A FAST AND ACCURATE SPLITTING METHOD FOR OPTIMAL TRANSPORT: ANALYSIS AND IMPLEMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We develop a fast and reliable method for solving large-scale optimal transport (OT) problems at an unprecedented combination of speed and accuracy. Built on the celebrated Douglas-Rachford splitting, our method tackles the original OT problem directly instead of solving approximate regularized problems, as many state-of-the-art techniques do. This allows us to provide sparse transport plans and avoid numerical issues of methods that use entropic regularization. The algorithm has the same cost per iteration as the popular Sinkhorn method, and each iteration can be executed efficiently, in parallel. The proposed method inherits the strong convergence guarantees of the Douglas-Rachford splitting family. In addition, we establish a linear convergence rate for our formulation of the OT problem. We discuss efficient implementation of the proposed method on GPUs, including how to incorporate stopping criteria without any extra cost. Substantial experiments demonstrate a superior performance of our method, both in terms of computation times and robustness, compared to the state-of-the art.

1 INTRODUCTION

We study the discrete optimal transport (OT) problem:

$$\begin{array}{ll} \underset{X \geq 0}{\text{minimize}} & \langle C, X \rangle \\ \text{subject to} & X\mathbf{1}_n = p \\ & X^\top \mathbf{1}_m = q, \end{array} \tag{1}$$

where $X \in \mathbb{R}_+^{m \times n}$ is the transport plan, $C \in \mathbb{R}_+^{m \times n}$ is the cost matrix, and $p \in \mathbb{R}_+^m$ and $q \in \mathbb{R}_+^n$ are two discrete probability measures. OT has a very rich history in mathematics and operations research dated back to at least the 18th century. By exploiting geometric properties of the underlying ground space, OT provides a powerful and flexible way to compare probability measures. It has quickly become a central topic in machine learning and has found applications in countless learning tasks such as deep generative models Arjovsky et al. (2017), domain adaptation Courty et al. (2016), inference of high-dimensional cell trajectories in genomics Schiebinger et al. (2019); we refer to Peyré & Cuturi (2019) for a more comprehensive survey and applications of OT. However, the power of OT comes at the price of an enormous computational cost for computing the optimal transport plan. Standard methods for solving linear programs (LPs) suffer from the super-linear time complexity in term of the problem size. Such methods are also significantly more challenging to parallelize on modern processing hardware. Therefore, there has been substantial research in developing new efficient methods for OT.

1.1 RELATED WORK

Below we review some of the topics most closely related to our work.

Sinkhorn method The Sinkhorn (SK) method (Cuturi, 2013) aims to solve an approximation of (1) in which the objective is replaced by a regularized version of the form$\langle C, X \rangle - \eta H(X)$. Here, $H(X) = -\sum_{ij} X_{ij} \log(X_{ij})$ is an entropy function and $\eta > 0$ is a regularization parameter. The Sinkhorn method defines the quantity $K = \exp(-C/\eta)$ and repeats the following steps

$$u_k = p/(Kv_{k-1}) \quad \text{and} \quad v_k = q/(K^\top u_k),$$

until $\|u_k \odot (Kv_k) - p\| + \|v_k \odot (K^\top u_k) - q\|$ becomes small, then returns $\operatorname{diag}(u_k)K \operatorname{diag}(v_k)$. The division and multiplication operators between two vectors are to be understood entrywise. Each SK iteration is built from matrix-vector multiplies and element-wise arithmetic operations, and is hence readily parallelized on multi-core CPUs and GPUs. However, due to the entropic regularization, SK suffers heavily from numerical issues and can struggle to find even moderately accurate solutions. The problem is even more prevalent for GPU implementations as most modern GPUs are built and optimized for single-precision arithmetic. Substantial care is therefore needed to select an appropriate η that is small enough to provide a meaningful approximation, while avoiding numerical issues. Moreover, the entropy term enforces a dense solution, which can be undesirable when the optimal transportation plan itself is of interest (Blondel et al., 2018). We mention that there has been a substantial research in improving the performance of SK (Alaya et al., 2019; Altschuler et al., 2017; Blondel et al., 2018; Lin et al., 2019). Most of them aim to improve certain aspects of SK. Some modifications result in more stable but much slower methods, while others allow to produce sparse solutions but at a much higher cost per iteration than SK. Also, some sophisticated changes make parallelization challenging due to the many branching conditions that they introduce.

Operator splitting solvers for general LP With relatively low per-iteration cost and the ability to exploit sparsity in the problem data, operator splitting methods such as Douglas-Rachford splitting (Douglas & Rachford, 1956) and ADMM (Gabay & Mercier, 1976), are particularly suitable for large-scale optimization. Such algorithms can quickly produce solutions of moderate accuracy and have been the engine of several successful first-order solvers (O'donoghue et al., 2016; Stellato et al., 2020; Garstka et al., 2021). As OT can be cast as an LP, it can be solved by these splitting-based solvers. However, there has not been much success reported in this context, probably due to the *memory-bound* nature of large-scale OTs. For OTs, the main bottleneck is not floating point computations, but rather time-consuming memory operations on large two-dimensional arrays. Even an innocent update like $X \leftarrow X - C$ is more expensive than two matrix-vector multiplies in SK. To design a high-performance splitting method, it is thus crucial to minimize the memory operations associated with large arrays. In addition, since these solvers target general LPs, they often solve a linear system at each iteration, which is prohibitively expensive for many OT applications.

Convergence rates of DR for LP Many splitting methods, including Douglas-Rachford, are known to converge linearly under strong convexity (see e.g. Giselsson & Boyd (2016)). Recently, it has been shown that algorithms based on DR/ADMM based often enjoy similar convergence properties also in the absence of strong convexity. For example, Liang et al. (2017), derived local linear convergence guarantees under mild assumptions; Applegate et al. (2021) established global linear convergence for Primal-Dual methods for LPs by using restart strategies; and, Wang & Shroff (2017) established linear convergence for an ADMM-based algorithm on general LPs. Yet, these frameworks quickly become intractable on large OT problems, due to the many memory operations required in each iteration.

1.2 Contributions

We demonstrate that DR splitting, when properly applied and implemented, can solve large-scale OT problems to modest accuracy reliably and quickly, while retaining the excellent memory footprint and parallelization of the Sinkhorn method. Concretely, we make the following contributions:

- We develop a DR splitting algorithm that solves the original OT directly, avoiding any numerical issues of SK and forgoing the need for solving linear systems of general solvers. We perform simplifications to eliminate variables so that the final algorithm can be executed with only one matrix variable, while maintaining the same degree of parallelization. Our method implicitly maintains a primal-dual pair, which facilitates a simple evaluation of stopping criteria.
- The algorithm inherits the strong convergence guarantees of the DR splitting family, including the ADMM method and more broadly proximal point methods(Rockafellar, 1976). We derive a sublinear rate $O(1/k)$ and a global linear rate that holds independently of the initialization. Moreover, we compare the bounds and discuss when the sublinear rate dominates the linear rate.
- We provide an efficient GPU implementation that fuses many immediate steps into one and performs several on-the-fly reductions between a read and write of the matrix variable. We discuss

how stopping criteria can be incorporated at no extra cost. This gives practitioners a fast and robust OT solver also for applications where regularized OT is not suitable.

As a by-product of solving the original OT problem, our approximate solution is guaranteed to be sparse. Indeed, it is known that DR can identify an optimal support in a finite time, even before reaching a solution (see, e.g., Iutzeler & Malick (2020)). To avoid cluttering the presentation, we focus on the primal OT, but note that the approach applies equally well to the dual form. Moreover, our implementation is readily extended to other splitting schemes (e.g. Chambolle & Pock (2011)).

2 Background

Notation For any $x, y \in \mathbb{R}^n$, we denote by $\langle x, y\rangle$ the Euclidean inner product of x and y; the ℓ_2-norm is denoted by $\|\cdot\|_2$. For matrices $X, Y \in \mathbb{R}^{m\times n}$, $\langle X, Y\rangle = \operatorname{tr}(X^\top Y)$ denotes their inner product; the induced Frobenius norm is then defined as $\|\cdot\|_{\mathrm{F}} = \sqrt{\langle\cdot,\cdot\rangle}$. We shall use $\|\cdot\|$ to denote either the ℓ_2-norm or the Frobenius norm. For a closed and convex set $\mathcal{X}$, the distance and the projection map are given respectively by: $\operatorname{dist}(x, \mathcal{X}) = \min_{z\in\mathcal{X}} \|z - x\|$ and $\mathsf{P}_{\mathcal{X}}(x) = \operatorname{argmin}_{z\in\mathcal{X}} \|z - x\|$. The Euclidean projection of $x \in \mathbb{R}^n$ onto the nonnegative orthant is denoted by $[x]_+ = \max(x, 0)$. The $(n-1)$-dimensional probability simplex is denoted by $\Delta_n = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_i = 1\}$.

