A Direct $\tilde{O}(1/\epsilon)$ Iteration Parallel Algorithm for Optimal Transport

Arun Jambulapati, Aaron Sidford, and Kevin Tian
Stanford University
{jmblpati,sidford,kjtian}@stanford.edu

Abstract

Optimal transportation, or computing the Wasserstein or "earth mover's" distance between two n-dimensional distributions, is a fundamental primitive which arises in many learning and statistical settings. We give an algorithm which solves the problem to additive ϵ accuracy with $\tilde{O}(1/\epsilon)$ parallel depth and $\tilde{O}\left(n^2/\epsilon\right)$ work. [BJKS18, Qua19] obtained this runtime through reductions to positive linear programming and matrix scaling. However, these reduction-based algorithms use subroutines which may be impractical due to requiring solvers for second-order iterations (matrix scaling) or non-parallelizability (positive LP). Our methods match the previous-best work bounds by [BJKS18, Qua19] while either improving parallelization or removing the need for linear system solves, and improve upon the previous best first-order methods running in time $\tilde{O}(\min(n^2/\epsilon^2, n^{2.5}/\epsilon))$ [DGK18, LHJ19]. We obtain our results by a primal-dual extragradient method, motivated by recent theoretical improvements to maximum flow [She17].

1 Introduction

Optimal transport is playing an increasingly important role as a subroutine in tasks arising in machine learning [ACB17], computer vision [BvdPPH11, SdGP+15], robust optimization [EK18, BK17], and statistics [PZ16]. Given these applications for large scale learning, designing algorithms for efficiently approximately solving the problem has been the subject of extensive recent research [Cut13, AWR17, GCPB16, CK18, DGK18, LHJ19, BJKS18, Qua19].

Given two vectors r and c in the n-dimensional probability simplex Δ^n and a cost matrix $C \in \mathbb{R}_{\geq 0}^{n \times n}$[1], the optimal transportation problem is

$$\min_{X \in \mathcal{U}_{r,c}} \langle C, X \rangle, \quad \text{where} \quad \mathcal{U}_{r,c} \overset{\text{def}}{=} \left\{ X \in \mathbb{R}_{\geq 0}^{n \times n},\ X\mathbf{1} = r,\ X^\top \mathbf{1} = c \right\}. \tag{1}$$

This problem arises from defining the *Wasserstein* or *Earth mover's* distance between discrete probability measures r and c, as the cheapest coupling between the distributions, where the cost of the coupling $X \in \mathcal{U}_{r,c}$ is $\langle C, X \rangle$. If r and c are viewed as distributions of masses placed on n points in some space (typically metric), the Wasserstein distance is the cheapest way to move mass to transform r into c. In (1), X represents the transport plan (X_{ij} is the amount moved from r_i to c_j) and C represents the cost of movement (C_{ij} is the cost of moving mass from r_i to c_j).

Throughout, the value of (1) is denoted OPT. We call $\hat{X} \in \mathcal{U}_{r,c}$ an *ϵ-approximate transportation plan* if $\langle C, \hat{X} \rangle \leq \text{OPT} + \epsilon$. Our goal is to design an efficient algorithm to produce such a $\hat{X}$.

---

[1] Similarly to earlier works, we focus on square matrices; generalizations to rectangular matrices are straightforward.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1.1 Our Contributions

Our main contribution is an algorithm running in $\tilde{O}(\|C\|_{\max}/\epsilon)$ parallelelizable iterations[2] and $\tilde{O}(n^2\|C\|_{\max}/\epsilon)$ total work producing an ϵ-approximate transport plan.

Matching runtimes were given in the recent work of [BJKS18, Qua19]. Their runtimes were obtained via reductions to matrix scaling and positive linear programming, each well-studied problems in theoretical computer science. However, the matrix scaling algorithm is a second-order Newton-type method which makes calls to structured linear system solvers, and the positive LP algorithm is not parallelizable (i.e. has depth polynomial in dimension). These features potentially limit the practicality of these algorithms. The key remaining open question this paper addresses is, *is there an efficient first-order, parallelizable algorithm for approximating optimal transport?* We answer this affirmatively and give an efficient, parallelizable primal-dual first-order method; the only additional overhead is a scheme for implementing steps, incurring roughly an additional $\log \epsilon^{-1}$ factor.

Our approach heavily leverages the recent improvement to the maximum flow problem, and more broadly two-player games on a simplex (ℓ_1 ball) and a box (ℓ_∞ ball), due to the breakthrough work of [She17]. First, we recast (1) as a minimax game between a box and a simplex, proving correctness via a rounding procedure known in the optimal transport literature. Second, we show how to adapt the dual extrapolation scheme under the weaker convergence requirements of area-convexity, following [She17], to obtain an approximate minimizer to our primal-dual objective in the stated runtime. En route, we slightly simplify analysis in [She17] and relate it more closely to the existing extragradient literature.

Finally, we give preliminary experimental evidence showing our algorithm can be practical, and highlight some open directions in bridging the gap between theory and practice of our method, as well as accelerated gradient schemes [DGK18, LHJ19] and Sinkhorn iteration.

1.2 Previous Work

Optimal Transport. The problem of giving efficient algorithms to find ϵ-approximate transport plans $\hat{X}$ which run in nearly linear time[3] has been addressed by a line of recent work, starting with [Cut13] and improved upon in [GCPB16, AWR17, DGK18, LHJ19, BJKS18, Qua19]. We briefly discuss their approaches here.

Works by [Cut13, AWR17] studied the Sinkhorn algorithm, an alternating minimization scheme. Regularizing (1) with an η^{-1} multiple of entropy and computing the dual, we arrive at the problem

$$\min_{x,y\in\mathbb{R}^n} \mathbf{1}^\top B_{\eta C}(x,y)\mathbf{1} - r^\top x - c^\top y \quad \text{where} \quad B_{\eta C}(x,y)_{ij} = e^{x_i+y_j-\eta C_{ij}}.$$

This problem is equivalent to computing diagonal scalings X and Y for $M = \exp(-\eta C)$ such that XMY has row sums r and column sums c. The Sinkhorn iteration alternates fixing the row sums and the column sums by left and right scaling by diagonal matrices until an approximation of such scalings is found, or equivalently until XMY is close to being in $\mathcal{U}_{r,c}$.

As shown in [AWR17], we can round the resulting almost-transportation plan to a transportation plan which lies in $\mathcal{U}_{r,c}$ in linear time, losing at most $2\|C\|_{\max}(\|X\mathbf{1}-r\|_1 + \left\|X^\top\mathbf{1}-c\right\|_1)$ in the objective. Further, [AWR17] showed that $\tilde{O}(\|C\|^3_{\max}/\epsilon^3)$ iterations of this scheme sufficed to obtain a matrix which $\epsilon/\|C\|_{\max}$-approximately meets the demands in ℓ_1 with good objective value, by analyzing it as an instance of mirror descent with an entropic regularizer. The same work proposed an alternative algorithm, Greenkhorn, based on greedy coordinate descent. [DGK18, LHJ19] showed that $\tilde{O}\left(\|C\|^2_{\max}/\epsilon^2\right)$ iterations, corresponding to $\tilde{O}\left(n^2\|C\|^2_{\max}/\epsilon^2\right)$ work, suffice for both Sinkhorn and Greenkhorn, the current state-of-the-art for this line of analysis.

An alternative approach based on first-order methods was studied by [DGK18, LHJ19]. These works considered minimizing an entropy-regularized Equation 1; the resulting weighted softmax function is prevalent in the literature on approximate linear programming [Nes05], and has found similar

---

[2] Our iterations consist of vector operations and matrix-vector products, which are easily parallelizable. Throughout $\|C\|_{\max}$ is the largest entry of C.

[3] We use "nearly linear" to describe complexities which have an $n^2\text{polylog}(n)$ dependence on the dimension (where the size of input C is n^2), and polynomial dependence on $\|C\|_{\max}, \epsilon^{-1}$.

applications in near-linear algorithms for maximum flow [She13, KLOS14, ST18] and positive linear programming [You01, AO15]. An unaccelerated algorithm, viewable as ℓ_∞ gradient descent, was analyzed in [DGK18] and ran in $\tilde{O}(\|C\|_{\max}/\epsilon^2)$ iterations. Further, an accelerated algorithm was discussed, for which the authors claimed an $\tilde{O}(n^{1/4}\|C\|_{\max}^{0.5}/\epsilon)$ iteration count. [LHJ19] showed that the algorithm had an additional dependence on a parameter as bad as $n^{1/4}$, roughly due to a gap between the ℓ_2 and ℓ_∞ norms. Thus, the state of the art runtime in this line is the better of $\tilde{O}\left(n^{2.5}\|C\|_{\max}^{0.5}/\epsilon\right)$, $\tilde{O}\left(n^2\|C\|_{\max}/\epsilon^2\right)$ operations. The dependence on dimension of the former of these runtimes matches that of the linear programming solver of [LS14, LS15], which obtain a polylogarithmic dependence on ϵ^{-1}, rather than a polynomial dependence; thus, the question of obtaining an accelerated ϵ^{-1} dependence without worse dimension dependence remained open.

This was partially settled in [BJKS18, Qua19], which studied the relationship of optimal transport to fundamental algorithmic problems in theoretical computer science, namely positive linear programming and matrix scaling, for which significantly-improved runtimes have been recently obtained [AO15, ZLdOW17, CMTV17]. In particular, they showed that optimal transport could be reduced to instances of either of these objectives, for which $\tilde{O}\left(\|C\|_{\max}/\epsilon\right)$ iterations, each of which required linear $O(n^2)$ work, sufficed. However, both of these reductions are based on black-box methods for which practical implementations are not known; furthermore, in the case of positive linear programming a parallel $\tilde{O}(1/\epsilon)$-iteration algorithm is not known. [BJKS18] also showed any polynomial improvement to the runtime of our paper in the dependence on either ϵ or n would result in maximum-cardinality bipartite matching in dense graphs faster than $\tilde{O}(n^{2.5})$ without fast matrix multiplication [San09], a fundamental open problem unresolved for almost 50 years [HK73].

