ANNEALED SINKHORN WITH MOMENTUM: CERTIFIED UNREGULARIZED OPTIMAL TRANSPORT IN LINEAR MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

We characterize Bregman Douglas–Rachford splitting (BDRS) for unregularized discrete optimal transport and develop an anytime primal–dual certificate in linear memory. We first establish that BDRS coincides with warm-started Inexact Proximal point method for exact Optimal Transport (IPOT) using a single inner Sinkhorn iteration. By eliminating the primal transport plan from the updates, we derive an equivalent dual formulation that reveals BDRS as annealed Sinkhorn under an implicit inverse-linear temperature schedule, with an additional log-scaling momentum term and a cooler kernel. While this explains the role of the temperature parameter in BDRS as an initial temperature, it also reduces the solver's memory requirement from quadratic to linear. Utilizing this annealing perspective, we introduce overrelaxed BDRS, which combines annealing and overrelaxed scaling within a single recursion. We derive a primal–dual certificate for both methods that can be evaluated in linear memory without transport plan construction, thus providing a computable stopping rule. On the DOTmark benchmark, combining momentum with the cooler kernel produces substantially smaller optimality gaps than annealed Sinkhorn under the same schedule. For pixel-level color transfer between 1024×1024 images, BDRS attains a lower repaired transport cost than MDOT-TNT with a $9\times$ speed up, reaching a relative duality gap of 1.59% in 24 minutes. We further demonstrate a color transfer with 4238×2365 images, yielding 10 million pixels per image and approximately one hundred trillion implicit transport entries, reaching a best relative duality gap of 2.41% and 2.80% within 35 hours in each direction on a single NVIDIA L40S GPU.

1 INTRODUCTION

The discrete optimal transport (OT) problem is a fundamental optimization problem underlying applications across machine learning, computer vision, economics, and computational biology, dating back to Monge (1781) and its modern linear programming formulation by Kantorovich (1942). The *unregularized* problem is of particular interest because it is free of entropic bias: it admits sparse transport plans and exact transport costs. Classical methods for unregularized OT include network simplex, combinatorial min-cost-flow methods (Pele & Werman, 2009), and interior-point methods (Lee & Sidford, 2014). Recent algorithms also achieve near-linear complexity under assumptions on the input geometry (Agarwal et al., 2024). For dense, large-scale applications, computational and memory efficiency remain central practical considerations. Entropic OT (EOT), popularized by Cuturi (2013), restored scalability through the Sinkhorn algorithm (Sinkhorn & Knopp, 1967), whose iterations reduce to computing dense matrix–vector products, ideally suited to GPUs.

The efficiency of EOT comes with an inherent speed–accuracy trade-off: as the temperature decreases, the EOT minimizer approaches the unregularized solution, but iteration counts grow and numerical instability arises. A rich line of work navigates this trade-off. Altschuler et al. (2017) obtain near-linear-time approximation guarantees via Sinkhorn iterations followed by a rounding step that restores exact marginal feasibility; Dvurechensky et al. (2018) establish an $\widetilde{\mathcal{O}}(n^2/\varepsilon^2)$ complexity bound for Sinkhorn and propose the accelerated APDAGD. Schmitzer (2019) combines log-domain stabilization with ε-scaling to warm-start Sinkhorn across decreasing temperatures, while Thibault et al. (2021) accelerate convergence at a fixed temperature via overrelaxed projections. Most re-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

cently, Chizat (2024) characterized the temperature schedules under which *annealed* Sinkhorn converges to unregularized OT, quantified the relaxation error induced by annealing, and proposed a debiased variant. These methods reduce, but do not eliminate, the underlying trade-off.

A complementary line of work applies proximal point algorithms (PPAs) to unregularized OT directly (see Eckstein, 1993; Parikh & Boyd, 2014; Rockafellar, 1976). When the Bregman divergence generated by the entropy is used, each proximal subproblem is an EOT problem and can be addressed with Sinkhorn iterations: Inexact Proximal point method for exact Optimal Transport (IPOT) (Xie et al., 2020) runs a small fixed number of inner iterations (typically one), although its convergence analysis requires subproblem accuracies that a single iteration is not known to satisfy. Subsequent work therefore certifies or accelerates the inner solves: inexact stopping criteria and inertial variants (Yang & Toh, 2022), dual block-coordinate solvers (Chu et al., 2023), mirror-descent formulations with adaptive temperature schedules (Kemertas et al., 2025a;b), and Newton-type inner solvers (Tang & Qiu, 2024; Wu et al., 2026; Pan et al., 2026).

