Final Report: Algorithms for large-scale optimal transport

汪祎非* 朱峰*
1500010611 1600010643

Statement

The main content of this project is not used as course project or assignment in other courses.

1 Introduction

According to [4], optimal transport is a mathematical gem at the interface between probability, analysis and optimization. The theoretical importance of OT is that it defines a metric between probability distributions, namely, Wasserstein metric. It reveals a canonical geometric structure with rich properties to be exploited. The earliest contribution to OT originated from Monge in 18th century. Kantorovich rediscovered it under a different formalism, namely the Linear Programming formulation of OT. With the development of scalable solvers, OT is widely applied to problems in imaging sciences(color and texture processing), computer vision and machine learning(regression).

In this report, we focus on the LP formulation and the Entropy Regularization of optimal transport. Our report is organized as follows. In Section 2, we propose three first-order methods and introduce Bregman ADMM. In Appendix we also introduce a splitting method with penalty functions. The Entropy Regularization of optimal transport and Sinkhorn's algorithm is presented in Section 3. We further propose an improvement of Sinkhorn's

*The authors are arranged lexicographically.

algorithm and ADMM for Entropy Regularization. Section 4 presents five datasets and contains comprehensive numerical experiments. In Section 5, we conclude our work.

Question 1(a) is answered in Section 4.2. Question 1(b) is answered in Section 2. Question 2 is answered in Section 3. The numerical results of all questions are provided in Section 4. The division of our work is listed in Acknowledgement (in the end of our report).

2 First Order Methods for Optimal Transport

We consider to solve the standard form of Linear Programming (LP).

$$\begin{aligned}\min_{\pi\in\mathbb{R}^{m\times n}} & \sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j}\pi_{i,j} \\ s.t. & \sum_{j=1}^{n}\pi_{i,j} = \mu_i,\ \forall i = 1,2,\dots m \\ & \sum_{i=1}^{m}\pi_{i,j} = \nu_j,\ \forall j = 1,2,\dots n \\ & \pi_{i,j} \geqslant 0\end{aligned} \tag{1}$$

To ensure that the feasible set for problem 1 is non-empty, we assume that $\mu_i \geqslant 0, \nu_j \geqslant 0$ and $\sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j$. Without loss of generality, we further assume $C_{i,j} \geqslant 0$ and $\sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j = 1$. Else we could add a positive constant on $C_{i,j}$ and divide μ_i and ν_j by $\sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j$.

2.1 ADMM: Complete Splitting

2.1.1 ADMM for primal

We introduce a new variable $\tilde{\pi} \in \mathbb{R}^{m\times n}$ and split the constraints into two sets $\{\pi \in \mathbb{R}^{m\times n} | \sum_{j=1}^n \pi_{i,j} = \mu_i, \sum_{i=1}^m \pi_{i,j} = \nu_j\}$ and $\{\tilde{\pi} \in \mathbb{R}^{m\times n} | \tilde{\pi} \geq 0\}$. Then we rewrite the LP problem 1 in the following form,

$$\begin{aligned}\min_{\pi\in\mathbb{R}^{m\times n}} & \sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j}\pi_{i,j} \\ \text{s.t.} & \sum_{j=1}^{n}\pi_{i,j} = \mu_i, \quad \sum_{i=1}^{m}\pi_{i,j} = \nu_j \\ & \pi_{i,j} = \tilde{\pi}_{i,j}, \quad \tilde{\pi}_{i,j} \geqslant 0\end{aligned} \tag{2}$$

Then we write down its augmented Lagrangian function as follows,

$$
\begin{aligned}
&L_t(\pi, \tilde{\pi}, \gamma, \lambda, \omega) \\
&= \sum_{i,j} C_{i,j}\pi_{i,j} + \sum_i \gamma_i(\mu_i - \sum_j \pi_{i,j}) + \sum_j \lambda_j(\nu_j - \sum_i \pi_{i,j}) + \sum_{i,j} \omega_{i,j}(\tilde{\pi}_{i,j} - \pi_{i,j}) \\
&\quad + \frac{t}{2}\left(\sum_i (\mu_i - \sum_j \pi_{i,j})^2 + \sum_j (\nu_j - \sum_i \pi_{i,j})^2 + \sum_{i,j} (\tilde{\pi}_{i,j} - \pi_{i,j})^2\right)
\end{aligned} \tag{3}
$$

where $\gamma \in \mathbb{R}^{m\times 1}, \lambda \in \mathbb{R}^{1\times n}, \omega \in \mathbb{R}^{m\times n}, \tilde{\pi} \in \mathbb{R}^{m\times n}, \tilde{\pi}_{i,j} \geqslant 0$.

For the update of $\pi_{i,j}$, since L is strongly convex with $\pi_{i,j}$, we set $\frac{\partial L}{\partial \pi_{ij}}$ to zero and yields

$$
\pi_{i,j} + \sum_k \pi_{i,k} + \sum_k \pi_{k,j} = \frac{1}{t}(\gamma_i + \lambda_j + \omega_{i,j} - c_{i,j}) + \mu_i + \nu_j + \tilde{\pi}_{i,j} \triangleq x_{i,j} \tag{4}
$$

Though a bit complicated, (4) can be solved in closed form,

$$
\begin{aligned}
\pi_{i,j} \leftarrow x_{i,j} &- \frac{1}{m+1}\left(\sum_k x_{k,j} - \frac{1}{m+n+1}\sum_{k,l} x_{k,l}\right) \\
&- \frac{1}{n+1}\left(\sum_k x_{i,k} - \frac{1}{m+n+1}\sum_{k,l} x_{k,l}\right)
\end{aligned} \tag{5}
$$

To update $\tilde{\pi}_{i,j}$, it's easy to compute that

$$
\tilde{\pi}_{i,j} \leftarrow (\pi_{i,j} - \frac{\omega_{i,j}}{t})_+ \tag{6}
$$

For the update of multipliers, we have

$$
\begin{aligned}
\gamma_i &\leftarrow \gamma_i + t(\mu_i - \sum_k \pi_{i,k}) \\
\lambda_j &\leftarrow \lambda_j + t(\nu_j - \sum_k \pi_{k,j}) \\
\omega_{i,j} &\leftarrow \omega_{i,j} + t(\tilde{\pi}_{i,j} - \pi_{i,j})
\end{aligned} \tag{7}
$$

We thus can obtain the following Algorithm 1.

Algorithm 1 ADMM for Optimal Transport

```
Input: $C \in \mathbb{R}^{m\times n}$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^n$, $t \in \mathbb{R}^+$
1: $\pi, \tilde{\pi}, \gamma, \lambda, \omega \leftarrow 0$
2: while not converge do
3:     Update $\pi$ by 5
4:     Update $\tilde{\pi}$ by 6
5:     Update $\gamma, \lambda, \omega$ by 7
6: end while
7: return $\pi$
```

2.1.2 ADMM for dual

We also derive and implement an ADMM for dual problem of Optimal Transport. The Lagrangian of 1 is given by

$$\begin{aligned}&\min_{\pi}\max_{\omega\geqslant 0,\gamma,\lambda} L(\pi,\gamma,\lambda,\omega)\\ =&\sum_{i,j} C_{i,j}\pi_{i,j} - \sum_i \gamma_i(\mu_i - \sum_j \pi_{i,j}) - \sum_j \lambda_j(\nu_j - \sum_i \pi_{i,j}) - \sum_{i,j}\omega_{i,j}\pi_{i,j}\end{aligned} \tag{8}$$

However, the first two constraints in 2 are linearly dependent. More rigorously, one constraint can be removed since $\sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j$. Thus, we impose another constraint on the dual variables in 8, namely, $\lambda_n = 0$.

Calculating $\max_{\omega\geqslant 0,\gamma,\lambda}\min_{\pi_{i,j}} L(\pi,\gamma,\lambda,\omega)$ yields the dual problem

$$\begin{aligned}\max_{\gamma,\lambda}\ & -\sum_i \gamma_i\mu_i - \sum_j \lambda_j\nu_j\\ \text{s.t. } & C_{i,j} + \gamma_i + \lambda_j \geqslant 0\\ & \lambda_n = 0\end{aligned} \tag{9}$$

which is equivalent to

$$\begin{aligned}\min_{\gamma,\lambda}\ & \sum_i \gamma_i\mu_i + \sum_j \lambda_j\nu_j\\ \text{s.t. } & C_{i,j} + \gamma_i + \lambda_j \geqslant 0\\ & \lambda_n = 0\end{aligned} \tag{10}$$

Similar to that in Section (2.1.1), we introduce a new variable $z_{i,j} \in \mathbb{R}^{m\times n}$ and split the constraints into two sets $\{\gamma \in \mathbb{R}^{m\times 1}, \lambda \in \mathbb{R}^{1\times n} | C_{i,j} + \gamma_i + \lambda_j = z_{i,j}\}$ and $\{z_{i,j} \in \mathbb{R}^{m\times n} | z_{i,j} \geqslant 0\}$.

We can now write down its augmented Lagrangian function,

$$\begin{aligned} L_t(\gamma, \lambda, \omega, z, \tilde{z}) = &\sum_i \gamma_i \mu_i + \sum_j \lambda_j \nu_j + \omega_{i,j}(C_{i,j} + \gamma_i + \lambda_j - z_{i,j}) \\ &+ \frac{t}{2} \sum_{i,j} (C_{i,j} + \gamma_i + \lambda_j - z_{i,j})^2 \end{aligned} \tag{11}$$

where $\gamma \in \mathbb{R}^{m\times 1}, \lambda \in \mathbb{R}^{1\times n}, \omega \in \mathbb{R}^{m\times n}, z \in \mathbb{R}^{m\times n}, z_{i,j} \geqslant 0, \lambda_n = 0$. In LP problems, “dual of dual" is primal, and ω has direct relation with π. In fact, if we ignore the quadratic term in 11, we can find that $\omega = -\pi$. This imples that $-\omega$ can be interpreted as an approximation to π in the primal problem.

For the update of $z_{i,j}$, we have

$$z_{i,j} \leftarrow \max\{C_{i,j} + \gamma_i + \lambda_j + \frac{\omega_{i,j}}{t}, 0\} \tag{12}$$

For the update of γ and λ, we set $\frac{\partial L}{\partial \gamma_i}$ and $\frac{\partial L}{\partial \lambda_j}$ to zero and yield

$$\begin{aligned} n\gamma_i + \sum_j \lambda_j = \sum_j z_{i,j} - \sum_j C_{i,j} - \frac{1}{t}(\mu_i + \sum_j \omega_{i,j}) \triangleq a_i \ (i = 1, ..., m) \\ m\lambda_j + \sum_i \gamma_i = \sum_i z_{i,j} - \sum_i C_{i,j} - \frac{1}{t}(\nu_j + \sum_i \omega_{i,j}) \triangleq b_j \ (j = 1, ..., n-1) \end{aligned} \tag{13}$$

Thanks for $\lambda_n = 0$, 13 has a unique solution:

$$\begin{aligned} \gamma_i &\leftarrow \frac{1}{n}\left(a_i - \sum_i a_i + \sum_j b_j\right) \ (i = 1, ..., m) \\ \lambda_j &\leftarrow \frac{1}{m}\left(b_j - \frac{n}{m}\sum_j b_j + \frac{n-1}{m}\sum_i a_i\right) \ (j = 1, ..., n-1). \end{aligned} \tag{14}$$

For the update of multipliers, it’s easy to obtain that

$$\omega_{i,j} \leftarrow \omega_{i,j} + t(C_{i,j} + \gamma_i + \lambda_j - z_{i,j}). \tag{15}$$

We thus can obtain the following Algorithm 2.

Algorithm 2 ADMM for Dual of Optimal Transport

```
Input: $C \in \mathbb{R}^{m\times n}$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^n$, $t \in \mathbb{R}^+$
1: $\gamma, \lambda, \omega \leftarrow 0$
2: while not converge do
3:     Update $\gamma, \lambda$ by 14
4:     Update $z$ by 12
5:     Update $\omega$ by 15
6: end while
7: return $-\omega$
```

2.2 ADMM: Simplex Splitting

We still focus on the primal problem, but this time, we utilize another splitting method. We rewrite the problem 1 as follows,

$$\begin{aligned} \min_{\pi\in\mathbb{R}^{m\times n}} \ & \sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j}\pi_{i,j} \\ \text{s.t.} \ & \sum_j \pi_{i,j} = \mu_i, \ \sum_i \tilde{\pi}_{i,j} = \nu_j \\ & \pi_{i,j} = \tilde{\pi}_{i,j} \geqslant 0 \end{aligned} \tag{16}$$

The augmented Lagrangian function can be written as

$$\begin{aligned} L_t(\pi,\tilde{\pi},\omega) = & \sum_{i,j} C_{i,j}\frac{\pi_{i,j}+\tilde{\pi}_{i,j}}{2} + \sum_{i,j}\omega_{i,j}(\pi_{i,j}-\tilde{\pi}_{i,j}) + \frac{t}{2}\sum_{i,j}(\pi_{i,j}-\tilde{\pi}_{i,j})^2 \\ \text{s.t.} \ & \sum_j \pi_{i,j} = \mu_i, \ \pi_{i,j} \geqslant 0 \\ & \sum_i \tilde{\pi}_{i,j} = \nu_j, \ \tilde{\pi}_{i,j} \geqslant 0 \end{aligned} \tag{17}$$

For update of π, we need to solve the following subproblem, i.e., the projection on a simplex,

$$\begin{aligned} \min_{\pi} \ & \sum_{i,j}\left(\pi_{i,j} - \tilde{\pi}_{i,j} + \frac{1}{t}\left(\omega_{i,j} + \frac{C_{i,j}}{2}\right)\right)^2 \\ \text{s.t.} \ & \sum_j \pi_{i,j} = \mu_i, \ \pi_{i,j} \geqslant 0 \end{aligned} \tag{18}$$

From standard duality theory and with some calculation, we can derive that the solution must be the form

$$\pi_{i,j} \leftarrow \max\{\tilde{\pi}_{i,j} - \frac{1}{t}\left(\omega_{i,j} + \frac{C_{i,j}}{2}\right) + \lambda_i, 0\} \triangleq \pi_{i,j}(\lambda_i) \tag{19}$$

where λ_i is chosen such that

$$\sum_j \pi_{i,j}(\lambda_i) = \mu_i. \tag{20}$$

Similarly, the update of $\tilde{\pi}$ can be written as

$$\tilde{\pi}_{i,j} \leftarrow \max\{\pi_{i,j} - \frac{1}{t}\left(\omega_{i,j} - \frac{C_{i,j}}{2}\right) + \tilde{\lambda}_j, 0\} \triangleq \tilde{\pi}_{i,j}(\tilde{\lambda}_j) \tag{21}$$

where $\tilde{\lambda}_j$ is chosen such that

$$\sum_i \tilde{\pi}_{i,j}(\tilde{\lambda}_j) = \nu_j. \tag{22}$$

We now briefly describe an algorithm mentioned in [3] to find the exact value of λ. For simplicity of statement, we consider the following problem, which is the canonical form of 18,

$$\begin{aligned} &\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|_2^2 \\ &\text{s.t. } \|\mathbf{x}\|_1 = z, \ \mathbf{x} \geqslant 0, \end{aligned} \tag{23}$$

where $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$, $z \geqslant 0$. First, we sort $\mathbf{x}$ into descending order, namely

$$y_1 \geqslant y_2 \geqslant \ldots \geqslant y_n. \tag{24}$$

Then we set

$$\begin{aligned} &\rho \leftarrow \max\left\{j \in [n] : y_j - \frac{1}{j}\left(\sum_{r=1}^{j} y_r - z\right) > 0\right\} \\ &\lambda \leftarrow \frac{1}{\rho}\left(z - \sum_{r=1}^{\rho} y_r\right) \\ &\mathbf{x} \leftarrow \max\{\mathbf{x} - \lambda, 0\} \end{aligned} \tag{25}$$

if $z > 0$. If $z = 0$, then $\mathbf{x}$ is automatically $\mathbf{0}$.

