorn algorithm and our algorithm with different numbers of grid points N. Employing linear regression, we obtain the empirical complexities of both algorithms: Our algorithm exhibits $\mathcal{O}(N^{1.01})$ complexity, and the Sinkhorn algorithm shows $\mathcal{O}(N^{3.01})$ complexity. Additionally, Fig. 1(b) illustrates the computational time required to achieve corresponding marginal errors for $N = 80$ under different ε, which emphasizes the overwhelming superiority of our algorithm in speed under different regularization parameters.

5.2   The Ricker wavelet

The Ricker wavelet is widely used in seismology for modeling the source time function, which has a form of

$$R(t) = A\left(1 - 2\pi^2 F_0^2 t^2\right)e^{-\pi^2 F_0^2 t^2}.$$

Here A denotes the wave amplitude, and F_0 is the dominant frequency.

We evaluate the performances of the original Sinkhorn algorithm and our algorithm in measuring the discrepancy between three Ricker wavelets

$$f_1(t) = R(t - \tau_1), \quad f_2(t) = R(t - \tau_2), \quad f_3(t) = R(t - \tau_3)$$

based on the Wasserstein-1 distance. Without loss of generality, we set $A = 1, F_0 = 1$. To transform the Ricker wavelet to probability distribution function, we apply the normalization method introduced in [27] and consider the following distance:

$$D(f_1, f_2, f_3) = W_\varepsilon \left(\frac{f_1^2 / \|f_1^2\|_1 + \delta}{1 + L\delta}, \frac{f_2^2 / \|f_2^2\|_1 + \delta}{1 + L\delta}, \frac{f_3^2 / \|f_3^2\|_1 + \delta}{1 + L\delta} \right), \tag{5.1}$$

where L is used for normalization, and δ is a given parameter to improve numerical stability.

In the experiment, we consider uniform grid points on the interval $[-2,2]$, and set parameters $\tau_1 = 0$, $\tau_2 = 0.75$, $\tau_3 = 1.5$, $\delta = 10^{-3}$. In Table 8, we output the computational time of both algorithms and the difference between the transport plans computed by both algorithms. The computational time required to achieve corresponding marginal errors for $N = 80$ under different ε is shown in Fig. 2(a). As discussed in Section 5.1, our algorithm maintains significant efficiency while preserving high computational precision compared to the Sinkhorn algorithm.

In Fig. 2(b), we show the importance of applying the log-domain stabilization technique when regularization parameters are relatively small, e.g. $\varepsilon = 0.001$. Without the log-domain stabilization, our algorithm terminates at the 89-th iteration due to exceeding the system threshold. However, with the log-domain stabilization applied, our algorithm maintains its efficiency without termination.

Table 8: The Ricker wavelet problem. The comparison between the Sinkhorn algorithm and our algorithm with different numbers of grid points N. P_O and P_S are the transport plans of our algorithm and the Sinkhorn algorithm, respectively.

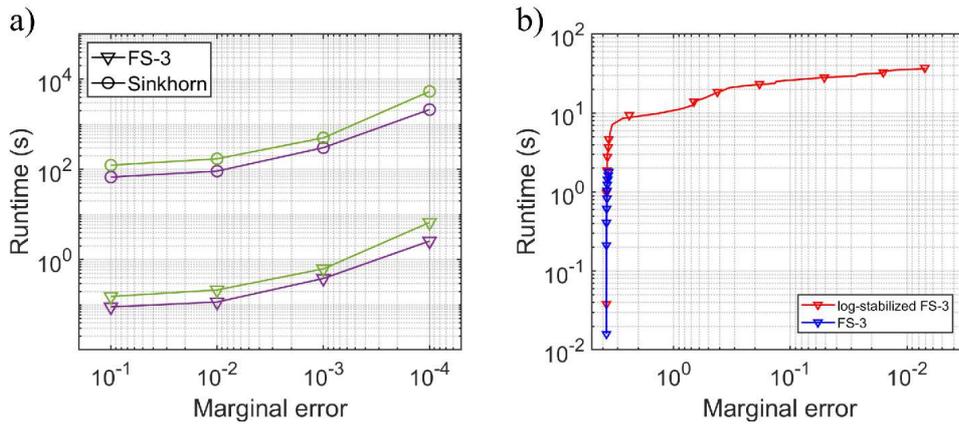| N	Computational time (s)		Speed-up ratio	$\|P_O - P_S\|_F$
	Ours	Sinkhorn		
10	5.07×10^{-1}	6.78×10^0	1.34×10^1	9.39×10^{-17}
20	1.06×10^0	5.27×10^1	4.97×10^1	1.03×10^{-16}
40	2.12×10^0	4.20×10^2	2.68×10^2	1.93×10^{-16}
80	4.20×10^0	3.40×10^3	8.10×10^2	7.36×10^{-17}
160	8.52×10^0	2.71×10^4	3.18×10^3	1.16×10^{-16}