OT and optimality conditions Let $e = \mathbf{1}_n$ and $f = \mathbf{1}_m$, and consider the linear mapping $\mathcal{A}$:

$$\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m+n} : X \mapsto (Xe, X^\top f),$$

and its adjoint $\mathcal{A}^* : \mathbb{R}^{m+n} \to \mathbb{R}^{m\times n} : (y, x) \mapsto ye^\top + fx^\top$. The projection onto the range of $\mathcal{A}$ is computed as Bauschke et al. (2021): $\mathsf{P}_{\operatorname{ran}A}((y, x)) = (y, x) - \alpha(f, -e)$, where $\alpha = (f^\top y - e^\top x)/(m + n)$. Problem (1) can be written as a linear program on the form

$$\underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} \;\; \langle C, X\rangle \quad \text{subject to} \;\; \mathcal{A}(X) = b, X \geq 0, \tag{2}$$

where $b = (p, q) \in \mathbb{R}^{m+n}$. Let $(\mu, \nu) \in \mathbb{R}^{m+n}$ be the dual variable associated the affine constraint in problem (2). Then, using the definition of $\mathcal{A}^*$, the dual problem of (2), or equivalently (1), reads

$$\underset{\mu,\nu}{\text{maximize}} \;\; p^\top\mu + q^\top\nu \quad \text{subject to} \;\; \mu e^\top + f\nu^\top \leq C. \tag{3}$$

OT is a bounded program and always admits a feasible solution. It is thus guaranteed to have an optimal solution, and the optimality conditions can be summarized as follows.

Proposition 1. *A pair X and (μ, ν) are primal and dual optimal if and only if: (i) $Xe = p, X^\top f = q, X \geq 0$, (ii) $\mu e^\top + f\nu^\top - C \leq 0$, (iii) $\langle X, C - \mu e^\top - f^\top\nu\rangle = 0$.*

These conditions mean that: (i) X is primal feasible; (ii) μ, ν are dual feasible, and (iii) the duality gap is zero $\langle C, X\rangle = p^\top\mu + q^\top\nu$, or equivalently, the complementary slackness condition. Thus solving the OT problem amounts to (approximately) finding such a primal-dual pair. To this end, we rely on the celebrated Douglas-Rachford splitting method Lions & Mercier (1979); Eckstein & Bertsekas (1992); Fukushima (1996); Bauschke & Combettes (2017).

Douglas-Rachford splitting Consider composite convex optimization problems on the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x) + g(x), \tag{4}$$

where f and g are proper closed and convex functions. To solve Problem (4), the Douglas-Rachford splitting algorithm starts from $y_0 \in \mathbb{R}^n$ and repeats for $k = 0, 1, \ldots,$ the following steps:

$$x_{k+1} = \operatorname{prox}_{\rho f}(y_k), \quad z_{k+1} = \operatorname{prox}_{\rho g}(2x_{k+1} - y_k), \quad y_{k+1} = y_k + z_{k+1} - x_{k+1}. \tag{5}$$

where $\rho > 0$ is the penalty parameter. The preceding procedure can be viewed as a fixed-point iteration $y_{k+1} = T(y_k)$, where $T(y) = y + \operatorname{prox}_{\rho g}\left(2\operatorname{prox}_{\rho f}(y) - y\right) - \operatorname{prox}_{\rho f}(y)$. It can also be derived from the ADMM method applied to an equivalent problem to (4) (see, Appendix C). Indeed, they are both special instances of the classical proximal point method in Rockafellar (1976). As for convergence, Lions & Mercier (1979) showed that T is a *firmly-nonexpansive* mapping, from which they obtained convergence of y_k. Moreover, the sequence x_k is guaranteed to converge to a minimizer of $f + g$ (assuming a minimizer exists) for any choice of $\rho > 0$. In particular, we have the following general convergence result, whose proof can be found in (Bauschke & Combettes, 2017, Corollary 28.3).

Lemma 2.1. *Consider the composite problem* (4) *and its Fenchel–Rockafellar dual defined as*

$$\underset{u\in\mathbb{R}^n}{\text{maximize}} \quad -f^*(u) - g^*(-u). \tag{6}$$

Let $\mathcal{P}^\star$ and $\mathcal{D}^\star$ be the sets of solutions to the primal (4) *and dual* (6) *problems, respectively. Let x_k, y_k, and z_k be generated by procedure* (5) *and let $u_k := (y_{k-1} - x_k)/\rho$. Then, there exists $y^\star \in \mathbb{R}^n$ such that $y_k \to y^\star$. Setting $x^\star = \operatorname{prox}_{\rho f}(y^\star)$ and $u^\star = (y^\star - x^\star)/\rho$, then it holds that (i) $x^\star \in \mathcal{P}^\star$ and $u^\star \in \mathcal{D}^\star$; (ii) $x_k - z_k \to 0$, $x_k \to x^\star$ and $z_k \to x^\star$; (iii) $u_k \to u^\star$.*

3 Douglas-Rachford splitting for optimal transport

To efficiently apply DR to OT, we need to specify the functions f and g as well as how to evaluate their proximal operations. We begin by introducing a very recent result in computing the projection onto the set of *real-valued* matrices with prescribed row and column sums Bauschke et al. (2021).

Lemma 3.1. *Let $e = \mathbf{1}_n$ and $f = \mathbf{1}_m$. Let $p \in \Delta_m$ and $q \in \Delta_n$. Then, the set $\mathcal{X}$ defined by:*

$$\mathcal{X} := \left\{ X \in \mathbb{R}^{m\times n} \,\middle|\, Xe = p \text{ and } X^\top f = q \right\}$$

is non-empty, and for every given $X \in \mathbb{R}^{m\times n}$, we have

$$\mathsf{P}_{\mathcal{X}}(X) = X - \frac{1}{n}\left((Xe - p)\, e^\top - \gamma f e^\top\right) - \frac{1}{m}\left(f(X^\top f - q)^\top - \gamma f e^\top\right),$$

where $\gamma = f^\top (Xe - p)/(m+n) = e^\top \left(X^\top f - q\right)/(m+n)$.

The result in the lemma follows immediately from (Bauschke et al., 2021, Corollary 3.4) and fact that $(p, q) = \mathsf{P}_{\operatorname{ran}\mathcal{A}}((p,\, q))$. The lemma implies that $\mathsf{P}_{\mathcal{X}}(X)$ can be carried out by basic linear algebra operations, like matrix-vector multiplies and rank-one updates, that can be effectively parallelized.

3.1 Algorithm derivation

Our algorithm is based on re-writing (2) on the standard form for DR-splitting (4), and choosing f and g so that the resulting iterations can be executed efficiently and the iterates converge rapidly. In particular, we propose to select f and g as follows:

$$f(X) = \langle C, X\rangle + \mathsf{I}_{\mathbb{R}_+^{m\times n}}(X) \quad \text{and} \quad g(X) = \mathsf{I}_{\{Y:\mathcal{A}(Y)=b\}}(X). \tag{7}$$

By Lemma 3.1, we readily have the explicit formula for the proximal operator of g, namely, $\operatorname{prox}_{\rho g}(\cdot) = \mathsf{P}_{\mathcal{X}}(\cdot)$. The proximal operator of f can also be evaluated explicitly as:

$$\operatorname{prox}_{\rho f}(X) = \mathsf{P}_{\mathbb{R}_+^{m\times n}}(X - \rho C) = [X - \rho C]_+.$$

The Douglas-Rachford splitting applied to the optimal transport problem then reads:

$$X_{k+1} = [Y_k - \rho C]_+, \quad Z_{k+1} = \mathsf{P}_{\mathcal{X}}(2X_k - Y_k), \quad Y_{k+1} = Y_k + Z_{k+1} - X_{k+1}. \tag{8}$$

Despite their simplicity, the updates in (8) are inefficient to execute in practice due to the many *memory operations* needed to operate on the large arrays X_k, Y_k, Z_k and C. To avoid this issue, we will now perform several simplifications to eliminate variables from (8). The resulting algorithm can be executed with only one matrix variable while maintaining the same degree of parallelization.