For multipliers, it's easy to have the following updates,

$$\omega_{i,j} \leftarrow \omega_{i,j} + t(\pi_{i,j} - \tilde{\pi}_{i,j}) \tag{26}$$

We thus have the following Algorithm 3:

Algorithm 3 ADMM for Primal of Optimal Transport: A second splitting method

```
Input: $C \in \mathbb{R}^{m\times n}$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^n$, $t \in \mathbb{R}^+$
1: $\pi, \tilde{\pi}, \gamma, \lambda, \omega \leftarrow 0$
2: while not converge do
3:     Update $\pi$ according to 19, 24 and 25
4:     Update $\tilde{\pi}$ according to 21, 24, and 25
5:     Update $\omega$ by 26
6: end while
7: return $\pi$
```

2.3 Bregman ADMM

[8] proposes Bregman ADMM, which uses Bregman divergence to replace the penalty term in ADMM. Let $\phi : \Omega \to \mathbb{R}$ be a continuously differentiable and strictly convex function on the relative interior of the convex set Ω. We define the Bregman divergence $B_\phi : \Omega \times \operatorname{relint}\Omega \to \mathbb{R}_+$ induced by ϕ as

$$B_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y\rangle \tag{27}$$

From the convexity of ϕ, $B_\phi(x, y) \geqslant 0$ and the equality holds if and only if $x = y$. To solve the LP problem 1, we consider to take $\phi(x) = \sum_i x_i \log(x_i)$. In this case, $B_\phi(x, y) = \mathrm{KL}(x, y)$ is the KL divergence between x and y.

Similar to that in Section 2.2, we introduce a new variable $\tilde{\pi} \in \mathbb{R}^{m\times n}$ to split the constraints into two simplex $S_\mu = \{\pi | \pi \geqslant 0, \pi\mathbb{1}_n = \mu\}$ and $S_\nu = \{\tilde{\pi} | \tilde{\pi} \geqslant 0, \tilde{\pi}^T\mathbb{1}_m = \mu\}$. Then problem 1 can be written in the following form,

$$\begin{cases} \min & \langle C, \pi\rangle \\ \text{s.t.} & \pi \in S_\mu, \tilde{\pi} \in S_\nu, \pi = \tilde{\pi} \end{cases} \tag{28}$$

We then introduce another variable $\sigma \in \mathbb{R}^{m\times n}$ for BADMM. Given $(\pi^{(l)}, \tilde{\pi}^{(l)}, \sigma^{(l)})$, we first update $\pi^{(l+1)}$ and $\tilde{\pi}^{(l+1)}$:

$$\pi^{(l+1)} = \arg\min_{\pi\in S_\mu} \langle C, \pi\rangle + \left\langle \tilde{\sigma}^{(l)}, \pi \right\rangle + t\mathrm{KL}(\pi, \tilde{\pi}^{(l)}) \tag{29}$$

$$\tilde{\pi}^{(l+1)} = \arg\min_{\tilde{\pi}\in S_\nu} \left\langle \sigma^{(l)}, -\tilde{\pi} \right\rangle + t\mathrm{KL}(\tilde{\pi}, \pi^{(l+1)}) \tag{30}$$

Both 29 and 30 have closed form solution

$$\pi_{i,j}^{(l+1)} = \frac{\tilde{\pi}_{i,j}^{(l)} \exp(-\frac{C_{i,j}+\sigma_{i,j}^{(l)}}{t})}{\sum_{j=1}^{n} \tilde{\pi}_{i,j}^{(l)} \exp(-\frac{C_{i,j}+\sigma_{i,j}^{(l)}}{t})}\mu_i, \quad \tilde{\pi}_{i,j}^{(l+1)} = \frac{\pi_{i,j}^{(l+1)} \exp(\frac{\sigma_{i,j}^{(l)}}{t})}{\sum_{i=1}^{m} \pi_{i,j}^{(l+1)} \exp(\frac{\sigma_{i,j}^{(l)}}{t})}\nu_j \tag{31}$$

We update $\sigma^{(l+1)}$ as

$$\sigma^{(l+1)} = \sigma^{(l)} + t(\pi^{(l+1)} - \tilde{\pi}^{(l+1)}) \tag{32}$$

Then we get the following Bregman ADMM, Algorithm 4.

Algorithm 4 Bregman ADMM for Optimal Transport

```
Input: $C \in \mathbb{R}^{m\times n}$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^n$, $t > 0$
1: Initialize $\tilde{\pi}^{(0)} = \mu\nu^T$, set $\sigma^{(0)} = \mathbf{0}_{n\times m}$, $l = 0$
2: while not converge do
3:     Update $(\pi^{(l+1)}, \tilde{\pi}^{(l+1)}, \sigma^{(l+1)})$ by 31 and 32, $l = l + 1$
4: end while
5: return $\frac{\pi^{(l)}+\tilde{\pi}^{(l)}}{2}$
```

[8] proves that when $m = n$, BADMM can be faster than ADMM by a factor of $O(n/\log(n))$.

3 Entropy Regularization of Optimal Transport

3.1 Background and Formulation

3.1.1 Kantorovich formulation of optimal transport

[4] introduces the Kantorovich formulation of optimal transport (Kantorovich's relaxation). For two vectors $\mu \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^n$, which satisfies $\mathbb{1}_n^T\mu = \mathbb{1}_m^T\nu$, we consider the following set of coupling matrices

$$U(\mu, \nu) \triangleq \left\{\pi \in \mathbb{R}_+^{m\times n} : \pi\mathbb{1}_n = \mu, \pi^T\mathbb{1}_m = \nu\right\} \tag{33}$$

where we use the following matrix-vector notation

$$\pi\mathbb{1}_n = (\sum_j \pi_{i,j})_i \in \mathbb{R}^m, \quad \pi\mathbb{1}_m = (\sum_i \pi_{i,j})_j \in \mathbb{R}^n,$$

The Kantorovich's optimal transport problem reads

$$L_{\mathbf{c}}(\mu,\nu) \triangleq \min_{\pi\in U(\mu,\nu)} \langle C,\pi\rangle \triangleq \sum_{i,j} C_{i,j}\pi_{i,j} \tag{34}$$

Therefore, problem 34 is equivalent to the standard form of LP 1.

3.1.2 Entropy regularization

For a coupling matrix $\pi \in U(\mu,\nu)$, the discrete entropy of π is defined as

$$H(\pi) \triangleq -\sum_{i,j} \pi_{i,j}(\log(\pi_{i,j}) - 1) \tag{35}$$

with an analogous definition for vectors, with the convention that $H(\pi) = -\infty$ if one of the entries $\pi_{i,j}$ is negative. Although discrete entropy is defined on matrices with positive entry, with $\lim\limits_{x\to 0+} x(\log x - 1) = 0$, we can extend its definition to matrices with non-negative entry. And we have

$$H(\pi) = -\sum_{\pi_{i,j}\geqslant 0} \pi_{i,j}(\log(\pi_{i,j}) - 1)$$

Suppose $M = \min\{\|\mu\|_\infty, \|\nu\|_\infty\}$, then we have $\pi_{i,j} \leqslant \min\{\mu_i,\nu_j\} \leqslant M$. The function $H(\pi)$ is then $1/M$-strongly convex, because its Hessian is $\partial^2 H(\pi) = -\mathrm{diag}(1/\pi_{i,j})$.

The idea of entropic regularization of optimal transport is to use $-H$ as a regularizing function to obtain approximate solutions to the Kantorovich's optimal transport problem 34.

$$L_C^{\epsilon}(\mu,\nu) \triangleq \min_{\pi\in U(\mu,\nu)} \langle C,\pi\rangle - \epsilon H(\pi) \tag{36}$$

The convergence of the solution of that regularized problem towards an optimal solution of the original linear program has been studied by [2].

Proposition 1 (Convergence with ϵ) *The unique solution π^ϵ of 36 converges to the optimal solution with maximal entropy within the set of all optimal solutions of the Kantorovich problem, namely*

$$\pi^\epsilon \to \arg\min_{\pi}\{-H(\pi) : \pi \in U(\mu,\nu), \langle \pi, C\rangle = L_C(\mu,\nu)\}, \quad \epsilon \to 0 \tag{37}$$

so that in particular

$$L_C^\epsilon(\mu,\nu) \to L_C(\mu,\nu), \quad \epsilon \to 0$$

Proof We consider a sequence $\{\epsilon_l\}$ such that $\epsilon_l \to 0$ and $\epsilon_l > 0$. We denote π^l the solution to 36 for $\epsilon = \epsilon_l$. Since $U(\mu, \nu)$ is bounded, we can extract a sequence (that we do not relabel for sake of simplicity) such that $\pi^l \to \pi^*$. Since $U(\mu, \nu)$ is closed, $\pi^* \in U(\mu, \nu)$. We consider any π such that $\langle C, \pi \rangle = L_C(\mu, \nu)$. By the optimality of π and π^l for their respective optimization problem, we have

$$0 \leqslant \left\langle C, \pi^l \right\rangle - \langle C, \pi \rangle \leqslant \epsilon_l (H(\pi^l) - H(\pi)) \tag{38}$$

Since H is continuous, taking the limit $l \to \infty$ in this expression shows that $\langle C, \pi^* \rangle = \langle C, \pi \rangle$ so that π^* is a feasible point of 37. Furthermore, we have $H(P^*) \geqslant H(P)$, which shows that P^* is the solution to 37. By strict convexity of $-H$, the solution is unique and the whole sequence is converging.

The following figures 1 and 2 illustrates the impact of ϵ on the coupling π between two Gaussian mixture models.

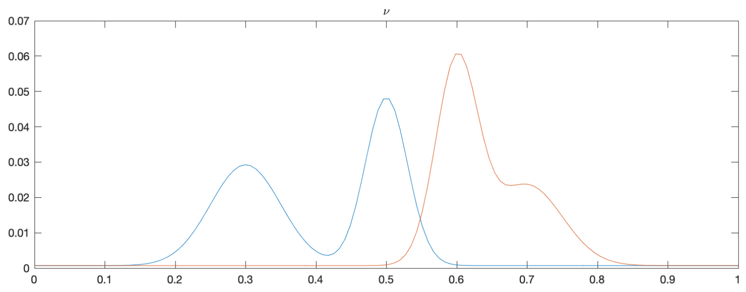

Figure 1 The probability distribution function of two Gaussian mixture models G_1 and G_2. G_1 (blue one) has mean $[0.3, 0.5]$, variance $[0.05^2, 0.03^2]$ and component proportion $[0.5, 0.5]$; G_2 (red one) has mean $[0.6, 0.7]$, variance $[0.03^2, 0.05^2]$ and component proportion $[0.6, 0.4]$.

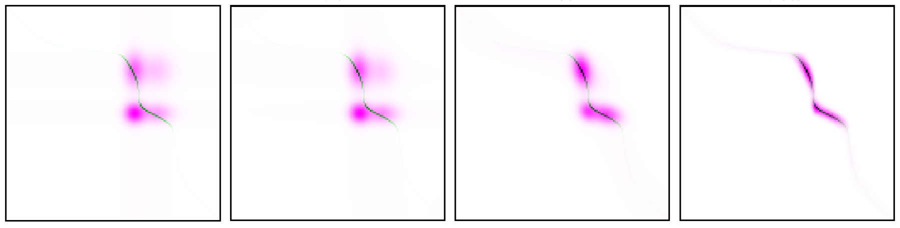

Figure 2 Impact of ϵ on the coupling between two gaussian mixture models 1. From left to right, $\epsilon = 1, 0.1, 0.01, 0.001$. The purple one is the solution to the entropy regularized LP. The green one is the solution to the original LP.

Define the Kullback-Leibler divergence between couplings as

$$\mathrm{KL}(\pi|K) \triangleq \sum_{i,j} \pi_{i,j} \log(\frac{\pi_{i,j}}{K_{i,j}}) - \pi_{i,j} + K_{i,j}$$

We take $K^{\epsilon}_{i,j} \triangleq e^{-\frac{C_{i,j}}{\epsilon}}$, the Gibbs kernel associated to the cost matrix C. The unique solution π^{ϵ} of 36 is a projection onto $U(\mu, \nu)$, namely

$$\pi^{\epsilon} \triangleq \arg \min_{\pi \in U(\mu,\nu)} \mathrm{KL}(\pi|K^{\epsilon})$$

3.1.3 Formulation of the problem

The optimization problem of the entropic regularization of optimal transport with parameter ϵ is given by:

$$\begin{cases} \min \langle C, \pi \rangle - \epsilon H(\pi) = \mathrm{KL}(\pi|K^{\epsilon}) \\ \text{s.t.}\, \pi \in U(\mu, \nu) \end{cases} \tag{39}$$

where $K^{\epsilon}_{i,j} = e^{-\frac{C_{i,j}}{\epsilon}}$.