Year	Author	Complexity	Approach	1st-order	Parallel
2015	[LS15]	$\tilde{O}(n^{2.5})$	Interior point	No	No
2017-19	[AWR17]	$\tilde{O}(n^2\|C\|_{\max}^2/\epsilon^2)$	Sink/Greenkhorn	Yes	Yes
2018	[DGK18]	$\tilde{O}(n^2\|C\|_{\max}^2/\epsilon^2)$	Gradient descent	Yes	Yes
2018-19	[LHJ19]	$\tilde{O}(n^{2.5}\|C\|_{\max}/\epsilon)$	Acceleration	Yes	Yes
2018	[BJKS18]	$\tilde{O}(n^2\|C\|_{\max}/\epsilon)$	Matrix scaling	No	Yes
2018-19	[BJKS18, Qua19]	$\tilde{O}(n^2\|C\|_{\max}/\epsilon)$	Positive LP	Yes	No
2019	This work	$\tilde{O}(n^2\|C\|_{\max}/\epsilon)$	Dual extrapolation	Yes	Yes

Table 1: Optimal transport algorithms. Algorithms using second-order information use potentially-expensive SDD system solvers; the runtime analysis of Sink/Greenkhorn is due to [DGK18, LHJ19].

Specializations of the transportation problem to ℓ_p metric spaces or arising from geometric settings have been studied [SA12, AS14, ANOY14]. These specialized approaches seem fundamentally different than those concerning the more general transportation problem.

Finally, we note recent work [ABRW18] showed the promise of using the Nyström method for low-rank approximations to achieve speedup in theory and practice for transport problems arising from specific metrics. As our method is based on matrix-vector operations, where low-rank approximations may be applicable, we find it interesting to see if our method can be combined with these improvements.

Remark. During the revision process for this work, an independent result [LMR19] was published to arXiv, obtaining improved runtimes for optimal transport via a combinatorial algorithm. The work obtains a runtime of $\tilde{O}(n^2\|C\|_{\max}/\epsilon + n\|C\|_{\max}^2/\epsilon^2)$, which is worse than our runtime by a low-order term. Furthermore, it does not appear to be parallelizable.

Box-simplex objectives. Our main result follows from improved algorithms for bilinear minimax problems over one simplex domain and one box domain developed in [She17]. This fundamental minimax problem captures ℓ_1 and ℓ_∞ regression over a simplex and box respectively, and inspired the development of conjugate smoothing [Nes05] as well as mirror prox / dual extrapolation [Nem04, Nes07]. These latter two approaches are extragradient methods (using two gradient operations per iteration rather than one) for approximately solving a family of problems, which includes convex minimization and finding a saddle point to a convex-concave function. These methods simulate backwards Euler discretization of the gradient flow, similar to how mirror descent simulates forwards

Euler discretization [DO19]. The role of the extragradient step is a fixed point iteration (of two steps) which is a good approximation of the backwards Euler step when the operator is Lipschitz.

Nonetheless, the analysis of [Nem04, Nes07] fell short in obtaining a $1/T$ rate of convergence without worse dependence on dimension for these domains, where T is the iteration count (which would correspond to a $\tilde{O}\left(1/\epsilon\right)$ runtime for approximate minimization). The fundamental barrier was that over a box, any strongly-convex regularizer in the ℓ_∞ norm has a dimension-dependent domain size (shown in [ST18]). This barrier can also be viewed as the reason for the worse dimension dependence in the accelerated scheme of [DGK18, LHJ19].

The primary insight of [She17] was that previous approaches attempted to regularize the schemes of [Nem04, Nes07] with separable regularizers, i.e. the sum of a regularizer which depends only on the primal block and one which depends only on the dual. If, say, the domain of the primal block was a box, then such a regularization scheme would run into the ℓ_∞ barrier and incur a worse dependence on dimension. However, by more carefully analyzing the requirements of these algorithms, [She17] constructed a non-separable regularizer with small domain size, satisfying a property termed *area-convexity* which sufficed for provable convergence of dual extrapolation [Nes07]. Interestingly, the property seems specialized to dual extrapolation and not mirror prox [Nem04].

2 Overview

First, in Section 2.1 we first describe a reformulation of (1) as a primal-dual objective, which we solve approximately in Section 3. Then in Section 2.2 we give additional notation critical for our analysis[4]. In Section 3 we leverage this to give an overview of our main algorithm.

2.1 ℓ_1-regression formulation

We adapt the view of [BJKS18, Qua19] of the objective (1) as a positive linear program. Let d be the (vectorized) cost matrix C associated with the instance and let Δ^{n^2} be the n^2 dimensional simplex[5]. We recall r, c are specified row and column sums with $\mathbf{1}^\top r = \mathbf{1}^\top c = 1$. The optimal transport problem can be written as, for $m = n^2$, and $A \in \{0,1\}^{2n \times m}, b \in \mathbb{R}^{2n}_{\geq 0}$, for A the (unsigned) *edge-incidence matrix* of the underlying bipartite graph and b the concatenation of r and c.

$$\min_{x \in \Delta^m, Ax = b} d^\top x. \tag{2}$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}.$$

Figure 1: Edge-incidence matrix A of a 3×3 bipartite graph and uniform demands.

In particular, A is the 0-1 matrix on $V \times E$ such that $A_{ve} = 1$ iff v is an endpoint of edge e. We summarize some additional properties of the constraint matrix A and vector b.

Fact 2.1. *A, b have the following properties.*

1. *$A \in \{0,1\}^{2n \times m}$ has 2-sparse columns and n-sparse rows. Thus $\left\lVert A \right\rVert_{1 \to 1} = 2$.*
2. *$b^\top = \begin{pmatrix} r^\top & c^\top \end{pmatrix}$, so that $\left\lVert b \right\rVert_1 = 2$.*
3. *A has $2n^2$ nonzero entries.*

---

[4] Because many of the objects defined in Section 2.2 are developed in Section 2.1, we postpone their statement, but refer the reader to Section 2.2 for any ambiguous definitions.

[5] We use d because C often arises from distances in a metric space, and to avoid overloading c.

Section 4 recalls the proof of the following theorem, which first appeared in [AWR17].

Theorem 2.2 (Rounding guarantee, Lemma 7 in [AWR17])**.** *There is an algorithm which takes* $\tilde{x}$ *with* $\|A\tilde{x} - b\|_1 \leq \delta$ *and produces* $\hat{x}$ *in* $O(n^2)$ *time, with*

$$A\hat{x} = b, \|\tilde{x} - \hat{x}\|_1 \leq 2\delta.$$

We now show how the rounding procedure gives a roadmap for our approach. Consider the following ℓ_1 regression objective over the simplex (a similar penalized objective appeared in [She13]):

$$\min_{x \in \Delta^m} d^\top x + 2\|d\|_\infty \|Ax - b\|_1 . \tag{3}$$

We show that the penalized objective value is still OPT, and furthermore any approximate minimizer yields an approximate transport plan.

Lemma 2.3 (Penalized ℓ_1 regression)**.** *The value of* (3) *is* OPT*. Also, given* $\tilde{x}$*, an* ϵ*-approximate minimizer to* (3)*, we can find* ϵ*-approximate transportation plan* $\hat{x}$ *in* $O(n^2)$ *time.*

Proof. Recall OPT $= \min_{x \in \Delta^m, Ax=b} d^\top x$. Let $\tilde{x}$ be the minimizing argument in (3). We claim there is some optimal $\tilde{x}$ with $A\tilde{x} = b$; clearly, the first claim is then true. Suppose otherwise, and let $\|A\tilde{x} - b\|_1 = \delta > 0$. Then, let $\hat{x}$ be the result of the algorithm in Theorem 2.2, applied to $\tilde{x}$, so that $A\hat{x} = b$, $\|\tilde{x} - \hat{x}\|_1 \leq 2\delta$. We then have

$$d^\top \hat{x} + 2\|d\|_\infty \|A\hat{x} - b\|_1 = d^\top(\hat{x} - \tilde{x}) + d^\top \tilde{x} \leq d^\top \tilde{x} + \|d\|_\infty \|\hat{x} - \tilde{x}\|_1 \leq d^\top \tilde{x} + 2\|d\|_\infty \delta.$$

The objective value of $\hat{x}$ is no more than of $\tilde{x}$, a contradiction. By this discussion, we can take any approximate minimizer to (3) and round it to a transport plan without increasing the objective. □

Section 3 proves Theorem 2.4, which says we can efficiently find an approximate minimizer to (3).

Theorem 2.4 (Approximate ℓ_1 regression over the simplex)**.** *There is an algorithm (Algorithm 1) taking input* ϵ*, which has* $O((\|d\|_\infty \log n \log \gamma)/\epsilon)$ *parallel depth for* $\gamma = \log n \cdot \|d\|_\infty / \epsilon$*, and total work* $O(n^2(\|d\|_\infty \log n \log \gamma)/\epsilon)$*, and obtains* $\tilde{x}$ *an* ϵ*-additive approximation to the objective in* (3)*.*

We will approach proving Theorem 2.4 through a primal-dual viewpoint, in light of the following (based on the definition of the ℓ_1 norm):

$$\min_{x \in \Delta^m} d^\top x + 2\|d\|_\infty \|Ax - b\|_1 = \min_{x \in \Delta^m} \max_{y \in [-1,1]^{2n}} d^\top x + 2\|d\|_\infty \left(y^\top Ax - b^\top y\right). \tag{4}$$

Further, a low-*duality gap* pair to (4) yields an approximate minimizer to (3).

Lemma 2.5 (Duality gap to error)**.** *Suppose* x, y *is feasible* ($x \in \Delta^m, y \in [-1,1]^{2n}$)*, and for any feasible* u, v*,*

$$\left(d^\top x + 2\|d\|_\infty \left(v^\top Ax - b^\top v\right)\right) - \left(d^\top u + 2\|d\|_\infty \left(y^\top Au - b^\top y\right)\right) \leq \delta.$$

Then, we have $d^\top x + 2\|d\|_\infty \|Ax - b\|_1 \leq \delta +$ OPT.