First, thanks to the linearity of $\mathsf{P}_{\mathcal{X}}(\cdot)$, the Z-variable can be easily eliminated, yielding the Y-update:

$$Y_{k+1} = X_{k+1} - n^{-1}\big(2X_{k+1}e - Y_k e - p - \gamma_k f\big)e^\top - m^{-1} f\big(2X_{k+1}^\top f - Y_k^\top f - q - \gamma_k e\big)^\top,$$

where $\gamma_k = f^\top (2X_{k+1}e - Y_k e - p)/(m+n) = e^\top \left(2X_{k+1}^\top f - Y_k^\top f - q\right)/(m+n)$. Next, we define the following quantities for Y_k: $a_k = Y_k e - p$, $b_k = Y_k^\top f - q$, and $\alpha_k = f^\top a_k/(m+n) = e^\top b_k/(m+n)$. Similarly, for X_k, we let $r_k = X_k e - p$, $s_k = X_k^\top f - q$, and $\beta_k = f^\top r_k/(m+n) = e^\top s_k/(m+n)$. Then, the preceding update can be written compactly as

$$Y_{k+1} = X_{k+1} + \phi_{k+1} e^\top + f\varphi_{k+1}^\top, \tag{9}$$

Algorithm 1 Douglas-Rachford Splitting for Optimal Transport (DROT)

Input: OT(C, p, q), initial point X_0, penalty parameter ρ
1: $\phi_0 = 0, \varphi_0 = 0$
2: $a_0 = X_0 e - p$, $b_0 = X_0^\top f - q$, $\alpha_0 = f^\top a_0/(m+n)$
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: $\quad X_{k+1} = \left[X_k + \phi_k e^\top + f\varphi_k^\top - \rho C\right]_+$
5: $\quad r_{k+1} = X_{k+1}e - p$, $s_{k+1} = X_{k+1}^\top f - q$, $\beta_{k+1} = f^\top r_{k+1}/(m+n)$
6: $\quad \phi_{k+1} = \left(a_k - 2r_{k+1} + (2\beta_{k+1} - \alpha_k)f\right)/n$
7: $\quad \varphi_{k+1} = \left(b_k - 2s_{k+1} + (2\beta_{k+1} - \alpha_k)e\right)/m$
8: $\quad a_{k+1} = a_k - r_{k+1}$, $b_{k+1} = b_k - s_{k+1}$, $\alpha_{k+1} = \alpha_k - \beta_{k+1}$
9: **end for**
Output: X_K

where

$$\phi_{k+1} = \left(a_k - 2r_{k+1} + (2\beta_{k+1} - \alpha_k)f\right)/n \quad \text{and} \quad \varphi_{k+1} = \left(b_k - 2s_{k+1} + (2\beta_{k+1} - \alpha_k)e\right)/m.$$

We can see that the Y-update can be implemented using 4 matrix-vector multiples (for computing $a_k, b_k, r_{k+1}, s_{k+1}$), followed by 2 rank-one updates. As a rank-one update requires a read from an input matrix and a write to an output matrix, it is typically twice as costly as a matrix-vector multiply (which only writes the output to a vector). Thus, it would involve 8 memory operations of large arrays. The current form is also not compatible with standard rank-one update subroutines like BLAS's `ger` $A \leftarrow A + xy^\top$[1] that requires the same array A as input and output.

Next, we show that the Y-update can be too removed. Noticing that updating Y_{k+1} from Y_k and X_{k+1} does not need the *full* matrix Y_k but only the ability to multiply it with some vectors. This allows us to use a single *physical* memory array to represent both the sequences X_k and Y_k. Suppose that we overwrite the matrix X_{k+1} as:

$$X_{k+1} \leftarrow X_{k+1} + \phi_{k+1}e^\top + f\varphi_{k+1}^\top,$$

then after the two rank-one updates, the X-array holds the value of Y_{k+1}. We can access to the actual X-value again in the next update, which now reads: $X_{k+2} \leftarrow \left[X_{k+1} - \rho C\right]_+$. It thus remains to show that the two matrix-vector multiplies associated with Y_k can be computed efficiently. By multyplying both sides of (9) by e and subtracting the result by p, we obtain

$$Y_{k+1}e - p = X_{k+1}e - p + \phi_{k+1}e^\top e + (\varphi_{k+1}^\top e)f$$

Since $e^\top e = n$ and $(b_k - 2s_{k+1})^\top e = (m+n)(\alpha_k - 2\beta_{k+1})$, it holds that $(\varphi_{k+1}^\top e)f = (\alpha_k - 2\beta_{k+1})f$. We also have $\phi_{k+1}e^\top e = a_k - 2r_{k+1} + (2\beta_{k+1} - \alpha_k)f$. Therefore, we end up with an extremely simple recursive form for updating a_k:

$$a_{k+1} = a_k - r_{k+1}.$$

Similarly, we have $b_{k+1} = b_k - s_{k+1}$ and $\alpha_{k+1} = \alpha_k - \beta_{k+1}$. In summary, the DROT method can be implemented with a single matrix variable and the full algorithm is summarized in Algorithm 1.

Stopping criteria It is interesting to note that while DROT directly solves to the primal problem (1), it maintains a pair of *vectors* that *implicitly* plays the role of the dual variables μ and ν in the dual problem (3). To get a feel for this, we note that the optimality conditions in Proposition 1 are equivalent to the existence of a pair $X^\star$ and $(\mu^\star, \nu^\star)$ such that

$$(X^\star e, X^{\star\top} f) = (p, q) \quad \text{and} \quad X^\star = \left[X^\star + \mu^\star e^\top + f\nu^{\star\top} - C\right]_+.$$

This is a direct consequence of a well-known result in orthogonal projection onto convex cones (Deutsch & Deutsch, 2001, Theorem 5.6). Now, compared to Step 4 in DROT, the second condition above suggests that ϕ_k and φ_k are such implicit variables. To see why this is indeed the case, let $U_{k+1} = (Y_k - X_{k+1})/\rho$. Then it is easy to verify that

$$(Z_{k+1} - X_{k+1})/\rho = (X_{k+1} - Y_k + \phi_k e^\top + f\varphi_k^\top - \rho C)/\rho = -U_{k+1} + (\phi_k/\rho)e^\top + f(\varphi_k/\rho)^\top.$$

[1] http://www.netlib.org/blas

By Lemma 2.1, we have $Z_{k+1} - X_{k+1} \to 0$ and $U_{k+1} \to U^\star \in \mathbb{R}^{m\times n}$, by which it follows that

$$(\phi_k/\rho)e^\top + f(\varphi_k/\rho)^\top \to U^\star.$$

Finally, by evaluating the conjugate functions f^* and g^* in Lemma 2.1, it can be shown that $U^\star$ must have the form $\mu^\star e^\top + f\nu^{\star\top}$, where $\mu^\star \in \mathbb{R}^m$ and $\nu^\star \in \mathbb{R}^n$ satisfying $\mu^\star e^\top + f\nu^{\top\star} \leq C$. With such primal-dual pairs at our disposal, we can now explicitly evaluate their distance to the set of solutions laid out in Proposition 1 by considering: $r_{\text{primal}} = (\|r_k\|^2 + \|s_k\|^2)^{1/2}$, $r_{\text{dual}} = \|[(\phi_k/\rho)e^\top + f(\varphi_k/\rho)^\top - C]_+\|$, and $\text{gap} = \left|\langle C, X_k\rangle - (p^\top\phi_k + \varphi_k^\top q)/\rho\right|$. As problem (1) is feasible and bounded, Lemma 2.1 and strong duality guarantees that all the three terms will converge to zero. Thus, we terminate DROT when these become smaller than some user-specified tolerances.

3.2 Convergence rates

Since DR is equivalent to ADMM up to changes of variables and swapping of the iteration order, it is expected that DR can attain the ergodic rate $O(1/t)$ of ADMM, both for the objective gap and the constraint violation (He & Yuan, 2012; Beck, 2017). However, mapping the convergence proof of ADMM to a specific instance of DR is tedious and error prone. We thus give a direct proof for completeness. The proof relies on the following Lyapunov function

$$V(X, Z, U) = f(X) + g(Z) + \frac{1}{\rho}\langle U, Z - X\rangle,$$

which is defined for $X \in \mathbb{R}_+^{m\times n}$, $Z \in \mathcal{X}$ and $U \in \mathbb{R}^{m\times n}$. We can now state the first result.

Theorem 1 (Sublinear rate)**.** *Let X_k, Y_k, Z_k be the generated by procedure* (8) *and let $U_{k+1} = Y_k - X_{k+1}$. Let $X^\star$ be a solution of* (1) *defined from $Y^\star$ in the manner of Lemma 2.1. Then*

$$V(X_{k+1}, Z_{k+1}, Y^\star - X^\star) - V(X^\star, X^\star, U_{k+1}) \leq \frac{1}{2\rho}\|Y_k - Y^\star\|^2 - \frac{1}{2\rho}\|Y_{k+1} - Y^\star\|^2.$$

Furthermore, if we let $\bar{X}_k = \sum_{i=1}^k X_k/i$ and $\bar{Z}_k = \sum_{i=1}^k Z_k/i$, it holds that

$$\left\langle C, \bar{X}_k\right\rangle - \langle C, X^\star\rangle \leq O(\|Y_0 - Y^\star\|^2/k) \quad \textit{and} \quad \frac{1}{2\rho}\left\|\bar{X}_k - \bar{Z}_k\right\|^2 \leq O(\|Y_0 - Y^\star\|^2/k).$$

The theorem implies that one can compute an ϵ-optimal solution satisfying $\langle C, X\rangle - \langle C, X^\star\rangle \leq \epsilon$ in $O(1/\epsilon)$ iterations. The best known iteration complexity for the Sinkhorn method is $O(1/\epsilon^2)$ (Lin et al., 2019). Note that the linearity of $\langle C, \cdot\rangle$ allows to update the *scalar* value $\left\langle C, \bar{X}_k\right\rangle$ recursively without ever needing to form the ergodic sequence $\bar{X}_k$. Yet, it turns out that this complexity is over-conservative, which follows from the second result:

Theorem 2 (Linear rate)**.** *Let X_k and Y_k be generated by* (8)*, $\mathcal{G}^* = \{Y \in \mathbb{R}^{m\times n} | \, T(Y) = Y\}$ and $\mathcal{S}^*$ be the set of optimal solutions to* (1)*. Then*

$$\text{dist}(Y_k, \mathcal{G}^*) \leq O(\|Y_0 - Y^\star\| r^k),$$
$$\text{dist}(X_k, \mathcal{S}^*) \leq O(\|Y_0 - Y^\star\| r^k),$$

where $r < 1$ and $Y^* \in \mathcal{G}^*$.