3.2 Sinkhorn's Algorithm

We start with the following proposition.

Proposition 2 *The solution to 39 is unique and has the form*

$$\forall 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n, \quad \pi^{\epsilon}_{i,j} = a_i K^{\epsilon}_{i,j} b_j \tag{40}$$

for two (unknown) scaling variable $(a, b) \in \mathbb{R}^m_+ \times \mathbb{R}^n_+$.

Proof Introducing two dual variable $f \in \mathbb{R}^m, g \in \mathbb{R}^n$ for each marginal constraint. The Lagrangian of 39 reads

$$L(\pi, f, g) = \langle \pi, C \rangle - \epsilon H(P) - \langle f, \pi \mathbb{1}_n - \mu \rangle - \left\langle g, \pi^T \mathbb{1}_m - \nu \right\rangle$$

The first order conditions yield

$$\frac{\partial L}{\partial \pi_{i,j}} = C_{i,j} + \epsilon log(P_{i,j}) - f_i - g_j = 0$$

This means that an optimal π coupling to the regularized problem can be written as $P_{i,j} = e^{f_i/\epsilon}e^{-C_{i,j}/\epsilon}e^{g_j/\epsilon}$. Let $a = e^{f_i/\epsilon}, b = e^{g_j/\epsilon}$, then we have

$$\pi_{i,j} = a_i K^{\epsilon}_{i,j} b_j$$

This shows that the solution of 39 has a specific form, which can be parameterized using $m + n$ variables.

The parameterization of the optimal solution in equation 40 can be written in the matrix form $\pi^{\epsilon} = \text{diag}(a)K^{\epsilon}\text{diag}(b)$. Then, a, b must satisfies:

$$\text{diag}(a)K^{\epsilon}\text{diag}(b)\mathbb{1}_m = \mu, \quad \text{diag}(a)K^{T}\text{diag}(b)\mathbb{1}_n = \nu \tag{41}$$

These two equations can be further simplified as

$$a \odot (K^{\epsilon}b) = \mu, \quad b \odot \left((K^{\epsilon})^T a\right) = \nu \tag{42}$$

where $\odot$ corresponds to entry-wise multiplication of vectors. The Sinkhorn's algorithm iteratively update $a^{(l+1)}$ and $b^{(l+1)}$ by

$$a^{(l+1)} = \frac{\mu}{K^{\epsilon}b^{(l)}}, \quad b^{l+1} = \frac{\nu}{(K^{\epsilon})^T a^{(l+1)}} \tag{43}$$

initialized with an arbitrary positive vector $b^{(0)} = \mathbb{1}_n$. The division operator used above between two vectors is to be understood entry-wise. The whole algorithm is given by:

Algorithm 5 Sinkhorn's algorithm

```
Input: $\epsilon, C \in \mathbb{R}^{m\times n}, \mu \in \mathbb{R}^m, \nu \in \mathbb{R}^n$.
1: Start with $b^{(0)} = \mathbb{1}_n, l = 0$
2: Calculate $K^{\epsilon}_{i,j} = e^{-\frac{C_{i,j}}{\epsilon}}$
3: while not converge do
4:     Update $a^{(l+1)}$ and $b^{(l+1)}$ by 43, $l = l + 1$
5: end while
6: $a = a^{(l)}, b = b^{(l)}$
7: return $\pi = \text{diag}(a)K^{\epsilon}\text{diag}(b)$
```

This algorithm was originally introduced with a proof of convergence by [7]. For simplicity, we assume $m = n$. [1] showed that by setting $\epsilon = \frac{\tau}{4\log(n)}$, $O(\|C\|^3_{\infty}\log(n)\tau^{-3})$ Sinkhorn iterations (with an additional rounding step to compute a valid coupling $\hat{\pi} \in U(\mu, \nu)$) are enough to ensure $\langle \hat{\pi}, C\rangle \leqslant L_C(\mu,\nu)+\tau$. This implies that Sinkhorn computes a τ-approximate solution of the unregularized OT problem in $O(n^2\log(n)\tau^{-3})$ operations. The rounding scheme consists in, given two vectors $a\in\mathbb{R}^m$, $b\in\mathbb{R}^n$ to carry out the following updates:

$$a' = a\odot \min\left(\frac{\mu}{a\odot(K^\epsilon b)}, \mathbb{1}_m\right),\quad b' = b\odot \min\left(\frac{\nu}{b\odot((K^\epsilon)^T a')}, \mathbb{1}_n\right)$$

$$\Delta_\mu = \mu - a'\odot(K^\epsilon b'),\quad \Delta_\nu = \nu - b'\odot\left((K^\epsilon)^T a'\right)$$

$$\hat{\pi} = \text{diag}(a')K^\epsilon\text{diag}(b') + \Delta_\mu\Delta_\nu^T/\|\Delta_\mu\|_1$$

This yields a matrix $\hat{\pi}\in U(\mu,\nu)$ such that the 1-norm of $\hat{\pi}-\text{diag}(a)K^\epsilon\text{diag}(b)$ is controlled by the marginal violations of $\text{diag}(a)K^\epsilon\text{diag}(b)$, namely

$$\|\hat{\pi}-\text{diag}(a)K^\epsilon\text{diag}(b')\|_1 \leqslant \|\mu - a\odot(K^\epsilon b)\|_1 + \|\nu - b\odot\left((K^\epsilon)^T a\right)\|_1$$

In actual implementation, this step is crucial to decrease the violation of constraints.

3.3 Our proposed algorithm

3.3.1 Sinkhorn's algorithm with numerical stability and continuation strategy

We denote $\|C\|_\infty = \max_{i,j}|C_{i,j}|$. In actual implementation of Sinkhorn's algorithm 5, because we have $K^\epsilon_{i,j} = e^{-\frac{C_{i,j}}{\epsilon}}$, when ϵ is small, namely $\epsilon < \frac{\|C\|_\infty}{806}$, the largest entry of K is smaller than $e^{-806}\approx 10^{-350}$. Most of the entry of K^ϵ would be rounded to 0 because they are smaller than the smallest positive number that the computer can restore. If we want to calculate K^ϵ, we would have a positive lower bound $\frac{\|C\|_\infty}{806}$ for ϵ. This is not desired and it would make the result from proposition 1 meaningless.

We propose a numerical stable version of Sinkhorn's algorithm with no lower bound of ϵ. A possible way to remove this lower bound of ϵ is to avoid directly calculating K^ϵ to update $a^{(l+1)}$ and $b^{(l+1)}$. Therefore, from the proof of proposition 2, we denote $a^{(l)} = e^{f^{(l)}/\epsilon}$, $b^{(l)} = e^{g^{(l)}/\epsilon}$, where $f^{(l)}\in R^m, g^{(l)}\in R^n$. Then, the update rule for $f^{(l+1)}$ and $g^{(l+1)}$ reads

$$f^{(l+1)} = \epsilon(\log(\mu) - \log(K^\epsilon e^{g^{(l)}/\epsilon})),\quad g^{(l+1)} = \epsilon(\log(\nu) - \log((K^\epsilon)^T e^{f^{(l+1)}/\epsilon})), \tag{44}$$

Our motivation is to caculate $\log(K^\epsilon e^{g^{(l)}/\epsilon})$ and $\log((K^\epsilon)^T e^{f^{(l+1)}/\epsilon})$ in update rule 44 in a numerically stable way. We start with the update of $f^{(l+1)}$. It is easy to verify that

$$f_i^{(l+1)} = \epsilon \log(\mu_i) - \epsilon \log(\sum_{j=1}^{n} e^{(-C_{i,j}+g_j^{(l)})/\epsilon})$$

Directly calculating $\sum_{j=1}^{n} e^{(-C_{i,j}+g_j^{(l)})/\epsilon}$ is dangerous because it might be rounded to 0. Let $\hat{f}_i^{(l+1)} = \max_j\{-C_{i,j} + g_j^{(l)}\}$. Then, to update f with numerical stability, we can update $f_i^{(l+1)}$ by

$$f_i^{(l+1)} = \epsilon \log(\mu_i) - \hat{f}_i^{(l)} - \epsilon \log(\sum_{j=1}^{n} e^{(-C_{i,j}+g_j^{(l)}-\hat{f}_i^{(l)})/\epsilon}) \tag{45}$$

Because $\sum_{j=1}^{n} e^{(-C_{i,j}+g_j^{(l)}-\hat{f}_i^{(l)})/\epsilon} \geqslant 1$, to update $f^{(l+1)}$ in this way is safe. Actually, we use $\hat{f}_i^{(l)}/\epsilon + \log(\sum_{j=1}^{n} e^{(-C_{i,j}+g_j^{(l)}-\hat{f}_i^{(l)})/\epsilon})$ to calculate $\log(K^\epsilon e^{g^{(l)}/\epsilon})$. Similarly, we let $\hat{g}_j^{(l+1)} = \max_i\{-C_{i,j} + f_i^{(l+1)}\}$, and update $g_j^{(l+1)}$ by

$$g_j^{(l+1)} = \epsilon \log(\nu_j) - \hat{g}_j^{(l+1)} - \epsilon \log(\sum_{i=1}^{n} e^{(-C_{i,j}+f_i^{(l+1)}-\hat{g}_j^{(l+1)})/\epsilon}) \tag{46}$$

In the end of the algorithm, we also use an additional rounding step to compute a valid coupling $\hat{\pi}$. Given two vectors $f \in \mathbb{R}^m, g \in \mathbb{R}^n$ and coupling matrix $\pi \in \mathbb{R}^{m\times n}$ with $\pi_{i,j} = e^{(f_i - C_{i,j}+g_j)/\epsilon}$, we compute $\hat{\pi}$ in the following way:

$$\begin{aligned}
&\hat{f}_i = \max_j\{-C_{i,j} + g_j\}, \quad f_i' = f_i + \min\{\epsilon(\log(\mu_i) - \sum_{j=1}^{n} e^{(-C_{i,j}+g_j-\hat{f}_i)/\epsilon}) - \hat{f}_i - f_i, 0\}\\
&\hat{g}_j = \max_i\{-C_{i,j} + f_i'\}, \quad g_j' = g_j + \min\{\epsilon(\log(\nu_j) - \sum_{i=1}^{m} e^{(-C_{i,j}+f_i'-\hat{g}_j)/\epsilon}) - \hat{g}_j - g_j, 0\}\\
&\hat{f}_i' = \max_j\{-C_{i,j} + g_j'\}, \quad (\Delta_\mu)_i = \mu_i - (\sum_{j=1}^{n} e^{(-C_{i,j}+g_j'-\hat{f}_i')/\epsilon}) e^{(f_i'+\hat{f}_i')/\epsilon}\\
&(\Delta_\nu)_j = \nu_j - (\sum_{i=1}^{n} e^{(-C_{i,j}+f_i'-\hat{g}_j')/\epsilon}) e^{(g_j'+\hat{g}_j')/\epsilon}, \quad \hat{\pi} = \pi + \Delta_\mu \Delta_\nu^T / \|\Delta_\mu\|_1
\end{aligned} \tag{47}$$

Then, we get the following Sinkhorn's algorithm with numerical stability. It shares the same idea with Log-domain Sinkhorn's algorithm [4].

We shall point out that $\epsilon > 0$ is vital in the iteration 45 and 46 because $f^{(l)}$ and $g^{(l)}$ will converge to the optimal solution to the dual problem of 39 by Proposition 2. If we let $\epsilon \to 0$

Algorithm 6 Sinkhorn's algorithm with numerical stability

```
Input: $\epsilon, C \in \mathbb{R}^{m\times n}, \mu \in \mathbb{R}^m, \nu \in \mathbb{R}^n$.
1: Start with $g^{(0)} = \mathbf{0}_n, l = 0$
2: while not converge do
3:     Calculate $\hat{f}_i^{(l+1)} = \max_j\{-C_{i,j} + g_j^{(l)}\}$ and update $f^{(l+1)}$ by 45
4:     Calculate $\hat{g}_i^{(l+1)} = \max_i\{-C_{i,j} + f_i^{(l+1)}\}$ and update $g^{(l+1)}$ by 46, $l = l + 1$
5: end while
6: Let $f = f^{(l)}, g = g^{(l)}$, calculate $\pi_{i,j} = e^{(f_i - C_{i,j} + g_j)/\epsilon}$, and compute $\hat{\pi}$ by 47
7: return $\hat{\pi}$
```

in 45 and 46, we would find that these updates become as follows:

$$f_i^{(l+1)} = -\hat{f}_i^{(l)} = \min_j\{C_{i,j} - g_j^{(l)}\}, \quad g_j^{(l+1)} = -\hat{g}_j^{(l+1)} = \max_i\{C_{i,j} - f_i^{(l+1)}\} \tag{48}$$

Nevertheless, $f^{(l)}$ and $g^{(l)}$ in this iteration 48 will not converge to the solution to the dual problem of 1. With $\epsilon = 0$, we lose the strong convexity of the objective function and, therefore, the solution to the dual problem of 1 is not unique. As a result, $f^{(l)}$ and $g^{(l)}$ may stuck in a plateau. This is also pointed out in Section 3.2 of [4].