Proof. The result follows from maximizing over v, and noting that for the minimizing u,

$$d^\top u + 2\|d\|_\infty \left(y^\top Au - b^\top y\right) \leq d^\top u + 2\|d\|_\infty \|Au - b\|_1 = \text{OPT}.$$

□

Correspondingly, Section 3 gives an algorithm which obtains (x, y) with bounded duality gap within the runtime of Theorem 2.4.

2.2 Notation

$\mathbb{R}_{\geq 0}$ is the nonnegative reals. $\mathbf{1}$ is the all-ones vector of appropriate dimension when clear. The probability simplex is $\Delta^d \stackrel{\text{def}}{=} \{v \mid v \in \mathbb{R}^d_{\geq 0}, \mathbf{1}^\top v = 1\}$. We say matrix X is in the simplex of appropriate dimensions when its (nonnegative) entries sum to one.

$\|\cdot\|_1$ and $\|\cdot\|_\infty$ are the ℓ_1 and ℓ_∞ norms, i.e. $\|v\|_1 = \sum_i |v_i|$ and $\|v\|_\infty = \max_i |v_i|$. When A is a matrix, we let $\|A\|_{p \to q}$ be the matrix operator norm, i.e. $\sup_{\|v\|_p = 1} \|Av\|_q$, where $\|\cdot\|_p$ is the ℓ_p norm. In particular, $\|A\|_{1 \to 1}$ is the largest ℓ_1 norm of a column of A.

Throughout $\log$ is the natural logarithm. For $x \in \Delta^d$, $h(x) = \sum_{i \in [d]} x_i \log x_i$ is (negative) entropy where $0 \log 0 = 0$ by convention. It is well-known that $\max_{x \in \Delta^d} h(x) - \min_{x \in \Delta^d} h(x) = \log d$.

We also use the Bregman divergence of a regularizer and the proximal operator of a divergence.

Definition 2.6 (Bregman divergence)**.** *For (differentiable) regularizer r and z, w in its domain, the* Bregman divergence *from z to w is*

$$V^r_z(w) \stackrel{\text{def}}{=} r(w) - r(z) - \langle \nabla r(z), w - z \rangle.$$

When r is convex, the divergence is nonnegative and convex in the argument (w in the definition).

Definition 2.7 (Proximal operator)**.** *For (differentiable) regularizer r, z in its domain, and g in the dual space (when the domain is in $\mathbb{R}^d$, so is the dual space), we define the* proximal operator *as*

$$\text{Prox}^r_z(g) \stackrel{\text{def}}{=} \text{argmin}_w \left\{ \langle g, w \rangle + V^r_z(w) \right\}.$$

Several variables have specialized meaning throughout. All graphs considered will be on $2n$ vertices with m edges, i.e. $m = n^2$. $A \in \mathbb{R}^{2n \times m}$ is the edge-incidence matrix. d is the vectorized cost matrix C. b is the constraint vector, concatenating row and column constraints r, c. In algorithms for solving (4), x and y are primal (in a simplex) and dual (in a box) variables respectively. In Section 3, we adopt the linear programming perspective where the decision variable $x \in \Delta^m$ is a vector. In Section 4, for convenience we take the perspective where X is an unflattened $n \times n$ matrix. $\mathcal{U}_{r,c}$ is the feasible polytope: when the domain is vectors, $\mathcal{U}_{r,c}$ is $x \mid Ax = b$, and when it is matrices, $\mathcal{U}_{r,c}$ is $X \mid X\mathbf{1} = r, X^\top \mathbf{1} = c$ (by flattening X this is consistent).

3 Main Algorithm

This section describes our algorithm for finding a primal-dual pair (x, y) with a small duality gap, with respect to the objective in (4), which we restate here for convenience:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} d^\top x + 2 \|d\|_\infty \left(y^\top A x - b^\top y \right), \; \mathcal{X} \stackrel{\text{def}}{=} \Delta^m, \; \mathcal{Y} \stackrel{\text{def}}{=} [-1, 1]^{2n}. \qquad \text{(Restatement of (4))}$$

Our algorithm is a specialization of the algorithm in [She17]. One of our technical contributions in this regard is an analysis of the algorithm which more closely relates it to the analysis of dual extrapolation [Nes07], an algorithm for finding approximate saddle points with a more standard analysis. In Section 3.1, we give the algorithmic framework and convergence analysis. In Section B.1, we provide analysis of an alternating minimization scheme for implementing steps of the procedure. The same procedure was used in [She17] which claimed without proof the linear convergence rate of the alternating minimization; we hope the analysis will make the method more broadly accessible to the optimization community. We defer many proofs to Appendix B.

3.1 Dual Extrapolation Framework

For an objective $F(x, y)$ convex in x and concave in y, the standard way to measure the duality gap is to define the *gradient operator* $g(x, y) = (\nabla_x F(x, y), -\nabla_y F(x, y))$, and show that for $z = (x, y)$ and any u on the product space, the *regret*, $\langle g(z), z - u \rangle$, is small. Correspondingly, we define

$$g(x, y) \stackrel{\text{def}}{=} \left(d + 2 \|d\|_\infty A^\top y, \; 2 \|d\|_\infty (b - Ax) \right).$$

The dual extrapolation framework [Nes07] requires a regularizer on the product space. The algorithm is simple to state; it takes two "mirror descent-like" steps each iteration, maintaining a state

s_t in the dual space[6]. A typical setup is a Lipschitz gradient operator and a regularizer which is the sum of canonical strongly-convex regularizers in the norms corresponding to the product space $\mathcal{X}, \mathcal{Y}$. However, recent works have shown that this setup can be greatly relaxed and still obtain similar rates of convergence. In particular, [She17] introduced the following definition.

Definition 3.1 (Area-convexity)**.** *Regularizer r is κ-area-convex with respect to operator g if for any points a, b, c in its domain,*

$$\kappa\left(r(a) + r(b) + r(c) - 3r\left(\frac{a+b+c}{3}\right)\right) \geq \langle g(b) - g(a), b - c\rangle. \tag{5}$$

Area-convexity is so named because $\langle g(b) - g(a), b - c\rangle$ can be viewed as measuring the "area" of the triangle with vertices a, b, c with respect to some Jacobian matrix. In the case of bilinear objectives, the left hand side in the definition of area-convexity is invariant to permuting a, b, c, whereas the sign of the right hand side can be flipped by interchanging a, c, so area-convexity implies convexity. However, it does not even imply the regularizer r is strongly-convex, a typical assumption for the convergence of mirror descent methods.

We state the algorithm for time horizon T; the only difference from [Nes07] is a factor of 2 in defining s_{t+1}, i.e. adding a $1/2\kappa$ multiple rather than $1/\kappa$. We find it of interest to explore whether this change is necessary or specific to the analysis of [She17].

Algorithm 1 $\bar{w} = \texttt{Dual-Extrapolation}(\kappa, r, g, T)$: Dual extrapolation with area-convex r.

```
Initialize $s_0 = 0$, let $\bar{z}$ be the minimizer of $r$.
for $t < T$ do
    $z_t \leftarrow \text{Prox}^r_{\bar{z}}(s_t)$.
    $w_t \leftarrow \text{Prox}^r_{\bar{z}}\left(s_t + \frac{1}{\kappa} g(z_t)\right)$.
    $s_{t+1} \leftarrow s_t + \frac{1}{2\kappa} g(w_t)$.
    $t \leftarrow t + 1$.
end for
return $\bar{w} \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t\in[T]} w_t$.
```

Lemma 3.2 (Dual extrapolation convergence)**.** *Suppose r is κ-area-convex with respect to g. Further, suppose for some u, $\Theta \geq r(u) - r(\bar{z})$. Then, the output $\bar{w}$ to Algorithm 1 satisfies*

$$\langle g(\bar{w}), \bar{w} - u\rangle \leq \frac{2\kappa\Theta}{T}.$$

In fact, by more carefully analyzing the requirements of dual extrapolation we have the following.

Corollary 3.3. *Suppose in Algorithm 1, the proximal steps are implemented with $\epsilon'/4\kappa$ additive error. Then, the upper bound of the regret in Lemma 3.2 is $2\kappa\Theta/T + \epsilon'$.*

We now state a useful second-order characterization of area-convexity involving a relationship between the Jacobian of g and the Hessian of r, which was proved in [She17].

Theorem 3.4 (Second-order area-convexity, Theorem 1.6 in [She17])**.** *For bilinear minimax objectives, i.e. whose associated operator g has Jacobian*

$$J = \begin{pmatrix} 0 & M^\top \\ -M & 0 \end{pmatrix},$$

and for twice-differentiable r, if for all z in the domain,

$$\begin{pmatrix} \kappa\nabla^2 r(z) & -J \\ J & \kappa\nabla^2 r(z) \end{pmatrix} \succeq 0,$$

then r is 3κ-area-convex with respect to g.

---

[6] In this regard, it is more similar to the "dual averaging" or "lazy" mirror descent setup [Bub15].

Finally, we complete the outline of the algorithm by stating the specific regularizer we use, which first appeared in [She17]. We then prove its 3-area-convexity with respect to g by using Theorem 3.4.

$$r(x, y) = 2\left\|d\right\|_\infty \left(10 \sum_{j \in [n]} x_j \log x_j + x^\top A^\top (y^2)\right), \tag{6}$$

where (y^2) is entry-wise. To give some motivation for this regularizer, one ℓ_∞-strongly convex regularizer is $\frac{1}{2}\left\|y\right\|_2^2$, but over the ℓ_∞ ball, this regularizer has large range. The term $x^\top A^\top (y^2)$ in (6) captures the curvature required for strong-convexity locally, but has a smaller range due to the restrictions on x, A. The constants chosen were the smallest which satisfy the assumptions of the following Lemma 3.5.

Lemma 3.5 (Area-convexity of the Sherman regularizer)**.** *For the Jacobian J associated with the objective in* (4) *and the regularizer r defined in* (6)*, we have*

$$\begin{pmatrix} \nabla^2 r(z) & -J \\ J & \nabla^2 r(z) \end{pmatrix} \succeq 0.$$

We now give the proof of Theorem 2.4, requiring some claims in Appendix B.1 for the complexity of Algorithm 1. In particular, Appendix B.1 implies that although the minimizer to the proximal steps cannot be computed in closed form because of non-separability, a simple alternating scheme converges to an approximate-minimizer in near-constant time.