A third family attacks OT through operator splitting. Douglas–Rachford splitting (DRS) has a long history in convex optimization (Eckstein & Bertsekas, 1992), and has been specialized to OT in several forms: DROT (Mai et al., 2021) applies Euclidean DRS to unregularized OT, PDOT (Lu & Yang, 2024) applies restarted primal–dual hybrid gradient, and Lindbäck et al. (2023) adapt DRS to EOT on GPUs; Burns & Liang (2026, Table 2) catalogue first-order methods for OT in detail. Within this family, Ma et al. (2025) recently showed that a special case of Bregman Douglas–Rachford Splitting (BDRS) can be applied directly to unregularized OT; hereafter we use BDRS to refer to this specific case. Overall, BDRS appears to be a distinct method, unrelated to the proximal point and Sinkhorn-based algorithms above. Moreover, Ma et al. (2025, Remark 5.1) observe empirically that BDRS performs well *without* requiring the regularization parameter to be tuned toward zero — an observation that has so far lacked an explanation.

In this paper, we present, among others, an exact algebraic characterization of BDRS that closes this gap. We first show that BDRS coincides with IPOT run with a single inner Sinkhorn iteration, connecting the operator splitting and proximal point views of the same matrix-scaling iteration. We then eliminate the primal transport plan from the iteration entirely and derive an equivalent *dual* formulation: the resulting updates are precisely those of annealed Sinkhorn under an inverse-linear temperature schedule, augmented by a momentum-type extrapolation of the dual potentials. In other words, BDRS *is* Annealed Sinkhorn (AS) with momentum and a cooler kernel. This characterization reveals its hidden cooling even when its temperature parameter is held fixed. We further propose a principled overrelaxation and an anytime optimality certificate that can be computed with linear memory. Furthermore, it has three practical consequences that we develop in the remainder of the paper: the dual form needs only linear memory, the annealing view suggests a principled overrelaxation, and every sweep admits a computable optimality certificate. Our contributions are as follows.

(1) Exact characterization and linear memory. We establish that BDRS is equivalent to IPOT (Xie et al., 2020) with a single inner Sinkhorn iteration and derive a dual formulation of BDRS, revealing it to be AS (Chizat, 2024) under an implicit inverse-linear cooling schedule with a momentum term and cooler kernel. In Dual BDRS, η controls the initial temperature and eliminates the $\mathcal{O}(N^2)$ primal transport plan from the iteration, reducing persistent solver memory to $\mathcal{O}(N)$.

(2) Overrelaxation. Guided by the annealing interpretation, we introduce Overrelaxed BDRS (O-BDRS), governed by a single parameter $\lambda \in (1, 2)$ that steepens the implicit cooling schedule. Its dual formulation unifies the mechanics of annealed and overrelaxed Sinkhorn (Thibault et al., 2021) in a single recursion, and setting $\lambda = 1$ recovers BDRS.

(3) Anytime certificate in linear memory. We derive a computable primal–dual certificate for both Dual BDRS and O-BDRS without the need to compute the primal transport plan. The duality gap certifies additive τ-optimality for the unregularized problem and provides a stopping rule that preserves linear memory. The certificate is valid independently of any convergence guarantee.

(4) Scale. For pixel-level color transfer between 1024×1024 images, Dual BDRS reaches a relative duality gap of 1.6% in 24 minutes, attaining a lower feasible transport cost than MDOT-TNT (Kemertas et al., 2025a) with a $9\times$ speedup. We further demonstrate color transfer between 4238×2365 images, solving a single global OT problem with more than 10 million atoms per image and 10^{14} implicit transport entries. Both 1000-iteration runs in each direction take approximately 35 hours on a single NVIDIA L40S GPU, attaining best relative duality gaps of 2.41% and 2.80% respectively.