We can also apply continuation strategy in Sinkhorn's algorithm with numerical stability. We have three additional parameters ϵ_0, α, M for continuation. Namely, we start with a large ϵ_0 and $k = 0$. Then, until $\epsilon_k = \epsilon$, we gradually decrease the value of $\epsilon_{k+1} = \max\{\alpha\epsilon_k, \epsilon\}$ and set $k = k + 1$ after M iterations. In each iteration, $f_i^{(l+1)}$ and $g_j^{(l+1)}$ are updated as follows:

$$f_i^{(l+1)} = \epsilon_k \log(\mu_i) - \hat{f}_i^{(l)} - \epsilon_k \log(\sum_{j=1}^{n} e^{(-C_{i,j} + g_j^{(l)} - \hat{f}_i^{(l)})/\epsilon_k}) \tag{49}$$

$$g_j^{(l+1)} = \epsilon_k \log(\nu_j) - \hat{g}_j^{(l+1)} - \epsilon_k \log(\sum_{i=1}^{n} e^{(-C_{i,j} + f_i^{(l+1)} - \hat{g}_j^{(l+1)})/\epsilon_k}) \tag{50}$$

In the end, we still use an additional rounding step to compute a valid coupling $\hat{\pi}$. The algorithm is given below:

Algorithm 7 Sinkhorn's algorithm with numerical stability and continuation strategy

```
Input: $\epsilon, C \in \mathbb{R}^{m\times n}, \mu \in \mathbb{R}^m, \nu \in \mathbb{R}^n$, continuation parameter $\epsilon_0, \alpha, M$.
 1: Start with $g^{(0)} = \mathbf{0}_n$, $k = 0$.
 2: while $\epsilon_k > \epsilon$ do
 3:     for $l = 0 : M - 1$ do
 4:         Calculate $\hat{f}_i^{(l+1)} = \max_j\{-C_{i,j} + g_j^{(l)}\}$ and update $f^{(l+1)}$ by 49
 5:         Calculate $\hat{g}_i^{(l+1)} = \max_i\{-C_{i,j} + f_i^{(l+1)}\}$ and update $g^{(l+1)}$ by 50
 6:     end for
 7:     Set $\epsilon_{k+1} = \max\{\alpha\epsilon_k, \epsilon\}$, $f^{(0)} = f^{(M)}$, $g^{(0)} = g^{(M)}$ and $k = k + 1$
 8: end while
 9: for $l = 0 : M - 1$ do
10:     Calculate $\hat{f}_i^{(l+1)} = \max_j\{-C_{i,j} + g_j^{(l)}\}$ and update $f^{(l+1)}$ by 49
11:     Calculate $\hat{g}_i^{(l+1)} = \max_i\{-C_{i,j} + f_i^{(l+1)}\}$ and update $g^{(l+1)}$ by 50
12: end for
13: Let $f = f^{(l)}, g = g^{(l)}$, calculate $\pi_{i,j} = e^{(f_i - C_{i,j} + g_j)/\epsilon}$, and compute $\hat{\pi}$ by 47
14: return $\hat{\pi}$
```

3.4 First-Order Method: ADMM

To implement a first-order algorithm to solve Entropy Regularization of OT 39, we again choose ADMM. We reformulate the problem as follows,

$$\begin{aligned} &\min_{\pi,\tilde{\pi}} \ \langle C, \pi\rangle - \epsilon H(\tilde{\pi}) \\ &\text{s.t. } \pi = \tilde{\pi} \in U(\mu, \nu) \end{aligned} \tag{51}$$

which is, equivalently,

$$\begin{aligned} &\min_{\pi,\tilde{\pi}} \ \sum_{i,j} C_{i,j}\pi_{i,j} + \epsilon \sum_{i,j} \tilde{\pi}_{i,j}(\log(\tilde{\pi}_{i,j}) - 1) \\ &\text{s.t. } \sum_j \pi_{i,j} = \mu_i, \ \sum_i \pi_{i,j} = \nu_j, \ \pi = \tilde{\pi} \geqslant 0. \end{aligned} \tag{52}$$

The augmented Lagrangian function is

$$\begin{aligned}
&L_t(\pi, \tilde{\pi}, \gamma, \lambda, \omega) \\
&= \sum_{i,j} C_{i,j}\pi_{i,j} + \sum_i \gamma_i \left(\mu_i - \sum_j \pi_{i,j}\right) + \sum_j \lambda_j \left(\nu_j - \sum_i \pi_{i,j}\right) + \sum_{i,j} \omega_{i,j}(\tilde{\pi}_{i,j} - \pi_{i,j}) \\
&\quad + \frac{t}{2}\left(\sum_i \left(\mu_i - \sum_j \pi_{i,j}\right)^2 + \sum_j \left(\nu_j - \sum_i \pi_{i,j}\right)^2 + \sum_{i,j}(\pi_{i,j} - \tilde{\pi}_{i,j})^2\right) \\
&\quad + \epsilon \sum_{i,j} \tilde{\pi}_{i,j}(\log(\tilde{\pi}_{i,j}) - 1)
\end{aligned} \tag{53}$$

For the update of π, which is similar to 5 in the previous sections, we have

$$\begin{aligned}
\pi_{i,j} \leftarrow x_{i,j} &- \frac{1}{m+1}\left(\sum_k x_{k,j} - \frac{1}{m+n+1}\sum_{k,l} x_{k,l}\right) \\
&- \frac{1}{n+1}\left(\sum_k x_{i,k} - \frac{1}{m+n+1}\sum_{k,l} x_{k,l}\right)
\end{aligned} \tag{54}$$

where

$$x_{i,j} = \frac{1}{t}(\gamma_i + \lambda_j + \omega_{i,j} - c_{i,j}) + \mu_i + \nu_j + \tilde{\pi}_{i,j} \tag{55}$$

For the update of $\tilde{\pi}$, setting $\frac{\partial L_t}{\partial \tilde{\pi}} = 0$, we need to solve the following problem for each $\tilde{\pi}_{i,j}$,

$$\omega_{i,j} + t(\tilde{\pi}_{i,j} - \pi_{i,j}) + \epsilon \log \tilde{\pi}_{i,j} = 0 \tag{56}$$

Such equation has no closed form solution, though the left hand side is strictly increasing with $\tilde{\pi}_{i,j}$ and its range is $\mathbb{R}$(t and ϵ are both positive numbers). Therefore, we seek to update one step ahead with Newton Method in each iteration as well as sustain the nonnegativity, i.e.,

$$\tilde{\pi}_{i,j} \leftarrow \max\left\{\tilde{\pi}_{i,j} - \frac{\omega_{i,j} + t(\tilde{\pi}_{i,j} - \pi_{i,j}) + \epsilon \log \tilde{\pi}_{i,j}}{t + \epsilon/\tilde{\pi}_{i,j}}, 0\right\} \tag{57}$$

However, $\tilde{\pi}_{i,j}$ may possibly be zero, which may lead to numerical explosion in 57. Thus, every time before 57, we let

$$\tilde{\pi}_{i,j} \leftarrow \tilde{\pi}_{i,j} + \tilde{\epsilon}, \tag{58}$$

where $\tilde{\epsilon}$ is a very small number. In our implementation, we let $\tilde{\epsilon}$ be 1e-16.

Update of multipliers are as follows,

$$\begin{aligned} \gamma_i &\leftarrow \gamma_i + t(\mu_i - \sum_k \pi_{i,k}) \\ \lambda_j &\leftarrow \lambda_j + t(\nu_j - \sum_k \pi_{k,j}) \\ \omega_{i,j} &\leftarrow \omega_{i,j} + t(\tilde{\pi}_{i,j} - \pi_{i,j}) \end{aligned} \tag{59}$$

We thus have the following Algorithm 8.

Algorithm 8 ADMM for Entropy Regularization

```
Input: $C \in \mathbb{R}^{m\times n}$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^n$, $t, \epsilon, \tilde{\epsilon} \in \mathbb{R}^+$
1: $\pi, \tilde{\pi}, \gamma, \lambda, \omega \leftarrow 0$
2: while not converge do
3:     Update $\pi$ according to 54, 55
4:     Update $\tilde{\pi}$ according to 58, 57
5:     Update $\gamma$, $\lambda$ and $\omega$ by 59
6: end while
7: return $\pi$
```

4 Numerical Experiment

In this section, we apply various methods to solve the LP form of optimal transport 1 and the entropic regularization of optimal transport 39. All of the numerical experiments are conducted on an Intel Core i7-6500U Processor with 2 cores and 4 threads.

4.1 Datasets of Optimal Transport

We perform the numerical experiments on various datasets, including randomly generated data, DOTmark[6], ellipse example [5], Caffarelli's example [5] and Gaussian mixture model.

4.1.1 Randomly generated data

In order to make the feasible set not empty, we first generate a random matrix $R \in \mathbb{R}^{m\times n}$, where $R_{i,j} \sim \mathcal{N}(0,1)\, i.i.d..$ and then generate μ_i, ν_j by

$$\mu_i = \frac{\sum_{j=1}^n |R_{i,j}|}{\sum_{i=1}^m \sum_{j=1}^n |R_{i,j}|}, \forall i = 1, 2, \dots m$$

$$\nu_j = \frac{\sum_{i=1}^m |R_{i,j}|}{\sum_{i=1}^m \sum_{j=1}^n |R_{i,j}|}, \forall j = 1, 2, \dots n$$

It is easy to verify that $\sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j = 1$. We then generate $C_{i,j} \sim \mathcal{N}(0,1)\, i.i.d..$ We denote $C_{\min} = \min_{i,j} C_{i,j}$ and let $C_{i,j} = C_{i,j} - C_{\min}$ to ensure $C_{i,j} \geq 0$.

4.1.2 DOTmark

DOTmark [6] provides 10 classes of 10 different images, each of which is available at the different resolutions from 32×32 to 512×512. Suppose we select two images x and y with same resolution $l \times l$ from one class. The corresponding parameter for the standard form of LP is given in the following way: $m = n = l^2$, $\mu_i = x_i$, $\nu_j = y_j$ and $C_{i,j} = \|x_i - y_j\|_p^p$, where $(x_i)_{1\leq i\leq m}$ and $(y_j)_{1\leq j\leq n}$ form a regular square grid in $\mathbb{R}^2$. In practice, we take $p = 2$. In solving this LP, we actually calculate the Wasserstein distance between x and y.

4.1.3 Ellipse Example

The ellipse example consists of two uniform samples, source and target data set, of size n from the unit circle with normal distributed noise added with zero mean and deviation 0.1. Slightly different from that in the paper [5], we construct the source data example by scaling in the x-Axis by 2.0 and in the y-Axis by 0.5. The target data sample is then scaled in the x-Axis by 0.5 and y-Axis by 2.0. The corresponding parameter for the standard form of LP is as follows. $m = n$, $\mu_i = 1, \nu_i = 1 (i = 1, ..., n)$ and $C_{i,j} = \|x_i - y_j\|_2$, which is the Euclidean distance between two points x_i and y_j. Figure 3 gives an example.

4.1.4 Caffarelli's Example

This example, mentioned in [5], consists of two uniform samples on $[-1, 1]^2$ of size n. Then any point outside the unit circle are then discarded. The source and target data sample is split along the x-Axis at 0 and shifted by -2 and +2, respectively. Note that the true number of points may be less than n. When n is large, about $\frac{\pi}{4}n$ points are preserved. For ease of

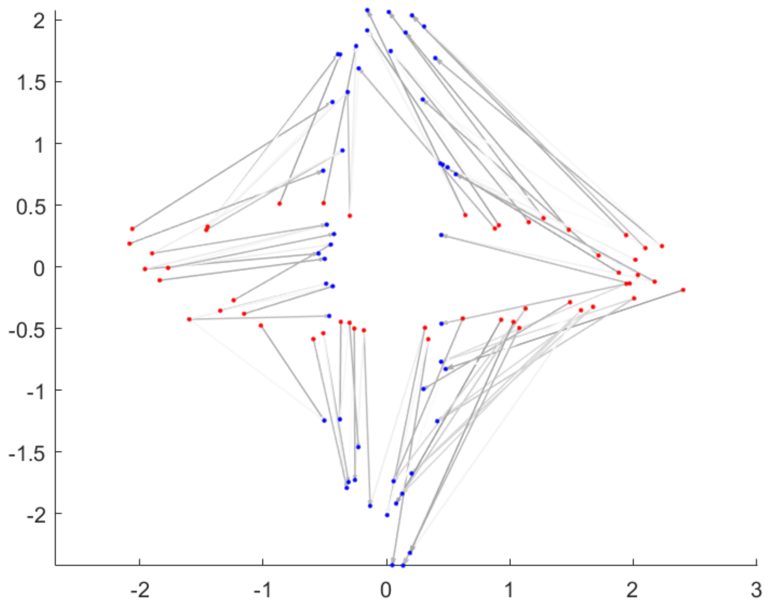

Figure 3 An Ellipse Example with sample size $n = 50$. The darker the line, the greater the $\pi_{i,j}$.

notation, we still assume there are n points. The corresponding parameter for the standard form of LP is as follows. $m = n$, $\mu_i = 1, \nu_i = 1 (i = 1, ..., n)$ and $C_{i,j} = \|x_i - y_j\|_2$, which is the Euclidean distance between two points x_i and y_j. Figure 4 gives an example.

4.1.5 Gaussian Mixture Model

Suppose p_μ and p_ν are the probability density function of two different Gaussian mixture models and N is the number of discrete points. In this example, μ and ν are the normalized discrete distribution of Gaussian mixture models, which satisfy

$$\mu_i = \frac{p_\mu(\frac{i-1}{N-1})}{\sum_{i=1}^{N} p_\mu(\frac{i-1}{N-1})}, \quad \nu_j = \frac{p_\nu(\frac{j-1}{N-1})}{\sum_{j=1}^{N} p_\nu(\frac{j-1}{N-1})}$$

And $C_{i,j} = |i - j|^2$.