Proof of Theorem 2.4. The algorithm is Algorithm 1, using the regularizer r in (6). Clearly, in the feasible region the range of the regularizer is at most $20\left\|d\right\|_\infty \log n + 4\left\|d\right\|_\infty$, where the former summand comes from the range of entropy and the latter $\left\|A^\top\right\|_\infty = 2$. We may choose $\Theta = O(\left\|d\right\|_\infty \log n)$ to be the range of r to satisfy the assumptions of Lemma 3.2, since for all u, $\langle \nabla r(\bar{z}), \bar{z} - u\rangle \leq 0 \Rightarrow V^r_{\bar{z}}(u) \leq r(u) - r(\bar{z})$.

By Theorem 3.4 and Lemma 3.5, r is 3-area-convex with respect to g. By Corollary 3.3, $T = 12\Theta/\epsilon$ iterations suffice, implementing each proximal step to $\epsilon/12$-additive accuracy. Finally, using Theorem B.5 to bound this implementation runtime concludes the proof. $\square$

4 Rounding to $\mathcal{U}_{r,c}$

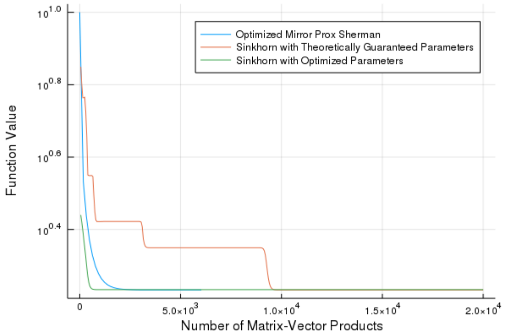
We state the rounding procedure in [AWR17] for completeness here, which takes a transport plan $\tilde{X}$ close to $\mathcal{U}_{r,c}$ and transforms it into a plan which exactly meets the constraints and is close to $\tilde{X}$ in ℓ_1, and then prove its correctness in Appendix C. Throughout $r(X) \stackrel{\text{def}}{=} X\mathbf{1}, c(X) \stackrel{\text{def}}{=} X^\top \mathbf{1}$.

Algorithm 2 $\hat{X} = \texttt{Rounding}(\tilde{X}, r, c)$: Rounding to feasible polytope

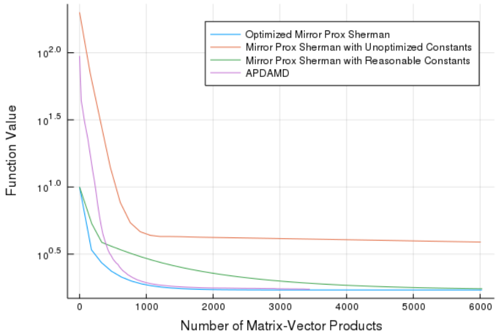
```
$X' \leftarrow \textbf{diag}\left(\min\left(\frac{r}{r(\tilde{X})}, 1\right)\right)\tilde{X}$.
$X'' \leftarrow X'\textbf{diag}\left(\min\left(\frac{c}{c(X')}, 1\right)\right)$.
$e_r \leftarrow r - \mathbf{1}^\top r(X''), e_c \leftarrow c - \mathbf{1}^\top c(X''), E \leftarrow \mathbf{1}^\top e_r$.
$\hat{X} \leftarrow X'' + \frac{1}{E} e_r e_c^\top$.
return $\hat{X}$.
```

5 Experiments

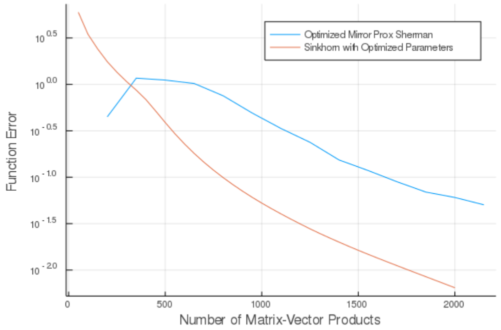
We show experiments illustrating the potential of our algorithm to be useful in practice, by considering its performance on computing optimal transport distances on the MNIST dataset and comparing against algorithms in the literature including APDAMD [LHJ19] and Sinkhorn iteration. All comparisons are based on the number of matrix-vector multiplications (rather than iterations, due to our algorithm's alternating subroutine), the main computational component of all algorithms considered.



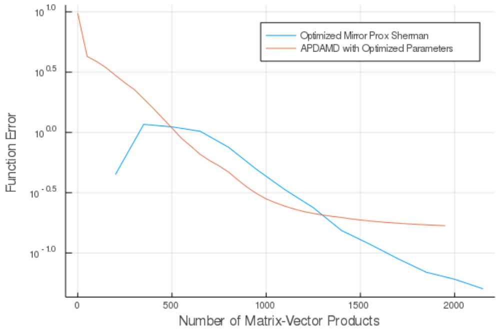
(a) Comparison with Sinkhorn iteration with different parameters.



(b) Comparison with APDAMD [LHJ19] with different parameters.



(a) Comparison with Sinkhorn iteration on 20 randomly chosen MNIST digit pairs.



(b) Comparison with APDAMD [LHJ19] on 20 randomly chosen MNIST digit pairs.

While our unoptimized algorithm performs poorly, slightly optimizing the size of the regularizer and step sizes used results in an algorithm with competitive performance to APDAMD, the first-order method with the best provable guarantees and observed practical performance. Sinkhorn iteration outperformed all first-order methods experimentally; however, an optimized version of our algorithm performed better than conservatively-regularized Sinkhorn iteration, and was more competitive with variants of Sinkhorn found in practice than other first-order methods.

As we discuss in our implementation details (Appendix D), we acknowledge that implementations of our algorithm illustrated are not the same as those with provable guarantees in our paper. However, we believe that our modifications are justifiable in theory, and consistent with those made in practice to existing algorithms. Further, we hope that studying the modifications we made (step size, using mirror prox [Nem04] for stability considerations), as well as the consideration of other numerical speedups such as greedy updates [AWR17] or kernel approximations [ABRW18], will become fruitful for understanding the potential of accelerated first-order methods in both the theory and practice of computational optimal transport.

Acknowledgements

We thank Jose Blanchet and Carson Kent for helpful conversations. AJ was supported by NSF Graduate Fellowship DGE-114747. AS was supported by NSF CAREER Award CCF-1844855. KT was supported by NSF Graduate Fellowship DGE-1656518.

References

[ABRW18] Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Weed. Approximating the quadratic transportation metric in near-linear time. *CoRR*, abs/1810.10046, 2018. 1.2, 5, D

[ACB17] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 214–223, 2017. 1

[ANOY14] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014. 1.2

[AO15] Zeyuan Allen Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 229–236, 2015. 1.2

[AS14] Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 555–564, 2014. 1.2

[AWR17] 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 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1961–1971, 2017. 1, 1.2, **??**, 2.1, 2.2, 4, 5, 2.2, D

[BJKS18] Jose Blanchet, Arun Jambulapati, Carson Kent, and Aaron Sidford. Towards optimal running times for optimal transport. *CoRR*, abs/1810.07717, 2018. (document), 1, 1.1, 1.2, **??**, **??**, 2.1

[BK17] Jose H. Blanchet and Yang Kang. Distributionally robust groupwise regularization estimator. In *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017, Seoul, Korea, November 15-17, 2017.*, pages 97–112, 2017. 1

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015. 6

[BvdPPH11] Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph.*, 30(6):158:1–158:12, 2011. 1

[CK18] Deeparnab Chakrabarty and Sanjeev Khanna. Better and simpler error analysis of the sinkhorn-knopp algorithm for matrix scaling. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 4:1–4:11, 2018. 1

[CMTV17] Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 902–913, 2017. 1.2

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2292–2300, 2013. 1, 1.2

[DGK18] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1366–1375, 2018. (document), 1, 1.1, 1.2, **??**, 1, 1.2

[DO19] Jelena Diakonikolas and Lorenzo Orecchia. The approximate duality gap technique: A unified theory of first-order methods. *SIAM Journal on Optimization*, 29(1):660–689, 2019. 1.2

[EK18] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *Math. Program.*, 171(1-2):115–166, 2018. 1

[GCPB16] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis R. Bach. Stochastic optimization for large-scale optimal transport. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3432–3440, 2016. 1, 1.2

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. 1.2

[KLOS14] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226, 2014. 1.2

[LHJ19] Tianyi Lin, Nhat Ho, and Michael I. Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. *CoRR*, abs/1901.06482, 2019. (document), 1, 1.1, 1.2, **??**, 1, 1.2, 5, 2b, 3b, D

[LMR19] Nathaniel Lahn, Deepika Mulchandani, and Sharath Raghvendra. A graph theoretic additive approximation of optimal transport. *CoRR*, abs/1905.11830, 2019. 1.2

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433, 2014. 1.2

[LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 230–249, 2015. 1.2, **??**

[Nem04] Arkadi Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004. 1.2, 5, D

[Nes05] Yurii Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005. 1.2, 1.2

[Nes07] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Math. Program.*, 109(2-3):319–344, 2007. 1.2, 3, 3.1, 3.1

[PZ16] Victor M. Panaretos and Yoav Zemel. Amplitude and phase variation of point processes. *Annals of Statistics*, 44(2):771–812, 2016. 1

[Qua19] Kent Quanrud. Approximating optimal transport with linear programs. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 6:1–6:9, 2019. (document), 1, 1.1, 1.2, **??**, 2.1

[SA12] R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 385–394, 2012. 1.2

[San09] Piotr Sankowski. Maximum weight bipartite matching in matrix multiplication time. *Theor. Comput. Sci.*, 410(44):4480–4488, 2009. 1.2

[SdGP$^+$15] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas J. Guibas. Convolutional wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, 2015. 1

[She13] Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 263–269, 2013. 1.2, 2.1

[She17] Jonah Sherman. Area-convexity, l_∞ regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 452–460, 2017. (document), 1.1, 1.2, 3, 3.1, 3.1, 3.1, 3.4, 3.1

[ST18] Aaron Sidford and Kevin Tian. Coordinate methods for accelerating ℓ_∞ regression and faster approximate maximum flow. In *59th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2018, 7-9 October, 2018, Paris, France*, 2018. 1.2, 1.2

[You01] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 538–546, 2001. 1.2

[ZLdOW17] Zeyuan Allen Zhu, Yuanzhi Li, Rafael Mendes de Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 890–901, 2017. 1.2

A Algorithm

We give the complete algorithm for approximating optimal transport distance to additive ϵ here. We assume $C \in \mathbb{R}_{\geq 0}^{n \times n}$ and $r, c \in \Delta^n$. Finally, we refer to blocks of variable z on a product space as z^x, z^y, i.e. $z = (z^x, z^y)$. Again $r(X) \overset{\text{def}}{=} X\mathbf{1}$, $c(X) \overset{\text{def}}{=} X^\top \mathbf{1}$.