2 Background and BDRS Preliminaries

We are interested in solving the unregularized discrete OT problem. For brevity, all references to the OT problem in this paper refer to this variant unless otherwise stated. The equations are given by (Kantorovich, 1942; Peyré & Cuturi, 2020):

$$p^\star = \min_{X \in \Pi(r,c)} \langle C, X \rangle, \quad \Pi(r,c) = \{X \geq 0 : X\mathbf{1}_n = r,\ X^\top \mathbf{1}_m = c\}, \tag{1}$$

where $X \in \mathbb{R}_+^{m \times n}$ is the transportation plan, $r \in \mathbb{R}_{++}^m$ and $c \in \mathbb{R}_{++}^n$ are the source and target marginals and have equal total mass of one, Π is the transportation polytope and $C \in \mathbb{R}^{m \times n}$ is the transport cost matrix. Entropic OT replaces this objective by $\langle C, X \rangle - \eta H(X)$, where η is a temperature that controls the approximation to the unregularized version and $H(X) := -\sum_{i=1}^m \sum_{j=1}^n X_{ij} \log X_{ij}$. The dual of equation 1 is given by:

$$p^\star = \max_{f \in \mathbb{R}^m,\, g \in \mathbb{R}^n} \{\langle r, f \rangle + \langle c, g \rangle : f_i + g_j \leq C_{ij} \text{ for all } i, j\}. \tag{2}$$

Here, f and g are commonly called the source and target Kantorovich (or dual) potentials and every dual feasible pair (f, g) provides a lower bound on $p^\star$. Ma et al. (2025) proposes a special case of BDRS that targets the dual form of unregularized discrete OT. The update equations admit a natural primal-dual matrix-scaling interpretation and are given as follows:

$$u^k := \frac{r}{(X^k \odot K)v^{k-1}} \tag{3a}$$

$$v^k := \frac{c}{(X^k \odot K)^\top u^k} \tag{3b}$$

$$X^{k+1} := \text{diag}(u^k)(X^k \odot K)\text{diag}(v^k) \tag{3c}$$

where $K = \exp(-C/\eta)$ and $\odot$ is element-wise multiplication. That is to say, that $K^{\odot(k+1)} = \exp(-C/\eta_{k+1})$, where $\eta_{k+1} = \frac{\eta}{k+1}$, is the kernel K cooled by temperature η_{k+1}. Throughout, division between vectors is understood element-wise, using the symbol $\oslash$ or as a fraction. Let us define the cumulative row and column scaling vectors up to step k as $a^k = \bigodot_{t=0}^k u^t$ and $b^k = \bigodot_{t=0}^k v^t$. Throughout this section, we adopt the initializations $a^{-1} = u^{-1} = \mathbf{1}_m$, $b^{-1} = v^{-1} = \mathbf{1}_n$, and $X^0 = \mathbf{1}_{m \times n}$. It is important to note that by definition, $(X^{k+1})^\top \mathbf{1}_m = c$, which is a property that we will be used extensively in the paper. We next state a result that underlies both the dual formulation of BDRS and the certified primal–dual bound developed later.

Lemma 1 (Non-recursive Primal Update)**.** *The matrix X^k generated by the BDRS iterations at the start of any iteration $k \geq 0$ takes the following non-recursive form:*

$$X^{k+1} = \textit{diag}(a^k) K^{\odot(k+1)} \textit{diag}(b^k). \tag{4}$$

Equivalently,

$$X_{ij}^{k+1} = \exp\left(\log a_i^k + \log b_j^k - \frac{k+1}{\eta} C_{ij}\right). \tag{5}$$

where X^k no longer relies on a recursive update.

We defer the proof to Appendix A and state two important intuitions. First, substituting the factorization into the BDRS updates u^k and v^k eliminates the intermediate X^k from the algorithm, leading to the dual formulation that we derive in Section 3.2. Second, the entrywise exponential form, together with column feasibility of X^{k+1}, enables the construction of a dual feasible pair (f, g) and hence a certified dual lower bound in Section 3.4.

3 Main Results

We present four main results in this section. First, we establish that BDRS is equivalent to IPOT (Xie et al., 2020) with a single Sinkhorn iteration. Second, we show that primal-dual BDRS can be formulated as dual BDRS, which exposes its connection to AS (Chizat, 2024) and enables an $\mathcal{O}(n)$ memory implementation. Third, we introduce O-BDRS and dual O-BDRS, which unifies overrelaxed and annealed Sinkhorn. Finally, we construct a stopping criterion based on primal–dual bounds instead of relying on row and column marginal errors.