4.2 Different methods on Mosek and Gurobi

'prim', 'dual', 'int' represent primal simplex, dual simplex and interior point method respectively. '(M)' means Mosek, '(G)' means Gurobi. For each method, we record the time and the number of iteration it takes and evaluate their performance. We examine the accuracy

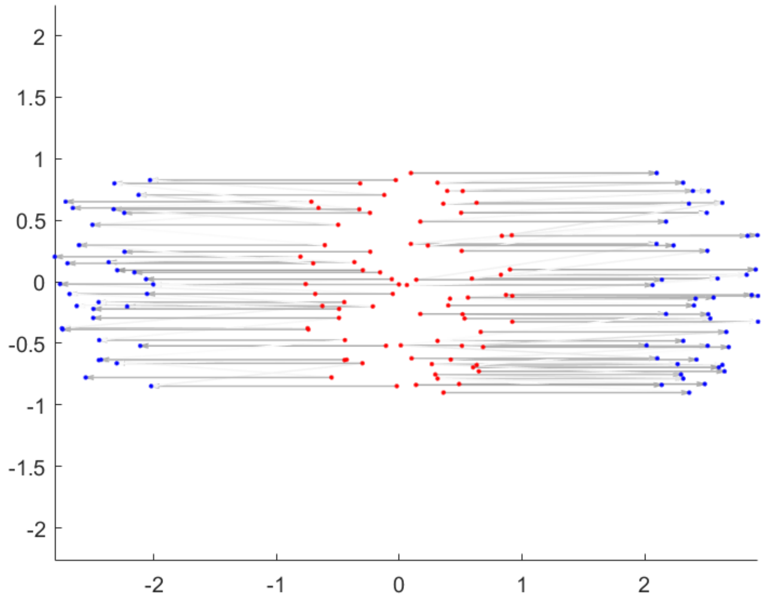

Figure 4 An Caffarelli's Example with sample size $n = 50$. The darker the line, the greater the $\pi_{i,j}$.

of the solution π by the value of the objective function ('objval')

$$\sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j}\pi_{i,j}$$

and the violation of the constraints ('vltcst')

$$\sum_{i=1}^{m} |\mu_i - \sum_{j=1}^{n} \pi_{i,j}| + \sum_{j=1}^{n} |\nu_j - \sum_{i=1}^{m} \pi_{i,j}|$$

In the table, we only give the exact value of 'objval' for prim(M). In 'objval' of other methods, we give its relative difference to prim(M). Namely, if 'objval' for prim(M) is f_1 and 'objval' for prim(G) is f_2, in 'objval' of prim(G), we give $\frac{f_1 - f_2}{f_1}$.

4.2.1 Randomly generated data

For simplicity, we only consider the case when $m = n$. We take $m = 128, 256, 512, 1024$.

From Table 1, we can see on Mosek, dual simplex method and interior point method take much more time than primal simplex does, especially when m, n are large. But on Gurobi, dual simplex method is the fastest method and the time primal simplex method takes is close to the time interior point takes. Interior point method takes much smaller number of iterations

Table 1 Perfomance of mosek and gurobi on randomly generated data

method		prim(M)	dual(M)	int(M)	prim(G)	dual(G)	int(G)
$m = 128$ $n = 128$	time(s)	0.05	0.04	0.08	0.07	0.03	0.08
	iter	1488	220	13	3101	204	13
	objval	1.36e+00	-6.08e-06	-6.08e-06	-6.08e-06	-6.08e-06	-6.08e-06
	vltcst	1.11e-05	5.55e-17	1.14e-10	4.94e-17	4.94e-17	4.94e-17
$m = 256$ $n = 256$	time(s)	0.21	0.18	0.33	0.20	0.14	0.30
	iter	4708	480	13	9252	437	15
	objval	1.63e+00	-1.90e-05	-1.90e-05	-1.90e-05	-1.90e-05	-1.90e-05
	vltcst	3.62e-05	4.99e-17	1.05e-10	7.81e-18	6.51e-18	7.81e-18
$m = 512$ $n = 512$	time(s)	0.95	1.38	1.99	1.17	0.63	2.06
	iter	11742	472	15	32211	2266	19
	objval	1.57e+00	-4.76e-05	-4.76e-05	-4.76e-05	-4.76e-05	-4.76e-05
	vltcst	9.14e-05	1.73e-16	1.44e-12	7.42e-17	7.42e-17	7.42e-17
$m = 1024$ $n = 1024$	time(s)	5.39	11.95	11.37	6.28	3.13	12.32
	iter	28914	2299	18	108153	6020	23
	objval	1.94e+00	-1.21e-04	-1.21e-04	-1.21e-04	-1.21e-04	-1.21e-04
	vltcst	2.31e-04	3.51e-16	2.85e-11	4.68e-17	4.66e-17	4.68e-17

than We can observe that compared to other methods, primal simplex method on Mosek gets the largest value of objective function and the largest value of the violation of constraints. On Mosek, dual simplex methods achieves the lowest value of ‘vltcst’, which partly explains why it takes the longest time. On Gurobi, the solution from all methods satisfies constraints perfectly.

4.2.2 DOTmark

We then test the performance of Mosek and Gurobi on DOTmark [6]. We number the classes of DOTmark with the order of alphabet in the following way.

CauchyDensity	ClassicImages	GRFmoderate	GRFrough	GRFsmooth
1	2	3	4	5
LogGRF	LogitGRF	MicroscopyImages	Shapes	WhiteNoise
6	7	8	9	10

We test on pictures with resolution 32×32.

From Table 2, we observe that dual simplex method on Mosek takes much longer time than other methods. The cpu-time of primal simplex method on Mosek and primal simplex method and dual simplex method on Gurobi are close. On Mosek and Gurobi, the iteration number vary with different classes of DOTmark. Compared to other methods, primal simplex method on Mosek gets the largest ‘vltcst’ and the largest ‘vltcst’. On Mosek, dual simplex

Table 2 Perfomance of mosek and gurobi on DOTmark

Class No.	method	prim(M)	dual(M)	int(M)	prim(G)	dual(G)	int(G)
1	time(s)	5.53	52.84	11.34	5.17	6.49	10.51
	iter	23489	13515	16	74691	58692	17
	objval	1.75e+01	-1.54e-05	-1.53e-05	-1.53e-05	-1.54e-05	-1.53e-05
	vltcst	2.34e-05	3.28e-16	5.53e-12	1.48e-17	2.11e-17	2.53e-17
2	time(s)	5.76	26.44	11.39	4.95	4.57	11.12
	iter	22163	7634	16	85161	24036	17
	objval	6.27e+00	6.43e-07	6.43e-07	6.43e-07	6.43e-07	6.43e-07
	vltcst	2.67e-05	2.37e-16	1.92e-11	8.57e-18	1.01e-17	8.81e-18
3	time(s)	5.12	26.71	11.12	4.76	3.79	10.43
	iter	21240	7395	16	81109	16455	15
	objval	4.07e+00	2.50e-05	2.50e-05	2.50e-05	2.50e-05	2.50e-05
	vltcst	2.89e-05	3.56e-17	6.41e-14	1.11e-17	8.73e-18	1.12e-17
4	time(s)	5.09	19.64	11.11	5.10	3.01	10.73
	iter	22063	4493	17	88412	7513	16
	objval	1.48e+00	1.69e-05	1.69e-05	1.69e-05	1.69e-05	1.69e-05
	vltcst	2.77e-05	9.29e-17	9.24e-11	1.56e-17	1.38e-17	1.59e-17
5	time(s)	5.17	37.87	10.59	4.62	6.29	10.41
	iter	21509	11384	14	85058	43743	15
	objval	2.14e+01	-3.71e-06	-3.71e-06	-3.71e-06	-3.79e-06	-3.71e-06
	vltcst	2.98e-05	1.16e-16	3.72e-10	1.04e-17	7.75e-18	1.03e-17
6	time(s)	5.70	27.97	10.92	4.88	5.69	10.63
	iter	22635	8991	14	82331	40023	15
	objval	1.92e+01	5.65e-06	5.65e-06	5.65e-06	5.65e-06	5.65e-06
	vltcst	2.49e-05	2.97e-16	2.03e-11	2.20e-17	2.21e-17	1.95e-17
7	time(s)	5.32	28.00	10.98	5.07	6.10	10.14
	iter	22695	9302	14	85918	36763	15
	objval	1.69e+01	2.69e-06	2.69e-06	2.69e-06	2.69e-06	2.69e-06
	vltcst	2.57e-05	2.29e-16	1.57e-09	1.22e-17	1.35e-17	1.66e-17
8	time(s)	3.40	6.98	5.49	4.11	3.17	4.81
	iter	15322	4493	17	77555	14763	17
	objval	1.09e+01	-1.45e-05	-1.45e-05	-1.45e-05	-1.45e-05	-1.45e-05
	vltcst	1.56e-05	2.19e-16	7.59e-12	4.33e-17	4.89e-17	5.28e-17
9	time(s)	2.62	7.51	3.78	2.12	2.21	2.28
	iter	12983	5418	9	4616	2075	10
	objval	2.44e+01	1.69e-05	1.70e-05	1.69e-05	1.69e-05	1.69e-05
	vltcst	2.04e-05	7.50e-17	1.87e-08	2.08e-17	2.08e-17	2.08e-17
10	time(s)	5.41	13.07	10.51	5.08	2.60	11.20
	iter	23237	2545	14	92953	4390	17
	objval	7.09e-01	3.08e-05	3.12e-05	3.08e-05	3.08e-05	3.08e-05
	vltcst	2.60e-05	3.13e-16	1.60e-09	2.42e-17	2.17e-17	2.59e-17

methods achieves the smallest 'vltcst'. On Gurobi , the solution π from all methods satisfies the constraints perfectly.

4.2.3 Ellipse Example & Caffarelli's Example

Similar to randomly generated data, we only consider the case when $m = n$. We take $m = 128, 256, 512, 1024$.

Table 3 Perfomance of mosek and gurobi on Ellipse Example

method		prim(M)	dual(M)	int(M)	prim(G)	dual(G)	int(G)
$m = 128$ $n = 128$	time(s)	0.10	0.13	0.08	0.05	0.14	0.08
	iter	1675	1497	12	2965	1302	16
	objval	2.57e+00	-6.86e-06	-6.86e-06	-6.86e-06	-6.86e-06	-6.86e-06
	vltcst	1.33e-05	1.04e-17	7.49e-10	5.20e-18	6.07e-18	5.20e-18
$m = 256$ $n = 256$	time(s)	0.18	0.86	0.40	0.19	1.38	0.32
	iter	4112	3968	16	8966	3664	16
	objval	2.48e+00	-1.77e-05	-1.77e-05	-1.77e-05	-1.77e-05	-1.77e-05
	vltcst	3.48e-05	2.13e-17	1.11e-10	1.39e-17	1.47e-17	1.47e-17
$m = 512$ $n = 512$	time(s)	0.86	9.26	2.16	0.92	2.05	1.95
	iter	9528	12791	19	25763	42487	17
	objval	2.28e+00	-4.32e-05	-4.32e-05	-4.32e-05	-4.32e-05	-4.32e-05
	vltcst	8.55e-05	1.00e-16	3.09e-12	6.20e-17	6.14e-17	6.18e-17
$m = 1024$ $n = 1024$	time(s)	5.25	96.98	12.30	4.97	11.51	11.05
	iter	21756	33591	20	79738	152520	19
	objval	2.25e+00	-9.83e-05	-9.83e-05	-9.83e-05	-9.83e-05	-9.83e-05
	vltcst	1.96e-04	9.45e-17	4.96e-13	2.63e-17	2.69e-17	2.65e-17

Different from the results on randomly generated data, Table 3 shows that on Ellipse Example, dual simplex method takes much longer time and more iterations than primal simplex method, no matter on Mosek or on Gurobi. Similar to Ellipse Example, Table 4 shows that dual simplex method takes much longer time than the primal simplex method, no matter on Mosek or on Gurobi. The iteration number of primal simplex method and dual simplex method are close. Similarly, compared to other methods, primal simplex method on Mosek gets the largest 'vltcst' and the largest 'vltcst'. On Mosek, dual simplex methods achieves the smallest 'vltcst'. On Gurobi , the solution π from all methods satisfies the constraints perfectly.

4.3 First order methods

'prim(M)' means primal method of Mosek. 'ADMM-p' means ADMM primal method 1. 'ADMM-d' means ADMM dual method 2. 'ADMM-s' means another ADMM splitting method for primal problem 3. BADMM means Bregman ADMM 4.

Table 4 Perfomance of mosek and gurobi on Caffarelli's Example

method		prim(M)	dual(M)	int(M)	prim(G)	dual(G)	int(G)
$m = 128$ $n = 128$	time(s)	0.04	0.06	0.06	0.05	0.05	0.06
	iter	1370	795	14	2061	820	15
	objval	4.00e+00	-5.58e-06	-5.58e-06	-5.58e-06	-5.58e-06	-5.58e-06
	vltcst	1.09e-05	3.12e-17	6.02e-12	3.12e-17	3.12e-17	3.12e-17
$m = 256$ $n = 256$	time(s)	0.11	0.26	0.21	0.10	0.35	0.19
	iter	3016	1996	15	6373	2157	15
	objval	4.00e+00	-1.34e-05	-1.34e-05	-1.34e-05	-1.34e-05	-1.34e-05
	vltcst	2.60e-05	6.77e-17	5.81e-14	6.42e-17	6.42e-17	6.42e-17
$m = 512$ $n = 512$	time(s)	0.45	2.33	1.11	0.52	0.88	0.96
	iter	7536	5588	18	21406	19906	17
	objval	4.01e+00	-3.40e-05	-3.40e-05	-3.40e-05	-3.40e-05	-3.40e-05
	vltcst	6.70e-05	7.11e-17	1.13e-12	4.08e-17	4.08e-17	4.08e-17
$m = 1024$ $n = 1024$	time(s)	2.27	24.16	5.87	2.53	4.70	4.82
	iter	17237	15464	19	62128	68342	18
	objval	4.00e+00	-7.75e-05	-7.75e-05	-7.75e-05	-7.75e-05	-7.75e-05
	vltcst	1.54e-04	9.50e-17	4.83e-12	4.29e-17	4.29e-17	4.29e-17

4.3.1 Art of Tuning Parameters

The success of ADMM always lies in an appropriate choice of the coefficient t in the quadratic term of the augmented Lagrangian function. Thanks to the normalization of constraints, i.e., letting $\sum_{i=1}^{m} \mu_i = \sum_{j=1}^{n} \nu_j = 1$, the tuning of parameters is relatively easier. Intuitively, the value of t should be dependent on the problem size, m and n, and the objective coefficients C. In our implementation, for primal problem, we let t to be proportional to $(m+n)\bar{C}$, where $\bar{C}$ is the mean of the coefficient matrix C ($\frac{1}{mn}\sum_{i,j} C_{i,j}$). For dual problem, we set t to be inversely proportional to $(m+n)\bar{C}$. For the Bregman ADMM, we set t to be proportional to $\bar{C}$. The detailed proportion will be clear in the following. Empirically, we found that such strategy performs well in our experiments.

4.3.2 Randomly Generated Data

In Table 5, we present the numerical results on random generated data. Detailed tuning parameters is listed in Table 6.