Algorithm 3 $\hat{X} =$ `Optimal-Transport`(C, ϵ, r, c): Produces ϵ-approximate transportation plan

```
Vectorize $C$ to produce $d$.
Let $b$ be $r, c$ concatenated; let $A$ be the incidence matrix of a complete $n \times n$ bipartite graph.
$t \leftarrow 0$.
$x_0 \leftarrow \frac{1}{n^2}\mathbf{1}$, $y_0 \leftarrow \mathbf{0_{2n}}$.
$s_0^x \leftarrow \mathbf{0_{n^2}}$, $s_0^y \leftarrow \mathbf{0_{2n}}$.
$\Theta \leftarrow 20\left\|d\right\|_\infty \log n + 4\left\|d\right\|_\infty$.
while $d^\top x_{t+\frac{1}{2}} + 2\left\|d\right\|_\infty \left\|Ax_{t+\frac{1}{2}} - b\right\|_1 \leq -2\left\|d\right\|_\infty b^\top y_{t+\frac{1}{2}} + \max_j \left[d + 2\left\|d\right\|_\infty A^\top y_{t+\frac{1}{2}}\right]_j + \epsilon$
do
  $t \leftarrow t + 1$.
  $k \leftarrow 0$.
  $x'_0 \leftarrow x_{t-\frac{1}{2}}$, $y'_0 \leftarrow y_{t-\frac{1}{2}}$.
  for $0 \leq k < \left\lceil 24 \log\left(\left(\frac{88\left\|d\right\|_\infty}{\epsilon^2} + \frac{2}{\epsilon}\right)\Theta\right)\right\rceil$ do
    $x'_k \leftarrow \exp\left(\frac{1}{20\left\|d\right\|_\infty} s_t^x + \frac{1}{10} A^\top (y'_{k-1})^2\right)$, $x'_k \leftarrow x'_k / \left\|x'_k\right\|_1$.
    $y'_k \leftarrow \min\left(1, \max\left(-1, \frac{-s_t^y}{4\left\|d\right\|_\infty Ax'_k}\right)\right)$. Operations are element-wise.
  end for
  $x_t \leftarrow x'_k$, $y_t \leftarrow y'_k$.
  $s_{t+\frac{1}{2}}^x \leftarrow s_t^x + \frac{1}{3}\left(d + 2\left\|d\right\|_\infty A^\top y_t\right)$.
  $s_{t+\frac{1}{2}}^y \leftarrow s_t^y + \frac{1}{3}\left(2\left\|d\right\|_\infty (b - Ax_t)\right)$.
  $k \leftarrow 0$.
  $x'_0 \leftarrow x_t$, $y'_0 \leftarrow y_t$.
  for $0 \leq k < \left\lceil 24 \log\left(\left(\frac{88\left\|d\right\|_\infty}{\epsilon^2} + \frac{2}{\epsilon}\right)\Theta\right)\right\rceil$ do
    $x'_k \leftarrow \exp\left(\frac{1}{20\left\|d\right\|_\infty} s_{t+\frac{1}{2}}^x + \frac{1}{10} A^\top (y'_{k-1})^2\right)$, $x'_k \leftarrow x'_k / \left\|x'_k\right\|_1$.
    $y'_k \leftarrow \min\left(1, \max\left(-1, \frac{-s_{t+\frac{1}{2}}^y}{4\left\|d\right\|_\infty Ax'_k}\right)\right)$. Operations are element-wise.
  end for
  $x_{t+\frac{1}{2}} \leftarrow x'_k$, $y_{t+\frac{1}{2}} \leftarrow y'_k$.
  $s_{t+1}^x \leftarrow s_t^x + \frac{1}{6}\left(d + 2\left\|d\right\|_\infty A^\top y_{t+\frac{1}{2}}\right)$.
  $s_{t+1}^y \leftarrow s_t^y + \frac{1}{6}\left(2\left\|d\right\|_\infty (b - Ax_{t+\frac{1}{2}})\right)$.
end while
Un-vectorize $x$ to produce $\tilde{X}$.
$X' \leftarrow \textbf{diag}\left(\min\left(\frac{r}{r(\tilde{X})}, 1\right)\right)\tilde{X}$.
$X'' \leftarrow X'\textbf{diag}\left(\min\left(\frac{c}{c(X')}, 1\right)\right)$.
$e_r \leftarrow r - \mathbf{1}^\top r(X''), e_c \leftarrow c - \mathbf{1}^\top c(X''), E \leftarrow \mathbf{1}^\top e_r$.
$\hat{X} \leftarrow X'' + \frac{1}{E} e_r e_c^\top$.
return $\hat{X}$.
```

We remark that there are a variety of termination conditions that can be useful in practice for the alternating minimization procedure. For example, a standard early-stopping condition based on the observed movement of consecutive iterates was very successful in practice (Appendix D).

B Missing proofs from Section 3

In this section, we state missing proofs from Section 3. We provide the efficient implementation of the proximal steps required by Algorithm 1 in Appendix B.1.

Proof of Lemma 3.2. Our first step is to prove the following inequality:

$$\frac{1}{2\kappa}\langle g(w_t), w_t - \bar{z}\rangle \le \langle s_{t+1}, z_{t+1} - \bar{z}\rangle + V^r_{\bar{z}}(z_{t+1}) - \langle s_t, z_t - \bar{z}\rangle - V^r_{\bar{z}}(z_t). \tag{7}$$

Let $c_t = \frac{z_t + w_t + z_{t+1}}{3}$. The proof follows from minimality of z_t with respect to c_t, minimality of w_t with respect to z_{t+1}, and area-convexity (5) with respect to z_t, w_t, and z_{t+1}. Respectively,

$$\begin{aligned}
\langle s_t, z_t\rangle + r(z_t) &\le \langle s_t, c_t\rangle + r(c_t)\\
\langle s_t, w_t\rangle + \frac{1}{\kappa}\langle g(z_t), w_t\rangle + r(w_t) &\le \langle s_t, z_{t+1}\rangle + \frac{1}{\kappa}\langle g(z_t), z_{t+1}\rangle + r(z_{t+1})\\
\frac{1}{\kappa}\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle &\le r(z_t) + r(w_t) + r(z_{t+1}) - 3r\left(c_t\right).
\end{aligned} \tag{8}$$

Adding three times the first equation to the third, rearranging, and using the definition of c_t, we have

$$\frac{1}{\kappa}\langle g(w_t) - g(z_t), w_t - z_{t+1}\rangle \le r(w_t) + r(z_{t+1}) - 2r(z_t) + \langle s_t, w_t + z_{t+1} - 2z_t\rangle.$$

Rearranging the second equation, we have

$$\frac{1}{\kappa}\langle g(z_t), w_t - z_{t+1}\rangle \le r(z_{t+1}) - r(w_t) + \langle s_t, z_{t+1} - w_t\rangle.$$

Adding these two equations, we have

$$\frac{1}{\kappa}\langle g(w_t), w_t - z_{t+1}\rangle \le 2r(z_{t+1}) - 2r(z_t) + \langle s_t, 2z_{t+1} - 2z_t\rangle.$$

Dividing by 2 and adding $\frac{1}{2\kappa}\langle g(w_t), z_{t+1} - \bar{z}\rangle$ to both sides, we obtain the desired (7). Now, define the potential function

$$\Phi_k = \frac{1}{2\kappa}\sum_{t=0}^{k-1}\langle g(w_t), w_t - \bar{z}\rangle - \langle s_k, z_k - \bar{z}\rangle - V^r_{\bar{z}}(z_k)$$

Then, by (7), Φ_k is nonincreasing in k. Therefore for any u, by the definition of Θ,

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\langle g(w_t), w_t - u\rangle &\le \frac{1}{T}\sum_{t=0}^{T-1}\langle g(w_t), w_t - \bar{z}\rangle + \frac{1}{T}\sum_{t=0}^{T-1}\langle g(w_t), \bar{z} - u\rangle + \left(\frac{2\kappa\Theta}{T} - \frac{2\kappa V_{\bar{z}}(u)}{T}\right)\\
&\le \frac{1}{T}\sum_{t=0}^{T-1}\langle g(w_t), w_t - \bar{z}\rangle + \frac{1}{T}\sum_{t=0}^{T-1}\langle g(w_t), \bar{z} - z_T\rangle + \left(\frac{2\kappa\Theta}{T} - \frac{2\kappa V_{\bar{z}}(z_T)}{T}\right)\\
&= \frac{2\kappa}{T}\Phi_T + \frac{2\kappa\Theta}{T} \le \frac{2\kappa}{T}\Phi_0 + \frac{2\kappa\Theta}{T} = \frac{2\kappa\Theta}{T}.
\end{aligned}$$

The inequality on the second line used the definition of $z_T = \mathrm{Prox}^r_{\bar{z}}\left(\frac{1}{2\kappa}\sum_{t\in[T-1]} g(w_t)\right)$, and the last inequality is $\Phi_T \le \Phi_0$. The conclusion follows from the definition of g (because it is linear). □

Proof of Corollary 3.3. We see that (7) now holds up to $\frac{\epsilon'}{2\kappa}$ additive error, so that Φ_k is increasing by at most $\frac{\epsilon'}{2\kappa}$ each step. Thus, we obtain $\Phi_T \le \Phi_0 + \frac{T\epsilon'}{2\kappa}$, yielding the conclusion. □