This means that an ϵ-optimal solution can be computed in $O(\log \epsilon)$ iterations. However, it is, in general, difficult to estimate the convergence rate r, and it may in the worst case be close to one. In such settings, the sublinear rate will typically dominate for the first iterations.

3.3 Implementation

In this section, we detail our implementation of DROT to exploit parallel processing on GPUs. We review only the most basic concepts of GPU programming necessary to describe our kernel and refer to (Kirk & Wen-Mei, 2016; Cheng et al., 2014) for comprehensive treatments.

Thread hierarchy When a kernel function is launched, a large number of threads are generated to execute its statements. These threads are organized into a two-level hierarchy. A *grid* contains multiple blocks and a *block* contains multiple threads. Each block is scheduled to one of the streaming

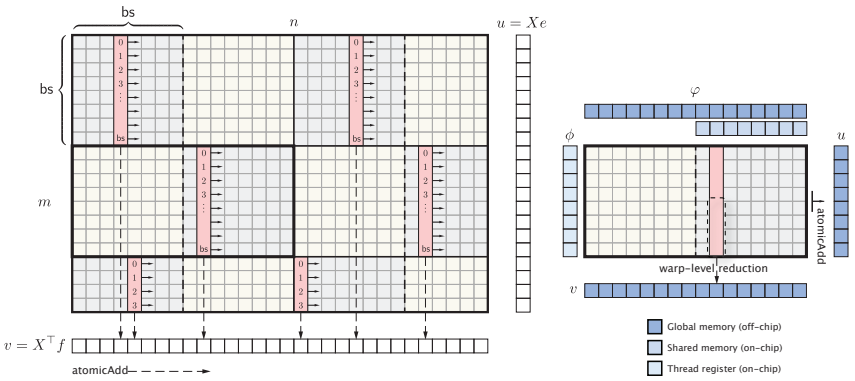


Figure 1: Left: Logical view of the main kernel. To expose sufficient parallelism to GPU, we organize threads into a 2D grid of blocks (3×2 in the figure), which allows several threads per row. The threads are then grouped in 1D blocks (shown in red) along the columns of X. This ensures that global memory access is aligned and *coalesced* to maximize bandwidth utilization. We use the parameter work-size ws to indicate how many elements of a row each thread should handle. For simplicity, this parameter represents multiples of the block size bs. Each arrow denotes the activity of a single thread in a thread block. Memory storage is assumed to be column-major. Right: Activity of a normal working block, which handles a submatrix of size $\mathsf{bs} \times (\mathsf{ws} \cdot \mathsf{bs})$.

multiprocessors (SMs) on the GPU concurrently or sequentially, depending on available hardware. While all threads in a thread block run *logically* in parallel, not all of them can run *physically* at the same time. As a result, different threads in a thread block may progress at a different pace. Once a thread block is scheduled to an SM, its threads are further partitioned into *warps*. A warp consists of 32 consecutive threads that execute the same instruction at *the same time*. Each thread has its own instruction address counter and register state, and carries out the current instruction on its own data.

Memory hierarchy *Registers* and *shared memory* (on-chip) are the fastest memory spaces on a GPU. Registers are private to each thread, while shared memory is visible to all threads in the same thread block. An automatic variable declared in a kernel is generally stored in a register. Shared memory is programmable, and users have full control over when data is moved into or evicted from the shared memory. It enables block-level communication, facilitates reuse of on-chip data, and can greatly reduce the global memory access of a kernel. However, there are typically only a couple dozen registers per thread and a couple of kBs shared memory per thread block.

The largest memory on a GPU card is *global memory*, located outside the compute chip. All threads can access global memory, but its latency is much higher, typically hundreds of clock cycles.[2] Therefore, minimizing global memory transactions is vital to a high-performance kernel. When all threads of a *warp* execute a load (store) instruction that access *consecutive* memory locations, these will be *coalesced* into as few transactions as possible. For example, if they access consecutive 4-byte words such as *float32* values, four coalesced 32-byte transactions will service that memory access.

Before proceeding further, we state the main result of the section.

Claim 3.1. *On average, an iteration of DROT, including all the stopping criteria, can be done using 2.5 memory operations of $m \times n$ arrays. In particular, this includes one read from and write to X, and one read from C in every other iteration.*

Main kernel Steps 4–5 and the stopping criteria are the main computational components of DROT, since they involve matrix operations. We will design an efficient kernel that: (i) updates X_k to X_{k+1}, (ii) computes $u_{k+1} := X_{k+1}e$ and $v_{k+1} = X_{k+1}^\top f$, (iii) evaluates $\langle C, X_{k+1} \rangle$ in the duality gap expression. The kernel fuses many immediate steps into one and performs several on-the-fly reductions while updating X_k, thereby ensuring minimal access to the global memory.

Our kernel is designed so that each thread block handles a *submatrix* x of size $\mathsf{bs} \times (\mathsf{ws} \cdot \mathsf{bs})$, except for the corner blocks which will have fewer rows and/or columns. Since all threads in a block need the same values of φ, it is best to read these into shared memory once per block and then let threads access them from there. We therefore divide φ into chunks of the block size and set up a loop to let the threads collaborate in reading chunks in a coalesced fashion into shared memory. Since each thread works on a single row, it accesses the same element of ϕ throughout, and we can thus load and store that value directly in a register. These allow maximum reuse of φ and ϕ.

[2] https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

In the j-th step, the working block loads column j of x to the chip in a coalesced fashion. Each thread $i \in \{0, 1, \ldots, \mathsf{bs}-1\}$ uses the loaded value x_{ij} to compute and store x_{ij}^+ in a register:

$$x_{ij}^+ = \max(x_{ij} + \phi_i + \varphi_j - \rho c_{ij}, 0),$$

where c is the corresponding submatrix of C. As sum reduction is order-independent, it can be done locally at various levels, and local results can then be combined to produce the final value. We therefore reuse x_{ij}^+ and perform several such partial reductions. First, to compute the local value for u_{k+1}, at column j, thread i simply adds x_{ij}^+ to a running sum kept in a register. The reduction leading to $\langle C, X_{k+1}\rangle$ can be done in the same way. The vertical reduction to compute v_{k+1} is more challenging as coordination between threads is needed. We rely on *warp-level* reduction in which the data exchange is performed between registers. This way, we can also leverage efficient CUDA's built-in functions for collective communication at warp-level.[3] When done, the first thread in a warp holds the total reduced value and simply adds it atomically to a proper coordinate of v_{k+1}. Finally, all threads write x_{ij}^+ to the global memory. The process is repeated by moving to the next column.

When the block finishes processing the last column of the submatrix x, each thread adds its running sum atomically to a proper coordinate of v_{k+1}. They then perform one vertical (warp-level) reduction to collect their private sums to obtain the partial cost value $\langle c, x\rangle$. When all the submatrices have been processed, we are granted all the quantities in Items (i)–(iii), as desired. It is essential to notice that each entry of x is only read and written once during this process. To conclude Claim 3.1, we note that one can skip the load of C in every other iteration. Indeed, if in iteration k, instead of writing X_{k+1}, one writes the value of $X_{k+1} - \rho C$, then the next update is simply $X_{k+2} = \left[X_{k+1} + \phi_{k+1}e^\top + f\varphi_{k+1}^\top\right]_+$. Finally, all the remaining updates in DROT only involve vector and scalar operations, and can thus be finished off with a simple auxiliary kernel.

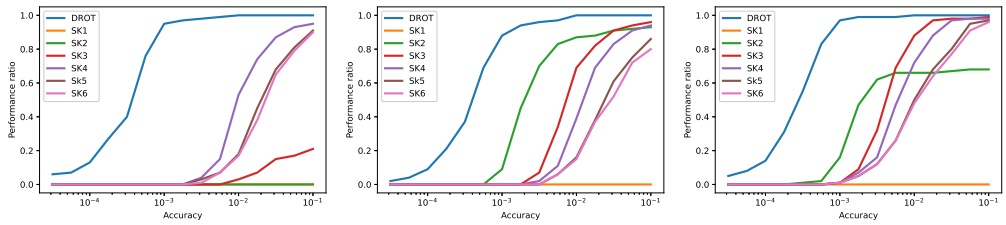
4 Experimental results

In this section, we perform experiments to validate our method and to demonstrate its efficiency both in terms of accuracy and speed. We focus on the Sinkhorn method, as implemented in the POT toolbox[4], due to its minimal cost per step and its GPU implementation is readily available. All runtime tests were carried out on an NVIDIA Tesla T4 GPU with 16GB of global memory.