3.1 BDRS IS EQUIVALENT TO IPOT

We first establish a connection between BDRS and IPOT (Xie et al., 2020) for discrete OT. Specifically, warm-started IPOT with a single step in the inner iteration is equivalent to BDRS. Although the two methods arise from fundamentally different optimization perspectives, they yield identical update equations when applied to unregularized OT. Specifically, Xie et al. (2020) derives IPOT from the primal perspective using an inexact Bregman proximal point method, whereas Ma et al. (2025) derives BDRS from the dual perspective using Bregman Douglas–Rachford splitting. Despite these distinct derivations, setting the number of IPOT inner iterations to $L = 1$ yields exactly the BDRS updates. This equivalence provides a useful bridge between the proximal-point and operator-splitting interpretations of unregularized OT and may provide insight into future convergence proofs under specific conditions, which is not the main focus of this paper. We refer the reader to Appendix B for a detailed derivation of the equivalence.

3.2 DUAL BDRS

In the following, we first use Lemma 1 to unroll BDRS into its dual form. Second, we show its connection to annealed Sinkhorn. Observe in equation 3 that evaluating the sequence requires storing X^k at each step, which can be prohibitively large. In this section, we establish that X^k is entirely redundant for the progression of the dual variables. By algebraically unrolling the BDRS sequence, we demonstrate that the algorithm can be fully decoupled from the primal state, reducing the updates to a highly efficient sequence of alternating projections over dual scaling vectors only.

Proposition 1 (Dual Formulation of BDRS)**.** *The BDRS iterations can be completely decoupled from the intermediate matrix X^k. The algorithm reduces exactly to a sequence of alternating projections over the cumulative scaling vectors a and b, with initial conditions $a^{-1} = b^{-1} = b^{-2} = \mathbf{1}$:*

$$a^k = \frac{r}{K^{\odot(k+1)}\left[\frac{(b^{k-1})^{\odot 2}}{b^{k-2}}\right]} \tag{6a}$$

$$b^k = \frac{c}{(K^{\odot(k+1)})^\top a^k} \tag{6b}$$

$$X^{k+1} = \mathit{diag}(a^k) K^{\odot(k+1)} \mathit{diag}(b^k) \tag{6c}$$

The proof is deferred to Appendix C.1. Dual BDRS indeed implicitly stores the cumulative scaling vectors a^k and b^k, and it is important to note that the variables are arbitrary and can be simply replaced with u^k and v^k such that the form is analogous to Sinkhorn.

Corollary 1 (Dual BDRS is annealed Sinkhorn with momentum)**.** *Relative to the standard annealed Sinkhorn recursion of Chizat (2024), BDRS makes exactly two modifications to the row update:*

$$\underbrace{\frac{r}{K^{\odot k} b^{k-1}}}_{\textit{Annealed Sinkhorn}} \quad \longrightarrow \quad \underbrace{\frac{r}{K^{\odot(k+1)}\left[(b^{k-1})^{\odot 2} \oslash b^{k-2}\right]}}_{\textit{Dual BDRS}}. \tag{7}$$

First, dual BDRS advances from $K^{\odot k}$ to $K^{\odot(k+1)}$. Second, it replaces the previous column scaling b^{k-1} by the extrapolated scaling

$$\widetilde{b}^{k-1} := (b^{k-1})^{\odot 2} \oslash b^{k-2}. \tag{8}$$

The column update and the completed matrix retain the annealed Sinkhorn form shown in Proposition 1. Moreover,

$$\log \widetilde{b}^{k-1} = \log b^{k-1} + \left(\log b^{k-1} - \log b^{k-2}\right), \tag{9}$$

so the second modification is unit inertial extrapolation in the column log scaling. In this precise sense, BDRS is annealed Sinkhorn with log scaling momentum on a cooler kernel.

The result follows by direct comparison with Proposition 1. The exact index matching and initialization relative to Chizat (2024) are provided in Appendix C.2.