Figure 2: The Ricker wavelet problem. (a): The computational time required to reach the corresponding marginal error for $N = 80$ under $\varepsilon = 0.1$ (purple) and $\varepsilon = 0.05$ (green). (b): The comparison between our algorithms with and without the log-domain stabilization for $N = 100, \varepsilon = 0.001$.

5.3   2D random distribution

In this subsection, we evaluate the performance of the Sinkhorn algorithm and our algorithm in computing the Wasserstein-1 metric between three discrete 2D random distributions on $[0,1] \times [0,1]$. Those distributions are represented by $N \times M$ dimensional random vectors, where each element follows a uniform distribution over $[0,1]$. We use the basic settings mentioned in Section 4.1, and for simplicity, we set $M = N$.

Similar to Section 5.1, we output the computational time of two algorithms and the difference between the transport plans computed by both algorithms in Table 9. Additionally, Fig. 3(a) illustrates the comparison between the Sinkhorn algorithm and our algorithm with different numbers of grid points N. It is notable that with increasing N, the computation time of the Sinkhorn algorithm becomes unacceptable, which reaches several weeks. In contrast, the computational cost of our algorithm remains affordable, which draws the same conclusion as we discussed in the sections above. After con-

Table 9: The 2D random distribution problem. The comparison between the Sinkhorn algorithm and our algorithm with different numbers of grid points N and $\varepsilon = 0.1$. P_O and P_S are the transport plans of our algorithm and the Sinkhorn algorithm, respectively. We use '–' to denote the computational time exceeding 2×10^6 s.

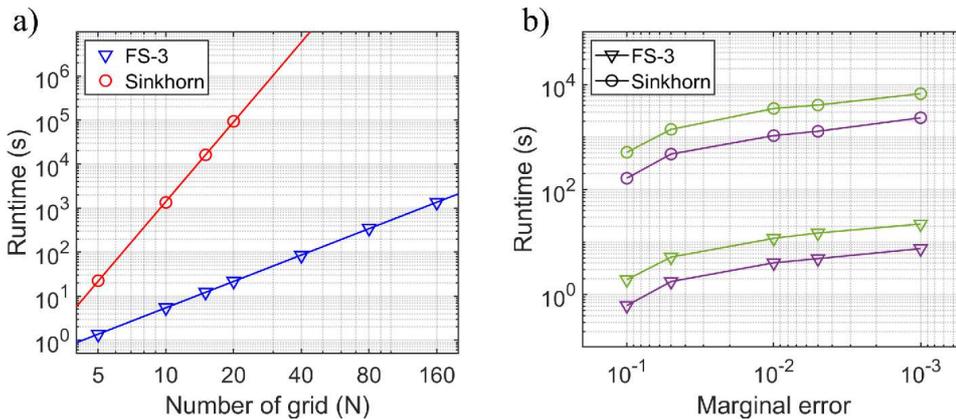| $N \times M$	Computational time (s)		Speed-up ratio	$\|P_O - P_S\|_F$
	Ours	Sinkhorn		
10×10	5.45×10^0	1.35×10^3	2.48×10^2	3.26×10^{-16}
20×20	2.19×10^1	9.36×10^4	4.27×10^3	7.24×10^{-16}
40×40	8.40×10^1	–	–	–
80×80	3.48×10^2	–	–	–
160×160	1.35×10^3	–	–	–

Figure 3: The 2D random distribution problem. (a): The comparison between the Sinkhorn algorithm and our algorithm with different numbers of grid points N. (b): The computational time required to reach the corresponding marginal error for $N = 10$ under $\varepsilon = 0.1$ (purple) and $\varepsilon = 0.01$ (green).

ducting linear regression, we obtain that the empirical complexity of our algorithm is $\mathcal{O}(N^{1.99})$ while that of the Sinkhorn algorithm is $\mathcal{O}(N^{6.01})$. With the growth of dimension, the advantage of our algorithm in efficiency becomes more prominent. Furthermore, the computational time required to achieve corresponding marginal errors for $N=10$ under different ε is shown in Fig. 3(b). Without doubt, our algorithm outperforms the Sinkhorn algorithm by a wide margin in terms of speed under different regularization parameters.