We consider six SKs (SK1-SK6) corresponding to $\eta = 10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}$, in that order. Given m and n, we generate source and target samples as x_s and x_t, whose entries are drawn from a 2D Gaussian distribution with randomized means and covariances $(\mu_\mathrm{s}, \Sigma_\mathrm{s})$ and $(\mu_\mathrm{t}, \Sigma_\mathrm{t})$. Here, $\mu_\mathrm{s} \in \mathbb{R}^2$ has normal distributed entries, and $\Sigma_\mathrm{s} = A_\mathrm{s}A_\mathrm{s}^\top$, where $A_\mathrm{s} \in \mathbb{R}^{2\times2}$ is a matrix with random entries in $[0, 1]$; $\mu_\mathrm{t} \in \mathbb{R}^2$ has entries generated from $\mathcal{N}(5, \sigma_\mathrm{t})$ for some $\sigma_\mathrm{t} > 0$, and $\Sigma_\mathrm{t} \in \mathbb{R}^{2\times2}$ is generated similarly to Σ_s. Given x_s and x_t, the cost matrix C represents pairwise squared Euclidean distance between samples: $C_{ij} = \|x_\mathrm{s}[i] - x_\mathrm{t}[j]\|_2^2$ for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$ and is normalized to have $\|C\|_\infty = 1$. The marginals p and q are set to $p = \mathbf{1}_m/m$ and $q = \mathbf{1}_n/n$. While there is a rich toolbox for setting ρ adaptively Boyd et al. (2011), we simply set it to $\rho_0/(m+n)$ for some constant ρ_0. Finally, DROT always starts at $x_0 = pq^\top$.

Performance profile For each subplot in Figure 2, we set $m = n = 512$ and generate 100 random problems. We give a constraint violation budget of 10^{-4}, allow up to 1000 iterations, and set $\rho_0 = 2$. A algorithm is terminated as soon as one of these conditions is satisfied. We then collect the objective values at termination and compute the relative function gaps $|f(X_K) - f(X^\star)|/f(X^\star)$, where $f(X^\star)$ is the optimal value. Figure 2 shows the percentage of problems that are successfully solved up to the accuracy level ε specified in the x-axis. We can see that DROT is consistently more accurate and robust. It reinforces that substantial care is needed to select the right η for SK. We can see that SK is extremely vulnerable in single-precision. Even in double-precision, a SK method that seems to work in one setting can still run into numerical issues in other cases. For example, by slightly changing a statistical property of the underlying data, nearly 40% of the problems in Fig. 2(c) cannot be solved by SK2 due to numerical errors, even though it works reasonably well in Fig. 2(b).

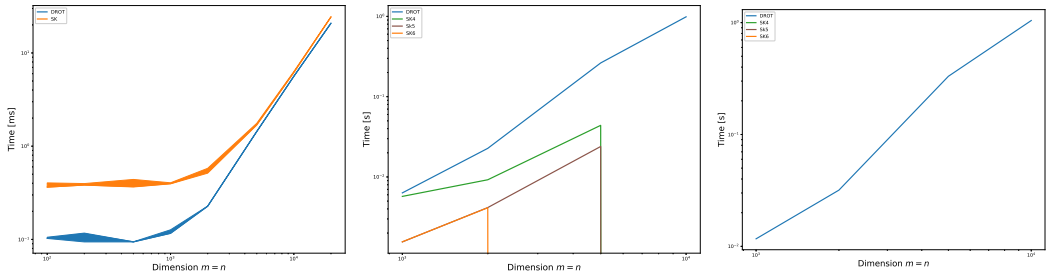
[3] https://developer.nvidia.com/blog/using-cuda-warp-level-primitives/
[4] https://pythonot.github.io



(a) Float 32, $\sigma_{\mathrm{t}} = 5$ (b) Float 64, $\sigma_{\mathrm{t}} = 5$ (c) Float 64, $\sigma_{\mathrm{t}} = 10$

Figure 2: Achievable accuracy and robustness

Runtime As we stride for the excellent per-iteration cost of SK, it would be incomplete if we do not compare these. Figure 3(a) shows the median of the per-iteration runtime and the 95% confidence interval, as a function of the dimensions. Here, m and n range from 100 to 20000 and each plot is obtained by performing 10 runs, and in each run the per-iteration runtime is the averaged runtime of 100 iterations. We noted that SKs call the highly optimized CuBLAS library to carry out its updates and we only evaluate the SK stopping criterion in every 10 iterations. The result confirms the efficacy of our kernel, and for very large problems their per-iteration time are almost identical. Finally, Fig 3(b)-(c) show the time requires for the total error (the sum of function gap and constraint violation) to reach different ε values. We have a very similar story here as in Fig. 2.



(a) Runtime per iteration (b) Time to $\varepsilon = 0.01$ (c) Time to $\varepsilon = 0.001$

Figure 3: Wall-clock time performance

Sparsity of transportation By design, DROT efficiently finds sparse transport plans. To illustrate this, we apply DROT to a color transferring problem between two images (see Blondel et al. (2018)). By doing so, we obtain a highly sparse plan ($> 98\%$ zeros), and in turn, a high-quality artificial image, visualized in Figure 4. In the experiment, we quantize each image with KMeans to reduce the number of distinct colors to 750. We subsequently use DROT to estimate an optimal color transfer between color distributions of the two images.

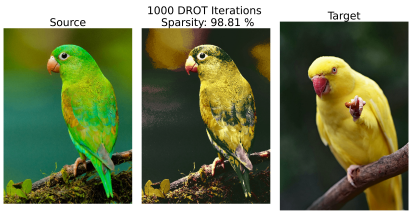


Figure 4: Color Transfer via DROT.

REFERENCES

Mokhtar Z Alaya, Maxime Berar, Gilles Gasso, and Alain Rakotomamonjy. Screening sinkhorn algorithm for regularized optimal transport. *arXiv preprint arXiv:1906.08540*, 2019.

Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *arXiv preprint arXiv:1705.09634*, 2017.

David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. Faster first-order primal-dual methods for linear programming using restarts and sharpness. *arXiv preprint arXiv:2105.12715*, 2021.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2nd edition, 2017.

Heinz H Bauschke, Shambhavi Singh, and Xianfu Wang. Projecting onto rectangular matrices with prescribed row and column sums. *arXiv preprint arXiv:2105.12222*, 2021.

Amir Beck. *First-order methods in optimization*, volume 25. SIAM, 2017.

Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and sparse optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pp. 880–889. PMLR, 2018.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

John Cheng, Max Grossman, and Ty McKercher. *Professional CUDA C programming*. John Wiley & Sons, 2014.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

Frank Deutsch and F Deutsch. *Best approximation in inner product spaces*, volume 7. Springer, 2001.

Jim Douglas and Henry H Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

Jonathan Eckstein and Dimitri P Bertsekas. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1): 293–318, 1992.

Masao Fukushima. The primal Douglas-Rachford splitting algorithm for a class of monotone mappings with application to the traffic equilibrium problem. *Mathematical Programming*, 72(1): 1–15, 1996.

Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1): 17–40, 1976.

Michael Garstka, Mark Cannon, and Paul Goulart. Cosmo: A conic operator splitting method for convex conic problems. *Journal of Optimization Theory and Applications*, pp. 1–32, 2021.

Pontus Giselsson and Stephen Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2016.

Bingsheng He and Xiaoming Yuan. On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

Franck Iutzeler and Jérôme Malick. Nonsmoothness in machine learning: specific structure, proximal identification, and applications. *Set-Valued and Variational Analysis*, 28(4):661–678, 2020.

David B Kirk and W Hwu Wen-Mei. *Programming massively parallel processors: a hands-on approach*. Morgan kaufmann, 2016.

Wu Li. Sharp lipschitz constants for basic optimal solutions and basic feasible solutions of linear programs. *SIAM journal on control and optimization*, 32(1):140–153, 1994.

Jingwei Liang, Jalal Fadili, and Gabriel Peyré. Local convergence properties of douglas–rachford and alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 172(3):874–913, 2017.

Tianyi Lin, Nhat Ho, and Michael Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pp. 3982–3991. PMLR, 2019.

Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

Brendan O'donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, pp. 1–36, 2020.

Sinong Wang and Ness Shroff. A new alternating direction method for linear programming. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1479–1487, 2017.