Proof of Lemma 3.5. We scale both r and J down by $2\left\|d\right\|_\infty$, which does not affect positive-semidefiniteness. By computation we have (recalling all columns of A have ℓ_1 norm of 2)

$$\nabla^2 r(x,y) = \begin{pmatrix} 5\left\|A_{:j}\right\|_1 \mathbf{diag}\left(\frac{1}{x_j}\right) & 2A^\top \mathbf{diag}\left(y_i\right)\\ 2\mathbf{diag}\left(y_i\right)A & 2\mathbf{diag}\left(A_i^\top x\right)\end{pmatrix}.$$

It suffices to show that for any vector $(a \quad b \quad c \quad d)$ we have

$$(a \quad b \quad c \quad d)\begin{pmatrix} 5\left\|A_{:j}\right\|_1 \textbf{diag}\left(\frac{1}{x_j}\right) & 2A^\top\textbf{diag}\left(y_i\right) & 0 & -A^\top \\ 2\textbf{diag}\left(y_i\right)A & 2\textbf{diag}\left(A_i^\top x\right) & A & 0 \\ 0 & A^\top & 5\left\|A_{:j}\right\|_1 \textbf{diag}\left(\frac{1}{x_j}\right) & 2A^\top\textbf{diag}\left(y_i\right) \\ -A & 0 & 2\textbf{diag}\left(y_i\right)A & 2\textbf{diag}\left(A_i^\top x\right) \end{pmatrix}\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

is nonnegative. Upon simplifying and gathering like terms, it suffices to show

$$\sum_{i,j} A_{ij}\left(\frac{5a_j^2}{x_j} + 4a_jb_iy_i + 2b_i^2x_j - 2a_jd_i + 2c_jb_i + \frac{5c_j^2}{x_j} + 4c_jd_iy_i + 2d_i^2x_j\right) \geq 0.$$

However, this is true for $y_i \in [-1, 1]$, since each coefficient groups into clearly nonnegative terms,

$$\left(\frac{4a_j^2}{x_j} + 4a_jb_iy_i + b_i^2x_j\right) + \left(\frac{a_j^2}{x_j} - 2a_jd_i + d_i^2x_j\right)$$
$$+\left(\frac{4c_j^2}{x_j} + 4c_jd_iy_i + d_i^2x_j\right) + \left(\frac{c_j^2}{x_j} + 2c_jb_i + b_i^2x_j\right).$$

□

B.1 Alternating Minimization Analysis

In this section, we give the convergence analysis of an alternating minimization procedure for minimizing a function of the form (throughout this section, $r(x, y)$ is as in (6))

$$f(x, y) \stackrel{\text{def}}{=} \langle \xi, x\rangle + \langle \eta, y\rangle + r(x, y) \tag{9}$$

which is the type of minimization problem arising from steps of the form $\text{Prox}_{\bar{z}}^{r}(g)$. As we will see, $f(x, y)$ is jointly convex. Throughout this section, let $x_{\text{OPT}}, y_{\text{OPT}}$ be the minimizer to f. Corollary 3.3 states that $O(\epsilon)$ additive error to f gives the same asymptotic convergence rate in Algorithm 1. We will show that a simple alternating minimization scheme enjoys a linear rate of convergence in our setting; thus, roughly $O(\log \epsilon^{-1})$ iterations suffice. We first give a proof of a general condition which suffices for linear convergence.

Lemma B.1. *Suppose $f(x, y)$ is twice-differentiable and jointly convex, over the product space $\mathcal{X} \times \mathcal{Y}$. Consider the alternating minimization scheme,*

1. $x_{k+1} \stackrel{\text{def}}{=} \text{argmin}_{x\in\mathcal{X}} f(x, y_k)$
2. $y_{k+1} \stackrel{\text{def}}{=} \text{argmin}_{y\in\mathcal{Y}} f(x_{k+1}, y)$

Further, suppose there are convex regions $\mathcal{X}_{k+1} \subseteq \mathcal{X}$, $\mathcal{Y}_k \subseteq \mathcal{Y}$ which contain x_{k+1}, y_k respectively, such that for any $x' \in \mathcal{X}_{k+1}$, $y', y'' \in \mathcal{Y}_k$, and for some $\sigma \geq 1$,

$$\nabla^2 f(x', y') \succeq \frac{1}{\sigma}\nabla^2_{yy} f(x_{k+1}, y''), \tag{10}$$

where ∇^2_{yy} is the Hessian with all but the yy block zeroed out. Then, for any $x^* \in \mathcal{X}_{k+1}$, $y^* \in \mathcal{Y}_k$,

$$f(x_{k+1}, y_k) - f(x_{k+1}, y_{k+1}) \geq \frac{1}{\sigma}\left(f(x_{k+1}, y_k) - f(x^*, y^*)\right).$$

Proof. Let $\tilde{y} = \left(1 - \frac{1}{\sigma}\right)y_k + \frac{1}{\sigma}y^*$. We will prove instead that

$$f(x_{k+1}, y_k) - f(x_{k+1}, \tilde{y}) \geq \frac{1}{\sigma}\left(f(x_{k+1}, y_k) - f(x^*, y^*)\right),$$

from which the conclusion will follow since $f(x_{k+1}, y_{k+1}) \leq f(x_{k+1}, \tilde{y})$. Note by definition of $\tilde{y}$, as well as optimality of x_{k+1} which implies $0 \geq \langle \nabla_x f(x_{k+1}, y_k), x_{k+1} - x^*\rangle$,

$$\langle \nabla_y f(x_{k+1}, y_k), y_k - \tilde{y}\rangle = \frac{1}{\sigma}\langle \nabla_y f(x_{k+1}, y_k), y_k - y^*\rangle \geq \frac{1}{\sigma}\langle \nabla f(x_{k+1}, y_k), z_{k+\frac{1}{2}} - z^*\rangle \tag{11}$$

where $z_{k+\frac{1}{2}} \overset{\text{def}}{=} (x_{k+1}, y_k)$ and $z^* \overset{\text{def}}{=} (x^*, y^*)$. Further, let $y_\alpha \overset{\text{def}}{=} (1-\alpha)y_k + \alpha y^*$, $\tilde{y}_\alpha \overset{\text{def}}{=} (1-\alpha)y_k + \alpha\tilde{y}$, and $x_\alpha \overset{\text{def}}{=} (1-\alpha)x_{k+1} + \alpha x^*$. Then, by Taylor expansion we have $f(x_{k+1}, y_k) - f(x_{k+1}, \tilde{y})$ equals

$$\begin{aligned}
&\langle \nabla_y f(x_{k+1}, y_k), y_k - \tilde{y}\rangle - \int_0^1 \int_0^\beta (\tilde{y} - y_k)^\top \nabla^2_{yy} f(x_{k+1}, \tilde{y}_\alpha)(\tilde{y} - y_k) d\alpha d\beta \\
&\geq \frac{1}{\sigma}\langle \nabla f(x_{k+1}, y_k), z_{k+\frac{1}{2}} - z^*\rangle - \frac{1}{\sigma^2}\int_0^1 \int_0^\beta (y^* - y_k)^\top \nabla^2_{yy} f(x_{k+1}, \tilde{y}_\alpha)(y^* - y_k) d\alpha d\beta \\
&\geq \frac{1}{\sigma}\left(\langle \nabla f(x_{k+1}, y_k), z_{k+\frac{1}{2}} - z^*\rangle - \int_0^1 \int_0^\beta (z^* - z_{k+\frac{1}{2}})^\top \nabla^2 f(x_\alpha, y_\alpha)(z^* - z_{k+\frac{1}{2}}) d\alpha d\beta\right) \\
&= \frac{1}{\sigma}\left(f(x_{k+1}, y_k) - f(x^*, y^*)\right).
\end{aligned}$$

In the first inequality, we used (11) and the definition of $\tilde{y}$, and in the second we used (10) (since $x_\alpha \in \mathcal{X}_{k+1}, y_\alpha, \tilde{y}_\alpha \in \mathcal{Y}_k$ by convexity). □

We now give a helper lemma specialized to the particular f in (9), which will be used in the proof of convergence.

Lemma B.2. *For some x_{k+1}, y_k, let $\mathcal{X}_{k+1} = \left\{x \mid x \geq \frac{1}{2}x_{k+1}\right\}$ where the inequality is entrywise, and let $\mathcal{Y}_k$ be the entire domain of y (i.e. $\mathcal{Y}$). Then for any $x' \in \mathcal{X}_{k+1}, y', y'' \in \mathcal{Y}_k$,*

$$\nabla^2 r(x', y') \succeq \frac{1}{12}\nabla^2_{yy} r(x_{k+1}, y'').$$

Proof. Recall that (since $\|A_{:j}\|_1 = 2$)

$$\nabla^2 r(x, y) = 2\|d\|_\infty \begin{pmatrix} 5\|A_{:j}\|_1 \,\mathbf{diag}\left(\frac{1}{x_j}\right) & 2A^\top \mathbf{diag}(y_i) \\ 2\mathbf{diag}(y_i) A & 2\mathbf{diag}\left(A_i^\top x\right)\end{pmatrix}.$$

Consider the diagonal approximation

$$D(x) = 2\|d\|_\infty \begin{pmatrix} \|A_{:j}\|_1 \,\mathbf{diag}\left(\frac{1}{x_j}\right) & 0 \\ 0 & \mathbf{diag}\left(A_i^\top x\right)\end{pmatrix}.$$

We claim for any y,

$$D(x) \preceq \nabla^2 r(x, y) \preceq 6D(x). \tag{12}$$

To see this, consider the quadratic forms with respect to some vector $(u \quad v)$:

$$\begin{aligned}
(u \quad v)\,\nabla^2 r(x, y)\begin{pmatrix} u \\ v\end{pmatrix} &= 2\|d\|_\infty \sum_{i,j} A_{ij}\left(\frac{5u_j^2}{x_j} + 4u_j v_i y_i + 2v_i^2 x_j\right), \\
(u \quad v)\,D(x)\begin{pmatrix} u \\ v\end{pmatrix} &= 2\|d\|_\infty \sum_{i,j} A_{ij}\left(\frac{u_j^2}{x_j} + v_i^2 x_j\right).
\end{aligned}$$

Now (12) follows because for any $y_i \in [-1, 1]$, it's easy to verify

$$\frac{u_j^2}{x_j} + v_i^2 x_j \leq \frac{5u_j^2}{x_j} + 4u_j v_i y_i + 2v_i^2 x_j \leq 6\left(\frac{u_j^2}{x_j} + v_i^2 x_j\right).$$

Therefore, to prove the lemma statement we can use

$$\nabla^2 r(x', y') \succeq D(x') \succeq \frac{1}{2}D(x_{k+1}) \succeq \frac{1}{12}\nabla^2_{yy} r(x_{k+1}, y'').$$

The inequality $D(x') \succeq \frac{1}{2}D(x_{k+1})$ followed from the definition of $\mathcal{X}_{k+1}$, and the last inequality followed from $D(x_{k+1})$ spectrally dominating $\frac{1}{6}\nabla^2 r(x_{k+1}, y'')$, and restrictions of $D(x_{k+1})$ to the yy block can only decrease the quadratic form. □

We now give the proof of the linear rate of convergence.