Remark 1. *Dual BDRS reveals that primal-dual BDRS implicitly follows an inverse-linear temperature schedule even when η is held fixed. This provides a potential explanation for the observation of*

Ma et al. (2025, Remark 5.1) that the algorithm performs well without requiring η to be close to zero. A related cumulative-scaling recurrence appears in Ma (2026), where $C = 0$ and $X^0 = M > 0$, contains the same second-order extrapolation. For general transport costs, our characterization exposes the evolving kernel $K^{\odot(k+1)}$ and thereby identifies the implicit annealing. Although Chizat (2024) noted similarities between AS and IPOT, we prove that BDRS and IPOT are equivalent when $L = 1$ and show precisely how AS transforms into BDRS/IPOT. We note that the convergence analysis from AS does not immediately transfer to BDRS and we defer this analysis to future work.

Memory Efficiency of Dual BDRS The bottleneck of primal-dual BDRS is its quadratic memory requirement, as the dense transport plan $X^k \in \mathbb{R}^{m\times n}$ must be explicitly stored throughout the iterations. For OT problems with source and target distributions of size N, this results in $\mathcal{O}(N^2)$ memory complexity. Dual BDRS eliminates this bottleneck by removing X^k from the intermediate updates and requires only the one-dimensional vectors. Consequently, the memory required for the algorithm is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. The $O(N)$ bound assumes that pairwise costs can be evaluated on demand from a linear-size representation. An explicitly stored dense cost matrix requires $O(N^2)$ memory. Regardless, each iteration still requires $O(N^2)$ pairwise work.

3.3 Overrelaxed BDRS

We established that BDRS is an annealed Sinkhorn algorithm with a built-in momentum term in Corollary 1. We now show that there is a principled manner to obtain a faster annealing rate. To do so, we first observe that the standard BDRS state update (equation 3c) can be factored as follows:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^k)^\top\right)$$

In this form, the mechanism is transparent: the update takes the previous state X^k and applies an element-wise multiplicative correction factor, defined as $(K \odot u^k(v^k)^\top)$. Building directly on this geometric insight, we introduce Overrelaxed BDRS (O-BDRS) by applying an overrelaxation parameter $\lambda \in [1, 2)$ to explicitly stretch this correction step. Exponentiating the correction factor element-wise yields the O-BDRS primal update:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^k)^\top\right)^{\odot\lambda} \tag{10}$$

This specific exponentiation holds a connection to Bregman Peaceman–Rachford Splitting (BPRS), the details of which we defer to Appendix D. Crucially, by altering only this primal update rule, we can again manipulate the overrelaxed algorithm into a dual formulation and show that it elegantly unifies overrelaxed and annealed Sinkhorn. All proofs in this section are deferred to Appendix E.

Proposition 2 (O-BDRS as Overrelaxed Annealed Sinkhorn)**.** *Define cumulative row and column scaling vectors exponentiated by λ up to step k as $\alpha^k = \bigodot_{t=0}^{k} (u^t)^{\odot\lambda}$ and $\beta^k = \bigodot_{t=0}^{k} (v^t)^{\odot\lambda}$, $\lambda \in [1, 2)$, with initial conditions $\alpha^{-1} = \beta^{-2} = \beta^{-1} = \mathbf{1}$. For $k \geq 0$, define $K_k^{\mathrm{O}} := K^{\odot(\lambda k+1)}$ and*

$$\widetilde{\beta}^{k-1} := \beta^{k-1} \odot \left(\beta^{k-1} \oslash \beta^{k-2}\right)^{\odot 1/\lambda}.$$

Then O-BDRS admits the following equations

$$\widehat{\alpha}^k := r \oslash \left(K_k^{\mathrm{O}} \widetilde{\beta}^{k-1}\right), \qquad \widehat{\beta}^k := c \oslash \left((K_k^{\mathrm{O}})^\top \widehat{\alpha}^k\right), \tag{11}$$

$$\alpha^k := (\alpha^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\alpha}^k)^{\odot\lambda}, \qquad \beta^k := (\beta^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\beta}^k)^{\odot\lambda}. \tag{12}$$

The corresponding O-BDRS state is

$$X^{k+1} = \mathrm{diag}(\alpha^k) K^{\odot\lambda(k+1)} \mathrm{diag}(\beta^k). \tag{13}$$

Consequently, observe how this is analogous to an annealed Sinkhorn iteration, followed by the overrelaxation form proposed by Thibault et al. (2021). Setting $\lambda = 1$ recovers Dual BDRS. The effect of λ is discussed in Section 4 and Appendix G.2.