A Proof of Theorem 1

We first recall the DROT method:

$$X_{k+1} = \left[Y_k - \rho C\right]_+ \tag{10a}$$

$$Z_{k+1} = \mathsf{P}_{\mathcal{X}}(2X_k - Y_k) \tag{10b}$$

$$Y_{k+1} = Y_k + Z_{k+1} - X_{k+1}. \tag{10c}$$

Since X_{k+1} is the projection of $Y_k - \rho C$ onto $\mathbb{R}_+^{m\times n}$, it holds that

$$\langle X_{k+1} - Y_k + \rho C, X - X_{k+1}\rangle \geq 0 \quad \forall X \in \mathbb{R}_+^{m\times n}. \tag{11}$$

Since $\mathcal{X}$ is a closed affine subspace, we have

$$\langle Z_{k+1} - 2X_{k+1} + Y_k, Z - Z_{k+1} \rangle = 0 \quad \forall Z \in \mathcal{X}. \tag{12}$$

Next we define for $X \in \mathbb{R}_+^{m\times n}$, $Z \in \mathcal{X}$ and $U \in \mathbb{R}^{m\times n}$ the function:

$$V(X, Z, U) = f(X) + g(Z) + \frac{1}{\rho} \langle U, Z - X \rangle .$$

Let $U_{k+1} := Y_k - X_{k+1}$, it holds that

$$\begin{aligned} V(X_{k+1}, Z_{k+1}, U_{k+1}) - V(X, Z, U_{k+1}) &\leq f(X_{k+1}) + g(Z_{k+1}) + \frac{1}{\rho} \langle U_{k+1}, Z_{k+1} - X_{k+1} \rangle \\ &\quad - f(X) + g(Z) + \frac{1}{\rho} \langle U_{k+1}, Z - X \rangle \\ &= \langle C, X_{k+1} \rangle - \langle C, X \rangle + \frac{1}{\rho} \langle Y_k - X_{k+1}, Z_{k+1} - Z + X - X_{k+1} \rangle . \end{aligned}$$

By (11), we have $\langle Y_k - X_{k+1}, X - X_{k+1} \rangle / \rho \leq \langle C, X \rangle - \langle C, X_{k+1} \rangle$, we thus arrive at

$$\begin{aligned} &V(X_{k+1}, Z_{k+1}, U_{k+1}) - V(X, Z, U_{k+1}) \leq \frac{1}{\rho} \langle Y_k - X_{k+1}, Z_{k+1} - Z \rangle \\ &= \frac{1}{\rho} \langle X_{k+1} - Z_{k+1}, Z_{k+1} - Z \rangle = - \|X_{k+1} - Z_{k+1}\|^2 + \frac{1}{\rho} \langle X_{k+1} - Z_{k+1}, X_{k+1} - Z \rangle , \end{aligned} \tag{13}$$

where the second step follows from (12). By writing $U = Y - Z$ for some $Y \in \mathbb{R}^{m\times n}$, it holds that

$$\begin{aligned} &V(X_{k+1}, Z_{k+1}, U) - V(X_{k+1}, Z_{k+1}, U_{k+1}) \\ &\qquad = \frac{1}{\rho} \langle Y - Z, Z_{k+1} - X_{k+1} \rangle - \frac{1}{\rho} \langle Y_k - X_{k+1}, Z_{k+1} - X_{k+1} \rangle \\ &\qquad = \frac{1}{\rho} \langle Y - Y_k, Z_{k+1} - X_{k+1} \rangle + \frac{1}{\rho} \langle X_{k+1} - Z, Z_{k+1} - X_{k+1} \rangle . \end{aligned} \tag{14}$$

Since $Z_{k+1} - X_{k+1} = Y_{k+1} - Y_k$, it holds that

$$\frac{1}{\rho} \langle Y - Y_k, Z_{k+1} - X_{k+1} \rangle = \frac{1}{2\rho} \|Y_k - Y\|^2 - \frac{1}{2\rho} \|Y_{k+1} - Y\|^2 + \frac{1}{2\rho} \|Y_{k+1} - Y_k\|^2 . \tag{15}$$

Combining (13)–(15) yields

$$V(X_{k+1}, Z_{k+1}, U) - V(X, Z, U_{k+1}) \leq \frac{1}{2\rho} \|Y_k - Y\|^2 - \frac{1}{2\rho} \|Y_{k+1} - Y\|^2 - \frac{1}{2\rho} \|X_{k+1} - Z_{k+1}\|^2 .$$

Finally, since U (and hence Y) is arbitrary, f and g are convex functions, a standard telescoping step yields the desired result.

B Proof of Theorem 2

We will employ a similar technique for proving linear convergence as Wang & Shroff (2017) used for ADMM on LPs. Although they, in contrast to our approach, handled the linear constraints via relaxation, we show that the linear convergence also holds for our OT-customized splitting algorithm which relies on polytope projections. The main difference is that the dual variables are given implicitly, rather than explicitly via the multipliers defined in the ADMM framework. As a consequence, their proof needs to be adjusted to apply to our proposed algorithm. The proof comprises two main components, which will be treated independently before Theorem 1 is proven. Namely:

- The geometry of the solution set and the Hoffman bound,
- Basic operator properties.

B.1 THE GEOMETRY OF THE SOLUTION SET

To facilitate the analysis of the given splitting algorithm, we will consider the following equivalent LP:

$$\begin{array}{ll} \underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} & \langle C, X \rangle \\ \text{subject to} & Ze = p \\ & Z^\top f = q \\ & X \geq 0 \\ & Z = X. \end{array} \tag{16}$$

with $e = \mathbf{1}_n$ and $f = \mathbf{1}_m$. Using strong duality, we can explicitly state a set of *linear conditions* for optimality:

$$Z^* e = p \tag{17a}$$
$$Z^{*\top} f = q \tag{17b}$$
$$X^* \geq 0 \tag{17c}$$
$$Z^* = X^* \tag{17d}$$
$$\mu^* e^\top + f \nu^{*\top} \leq C \tag{17e}$$
$$\langle C, X^* \rangle - p^\top \mu^* - q^\top \nu^* = 0. \tag{17f}$$

This means that set of optimal solutions, denoted $\mathcal{S}^*$, is a polyhedron, with

$$\mathcal{S}^* = \{\mathcal{Z} = (X^*, Z^*, \mu^*, \nu^*) | M(\mathcal{Z}) = t, M_+(\mathcal{Z}) \leq 0\}.$$

where M, M_+ denote the equality, and inequality constraints respectively of (17). Li (1994) proposed a uniform bound of the distance between a point such a polyhedron in terms of the constraint violation given by:

$$\text{dist}(\mathcal{Z}, \mathcal{S}^*) \leq \theta_{\mathcal{S}^*} \left\| \begin{array}{c} M(\mathcal{Z}) - t \\ (M_+(\mathcal{Z}))_+ \end{array} \right\|. \tag{18}$$

For the (X, Z)-variables generated by (8), (17a), (17b) (17c) will always hold due to the projections carried out in the subproblems. As a consequence, only (17d), (17e) (17f) will contribute to the bound (18). In particular:

$$\text{dist}((X_k, Z_k, \mu_k, \nu_k), \mathcal{S}^*) \leq \theta_{\mathcal{S}^*} \left\| \begin{array}{c} Z_k - X_k \\ \langle C, X_k \rangle - p^\top \mu_k - q^\top \nu_k \\ (\mu_k e^\top + f \nu_k^\top - C)_+ \end{array} \right\|. \tag{19}$$

The following Lemma allows us to express the RHS of (19) in terms of $Y_k - Y_{k-1}$.

Lemma B.1. *Let X_k, Z_k and Y_k be a sequence generated by* (8)*. It then holds that*

$$\begin{aligned} Z_{k+1} - X_{k+1} =& Y_{k+1} - Y_k \\ \mu_{k+1} e^\top + f \nu_{k+1}^\top - C =& \rho^{-1}(Y_{k+1} - Y_k) \\ \langle C, X_{k+1} \rangle - p^\top \mu_{k+1} - q^\top \nu_{k+1} =& -\rho^{-1} \langle Y_{k+1}, Y_{k+1} - Y_k \rangle \end{aligned}$$

Proof. The first equality follows directly from the Y-update in (8).

From the Z-update (8), we have the following optimality associated with the projection:

$$
\begin{aligned}
Z_{k+1} &= \mathsf{P}_{\mathcal{X}}(2X_{k+1} - Y_k) \\
\implies Z_{k+1} - (2X_{k+1} - Y_k) - \rho\mu_{k+1}e^\top - \rho f\nu^\top &= 0 \\
\implies Y_{k+1} - X_{k+1} &= \rho\mu_{k+1}e^\top + \rho f\nu_{k+1}^\top.
\end{aligned} \tag{20}
$$

for some $\mu_{k+1} \in \mathbb{R}^m$ and $\nu_{k+1} \in \mathbb{R}^n$. And from the X-update and the Y-update of (8), we have

$$
X_{k+1} = \max(0, Y_k - \rho C) \geq 0 \tag{21}
$$

which gives that

$$
\begin{aligned}
Y_{k+1} &= Y_k + Z_{k+1} - X_{k+1} \\
&= \min(Y_k + Z_{k+1}, Z_{k+1} + \rho C).
\end{aligned}
$$

This in turn implies

$$
Y_{k+1} - Z_{k+1} - \rho C = \min(0, Y_k - \rho C) \leq 0 \tag{22}
$$

and

$$
\begin{aligned}
Y_{k+1} - X_{k+1} &= \min(Y_k + Z_{k+1} - X_{k+1}, Z_{k+1} - X_{k+1} + \rho C) \\
&= \min(Y_{k+1}, Y_{k+1} - Y_k + \rho C) \\
&\leq Y_{k+1} - Y_k + \rho C.
\end{aligned} \tag{23}
$$