Lemma B.3. *For $f(x,y)$ defined in (9), the alternating minimization scheme*

1. $x_{k+1} \overset{\text{def}}{=} \text{argmin}_{x\in\mathcal{X}} f(x, y_k)$.
2. $y_{k+1} \overset{\text{def}}{=} \text{argmin}_{y\in\mathcal{Y}} f(x_{k+1}, y)$.

decreases the function error $f(x_k, y_k) - f(x_{\text{OPT}}, y_{\text{OPT}})$ by a factor of at least $1/24$ in each iteration.

Proof. We can apply Lemma B.1 with the sets defined in Lemma B.2, with $\sigma = 12$. On iteration k, consider picking the points $x^*, y^* = \frac{1}{2}(x_{k+1}+x_{\text{OPT}}), \frac{1}{2}(y_k+y_{\text{OPT}})$. Evidently, $x^* \in \mathcal{X}_{k+1}, y^* \in \mathcal{Y}_k$. Therefore, since $f(x_{k+1}, y_{k+1}) \geq f(x_{k+2}, y_{k+1})$,

$$f(x_{k+1}, y_k) - f(x_{k+2}, y_{k+1}) \geq f(x_{k+1}, y_k) - f(x_{k+1}, y_{k+1}) \geq \frac{1}{12}(f(x_{k+1}, y_k) - f(x^*, y^*)).$$

Furthermore, by convexity, we have

$$f(x_{k+1}, y_k) - f(x^*, y^*) \geq \frac{1}{2}(f(x_{k+1}, y_k) - f(x_{\text{OPT}}, y_{\text{OPT}})).$$

Finally, combining these two inequalities and rearranging,

$$\frac{23}{24}(f(x_{k+1}, y_k) - f(x_{\text{OPT}}, y_{\text{OPT}})) \geq f(x_{k+2}, y_{k+1}) - f(x_{\text{OPT}}, y_{\text{OPT}}).$$

Thus, by taking a y step and then an x step, we decrease the function error by a $1/24$ factor. $\square$

Finally, we show that steps of the alternating minimization can be implemented in linear time.

Lemma B.4. *For $f(x,y)$ defined in (9), we can implement the steps*

1. $x_{k+1} \overset{\text{def}}{=} \text{argmin}_{x} f(x, y_k)$.
2. $y_{k+1} \overset{\text{def}}{=} \text{argmin}_{y} f(x_{k+1}, y)$.

restricted to the relevant domains, in time $O(n^2)$.

Proof. Recall A has n^2 nonzero entries, so a matrix-vector multiplication can be performed in this time. Computing x in linear time is straightforward: it is defined by

$$\text{argmin}_x \langle \gamma, x\rangle + \sum_{j\in[n]} x_j \log x_j \text{ such that } x \in \Delta^m, \gamma \overset{\text{def}}{=} \frac{1}{20\|d\|_\infty}\xi + \frac{1}{10}A^\top(y^2).$$

By examining the KKT conditions, it is clear that the minimizing x is proportional to $\exp(-\gamma)$; computing γ takes $O(n^2)$ time, as does the simplex projection. Similarly, computing y in linear time is simple for fixed x: it is

$$\text{argmin}_y \langle \eta, y\rangle + \langle 2\|d\|_\infty Ax, y^2\rangle \text{ such that } y \in [-1,1]^{2n},$$

which is coordinate-wise decomposable as minimizing a quadratic over an interval. $\square$

Theorem B.5 (Complexity of alternating minimization)**.** *We can obtain an $\epsilon/2$-approximate minimizer to the proximal steps required by Algorithm 1 to $\epsilon/2$ accuracy, with the regularizer of* (6) *and $\kappa = 3$, in $O(\log\gamma)$ parallelizable iterations for $\gamma = \log n \cdot \|d\|_\infty \cdot \epsilon^{-1}$, and $O(n^2 \log\gamma)$ total work.*

Proof. By Lemmas B.3 and B.4, we can spend $O(n^2)$ parallelizable work to decrease the suboptimality gap by a $1/24$ factor, so it remains to argue that the initial error is at most $\text{poly}(\log n, \|d\|_\infty, \epsilon^{-1})$ to show that implementing the proximal steps to additive error $\epsilon/2$ can be done in $O(\log\gamma)$ iterations. We show that this is true for implementing the proximal step for z_t; a similar argument holds for w_t. To this end, note that by our setting of κ, for any z where we let $g(z) = (g^x(z), g^y(z))$,

$$\frac{1}{2\kappa}\left\|g^x(z)\right\|_\infty = \frac{1}{6}\left\|d + 2\left\|d\right\|_\infty A^\top y\right\|_\infty \leq \frac{\left\|d\right\|_\infty}{2},$$
$$\frac{1}{2\kappa}\left\|g^y(z)\right\|_1 = \frac{1}{6}\left\|2\left\|d\right\|_\infty (b - Ax)\right\|_1 \leq \frac{4\left\|d\right\|_\infty}{3}.$$

Therefore, for $s_t = (s_t^x, s_t^y)$, by the triangle inequality, and $t \leq 12\Theta/\epsilon$ the bound on the number of steps required where Θ is the range of r, we have

$$\left\|s_t^x\right\|_\infty \leq t \cdot \frac{1}{2\kappa}\left\|g^x(z)\right\|_\infty \leq \frac{6\left\|d\right\|_\infty \Theta}{\epsilon},$$
$$\left\|s_t^y\right\|_1 \leq t \cdot \frac{1}{2\kappa}\left\|g^y(z)\right\|_1 \leq \frac{16\left\|d\right\|_\infty \Theta}{\epsilon}.$$

A simple calculation yields $\Theta = 20\left\|d\right\|_\infty \log n + 4\left\|d\right\|_\infty$ upper bounds the range of r. Finally, let x_t^*, y_t^* be the minimizer of the proximal objective,

$$\langle s_t^x, x\rangle + \langle s_t^y, y\rangle + r(x, y).$$

For any initialization $x_{\text{init}}, y_{\text{init}}$ to the alternating minimization, the suboptimality gap is given by

$$\langle s_t^x, x_{\text{init}} - x_t^*\rangle + \langle s_t^y, y_{\text{init}} - y_t^*\rangle + r(x_{\text{init}}, y_{\text{init}}) - r(x_t^*, y_t^*)$$
$$\leq \left\|x_{\text{init}} - x_t^*\right\|_1 \left\|s_t^x\right\|_\infty + \left\|y_{\text{init}} - y_t^*\right\|_\infty \left\|s_t^y\right\|_1 + \Theta \leq \left(\frac{44\left\|d\right\|_\infty}{\epsilon} + 1\right)\Theta.$$

Therefore, the total number of iterations required is bounded by $24\log\left(\left(\frac{88\|d\|_\infty}{\epsilon^2} + \frac{2}{\epsilon}\right)\Theta\right)$ as desired. ☐

Numerical precision. We also make a brief comment on bit-complexity issues which may arise when scaling exponentials. In particular, each of our alternating minimization steps of the form

$$\langle g, x\rangle + \sum_j x_j \log x_j \tag{13}$$

require exponentiating a potentially large vector $\log x - g$, and rescaling the vector to be on the simplex. The following lemma shows that we can implement this step with $O(\log n)$ bit-complexity, with a small polynomial (say n^{-90}) loss in the objective value. Because we may assume that ϵ^{-1} is bounded by say, n^2, else an interior point method achieves our stated runtime, the cumulative loss in objective value over all iterations due to limited precision is significantly less than ϵ, and does not affect our asymptotic convergence rate. More precisely, we maintain x implicitly through a vector v which is $\log x$ up to a scaling, and show that by truncating v to have its range of coordinates bounded by $O(\log n)$, the resulting simplex variable remains a high-precision minimizer to (13).