3.4 Certified Primal–Dual Bounds in Linear Memory

Many existing algorithms monitor the marginal residuals, which measure proximity to the transport polytope but do not certify the transport cost. We now construct a primal–dual certificate for unregularized OT using the scalings already computed by Dual BDRS and Dual O-BDRS. The dual lower bound requires only vector operations. The primal upper bound additionally uses a cost-weighted reduction that can be evaluated alongside the column update. Both bounds preserve the linear-memory formulation but require additional computation of the same order as a solver iteration.

Scalings shared by both methods. For any iteration $k \geq 0$, we write the scalings and effective temperature at this iteration as

Method	A	B	t	ε
BDRS	a^k	b^k	$\widetilde{b}^{k-1}$	$\eta/(k+1)$
O-BDRS	$\widehat{\alpha}^k$	$\widehat{\beta}^k$	$\widetilde{\beta}^{k-1}$	$\eta/(\lambda k+1)$

(14)

where t is the incoming column scaling used in the row update:

$$\widetilde{b}^{k-1} := (b^{k-1})^{\odot 2} \oslash b^{k-2}, \qquad \widetilde{\beta}^{k-1} := \beta^{k-1} \odot (\beta^{k-1} \oslash \beta^{k-2})^{\odot 1/\lambda}.$$

The stated initializations give $t = \mathbf{1}_n$ at $k = 0$. With $K_\varepsilon := \exp(-C/\varepsilon)$, both methods satisfy

$$A = r \oslash (K_\varepsilon t), \qquad B = c \oslash (K_\varepsilon^\top A). \tag{15}$$

For O-BDRS, these are the unrelaxed scalings $\widehat{\alpha}^k$ and $\widehat{\beta}^k$, evaluated before overrelaxation at the effective temperature $\eta/(\lambda k + 1)$. Consider the implicit plan immediately after the row update,

$$Z := \operatorname{diag}(A) K_\varepsilon \operatorname{diag}(t). \tag{16}$$

By construction, its row marginal is r, and its column marginal is

$$s := Z^\top \mathbf{1}_m = t \odot (K_\varepsilon^\top A) = c \odot t \oslash B. \tag{17}$$

Observe that c, t, and B are already available, so the column marginal $s = c \odot t \oslash B$ of Z can be computed using only elementwise vector operations. One could instead use the column-feasible matrix $Y = \operatorname{diag}(A) K_\varepsilon \operatorname{diag}(B)$, but evaluating its row marginal $A \odot (K_\varepsilon B)$ would require an additional kernel–vector multiplication. We therefore use Z for the upper bound: its row constraint holds by construction, and its column error can be evaluated directly from the available scalings.

Proposition 3 (Primal–Dual Certificate)**.** *Let $r \in \mathbb{R}^m_{++}$ and $c \in \mathbb{R}^n_{++}$ have total mass one, let $C \in \mathbb{R}^{m \times n}$ have finite entries, and let $A, B, t > 0$ and $\varepsilon > 0$ satisfy equation 15. Let Ω be any known bound satisfying $\Omega \geq \operatorname{osc}(C)$. Define*

$$f := \varepsilon \log A, \qquad g := \varepsilon \log(B \oslash c), \tag{18}$$

$$\delta := \frac{1}{2}\|s - c\|_1, \qquad w := (C \odot K_\varepsilon)^\top A, \tag{19}$$

and

$$L := \langle r, f \rangle + \langle c, g \rangle, \qquad U := \langle t, w \rangle + \Omega\delta. \tag{20}$$

Then (f, g) is feasible for the unregularized Kantorovich dual. Furthermore, the intermediate plan Z admits a feasible repair $\widehat{Z} \in \Pi(r, c)$ such that

$$L \leq p^\star \leq \langle C, \widehat{Z} \rangle \leq U. \tag{21}$$

Consequently,

$$0 \leq \langle C, \widehat{Z} \rangle - p^\star \leq U - L. \tag{22}$$

The proof and the construction of $\widehat{Z}$, and the stabilized log-domain computation of U are given in Appendix F. The upper bound has two terms: $\langle t, w \rangle = \langle C, Z \rangle$ is the cost of the intermediate plan, and $\Omega\delta$ bounds the cost of restoring its column marginal. The repair is the row-feasible specialization of the rounding procedure of Altschuler et al. (2017). For costs in $[0, 1]$, one may take $\Omega = 1$ without computing the exact cost range.