By substituting (23) in (20), we obtain

$$
\mu_{k+1}e^\top + f\nu_{k+1}^\top - C \leq \rho^{-1}(Y_{k+1} - Y_k).
$$

For the 3rd relation, we note that (22) gives

$$
\begin{aligned}
0 \geq \min(0, Y_k - \rho C) &= Y_{k+1} - Z_{k+1} - \rho C \\
&= \rho\mu_{k+1}e^\top + \rho f\nu_{k+1}^\top + X_{k+1} - Z_{k+1} - \rho C \\
&= \rho\Big(\mu_{k+1}e^\top + f\nu_{k+1}^\top - \rho^{-1}(Y_{k+1} - Y_k) - C\Big).
\end{aligned}
$$

As a consequence, by (21), X_{k+1} and $\mu_{k+1}e^\top + f\nu_{k+1}^\top - \rho^{-1}(Y_{k+1} - Y_k) - C$ cannot be nonzero simultaneously. Equivalently we have

$$
\left\langle X_{k+1}, \mu_{k+1}e^\top + f\nu_{k+1}^\top - \rho^{-1}(Y_{k+1} - Y_k) - C \right\rangle = 0
$$

or

$$
\begin{aligned}
\langle C, X_{k+1} \rangle &= \left\langle X_{k+1}, \mu_{k+1}e^\top + f\nu_{k+1}^\top - \rho^{-1}(Y_{k+1} - Y_k) \right\rangle \\
&= \left\langle Z_{k+1} - (Y_{k+1} - Y_k), \mu_{k+1}e^\top + f\nu_{k+1}^\top - \rho^{-1}(Y_{k+1} - Y_k) \right\rangle \\
&= \left\langle Z_{k+1}, \mu_{k+1}e^\top + f\nu_{k+1}^\top \right\rangle - \left\langle \rho^{-1}Z_{k+1} + \mu_{k+1}e^\top + f\nu_{k+1}^\top, Y_{k+1} - Y_k \right\rangle \\
&\quad + \rho^{-1}\left\|Y_{k+1} - Y_k\right\|_{\mathrm{F}}^2 \\
&= \left\langle Z_{k+1}, \mu_{k+1}e^\top + f\nu_{k+1}^\top \right\rangle - \rho^{-1}\left\langle 2Y_{k+1} - Y_k, Y_{k+1} - Y_k \right\rangle \\
&\quad + \rho^{-1}\left\|Y_{k+1} - Y_k\right\|_{\mathrm{F}}^2 \\
&= \left\langle Z_{k+1}, \mu_{k+1}e^\top + f\nu_{k+1}^\top \right\rangle - \rho^{-1}\left\langle Y_{k+1}, Y_{k+1} - Y_k \right\rangle
\end{aligned} \tag{24}
$$

Note that:

$$\begin{aligned}\left\langle Z_{k+1}, \mu_{k+1}e^\top + f\nu_{k+1}^\top \right\rangle &= \mathrm{tr}(Z_{k+1}^\top \mu_{k+1} e^\top) + \mathrm{tr}(Z_{k+1}^\top f \nu_{k+1}^\top) \\ &= \mathrm{tr}(e^\top Z_{k+1}^\top \mu_{k+1}) + \mathrm{tr}(\nu_{k+1}^\top Z_{k+1}^\top) \\ &= \mathrm{tr}(p^\top \mu_{k+1}) + \mathrm{tr}(\nu_{k+1}^\top q) \\ &= p^\top \mu_{k+1} + q^\top \nu_{k+1} \end{aligned} \tag{25}$$

Substituting (25) to (24), we get

$$\langle C, X_{k+1}\rangle - p^\top \mu_{k+1} + q^\top \nu_{k+1} = -\rho^{-1} \langle Y_{k+1}, Y_{k+1} - Y_k\rangle .$$

□

By combining Lemma B.1 and (19), we obtain the following result:

Lemma B.2. *Let X_k, Z_k and Y_k be a sequence generated by* (8) *and μ_k, ν_k be defined by* (20)*, then:*

$$\mathrm{dist}((X_k, Z_k, \mu_k, \nu_k), \mathcal{S}^*) \leq \gamma \left\| Y_k - Y_{k-1} \right\|_{\mathrm{F}} .$$

Proof.

$$\begin{aligned}\mathrm{dist}((X_k, Z_k, \mu_k, \nu_k), \mathcal{S}^*) &\leq \theta_{\mathcal{S}^*} \left\| \begin{matrix} Y_k - Y_{k-1} \\ -\rho^{-1} \langle Y_k, Y_k - Y_{k-1}\rangle \\ (\rho^{-1}(Y_k - Y_{k-1}))_+ \end{matrix} \right\| . \\ &\leq \theta_{\mathcal{S}^*}(\|Y_k - Y_{k-1}\| + \rho^{-1} |\langle Y_k, Y_k - Y_{k-1}\rangle| + \rho^{-1}\|Y_k - Y_{k-1}\|) \\ &\leq \theta_{\mathcal{S}^*}(1 + \rho^{-1} + \rho^{-1}\|Y_k\|)\|Y_k - Y_{k-1}\| \\ &\leq \theta_{\mathcal{S}^*}(1 + \rho^{-1}(M+1))\|Y_k - Y_{k-1}\| \end{aligned}$$

where M is a positive constant such that $M \geq \|Y_k\|$ for all k. This exists due to the convergence of the algorithm. □

B.2 Operator properties

Consider the following operator, associated with the DR-algorithm:

$$T(y) = y + \mathrm{prox}_{\rho g}\left(2\mathrm{prox}_{\rho f}(y) - y\right) - \mathrm{prox}_{\rho f}(y) .$$

DR corresponds to finding a fixpoint to T, i.e. a point $y^* : y^* = T(y^*)$, or equivalently finding a point in the nullspace of G, where $G(y) = y - T(y)$. T, and thereby also G, are *firmly non-expansive* (see e.g. Lions & Mercier (1979)). That is, for all $y, y' \in \mathrm{dom}(T)$:

$$\begin{aligned}\langle T(y) - T(y'),\, y - y'\rangle &\geq \|T(y) - T(y')\|^2 \\ \langle G(y) - G(y'),\, y - y'\rangle &\geq \|G(y) - G(y')\|^2. \end{aligned} \tag{26}$$

Let $\mathcal{G}^*$ be the set of all fixpoints to T, i.e. $\mathcal{G}^* = \{y \in \mathrm{dom}(T) |\, G(y) = 0\}$. Then (26) implies the following bound:

Lemma B.3. *Let $y^* \in \mathcal{G}^*$ and $y_{k+1} = T(y_k)$, then:*

$$\|y_{k+1} - y^*\|^2 \leq \|y_k - y^*\|^2 - \|y_{k+1} - y_k\|^2.$$

Proof.

$$\begin{aligned}\|y_{k+1} - y^*\|^2 &= \|y_{k+1} - y_k + y_k - y^*\|^2 \\ &= \|y_k - y^*\|^2 + 2\langle y_{k+1} - y_k, y_k - y^*\rangle + \|y_k - y^*\|^2 \\ &= \|y_k - y^*\|^2 - 2\langle G(y_k) - G(y^*), y_k - y^*\rangle + \|y_k - y^*\|^2 \\ &\leq \|y_k - y^*\|^2 - 2\|G(y_k) - G(y^*)\|^2 + \|y_k - y^*\|^2 \\ &= \|y_k - y^*\|^2 - \|y_{k+1} - y_k\|^2\end{aligned}$$

□

Corollary B.1.

$$\|y_{k+1} - y_k\| \leq \|y_k - y^*\| \tag{27a}$$

$$(\mathrm{dist}(y_{k+1}, \mathcal{G}^*))^2 \leq (\mathrm{dist}(y_k, \mathcal{G}^*))^2 - \|y_{k+1} - y_k\|^2 \tag{27b}$$

Proof. The first inequality follows directly from the positivity of the LHS of Lemma B.3. The latter follows from:

$$(\mathrm{dist}(y_{k+1}, \mathcal{G}^*))^2 = \|y_{k+1} - \mathsf{P}_{\mathcal{G}^*}(y_{k+1})\|^2 \leq \|y_{k+1} - \mathsf{P}_{\mathcal{G}^*}(y_k)\|^2.$$

By applying the Lemma to the RHS with $y^* = \mathsf{P}_{\mathcal{G}^*}(y_k)$ we obtain the desired result. □

Recall that for DROT, $\mathrm{prox}_{\rho f}(\cdot) = \mathsf{P}_{\mathbb{R}_+^{m\times n}}(\cdot - \rho C)$ and $\mathrm{prox}_{\rho g}(\cdot) = \mathsf{P}_{\mathcal{X}}(\cdot)$. The following Lemma bridges the optimality conditions stated in (17) with $\mathcal{G}^*$.