Lemma B.6. *Let $v \in \mathbb{R}^m$, and let $x \in \Delta^m$ be such that $x \propto \exp(v)$. Consider the following operation: let $j^* = \text{argmin}_j v_j$, and j' be such that $v_{j'} < v_{j^*} - 100\log n$. Set $\hat{v} = v$ in every coordinate, except $\hat{v}_{j'} \leftarrow v_{j^*} - 100\log n$. Then, for $\hat{x} \propto \exp(\hat{v})$ in the simplex,*

$$\sum_j \hat{x}_j \log \hat{x}_j - \sum_j x_j \log x_j < n^{-95}, \langle g, \hat{x} - x\rangle < \left\|g\right\|_\infty n^{-95}.$$

Proof. Clearly, $\left\|\exp(v)\right\|_1 < \left\|\exp(\hat{v})\right\|_1$, since $\exp$ is monotone. Moreover,

$$\left\|\exp(\hat{v})\right\|_1 - \left\|\exp(v)\right\|_1 = \exp(v_{j^*} - 100\log n) - \exp(v_{j'}) < n^{-100}\exp(v_{j^*}) < n^{-100}\left\|\exp(v)\right\|_1.$$

Now, for every coordinate $j \neq j'$, this implies that $(1 - n^{-100})x_j < \hat{x}_j < x_j$. Thus,

$$\hat{x}_j \log \hat{x}_j - x_j \log x_j < -n^{-100} x_j \log x_j.$$

Furthermore, we have

$$\hat{x}_{j'} \log \hat{x}_{j'} - x_{j'} \log x_{j'} < -x_{j'} \log x_{j'} < 100 n^{-100},$$

by $-x \log x$ is increasing for small x and $x_{j'}$ is bounded by n^{-100}. Combining these estimates,

$$\sum_j \hat{x}_j \log \hat{x}_j - \sum_j x_j \log x_j < (100 + \log n) n^{-100} < n^{-95}$$

for $n > 10$. Similarly,

$$\langle g, \hat{x} - x \rangle \leq \|g\|_\infty \left(\hat{x}_{j'} + \sum_{j \neq j'} n^{-100} x_j \right) \leq \|g\|_\infty n^{-95}$$

for $n > 10$. ☐

Thus, repeatedly applying Lemma B.6 every time we need to truncate a coordinate of v due to finite bit precision, over the course of all iterations of the algorithm and all alternating minimization steps, the error incurred is negligible compared to the desired accuracy ϵ (where we also note $\|g\|_\infty$ for all g we encounter is bounded by a small polynomial in n).

C Missing proofs from Section 4

In this section, we give the proof to Theorem 2.2.

Theorem 2.2 (Rounding guarantee, Lemma 7 in [AWR17])**.** *There is an algorithm which takes* $\tilde{x}$ *with* $\|A\tilde{x} - b\|_1 \leq \delta$ *and produces* $\hat{x}$ *in* $O(n^2)$ *time, with*

$$A\hat{x} = b, \|\tilde{x} - \hat{x}\|_1 \leq 2\delta.$$

Proof. The algorithm is Algorithm 2. We adopt the alternative view of $\tilde{x}$ as a $n \times n$ matrix $\tilde{X}$ in the simplex, and define operations $r(X) = X\mathbf{1}, c(X) = X^\top \mathbf{1}$, recalling the first and last n entries of b are r, c, i.e. the row and column constraints. Recall we assume we have

$$\left\| r(\tilde{X}) - r \right\|_1 + \left\| c(\tilde{X}) - c \right\|_1 \leq \delta.$$

Clearly all operations in Algorithm 2 take $O(n^2)$ time. To explain briefly, X' is fixed so that its row sums are feasible (i.e. $X'\mathbf{1} \leq r$) and X'' is fixed so that its column sums are feasible. Further, entrywise $X'' \leq X' \leq \tilde{X}$, so X'' is feasible. We first bound

$$d \stackrel{\text{def}}{=} \left\| X'' - \tilde{X} \right\|_1 = \left(\sum_{i : r_i(\tilde{X}) > r_i} r_i(\tilde{X}) - r_i \right) + \left(\sum_{j : c_j(X') > c_j} c_j(X') - c_j \right).$$

Note $\left\| r(\tilde{X}) - r \right\|_1 \geq \sum_{i : r_i(\tilde{X}) > r_i} r_i(\tilde{X}) - r_i$. Further, by $X' \leq \tilde{X}$ entrywise,

$$\sum_{j : c_j(X') > c_j} c_j(X') - c_j \leq \left\| c(\tilde{X}) - c \right\|_1.$$

Thus $d \leq \delta$. $\hat{X} \in \mathcal{U}_{r,c}$, since $e_r, e_c \geq 0$ and $\mathbf{1}^\top e_r = \mathbf{1}^\top e_c = e$, so $\hat{X}\mathbf{1} = r,\ \hat{X}^\top \mathbf{1} = c$. Also,

$$\left\| \hat{X} - \tilde{X} \right\|_1 \leq \left\| X'' - \tilde{X} \right\|_1 + \left\| \hat{X} - X'' \right\|_1 \leq \delta + e.$$

Finally,

$$e = 1 - \mathbf{1}^\top X'' \mathbf{1} = 1 - \left(\mathbf{1}^\top \tilde{X} \mathbf{1} - d \right) = d.$$

Thus using $d \leq \delta$ proves the claim. ☐

D Experiment details

Here, we give the implementation details for the experimental results discussed in Section 5, and a brief justification of experimental decisions we made.

Dataset. For the first two figures in Section 5, we had the following experimental setup. We randomly sampled a pair of digits from the MNIST dataset corresponding to the digit 1, and added a small amount of background noise for numerical stability, as is standard in the literature [AWR17]. We downsampled the 28×28 pixel images to size 14×14 by skipping every other pixel to speed up experiments. Similar performances were observed across multiple random instances. Finally, the cost metric used was by Manhattan distance on the 2-dimensional grid.

For the second pair of figures, we randomly sampled 20 pairs of digits from the MNIST dataset where each pair corresponds to the same digit. As before we added a small amount of background noise for stability. As opposed to the previous comparison we ran all three algorithms on the true 28×28 pixel images. For each of the digit pairs we ran the Sinkhorn algorithm to high precision to obtain a baseline solution for comparison, and for each of the three algorithms tested we compared the value of the solutions obtained to this baseline. The plots compare the number of matrix-vector multiplies to the objective value error averaged over all 20 digit pairs. The metric is again the Manhattan distance over the 2-dimensional grid.

Objective value. For simplicity, in all cases we measured objective value by the overestimate presented in (4). By the proof of Lemma 2.3, this is an overestimate to the true objective after performing the rounding procedure in Algorithm 2. In practice, we observed that this overestimate was negligibly different from the objective after rounding.

Sinkhorn implementation details. We implemented the standard Sinkhorn algorithm, using different settings of η^{-1}. Sinkhorn iteration converges to an ϵ-approximate transportation plan in theory when η is very large, roughly $\log n/\epsilon$. However, in practice, it is observed that much smaller values of η suffice for rapid convergence. We tracked the convergence of Sinkhorn iteration for $\eta = 70$ and $\eta = 5$, which we considered close to a theoretically guaranteed parameter and a much less conservative practical parameter, respectively. The optimized Sinkhorn algorithm converged at rates much faster than the predicted ϵ^{-2} rate on all experiments, outperforming all other methods, which we believe merits further investigation. Significantly larger values of η led to numerical stability issues when computing $\exp(-\eta C)$.

APDAMD implementation details. We implemented the APDAMD algorithm (Algorithm 4 in [LHJ19]), with the quadratic regularizer (i.e. $\frac{1}{2\gamma} \|\lambda\|_2^2$). We observed that the amount of the quadratic regularizer added did not affect the practical convergence of the algorithm. A simple reason for this is because the algorithm builds in a more aggressive step-size strategy, because the pessimistic $\gamma = O(n)$ is often too conservative to be necessary in practice. The figure tracks APDAMD convergence with $\eta = 10^{-2}, \epsilon = 10^{-3}$.

Mirror prox. For numerical stability considerations, we implemented our algorithm as an instance of mirror prox [Nem04], another extragradient method which takes local iterations rather than accumulating a dual operator and taking steps with respect to some $\bar{z}$ (i.e. dual extrapolation). Although there is not a known proof of mirror prox convergence with an area-convex regularizer, we find this decision reasonable for several reasons. In general, variations of entropic mirror descent are well-known to be equivalent to their dual averaging versions; it is likely that a similar equivalence can be drawn between mirror prox and extragradient dual averaging, i.e. dual extrapolation. Furthermore, the standard proofs of dual extrapolation and mirror prox are quite similar; we believe it is likely that area-convexity results in convergence for mirror prox, although this merits further investigation.

Termination. We terminated our alternating minimization procedure when the movement of iterations in ℓ_1 was negligible. Typically, we observed that 3-5 alternating steps sufficed for convergence.

Step sizes. We varied two parameters in our experiments: the step size $\frac{1}{\kappa}$ used in our extragradient algorithm, and the amount of entropy used in our regularizer (in the paper, we used 10 times entropy compared to the quadratic component $x^\top A^\top (y^2)$). One reason this may be reasonable in practice is similar to the observed behavior of the Sinkhorn iteration tuning the η^{-1} parameter, and APDAMD performing a more-aggressive line search for the observed amount of regularizer necessary. We note that the need to tune the amount of entropy used in the regularizer is likely due to the analysis not being tight in the constants, and with a tighter analysis, it may not be necessary to tune this parameter. To this end, we plotted the performance of three algorithm settings.

- In the "unoptimized constants", we set the constants to roughly those with theoretical guarantees, i.e. 10 times entropy and step size 1.
- In the "reasonably optimized constants", we set the amount of entropy to be 4, and the step size to be $\|d\|_\infty/3$, to offset the $\|d\|_\infty$ multiple of the regularizer used in our iterations. For smaller values of ϵ, these settings compared favorably with APDAMD.
- In the "optimized constants", we set the amount of entropy at 3, and the step size at $\|d\|_\infty$. This setting outperformed APDAMD and was more competitive with Sinkhorn iteration.

Discussion. We believe multiple interesting avenues of exploration arise from our experiments.

- Sinkhorn with aggressively chosen η outperformed all other methods we benchmarked against, and converged at rates faster than suggested by its known analyses. It may prove fruitful to study if further assumptions about practical instances explain this discrepancy.
- Directly accelerated methods such as APDAMD also exhibit ϵ^{-1} convergence rates, at the cost of a worse dependence on dimension. However, this worst-case dependence can be mitigated if the instance is favorable in practice, i.e. by choosing $\gamma \approx O(1)$. This was observed to be the case in our experiments for the MNIST dataset. It is interesting to see if a similar adaptive tuning applies to our method with provable guarantees.
- Our method did not exhibit instability when changing the amount of entropy in the regularizer, but it did exhibit vastly-improved convergence. It is possible that the amount of regularizer needed is not quite so large, perhaps through a more careful analysis.
- We did not benchmark against the greedy Sinkhorn method of [AWR17], or consider numerical speedups such as those in [ABRW18]. It remains open to explore if these practical speedups are applicable to first-order methods such as ours as well.