In this setting, both 'ADMM-p' and 'ADMM-s' achieves relatively better results. They both satisfy the constraints quite well and achieve lower objective value. 'ADMM-p' spends moderate time in all problems because all computation has closed form. 'ADMM-s' performs quite well in 'objval' and 'vltcst', especially when the problem size is larger. However, such

Table 5 Perfomance of first order methods on randomly generated data

method		prim(M)	ADMM-p	ADMM-d	ADMM-s	BADMM
$m = 128$ $n = 128$	time(s)	0.10	2.44	1.01	3.50	1.91
	iter	1488	13405	5534	3466	7640
	objval	1.36e+00	-3.29e-06	1.32e-05	4.10e-06	3.26e-05
	vltcst	1.11e-05	5.00e-07	1.99e-04	8.29e-08	1.95e-05
$m = 256$ $n = 256$	time(s)	0.20	9.00	5.92	40.22	9.75
	iter	4708	17525	12309	11542	9986
	objval	1.63e+00	-9.92e-06	-6.07e-06	-8.76e-06	5.88e-05
	vltcst	3.62e-05	5.00e-07	2.00e-04	9.51e-08	1.94e-05
$m = 512$ $n = 512$	time(s)	0.97	47.85	41.07	187.57	76.40
	iter	11742	14719	15561	11863	12732
	objval	1.57e+00	-3.16e-05	-3.88e-05	-4.46e-05	8.87e-06
	vltcst	9.14e-05	5.00e-07	2.00e-04	9.89e-08	4.35e-05
$m = 1024$ $n = 1024$	time(s)	5.68	205.60	232.08	902.93	432.45
	iter	28914	14403	20000	13163	15564
	objval	1.94e+00	-1.10e-04	-1.18e-04	-1.18e-04	-9.02e-05
	vltcst	2.31e-04	5.00e-07	2.76e-04	9.94e-08	6.34e-05

Table 6 Tuned Parameters: Random Generated Data

Solver	t	Stopping Rule
ADMM-p	$5(m+n)\bar{C}$	'vltcst' $\leqslant$ 5e-07 or 'iter' $\geqslant$ 2e+04
ADMM-d	$1/(8(m+n)\bar{C})$	'vltcst' $\leqslant$ 2e-04 or 'iter' $\geqslant$ 2e+04
ADMM-s	$2(m+n)\bar{C}$	'vltcst' $\leqslant$ 1e-07 or 'iter' $\geqslant$ 2e+04
BADMM	$10\bar{C}$	$\|\pi - \tilde{\pi}\|_1 \leqslant$ 1e-06 or 'iter' $\geqslant$ 2e+04

accuracy is at the cost of much longer time. In fact, from our observation, much of the time by 'ADMM-s' is spent on the projection on the simplex(constraints). There exists more efficient algorithms for projection, but since time is limited, we do not implement them. The projection is quite essential in 'ADMM-s' because it imposes π to satisfy constraints, and this may explain why the constraints are easy to satisfy within tolerance.

It's worth noting that 'ADMM-d' has weakness in satisfying constraints. This is because in Section 2.1.2 we regard the "dual of dual" as an approximation of initial variables, which is numerically unstable. In our experiments, we found that 'ADMM-d' converges faster at the beginning to the optimal value, but then continuously vibrates around optimal value for a long time. 'BADMM' appears to perform worse than other three algorithms. The selection of t is inconsistent with other ADMM due to the property of KL divergence. If t is selected to be too big, then the exponential term in 31 may be so close to 1 that leads to bad precision. Furthermore, in our experiments, we found that when we increase t, 'BADMM' indeed has slightly better performance in ‘vltcst’ but deteriorates rapidly in ‘objval’. Our parameter is a balance between the two targets.

4.3.3 DOTmark

Some detailed tuning parameters is listed in Table 7.

Table 7 Tuned Parameters: DOTmark

Solver	t	Stopping Rule
ADMM-p	$5(m+n)\bar{C}$	‘vltcst’ $\leqslant$ 5e-07 or ‘iter’ $\geqslant$ 2e+04
ADMM-d	$1/(8(m+n)\bar{C})$	‘vltcst’ $\leqslant$ 2e-04 or ‘iter’ $\geqslant$ 2e+04
ADMM-s	$2(m+n)\bar{C}$	‘vltcst’ $\leqslant$ 4e-07 or ‘iter’ $\geqslant$ 2e+04
BADMM	$10\bar{C}$	$\|\pi - \tilde{\pi}\|_1 \leqslant$ 4e-06 or ‘iter’ $\geqslant$ 2e+04

Table 8 presents the numerical results on DOTmark.

As to constraints, all methods perform similar to that in the setting of random generated data, since our stopping rule is built on ‘vltcst’ or the norm of difference between π and $\tilde{\pi}$.

As to objective value, the three traditional ADMM methods are not stable on ‘objval’ and tend to achieve higher value than Mosek. This may be because of the different structure between random generated data and DOTmark. ‘BADMM’ tends to achieve lower objective value. However, the results are not stable and the constraints are not satisfied well.

As to iteration numbers and time spent, surprisingly, ‘ADMM-p’ and ‘ADMM-s’ need much less iterations than others. This phenomenon implies that we could tighten our stopping rule by decreasing the thresholds. Due to the time limit, further experiments could be conducted in the future.

4.3.4 Ellipse Example & Caffarelli’s Example

The detailed parameters is listed in Table 9.

Table 10 and 11 presents numerical results for Ellipse Example and Caffarelli’s Example, respectively.

The constraints are generally satisfied before we stop iterations. The solution from ‘BADMM’ does not satisfy the constraints very well. However, the optimal value is not within our expectation. The vibration of objective values around optimal values are observed in the three traditional ADMM methods. Among these algorithms, ‘ADMM-s’ performs relatively better on ‘iter’, ‘objval’, and ‘vltcst’, though it takes more time in each iteration. Nevertheless, ‘objval’ of ‘BADMM’ is relatively low.

Table 8 Perfomance of first order methods on DOTmark

method		prim(M)	ADMM-p	ADMM-d	ADMM-s	BADMM
1	time(s)	5.44	128.59	174.96	620.48	615.58
	iter	23489	8974	15247	10958	20000
	objval	1.75e+01	1.83e-03	2.49e-03	9.05e-04	-4.35e-03
	vltcst	2.34e-05	4.99e-07	2.00e-04	3.94e-07	8.22e-04
2	time(s)	5.52	128.18	154.83	448.70	607.74
	iter	22163	8942	13427	7746	20000
	objval	6.27e+00	2.51e-03	9.22e-05	1.94e-04	-1.03e-03
	vltcst	2.67e-05	5.00e-07	2.00e-04	3.87e-07	2.47e-04
3	time(s)	5.23	141.67	181.92	474.67	607.68
	iter	21240	9940	15782	8142	20000
	objval	4.07e+00	-2.90e-03	-6.66e-04	2.84e-04	7.53e-05
	vltcst	2.89e-05	4.99e-07	2.00e-04	3.91e-07	1.60e-04
4	time(s)	5.56	133.39	156.39	525.16	610.47
	iter	20163	9316	13560	8970	20000
	objval	1.48e+00	2.67e-03	3.33e-04	2.74e-04	3.77e-03
	vltcst	2.77e-05	5.00e-07	2.00e-04	3.88e-07	6.34e-05
5	time(s)	5.29	143.51	182.07	468.54	615.51
	iter	21509	10035	15795	8183	20000
	objval	2.14e+01	-1.62e-03	-3.89e-04	-3.46e-06	-1.56e-03
	vltcst	2.98e-05	5.00e-07	2.00e-04	3.22e-07	3.87e-04
6	time(s)	5.50	124.36	163.82	420.16	616.91
	iter	22635	8688	14282	7290	20000
	objval	1.92e+01	1.19e-03	3.89e-04	3.67e-04	-8.27e-04
	vltcst	2.49e-05	4.99e-07	2.00e-04	3.40e-07	3.38e-04
7	time(s)	5.51	139.22	165.99	511.25	614.00
	iter	22695	9726	14478	8894	20000
	objval	1.69e+01	-7.46e-04	2.79e-04	4.85e-05	-1.67e-03
	vltcst	2.57e-05	4.99e-07	2.00e-04	3.85e-07	3.54e-04
8	time(s)	3.60	120.51	157.94	399.11	619.88
	iter	15322	8401	13672	6812	20000
	objval	1.09e+01	1.89e-03	1.44e-03	1.46e-03	7.32e-05
	vltcst	1.56e-05	5.00e-07	2.00e-04	3.73e-07	7.87e-05
9	time(s)	2.68	93.18	121.59	115.47	207.71
	iter	12983	6514	10589	2000	20000
	objval	2.44e+01	-2.36e-04	1.09e-04	9.24e-04	7.93e-06
	vltcst	2.04e-05	4.97e-07	2.00e-04	3.69e-07	3.73e-05
10	time(s)	5.49	106.63	155.21	516.34	346.73
	iter	23237	7428	13515	8776	2508
	objval	7.09e-01	1.75e-03	1.31e-03	4.63e-04	2.87e-02
	vltcst	2.60e-05	5.00e-07	2.00e-04	3.81e-07	6.65e-05

Table 9 Tuning Parameters: Ellipse & Caffarelli

Solver	t	Stopping Rule
ADMM-p	$5(m+n)\bar{C}$	‘vltcst’ $\leqslant$ 5e-07 or ‘iter’ $\geqslant$ 2e+04
ADMM-d	$1/(8(m+n)\bar{C})$	‘vltcst’ $\leqslant$ 2e-04 or ‘iter’ $\geqslant$ 2e+04
ADMM-s	$2(m+n)\bar{C}$	‘vltcst’ $\leqslant$ 2e-07 or ‘iter’ $\geqslant$ 2e+04
BADMM	$10\bar{C}$	$\|\pi-\tilde{\pi}\|_1 \leqslant$ 4e-06 or ‘iter’ $\geqslant$ 2e+04

Table 10 Perfomance of first order methods on Ellipse Example

method		prim(M)	ADMM-p	ADMM-d	ADMM-s	BADMM
$m = 128$ $n = 128$	time(s)	0.10	2.62	1.73	9.34	3.37
	iter	1675	14454	9692	9515	10830
	objval	2.57e+00	3.22e-05	1.07e-04	4.49e-05	4.68e-04
	vltcst	1.33e-05	4.99e-07	9.99e-05	2.00e-07	1.95e-05
$m = 256$ $n = 256$	time(s)	0.19	4.32	4.51	34.93	30.86
	iter	4112	8287	9186	10075	18006
	objval	2.48e+00	1.02e-04	1.75e-04	5.46e-05	3.30e-04
	vltcst	3.48e-05	4.99e-07	9.96e-05	1.89e-07	6.35e-05
$m = 512$ $n = 512$	time(s)	0.88	22.48	29.94	265.18	155.61
	iter	9528	6863	11434	16927	20000
	objval	2.28e+00	1.39e-04	1.27e-04	1.30e-05	-3.60e-05
	vltcst	8.55e-05	4.99e-07	9.98e-05	1.97e-07	7.12e-04
$m = 1024$ $n = 1024$	time(s)	5.29	60.14	229.84	1161.15	616.73
	iter	21756	4178	20000	16527	20000
	objval	2.25e+00	1.58e-04	4.24e-06	-4.07e-05	-3.55e-03
	vltcst	1.96e-04	5.00e-07	1.14e-04	1.99e-07	6.16e-03

Table 11 Perfomance of first order methods on Caffarelli’s Example

method		prim(M)	ADMM-p	ADMM-d	ADMM-s	BADMM
$m = 128$ $n = 128$	time(s)	0.04	1.30	0.90	3.61	2.12
	iter	1370	8605	5910	5587	9588
	objval	4.00e+00	6.75e-05	2.54e-04	9.05e-05	4.19e-04
	vltcst	1.09e-05	4.99e-07	9.91e-05	1.61e-07	1.74e-05
$m = 256$ $n = 256$	time(s)	0.10	3.31	3.26	23.06	14.54
	iter	3016	9177	9444	10788	15936
	objval	4.00e+00	6.99e-05	1.67e-04	3.30e-05	2.38e-04
	vltcst	2.60e-05	5.00e-07	9.98e-05	1.95e-07	2.58e-05
$m = 512$ $n = 512$	time(s)	0.47	12.88	16.67	114.61	80.05
	iter	7536	6612	10983	12057	20000
	objval	4.01e+00	1.48e-04	1.51e-04	3.32e-05	2.43e-04
	vltcst	6.70e-05	4.99e-07	1.00e-04	1.96e-07	1.13e-04
$m = 1024$ $n = 1024$	time(s)	2.33	47.56	123.05	841.48	328.21
	iter	17237	5052	16104	20000	20000
	objval	4.00e+00	1.84e-04	6.40e-05	-3.31e-05	-4.43e-05
	vltcst	1.54e-04	4.99e-07	1.00e-04	6.56e-07	1.49e-03

4.4 Algorithms for entropic regularized OT

'sinkhorn' represents the Sinkhorn's algorithm with numerical stability and continuation strategy 7. 'ADMM' means the algorithm described in 8. Addition to 'time', 'iter', 'objval' and 'vltcst', we use 'entval' to denote the objective value with extropy regularization term. The values in 'objval' are, same as before, compared to 'prim(M)'.

The parameters and stopping criterions of our algorithms are given in Table 12. In 'sinkhorn', we apply continuation strategy. If the coefficient of the regularized term is ϵ, we select $10^i\epsilon(i = 4, 3, 2, 1, 0)$ as the sequence and run 4000 iterations in optimizing every subproblem.

Table 12 Tuned Parameters: algorithms for entropic regularized OT

method	t	Stopping Rule
ADMM	$5(m+n)\bar{C}$	'vltcst' $\leqslant$ 1e-07 or 'iter' $\geqslant$ 2e+04
sinkhorn	/	'iter' $\geqslant$ 2e+04

4.4.1 Gaussian Mixture Model: The effect of continuation strategy

We consider to test Sinkhorn's algorithm with numerical stability on Gaussian mixture model and test the effect of continuation strategy. We first take $N = 128$ as a first exmaple. Figure 5 gives the probability density function of two Gaussian mixture models.