5.4 Multiple image matching problem

Nowadays, image matching problem has emerged as a trending topic in optimal transport [9, 17, 31]. Our above experiments have already demonstrated the significant efficiency advantage of our algorithm over the traditional Sinkhorn algorithm, suggesting its considerable potential also in the realm of computer vision. Despite the extensive research on image matching, researchers are not satisfied with comparing two agents, but turn to multiple marginal matching problem which aims at learning transport plans to match a source domain to multiple target domains [11]. This multiple matching problem can be readily linked to the MMOT problem. Here we select three groups of images with different sizes from the DOTmark dataset which is specifically designed for discrete optimal transport [37] (see Fig. 4 for illustration).

We transform images into 2D distributions on $[0,1] \times [0,1]$, represented by $M \times N$ dimensional vectors. Without loss of generality, we set $M = N$. The Wasserstain-1 metric between three images are computed by our algorithm and the original Sinkhron algorithm. In Table 10, we present the computational time of the two algorithms and the difference between the transport plans computed by both algorithms. As mentioned in Section 5.3, the computational time of the Sinkhorn algorithm soon becomes unbearable as N increases. In contrast, the computational cost of our algorithm still remains affordable. The computational time required to achieve corresponding marginal errors for $N=16$ under different ε is shown in Fig. 5(a).

Finally, we show that the log-domain stabilization technique still works for two-dimensional problems with small regularization parameters. As shown in Fig. 5(b),

Figure 4: Illustration of images from DOTmark dataset.

our algorithm terminates at the 242-nd iteration without this technique. In contrast, by employing the log-domain stabilization technique, our algorithm maintains its efficacy, proven to be an efficient method for computing the MMOT problem.

Table 10: Multiple image matching problem. The comparison between the Sinkhorn algorithm and our algorithm with numbers of grid points N and $\varepsilon=0.1$. P_O and P_S are the transport plans of our algorithm and the Sinkhorn algorithm, respectively. We use "–" to denote the computational time exceeding 2×10^6s.

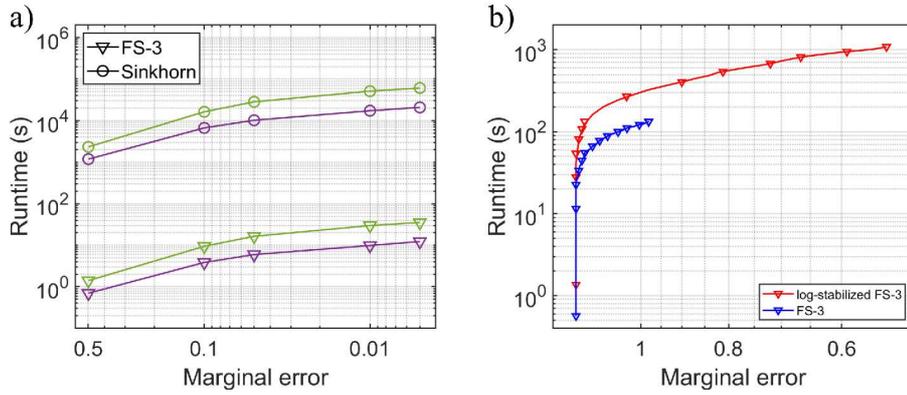| $N\times M$	Computational time (s)		Speed-up ratio	$\|P_O-P_S\|_F$
	Ours	Sinkhorn		
16×16	1.38×10^1	2.30×10^5	1.67×10^3	6.41×10^{-16}
32×32	5.48×10^1	1.48×10^6	2.70×10^4	1.06×10^{-15}
64×64	2.16×10^2	–	–	–
128×128	8.60×10^2	–	–	–
256×256	3.47×10^3	–	–	–

Figure 5: Multiple image matching problem. (a): The computational time required to reach the corresponding marginal error for $N=16$ under $\varepsilon=0.1$ (purple) and $\varepsilon=0.05$ (green). (b): The comparison between our algorithms with and without the log-domain stabilization for $N=32,\varepsilon=0.0005$.