3.5 Algorithm and Log-Domain Execution

Algorithm 1 synthesizes Dual O-BDRS directly in the log-domain using the LogSumExp (LSE) operator for numerical stability. In each iteration, we define an implicit effective temperature η_k and compute an extrapolated momentum variable $\log \tilde{\beta}^{k-1}$. The core iteration alternates between evaluating unrelaxed row and column targets ($\log \hat{\alpha}^k$ and $\log \hat{\beta}^k$) and applying geometric overrelaxation via $\lambda \in [1, 2)$. Here, LSE reduces over the columns of its matrix argument: $[\mathrm{LSE}(M)]_p = \log \sum_q \exp(M_{pq})$. The column update uses the same operator with the transposed cost matrix. Setting $\lambda = 1$ perfectly recovers Dual BDRS. When a convergence check is required, the anytime primal-dual certificate $U - L$ is evaluated periodically in linear memory using the available scalings, as detailed in Proposition 3 and Appendix F.

Algorithm 1 Dual O-BDRS (Log-Domain)

Initialize: $\log \alpha^{-1}, \log \beta^{-2}, \log \beta^{-1} \leftarrow \mathbf{0}$
for $k = 0, 1, 2, \ldots$ **do**
  $\eta_k \leftarrow \eta/(\lambda k + 1)$
  $\log \tilde{\beta}^{k-1} \leftarrow \log \beta^{k-1} + \frac{1}{\lambda}(\log \beta^{k-1} - \log \beta^{k-2})$
  $\log \hat{\alpha}^k \leftarrow \log r - \mathrm{LSE}\left(-\frac{C}{\eta_k} + \mathbf{1}_m (\log \tilde{\beta}^{k-1})^\top\right)$
  $\log \alpha^k \leftarrow (1 - \lambda) \log \alpha^{k-1} + \lambda \log \hat{\alpha}^k$
  $\log \hat{\beta}^k \leftarrow \log c - \mathrm{LSE}\left(-\frac{C^\top}{\eta_k} + \mathbf{1}_n (\log \hat{\alpha}^k)^\top\right)$
  $\log \beta^k \leftarrow (1 - \lambda) \log \beta^{k-1} + \lambda \log \hat{\beta}^k$
  if certificate requested **then**
    Compute gap $U - L$ via Proposition 3 (see Algorithm 2, Appendix F)
    if $U - L \leq \tau$ **then break**
  end if
end for
return $(\log \alpha^k, \log \beta^k, \log \hat{\alpha}^k, \log \hat{\beta}^k, \log \widetilde{\beta}^{k-1}, k)$

4 Experiments

We evaluate the two modifications that distinguish BDRS from annealed Sinkhorn, examine the contribution of overrelaxation, and assess the practical cost of certification. The DOTmark experiments use all 450 image pairs at resolution 64×64, with unit mass marginals and squared Euclidean costs normalized to $[0, 1]$. Reference optimal values (p^*) are computed using POT's network simplex solver. Unless otherwise stated, we use log-domain implementations in `float64` on a single NVIDIA H200 GPU, set $\eta = 1$, and allow up to 10^5 iterations. Appendix G gives the preprocessing, implementation, and evaluation details.

Momentum and kernel ablation. We isolate the current row-update kernel and log-scaling momentum in a 2×2 ablation, and additionally compare against Debiased Annealed Sinkhorn (DAS) and O-BDRS with $\lambda \in \{1.2, 1.5, 1.99\}$. Figure 1 shows that changing the kernel alone has little effect under the tested schedule. Momentum alone improves the result, but combining both modifications, as in BDRS, produces substantially smaller rounded primal optimality gaps. O-BDRS further improves the aggregate performance at fixed η. These comparisons use the inverse-linear temperature schedule intrinsic to BDRS; they do not compare individually optimized annealing schedules. Details and classwise results appear in Appendix G.1 and Appendix H.