Lemma B.4. *If $Y^* = X^* + \rho(\mu^* e^\top + f\nu^{*\top})$, where X^*, μ^*, and ν^* satisfy* (17)*, then $G(Y^*) = 0$.*

Proof. We need to prove that

$$G(Y^*) = \mathsf{P}_{\mathbb{R}_+^{m\times n}}(Y^* - \rho C) - \mathsf{P}_{\mathcal{X}}\Big(2\mathsf{P}_{\mathbb{R}_+^{m\times n}}(Y^* - \rho C) - Y^*\Big)$$

is equal to zero. Let us start with simplifying $\mathsf{P}_{\mathbb{R}_+^{m\times n}}(Y^* - \rho C)$. By complementary slackness, we know

$$\langle X^*, \mu^* e^\top + f\nu^{*\top} - C\rangle = 0,$$

and by dual feasibility

$$\mu^* e^\top + f\nu^{*\top} - C \leq 0.$$

Consequentially, we have:

$$\langle \mu^* e^\top + f\nu^{*\top} - C, X - X^*\rangle \leq 0, \quad \forall X \in \mathbb{R}_+^{m\times n}.$$

which defines of the normal cone of $\mathbb{R}_+^{m\times n}$, i.e.

$$\begin{aligned}&\mu^* e^\top + f\nu^{*\top} - C \in N_{\mathbb{R}_+^{m\times n}}(X^*) \\ \implies &Y^* - X^* - \rho C \in N_{\mathbb{R}_+^{m\times n}}(X^*) \\ \implies &X^* = \mathsf{P}_{\mathbb{R}_+^{m\times n}}(Y^* - \rho C).\end{aligned}$$

As a consequence, we have $\mathsf{P}_{\mathcal{X}}\Big(2\mathsf{P}_{\mathbb{R}_+^{m\times n}}(Y^* - \rho C) - Y^*\Big) = \mathsf{P}_{\mathcal{X}}(2X^* - Y^*)$. But by definition of Y^*, we have

$$\begin{aligned}0 &= Y^* - X^* - \rho(\mu^* e^\top + f\nu^{*\top}) \\ &= X^* - (2X^* - Y^*) - \rho\mu^* \times (\mathbf{1}^m)^\top \rho \mathbf{1}^n \times (\nu^*)^\top,\end{aligned}$$

which means that $X^* = \mathsf{P}_{\mathcal{X}}(2X^* - Y^*)$ (following from the optimality condition of the projection). By substituting all these quantities into the formula for $G(Y^*)$, we get that indeed $G(Y^*) = 0$. □

Lemma B.5. *There exists a $C' \geq 1$ such that*

$$\operatorname{dist}(Y_k, \mathcal{G}^*) \leq C' \|Y_k - Y_{k-1}\|$$

Proof. Let $Y^* = \mathsf{P}_{\mathcal{S}^*}(X_k) + \rho(\mathsf{P}_{\mathcal{S}^*}(\mu_k)e^\top + f(\mathsf{P}_{\mathcal{S}^*}(\nu_k))^\top)$. According to Lemma B.5, we have that $Y^* \in G^*$, then we have that:

$$\begin{aligned}
\operatorname{dist}(Y_k, \mathcal{G}^*) &= \|Y_k - \mathsf{P}_{G^*}(Y_k)\| \\
&\leq \|Y_k - Y^*\| \\
&= \|Y_k - \mathsf{P}_{\mathcal{S}^*}(X_k) - \rho(\mathsf{P}_{\mathcal{S}^*}(\mu_k)e^\top + f(\mathsf{P}_{\mathcal{S}^*}(\nu_k))^\top)\|.
\end{aligned}$$

By (20), $Y_k = X_k + \rho\mu_k e^\top + \rho f \nu_k^\top$, resulting in:

$$\begin{aligned}
\operatorname{dist}(Y_k, \mathcal{G}^*) &\leq \|X_k - \mathsf{P}_{\mathcal{S}^*}(X_k) + \rho(\mu_k - \mathsf{P}_{\mathcal{S}^*}(\mu_k))e^\top + \rho f(\nu_k - \mathsf{P}_{\mathcal{S}^*}(\nu_k))^\top\| \\
&\leq \|X_k - \mathsf{P}_{\mathcal{S}^*}(X_k)\| + \rho\|e\|\|\mu_k - \mathsf{P}_{\mathcal{S}^*}(\mu_k)\| + \rho\|f\|\|\nu_k - \mathsf{P}_{\mathcal{S}^*}(\nu_k)\| \\
&= \operatorname{dist}(X^*, \mathcal{S}^*) + \rho\|e\| \operatorname{dist}(\mu^*, \mathcal{S}^*) + \rho\|f\| \operatorname{dist}(\nu^*, \mathcal{S}^*) \\
&\leq \gamma(1 + \rho(\|e\| + \|f\|))\|Y_k - Y_{k-1}\|
\end{aligned}$$

where the last row follow from Lemma B.2. The fact that $C' = \gamma(1 + \rho(\|e\| + \|f\|) \geq 1$ follows from the first inequality of Corollary B.1. □

B.3 Proof of linear convergence

Proof. By combining Lemma B.5 with Corollary B.1, we obtain that:

$$\begin{aligned}
\operatorname{dist}(Y_{k+1}, \mathcal{G}^*))^2 &\leq (1 - \frac{1}{C'^2})(\operatorname{dist}(Y_k, \mathcal{G}^*))^2 \\
\implies \operatorname{dist}(Y_{k+1}, \mathcal{G}^*) &\leq \sqrt{1 - \frac{1}{C'^2}} \operatorname{dist}(Y_k, \mathcal{G}^*)
\end{aligned}$$

Letting $r = \sqrt{1 - \frac{1}{C'^2}} < 1$ and iterating the inequality k times gives

$$\begin{aligned}
\operatorname{dist}(Y_k, \mathcal{G}^*) &\leq \operatorname{dist}(Y_0, \mathcal{G}^*) r^k \\
&\leq O(\|Y_0 - Y^*\| \, r^k)
\end{aligned}$$

For the X-variable, we can use Lemma B.2 again

$$\begin{aligned}
\operatorname{dist}(X_{k+1}, \mathcal{S}^*) &\leq \gamma\|Y_{k+1} - Y_k\| \\
&\leq \gamma\|Y_k - \mathsf{P}_{\mathcal{G}^*}(Y_k)\| \\
&= \gamma \operatorname{dist}(Y_k, \mathcal{G}^*) \\
&\leq \gamma \operatorname{dist}(Y_0, \mathcal{G}^*) r^k.
\end{aligned}$$

Consequentially, we have

$$\operatorname{dist}(X_k, \mathcal{S}^*) \leq O(\|Y_0 - Y^*\| \, r^k).$$

□

B.4 A REMARK ON THE CONVERGENCE RATE

Since $e = \mathbf{1}_n$, $f = \mathbf{1}_m$, and $\gamma = \theta_{\mathcal{S}^*}(1 + \rho^{-1}(M+1))$, we have

$$\begin{aligned}
C' &= \gamma(1 + \rho(\|e\| + \|f\|) \\
&= \theta_{\mathcal{S}^*}(1 + \rho^{-1}(M+1))(1 + \rho(\sqrt{n} + \sqrt{m})) \\
&= \theta_{\mathcal{S}^*}(1 + (M+1)(\sqrt{n} + \sqrt{m}) + \rho^{-1}(M+1) + \rho(\sqrt{n} + \sqrt{m}))) \\
&\geq \theta_{\mathcal{S}^*}\left(1 + \sqrt{(M+1)(\sqrt{n} + \sqrt{m})}\right)^2
\end{aligned}$$

where the last inequality is tight when $\rho = \sqrt{(M+1)/(\sqrt{n} + \sqrt{m})}$.

C DR AND ITS CONNECTION TO ADMM

The Douglas-Rachford updates given in (5) can equivalently be formulated as:

$$\begin{aligned}
x_{k+1} &= \operatorname{prox}_{\rho f}(y_k) \\
z_{k+1} &= \operatorname{prox}_{\rho g}(2x_{k+1} + y_k) \\
y_{k+1} &= y_k + z_{k+1} - x_{k+1} \\
&\Leftrightarrow \\
z_{k+1} &= \operatorname{prox}_{\rho g}(2x_{k+1} + y_k) \\
y_{k+1} &= y_k + z_{k+1} - x_{k+1} \\
x_{k+2} &= \operatorname{prox}_{\rho f}(y_{k+1}) \\
&\Leftrightarrow \\
z_{k+1} &= \operatorname{prox}_{\rho g}(2x_{k+1} + y_k) \\
x_{k+2} &= \operatorname{prox}_{\rho f}(y_k + z_{k+1} - x_{k+1}) \\
y_{k+1} &= y_k + z_{k+1} - x_{k+1}.
\end{aligned}$$

If we let $w_{k+1} = \frac{1}{\rho}(y_k - x_{k+1})$, we get.

$$\begin{aligned}
z_{k+1} &= \operatorname{prox}_{\rho g}(x_{k+1} + \rho w_{k+1}) \\
x_{k+2} &= \operatorname{prox}_{\rho f}(z_{k+1} - \rho w_{k+1}) \\
w_{k+2} &= w_{k+1} + \frac{1}{\rho}(z_{k+1} - x_{k+2}).
\end{aligned}$$

By re-indexing the x and the w variables, we get:

$$\begin{aligned}
z_{k+1} &= \operatorname{prox}_{\rho g}(x_k + \rho w_k) \\
x_{k+1} &= \operatorname{prox}_{\rho f}(z_{k+1} - \rho w_k) \\
w_{k+1} &= w_k + \frac{1}{\rho}(z_{k+1} - x_{k+1}).
\end{aligned}$$

This is exactly ADMM applied to the composite convex problem on the form (4).