6   Conclusion

In this paper, we propose an efficient numerical algorithm for solving the L^1 norm based multi-marginal optimal transport problem with linear complexity relative to support size N. This method accelerates the repeated tensor-vector products in the Sinkhorn algorithm by decomposing it into a summation of multiple components, each computed recursively with $\mathcal{O}(N)$ additive and multiplicative operations. Furthermore, the log-domain stabilization technique is incorporated into our algorithm to avoid over- and underflow. In numerical experiments, we demonstrate that our algorithm achieves a significant speed advantage over the traditional method while maintaining accuracy. It reduces

the computation cost for l-marginal case from $\mathcal{O}(N^l)$ to $\mathcal{O}(N)$ for any l. We anticipate that our algorithm will lead to a notable acceleration in various fields such as computer vision, machine learning, and transportation operations.

Acknowledgments

References

[1] I. Abraham, R. Abraham, M. Bergounioux, and G. Carlier, *Tomographic reconstruction from a few views: A multi-marginal optimal transport approach*, Appl. Math. Optim., 75:55–73, 2017.

[2] M. Agueh and G. Carlier, *Barycenters in the Wasserstein space*, SIAM J. Math. Anal., 43:904–924, 2011.

[3] J. M. Altschuler and E. Boix-Adserà, *Wasserstein barycenters can be computed in polynomial time in fixed dimension*, J. Mach. Learn. Res., 22:44, 2021.

[4] A. Baradat and L. Monsaingeon, *Small noise limit and convexity for generalized incompressible flows, Schrödinger problems, and optimal transport*, Arch. Ration. Mech. Anal., 235:1357–1403, 2020.

[5] J.-D. Benamou and Y. Brenier, *A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem*, Numer. Math., 84:375–393, 2000.

[6] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré, *Iterative Bregman projections for regularized transportation problems*, SIAM J. Sci. Comput., 37:A1111–A1138, 2015.

[7] J.-D. Benamou, G. Carlier, and L. Nenna, *A numerical method to solve multi-marginal optimal transport problems with Coulomb cost*, in: Splitting Methods in Communication, Imaging, Science, and Engineering, Springer, 577–601, 2016.

[8] J.-D. Benamou, G. Carlier, and L. Nenna, *Generalized incompressible flows, multi-marginal transport and Sinkhorn algorithm*, Numer. Math., 142:33–54, 2019.

[9] M. Burger, M. Franek, and C.-B. Schönlieb, *Regularized regression and density estimation based on optimal transport*, Appl. Math. Res. Express, 2012(2):209–253, 2012.

[10] G. Buttazzo, L. De Pascale, and P. Gori-Giorgi, *Optimal-transport formulation of electronic density-functional theory*, Phys. Rev. A, 85:062502, 2012.

[11] J. Cao, L. Mo, Y. Zhang, K. Jia, C. Shen, and M. Tan, *Multi-marginal Wasserstein GAN*, Advances in Neural Information Processing Systems, Curran Associates, Inc., 32:1765–1775, 2019.

[12] Y. Chen, T. T. Georgiou, and M. Pavon, *On the relation between optimal transport and Schrödinger bridges: A stochastic control viewpoint*, J. Optim. Theory Appl., 169:671–691, 2016.

[13] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard, *Scaling algorithms for unbalanced optimal transport problems*, Math. Comp., 87:2563–2609, 2018.

[14] C. Cotar, G. Friesecke, and C. Klüppelberg, *Density functional theory and optimal transportation with Coulomb cost*, Comm. Pure Appl. Math., 66:548–599, 2013.

[15] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21:1253–1278, 2000.

[16] B. Engquist, B. D. Froese, and Y. Yang, *Optimal transport for seismic full waveform inversion*, Commun. Math. Sci., 14:2309–2330, 2016.

[17] B. Engquist, K. Ren, and Y. Yang, *The quadratic Wasserstein metric for inverse data matching*, Inverse Problems, 36(5):055001, 2020.

[18] W. Gangbo and A. Święch, *Optimal maps for the multidimensional Monge-Kantorovich problem*, Comm. Pure Appl. Math., 51:23–45, 1998.

[19] I. Haasler, A. Ringh, Y. Chen, and J. Karlsson, *Multimarginal optimal transport with a tree-structured cost and the Schrödinger bridge problem*, SIAM J. Control Optim., 59:2428–2453, 2021.

[20] I. Haasler, R. Singh, Q. Zhang, J. Karlsson, and Y. Chen, *Multi-marginal optimal transport and probabilistic graphical models*, IEEE Trans. Inf. Theory, 67:4647–4668, 2021.