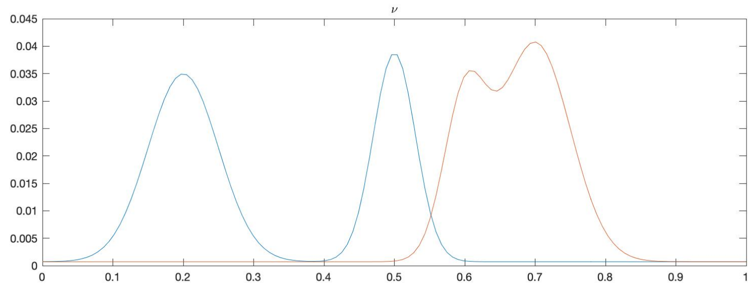

Figure 5 The probability distribution function of two Gaussian mixture models G_1 and G_2. G_1 (blue one) has mean $[0.2, 0.5]$, variance $[0.05^2, 0.03^2]$ and component proportion $[0.6, 0.4]$; G_2 (red one) has mean $[0.6, 0.7]$, variance $[0.03^2, 0.05^2]$ and component proportion $[0.3, 0.7]$.

We then take $\epsilon = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. If we use the vanilla Sinkhorn's algorithm 5 for $\epsilon = 10^{-4}, 10^{-5}$, K^ϵ would be a zero matrix. For Sinkhorn's algorithm with numerical stability 6, we take the maximum iteration number to be 10, 100, 1000, 4000 respectively. For Sinkhorn's algorithm with numerical stability and continuation strategy 7, we take $\alpha = 0.1$, $\epsilon_0 = \epsilon \times 10^5$ and the maximum iteration number to be 2, 20, 200, 800 respectively.

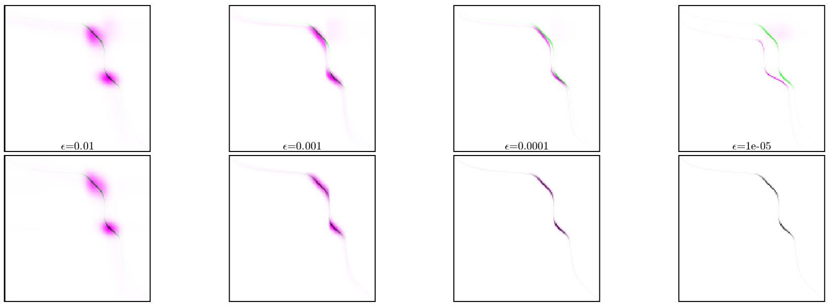

Figure 6 Impact of ϵ on the coupling between two gaussian mixture models 1. From left to right, $\epsilon = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. The upper row is without continuation strategy and the lower row is with continuatation strategy. The purple one is the solution to the entropic regularized LP. The green one is the solution to the original LP.

From Figure 6, we can find that with same iteration number, Sinkhorn's algorithm with numerical stability and continuation strategy converges faster than the one only with numerical stability. This shows that with proper choice of the parameter of continuation, we can achieve acceleration in Sinkhorn's algorithm with continuation strategy.

We then perform a comprehensive numerical experiment on Gaussian mixture model. Detailed parameters and the stopping criterions are listed in Table 12.

Table 13 gives the comprehensive numerical results for Gaussian Mixture Model. The setting is same as Figure 1, except that we test examples for different problem sizes, i.e., we let $N = 128, 256, 512, 1024$, respectively. We observe that, in most cases, although 'sinkhorn' takes much longer time and more iterations than 'ADMM', 'sinkhorn' outperforms 'ADMM' in both 'objval' and 'vltcst'.

4.4.2 Randomly generated data

The parameters and the stopping criterions are listed in Table 12. Table 14 presents numerical results on random generated data.

We find that the 'objval' and 'vltcst' of 'sinkhorn' is getting worse with the decrease of ϵ. This can be explained by the computation complexity of Sinkhorn's algorithm. As shown in Section 3.2, the τ-approximate solution of the unregularized OT problem takes $O(n^2 \log(n)\tau^{-3})$ operations where $\epsilon = \tau/\log(n)$. This means that if we decrease the ϵ by 0.1, theoretically we would need 10^3 times iteration to get a precise solution. Although our improvements of Sinkhorn's algorithm makes it adapt to small ϵ, but the computation is costly. If we want to use Sinkhorn's algorithm to get an approximation solution of the orignial problem 1, we should pay attention to choose an appropriate ϵ.

On the contrary, 'ADMM' appears robustness in satisfying constraints within acceptable

Table 13 Perfomance of algorithms for entropic regularization of OT on Gaussian mixture model

		$\epsilon = 10^{-2}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$	
method		sinkhorn	ADMM	sinkhorn	ADMM	sinkhorn	ADMM
$m = 128$ $n = 128$	time(s)	10.39	0.23	9.52	0.85	7.92	6.19
	iter	20000	647	20000	2398	20000	20000
	objval	7.25e-02	7.24e-02	6.77e-04	2.19e-03	1.76e-04	2.32e-04
	entval	-2.50e-02	-2.50e-02	5.20e-02	5.20e-02	5.26e-02	5.26e-02
	vltcst	1.27e-12	9.93e-08	6.15e-14	9.96e-08	5.14e-12	3.83e-07
$m = 256$ $n = 256$	time(s)	34.53	1.34	31.96	1.90	25.48	18.53
	iter	20000	1213	20000	1782	20000	18203
	objval	7.27e-02	7.26e-02	8.55e-04	1.85e-03	4.94e-04	2.74e-04
	entval	-3.89e-02	-3.89e-02	5.19e-02	5.19e-02	5.26e-02	5.26e-02
	vltcst	1.37e-12	9.96e-08	1.17e-13	9.91e-08	3.83e-12	9.98e-08
$m = 512$ $n = 512$	time(s)	140.15	12.56	130.25	11.82	104.53	56.03
	iter	20000	2228	20000	2100	20000	9939
	objval	7.27e-02	7.26e-02	8.95e-04	1.44e-03	2.07e-04	3.22e-04
	entval	-5.28e-02	-5.28e-02	5.17e-02	5.17e-02	5.26e-02	5.26e-02
	vltcst	1.42e-12	9.99e-08	8.56e-14	9.99e-08	1.58e-12	9.97e-08
$m = 1024$ $n = 1024$	time(s)	692.47	99.56	658.32	86.61	557.13	144.90
	iter	20000	4034	20000	3523	20000	5896
	objval	7.27e-02	7.24e-02	8.79e-04	7.20e-04	1.84e-04	4.07e-04
	entval	-6.67e-02	-6.67e-02	5.16e-02	5.16e-02	5.26e-02	5.26e-02
	vltcst	1.29e-12	9.99e-08	1.30e-13	1.00e-07	1.49e-12	1.00e-07

Table 14 Perfomance of algorithms for entropic regularization of OT on randomly generated data

		$\epsilon = 10^{-2}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$	
method		sinkhorn	ADMM	sinkhorn	ADMM	sinkhorn	ADMM
$m = 128$ $n = 128$	time(s)	10.06	0.82	8.36	6.16	6.84	6.21
	iter	20000	2223	20000	20000	20000	20000
	objval	2.40e-04	3.18e-04	1.23e-03	-6.35e-06	2.54e-03	-8.73e-06
	entval	1.30e+00	1.30e+00	1.36e+00	1.36e+00	1.36e+00	1.36e+00
	vltcst	6.48e-15	9.99e-08	6.60e-13	2.22e-07	9.86e-11	4.72e-07
$m = 256$ $n = 256$	time(s)	32.66	3.23	26.86	19.88	20.49	20.46
	iter	20000	3072	20000	20000	20000	20000
	objval	3.16e-04	3.49e-04	2.36e-03	-9.92e-06	4.52e-03	-1.24e-05
	entval	1.56e+00	1.56e+00	1.63e+00	1.62e+00	1.63e+00	1.63e+00
	vltcst	2.98e-15	9.99e-08	7.04e-13	5.62e-07	1.28e-10	6.90e-07
$m = 512$ $n = 512$	time(s)	134.57	17.43	108.35	111.25	83.80	111.31
	iter	20000	3045	20000	20000	20000	20000
	objval	3.48e-04	3.76e-04	1.55e-03	-4.66e-05	3.43e-03	-4.69e-05
	entval	1.49e+00	1.49e+00	1.57e+00	1.57e+00	1.57e+00	1.57e+00
	vltcst	9.34e-15	9.98e-08	7.69e-13	2.18e-07	8.42e-11	3.08e-07
$m = 1024$ $n = 1024$	time(s)	692.35	99.61	568.16	497.03	472.21	495.21
	iter	20000	4040	20000	20000	20000	20000
	objval	2.11e-04	2.04e-04	1.93e-03	-1.15e-04	3.54e-03	-1.23e-04
	entval	1.85e+00	1.85e+00	1.94e+00	1.93e+00	1.94e+00	1.94e+00
	vltcst	1.63e-14	9.97e-08	1.22e-12	2.18e-07	1.06e-10	2.99e-07

iteration number. 'vltcst' of all test samples are below 1e-06, though it's not comparable to 'sinkhorn'. 'objval' of ADMM appears to be gradually lower than 'prim(M)' when ϵ decreases. It's worth noting that in 'ADMM' we do not apply continuation strategy. In fact, we observe that with the carefully selected t, 'ADMM' converges quite fast to the optimal value at the beginning, whatever ϵ is, although vibration is unavoidable in the latter steps. Of course, for ϵ not large and small or moderate problem sizes, 'sinkhorn' performs better than 'ADMM'.

4.4.3 DOTmark

Detailed parameters are listed in 12. (The stopping rule for 'ADMM' changes to 'vltcst' $\leqslant$ 1e-07 or 'iter' $\geqslant$ 2e+04.) Table 15 and 16 present numerical results on DOTmark.

Same as before, 'objval' and 'vltcst' of 'sinkhorn' is getting worse with the decrease of ϵ. All of the results are within 1e-09 in 'vltcst', but 'objval' deteriorate rapidly when ϵ decreases. This can be again explained by the computation complexity of Sinkhorn's algorithm.

'ADMM' can achieve relatively lower objective value on average, and need moderate iterations before 'vltcst' arrives at a threshold. The performance is similar to random generated data setting with larger problem sizes. We should note that due to the different problem structures, it seems difficult sometimes for 'ADMM' to outperform 'prim(M)'.

4.4.4 Ellipse Example & Caffarelli's Example

Table 12 lists the detailed parameters.Table 17 and Table 18 present numerical results for Ellipse Example and Caffarelli's Example, respectively. Through our previous experiments, we observe that with the decrease of ϵ, 'ADMM' does not require so much increase of computational complexity as 'sinkhorn'. This can be explained by the limit when $\epsilon \to 0$. When $\epsilon \to 0$, 'ADMM' reduce to ADMM for the primal problem 1. But with $\epsilon \to 0$, we have shown in Section 3.3.1 that 'sinkhorn' will diverge due to the loss of strong convexity. And with the decrease of ϵ, the strong convexity of problem 39 decreases. This also explains why 'sinkhorn' requires such enormous iterations to find the optimal solution. In summary, we shall use 'sinkhorn' to solve the entropic regularization of OT 39 with moderate ϵ. With large ϵ, we shall use the original Sinkhorn's algorithm 5 because it is faster in computation. 'ADMM' seems to be suitable to arbitrary small ϵ.

Table 15 Perfomance of algorithms for entropic regularization of OT on DOTmark Part1

		$\epsilon = 10^{-2}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$	
Class No.	method	sinkhorn	ADMM	sinkhorn	ADMM	sinkhorn	ADMM
1	time(s)	593.54	335.53	502.26	369.18	440.68	371.82
	iter	20000	13674	20000	15058	20000	15062
	objval	2.01e-02	8.30e-04	3.36e-02	6.52e-04	2.11e+00	7.27e-04
	entval	1.78e+01	1.74e+01	1.81e+01	1.75e+01	5.44e+01	1.75e+01
	vltcst	2.87e-13	2.00e-07	2.38e-11	2.00e-07	6.03e-10	2.00e-07
2	time(s)	610.36	264.70	499.11	318.66	445.02	318.45
	iter	20000	10740	20000	13021	20000	13021
	objval	4.60e-02	-1.17e-03	7.43e-02	4.09e-04	5.86e+00	4.29e-04
	entval	6.47e+00	6.18e+00	6.74e+00	6.27e+00	4.30e+01	6.07e+00
	vltcst	1.46e-13	2.00e-07	1.69e-11	2.00e-07	3.80e-10	2.00e-07
3	time(s)	591.98	299.63	499.19	361.14	442.04	361.33
	iter	20000	12206	20000	14672	20000	14668
	objval	8.12e-02	3.74e-04	2.46e-01	-1.14e-03	6.45e+00	-1.11e-03
	entval	4.32e+00	3.99e+00	5.07e+00	4.07e+00	3.03e+01	4.07e+00
	vltcst	1.52e-13	2.00e-07	9.59e-12	2.00e-07	2.85e-10	2.00e-07
4	time(s)	591.00	268.63	499.50	330.66	441.70	331.92
	iter	20000	10923	20000	13433	20000	13467
	objval	2.18e-01	-1.26e-03	4.58e-01	-5.81e-04	9.30e+00	-5.05e-04
	entval	1.71e+00	1.39e+00	2.15e+00	1.47e+00	1.52e+01	1.48e+00
	vltcst	1.17e-13	2.00e-07	6.78e-12	2.00e-07	2.35e-10	2.00e-07
5	time(s)	589.61	295.34	499.33	368.54	442.19	368.14
	iter	20000	12018	20000	15052	20000	15047
	objval	8.86e-03	8.71e-04	3.55e-02	9.91e-04	2.46e+00	9.9 4e-04
	entval	2.15e+01	2.13e+01	2.22e+01	2.14e+01	7.41e+01	2.14e+01
	vltcst	1.97e-13	2.00e-07	3.58e-11	2.00e-07	5.87e-10	2.00e-07
6	time(s)	590.35	253.97	498.56	326.70	441.70	327.49
	iter	20000	10339	20000	13362	20000	13385
	objval	1.10e-02	2.37e-04	3.88e-02	-3.33e-04	2.17e+00	-3.35e-04
	entval	1.93e+01	1.91e+01	1.99e+01	1.92e+01	6.09e+01	1.92e+01
	vltcst	3.53e-13	2.00e-07	3.24e-11	2.00e-07	8.49e-10	2.00e-07
7	time(s)	590.12	296.02	499.00	339.12	441.04	339.56
	iter	20000	12054	20000	13862	20000	13877
	objval	1.60e-02	9.12e-04	4.17e-02	-4.18e-04	2.82e+00	-4.48e-04
	entval	1.71e+01	1.69e+01	1.76e+01	1.69e+01	6.48e+01	1.69e+01
	vltcst	3.62e-13	2.00e-07	3.62e-11	2.00e-07	6.20e-10	2.00e-07