[21] W. G. Horner, *A new method of solving numerical equations of all orders, by continuous approximation*, Philos. Trans. R. Soc. London, 308–335, 1819.

[22] Y. Hu, H. Chen, and X. Liu, *A global optimization approach for multimarginal optimal transport problems with Coulomb cost*, SIAM J. Sci. Comput., 45:A1214–A1238, 2023.

[23] Y. Hu, M. Li, X. Liu, and C. Meng, *Sampling-based approaches for multimarginal optimal transport problems with coulomb cost*, arXiv:2306.16763, 2023.

[24] T. Kailath and A. H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37:297–386, 1995.

[25] J. Kleinberg and E. Tardos, *Algorithm Design*, Pearson, 2006.

[26] W. Li, E. K. Ryu, S. Osher, W. Yin, and W. Gangbo, *A parallel method for earth mover's distance*, J. Sci. Comput., 75:182–197, 2018.

[27] Z. Li, Y. Tang, J. Chen, and H. Wu, *The quadratic Wasserstein metric with squaring scaling for seismic velocity inversion*, Numer. Math. Theory Methods Appl., 16(2):277–297, 2023.

[28] Q. Liao, J. Chen, Z. Wang, B. Bai, S. Jin, and H. Wu, *Fast Sinkhorn I: An O (N) algorithm for the Wasserstein-1 metric*, Commun. Math. Sci., 20:2053–2067, 2022.

[29] Q. Liao, Z. Wang, J. Chen, B. Bai, S. Jin, and H. Wu, *Fast Sinkhorn II: Collinear triangular matrix and linear time accurate computation of optimal transport*, J. Sci. Comput., 98:1, 2024.

[30] T. Lin, N. Ho, M. Cuturi, and M. I. Jordan, *On the complexity of approximating multimarginal optimal transport*, J. Mach. Learn. Res., 23:1–43, 2022.

[31] J. Maas, M. Rumpf, C. Schönlieb, and S. Simon, *A generalized model for optimal transport of images including dissipation and density modulation*, ESAIM Math. Model. Numer. Anal., 49(6):1745–1769, 2015.

[32] A. M. Metelli, A. Likmeta, and M. Restelli, *Propagating uncertainty in reinforcement learning via Wasserstein barycenters*, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates, Inc., Vol. 32, 4333–4345, 2019.

[33] L. Métivier, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux, *Measuring the misfit between seismograms using an optimal transport distance: Application to full waveform inversion*, Geophys. J. Int., 205:345–377, 2016.

[34] L. Nenna, *Numerical Methods for Multi-Marginal Optimal Transportation*, PhD Thesis, Université Paris Sciences et Lettres, 2016.

[35] O. Pele and M. Werman, *Fast and robust earth mover's distances*, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 460–467, 2009.

[36] G. Peyré et al., *Computational optimal transport: With applications to data science*, Found. Trends

Mach. Learn., 11:355–607, 2019.

[37] J. Schrieber, D. Schuhmacher, and C. Gottschlich, *DOTmark – A benchmark for discrete optimal transport*, IEEE Access, 5:271–282, 2017.

[38] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher, *Earth mover's distances on discrete surfaces*, ACM Trans. Graph., 33:1–12, 2014.

[39] Y. Takezawa, R. Sato, Z. Kozareva, S. Ravi, and M. Yamada, *Fixed support tree-sliced Wasserstein barycenter*, in: International Conference on Artificial Intelligence and Statistics, PMLR, 1120–1137, 2022.

[40] N. Tupitsa, P. Dvurechensky, A. Gasnikov, and C. A. Uribe, *Multimarginal optimal transport by accelerated alternating minimization*, in: 2020 59th IEEE Conference on Decision and Control (CDC), IEEE, 6132–6137, 2020.

[41] M. Yamada, Y. Takezawa, R. Sato, H. Bao, Z. Kozareva, and S. Ravi, *Approximating 1-Wasserstein distance with trees*, Trans. Mach. Learn. Res., 2022.

[42] S. Yang and H. Zha, *Estimating latent population flows from aggregated data via inversing multimarginal optimal transport*, in: Proceedings of the 2023 SIAM International Conference on Data Mining (SDM), SIAM, 181–189, 2023.

[43] Y. Yang, B. Engquist, J. Sun, and B. F. Hamfeldt, *Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion*, Geophysics, 83:R43–R62, 2018.