Table 16 Perfomance of algorithms for entropic regularization of OT on DOTmark Part2

		$\epsilon = 10^{-2}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$	
Class No.	method	sinkhorn	ADMM	sinkhorn	ADMM	sinkhorn	ADMM
8	time(s)	604.68	273.63	502.38	374.75	441.42	374.83
	iter	20000	11185	20000	15318	20000	15323
	objval	3.30e-02	6.25e-04	4.21e-02	-1.95e-04	2.01e+00	-2.27e-04
	entval	1.12e+01	1.08e+01	1.13e+01	1.09e+01	3.27e+01	1.09e+01
	vltcst	5.46e-13	2.00e-07	1.72e-11	2.00e-07	8.71e-10	2.00e-07
9	time(s)	603.00	249.23	501.40	299.38	439.78	264.61
	iter	20000	10177	20000	12241	20000	10814
	objval	1.69e-05	4.17e-05	1.69e-05	-7.33e-05	7.96e-01	9.68e-05
	entval	2.43e+01	2.43e+01	2.44e+01	2.44e+01	4.38e+01	2.24e+01
	vltcst	6.59e-13	1.98e-07	2.73e-11	1.99e-07	5.95e-10	1.99e-07
10	time(s)	590.49	237.95	500.10	281.92	441.94	282.65
	iter	20000	9693	20000	11524	20000	11549
	objval	4.20e-01	1.98e-03	1.25e+00	-1.02e-03	7.78e+00	-1.21e-04
	entval	9.22e-01	6.25e-01	1.60e+00	7.08e-01	6.23e+00	7.09e-01
	vltcst	2.17e-13	2.00e-07	2.16e-12	2.00e-07	1.18e-10	2.00e-07

Table 17 Perfomance of algorithms for entropic regularization of OT on Ellipse Example

		$\epsilon = 10^{-2}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$	
method		sinkhorn	ADMM	sinkhorn	ADMM	sinkhorn	ADMM
$m = 128$ $n = 128$	time(s)	10.04	0.36	8.91	6.23	7.18	6.21
	iter	20000	984	20000	20000	20000	20000
	objval	1.77e-03	2.24e-03	5.91e-04	1.58e-05	4.66e-03	7.81e-06
	entval	2.50e+00	2.50e+00	2.57e+00	2.57e+00	2.58e+00	2.57e+00
	vltcst	1.90e-14	9.95e-08	3.11e-13	3.85e-07	9.03e-11	5.63e-07
$m = 256$ $n = 256$	time(s)	33.01	0.81	29.70	19.50	22.52	20.14
	iter	20000	705	20000	19370	20000	20000
	objval	2.37e-03	2.72e-03	3.32e-04	2.71e-05	1.58e-03	1.97e-05
	entval	2.40e+00	2.40e+00	2.48e+00	2.48e+00	2.49e+00	2.48e+00
	vltcst	1.42e-14	9.91e-08	9.55e-13	9.97e-08	9.77e-11	2.43e-07
$m = 512$ $n = 512$	time(s)	134.58	3.57	119.84	91.35	91.55	112.25
	iter	20000	637	20000	16276	20000	20000
	objval	3.06e-03	3.27e-03	2.80e-04	3.83e-05	1.51e-03	1.62e-05
	entval	2.19e+00	2.19e+00	2.28e+00	2.28e+00	2.28e+00	2.28e+00
	vltcst	1.37e-14	9.92e-08	1.19e-12	9.99e-08	1.42e-10	2.36e-07
$m = 1024$ $n = 1024$	time(s)	677.72	15.72	625.73	386.85	514.66	491.65
	iter	20000	641	20000	15739	20000	20000
	objval	3.40e-03	3.48e-03	7.94e-05	-1.92e-05	1.61e-03	-3.87e-05
	entval	2.15e+00	2.15e+00	2.25e+00	2.25e+00	2.26e+00	2.25e+00
	vltcst	1.37e-14	1.00e-07	7.50e-13	9.99e-08	1.30e-10	1.46e-07

Table 18 Perfomance of algorithms for entropic regularization of OT on Caffarelli's Example

		$\epsilon = 10^{-2}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$	
method		sinkhorn	ADMM	sinkhorn	ADMM	sinkhorn	ADMM
$m = 128$ $n = 128$	time(s)	7.61	0.50	6.76	4.64	5.58	4.66
	iter	20000	1777	20000	20000	20000	20000
	objval	1.03e-03	1.45e-03	5.04e-04	9.71e-06	4.01e-04	6.89e-06
	entval	3.93e+00	3.93e+00	4.00e+00	4.00e+00	4.00e+00	4.00e+00
	vltcst	9.94e-15	9.98e-08	2.81e-12	1.63e-07	2.12e-10	2.94e-07
$m = 256$ $n = 256$	time(s)	21.63	0.84	19.52	13.12	15.29	13.09
	iter	20000	1154	20000	20000	20000	20000
	objval	1.40e-03	1.90e-03	4.87e-04	1.32e-05	8.98e-04	1.18e-05
	entval	3.93e+00	3.93e+00	4.00e+00	4.00e+00	4.01e+00	4.00e+00
	vltcst	1.18e-14	9.95e-08	1.47e-12	1.67e-07	1.88e-10	2.37e-07
$m = 512$ $n = 512$	time(s)	83.29	2.20	74.29	68.36	55.97	68.35
	iter	20000	644	20000	20000	20000	20000
	objval	1.73e-03	2.25e-03	1.40e-03	1.18e-05	1.78e-03	9.34e-06
	entval	3.92e+00	3.92e+00	4.01e+00	4.00e+00	4.01e+00	4.01e+00
	vltcst	1.44e-14	9.97e-08	1.21e-12	1.28e-07	1.70e-10	1.98e-07
$m = 1024$ $n = 1024$	time(s)	428.49	11.31	390.39	256.87	320.54	321.15
	iter	20000	704	20000	16004	20000	20000
	objval	2.01e-03	2.19e-03	6.65e-04	7.60e-06	1.32e-03	-1.46e-05
	entval	3.90e+00	3.90e+00	4.00e+00	4.00e+00	4.00e+00	4.00e+00
	vltcst	1.67e-14	9.86e-08	1.83e-12	9.99e-08	2.17e-10	1.48e-07

5 Conclusion

Our work for OT (Optimal Transport) ends up here, but new ideas for improvement and continuous passion for research will never fade. We now give a brief summary for what we have done and give some outlooks.

We first implemented several first-order methods, mainly 'ADMM's. Generally, they do not perform very well if compared to state-of-art algorithm packages such as Gurobi. However, the violation of constraints and optimality of objective values are within our tolerance. From our experiments, directly solving primal problems are much better than solving dual ones. Needless to say, it has been considered that first-order methods such as 'ADMM' are not suitable for large-scale OT problems. But we observe that first-order methods could also achieve relatively satisfying results, if we choose smart splitting schemes and tuning parameters.

We then choose Entropy Regularization as our main topic. Sinkhorn is a theoretically and practically elegant algorithm. However, numerical stability and convergence speed cause difficulty in optimizing the regularized problem, especially when the coefficient of regularized term ϵ is small. We adopt a technique to improve the stability and the continuation strategy to boost convergence. Great improvement has been observed, but deterioration of performance when ϵ is very small still exists. It's worth mentioning that ADMM shows moderate results on Entropy Regularization.

Now we provide some future outlooks. For first order methods, we hope to implement the interior point algorithm to deal with constraints satisfaction. In addition, second-order algorithms like Newton methods and semi-smooth Newton methods can be tested out. Also, as is mentioned before, projection on a simplex in first order methods can be further simplified to reduce iteration time. Furthermore, though we have testified that our choice of parameters achieve good results, theoretical properties remained to be studied.

For Sinkhorn's method, we hope to continue on refining the numerical stability and boosting convergence. A simple method is to combine Sinkhorn with other methods such as ADMM. Additionally, if time has permitted, we might want to try implementing the algorithm in C++ in order to accelerate processing.

For numerical experiments, we hope to test on the whole DOTmark to have a more precise investigation on the accuracy and efficiency of all these algorithms.

Transportation problem is the basis for many high level applications. However, solving the problem is still tough even with numerous proposed methods. Of course, we hope that we can find more efficient methods, both in time and space. This will be left for future research.

Acknowledgement

We would like to thank Zeyu Jia a lot for providing his server to let us run our numerical experiments.

The division of our work is listed below. Yifei Wang is responsible for the writing and coding of Bregman ADMM (Section 2.3), Sinkhorn's algorithm and related methods (Section 3.1-3.3); the framework of coding and the conducting of most of the numerical experiments in Section 4. Feng Zhu is responsible for the writing and coding of ADMMs and related methods (Section 2.1-2.2, Section 3.4, Appendix); the writing of most of the experiment reports in Section 4.

References

[1] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1964–1974.

[2] Roberto Cominetti and Jaime San Martín. "Asymptotic analysis of the exponential penalty trajectory in linear programming". In: *Mathematical Programming* 67.1-3 (1994), pp. 169–187.

[3] John Duchi et al. "Efficient projections onto the l_1-ball for learning in high dimensions". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 272–279.

[4] Peyré Gabriel and Cuturi Marco. "Computational Optimal Transport". In: *arXiv: 1803.00567v1* (2018).

[5] Samuel Gerber and Mauro Maggioni. "Multiscale strategies for computing optimal transport". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 2440–2471.

[6] Jörn Schrieber, Dominic Schuhmacher, and Carsten Gottschlich. "DOTmark - A Benchmark for Discrete Optimal Transport". In: *arxiv: 1610.03368* (2017).

[7] Richard Sinkhorn. "A relationship between arbitrary positive matrices and doubly stochastic matrices". In: *The annals of mathematical statistics* 35.2 (1964), pp. 876–879.

[8] Huahua Wang and Arindam Banerjee. "Bregman alternating direction method of multipliers". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2816–2824.

Appendix: Splitting method with penalty functions

To impose more attention on constraints, we propose the following splitting method with penalty functions.

Let $F(\pi)$ and $G(\pi)$ be defined in the following way,

$$\begin{aligned} F(\pi) &= \sum_{i,j} C_{i,j}\pi_{i,j} + \frac{t}{2}\left(\sum_i (\mu_i - \sum_j \pi_{i,j})^2 + \sum_j (\nu_j - \sum_i \pi_{i,j})^2\right) \\ G(\pi) &= \begin{cases} 0, & \pi_{i,j} \geqslant 0,\ \forall i,j \\ +\infty, & \text{else} \end{cases} \end{aligned} \tag{60}$$

Now let the problem be

$$\min_{\pi}\ F(\pi) + G(\pi) \tag{61}$$

It's easy to obtain

$$\operatorname{prox}_{\frac{1}{w}G}(\pi) = \max\{\pi, 0\} \tag{62}$$

To compute $\operatorname{prox}_{\frac{1}{w}F}(\pi)$ is equivalent to solve

$$\begin{aligned} \min_{\tilde{\pi}} \tilde{L}(\tilde{\pi}) = &\sum_{i,j} C_{i,j}\tilde{\pi}_{i,j} + \frac{t}{2}\left(\sum_i (\mu_i - \sum_j \tilde{\pi}_{i,j})^2 + \sum_j (\nu_j - \sum_i \tilde{\pi}_{i,j})^2\right) \\ &+ \frac{w}{2}\left(\sum_{i,j} (\tilde{\pi}_{i,j} - \pi_{i,j})^2\right), \end{aligned} \tag{63}$$

We set $\frac{\partial \tilde{L}}{\partial \pi}$ to zero and solve the linear equations. This yields

$$w\tilde{\pi}_{i,j} + t\left(\sum_k \tilde{\pi}_{i,k} + \sum_k \tilde{\pi}_{k,j}\right) = -C_{i,j} + w\pi_{i,j} + t(\mu_i + \nu_j) \triangleq r_{i,j} \tag{64}$$

and

$$\begin{aligned} \tilde{\pi}_{i,j} = &\frac{1}{w}(r_{i,j} - \frac{t}{nt+w}\sum_k r_{i,k} - \frac{t}{mt+w}\sum_k r_{k,j} \\ &+ \frac{t^2}{mt+nt+w}(\frac{1}{mt+w} + \frac{1}{nt+w})\sum_{k,l} r_{k,l}) \end{aligned} \tag{65}$$

We now have the following algorithm 9. Here we apply Douglas-Rachford update scheme and Nesterov accelerating method to boost convergence.

Algorithm 9 Splitting method with penalty functions for Optimal Transport

```
Input: $C \in \mathbb{R}^{m\times n}$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^n$, $t \in \mathbb{R}^+$, $w \in \mathbb{R}^+$
1: Initialize $\pi^{(0)}$ and set $\pi^{(-1)} = \pi$, $l \leftarrow 0$
2: while not converge do
3:     $\pi \leftarrow \pi^{(l)} + \frac{i-1}{i+2}(\pi^{(l)} - \pi^{(l-1)})$
4:     Compute $\tilde{\pi}$ by 65
5:     Compute $\hat{\pi}$ by 62, with $\pi$ replaced by $2\tilde{\pi} - \pi$
6:     $\pi^{(l+1)} \leftarrow \hat{\pi} + \tilde{\pi} - \pi$
7:     $l \leftarrow l + 1$
8: end while
9: return $\pi^{(i)}$
```

In real implmentation, we found that t is hard to tune. If it is small, then the constraints are not satisfied well. If it is large, the objective value becomes unacceptable. We have tried to combine 9 with 2 to impose constraints, but the results are not satisfactory.