What explains the gain from overrelaxation? The comparisons in Appendix G.2 indicate that much of the fixed-η advantage is explained by reaching lower effective temperatures sooner. O-BDRS at approximately k/λ iterations closely reproduces BDRS's rounded objective values at iteration k at the later checkpoints. Lowering BDRS's initial η to match O-BDRS's terminal update-kernel temperature largely removes the objective difference at the same iteration count. Thus, O-BDRS offers a reduction in iteration count at a common η, while these results do not establish a

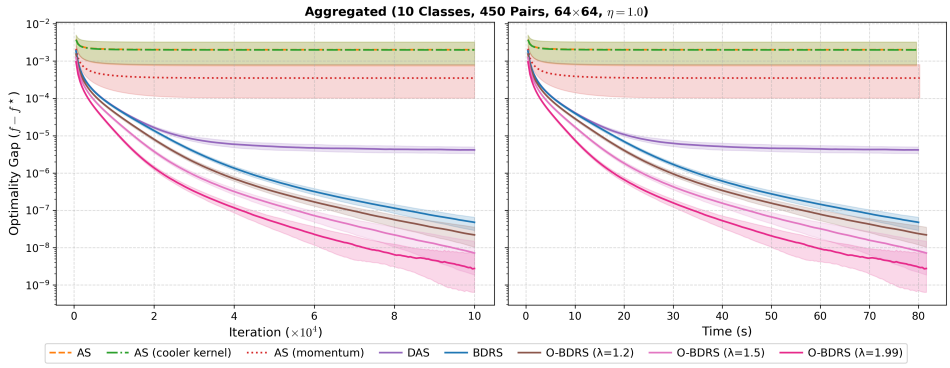


Figure 1: Momentum and kernel ablation on all 450 DOTmark pairs at resolution 64×64, with $\eta = 1$. The optimality gap $f - f^\star$ is shown against iteration count (left) and median solver time, excluding feasibility rounding (right). Here f is the cost after rounding and $f^\star$ is the reference value computed using POT's network simplex solver. Lines show medians across pairs; shaded bands indicate the interquartile range.

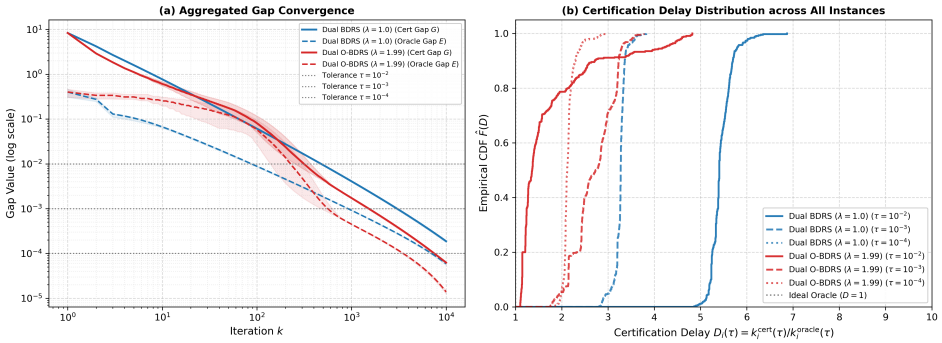


Figure 2: Anytime certification on 450 DOTmark pairs at resolution 64×64, comparing Dual BDRS ($\lambda = 1$) and Dual O-BDRS ($\lambda = 1.99$). Left: aggregated certified gaps $G^{\mathrm{cert}} = U^{\mathrm{best}} - L^{\mathrm{best}}$ (solid) and reference gaps $E^{\mathrm{oracle}} = U^{\mathrm{best}} - p^\star$ (dashed). Right: empirical cumulative distributions of the certification delay $D_i(\tau) = k_i^{\mathrm{cert}}(\tau)/k_i^{\mathrm{oracle}}(\tau)$ for $\tau \in \{10^{-2}, 10^{-3}, 10^{-4}\}$. Smaller delays indicate closer agreement with each method's own oracle stopping rule.

separate acceleration advantage over temperature-adjusted BDRS. The relevant tables can be found in Table 2(a) and (b) in Appendix G.2.

Effectiveness of the anytime certificate. This experiment uses a budget of 10^4 iterations. Figure 2 (left) shows that both methods reduce the certified gap during the run, with O-BDRS attaining a smaller aggregate certified gap at the later iterations. Of note is the region at 10^2 iterations, where BDRS actually has a slightly lower certified gap than O-BDRS on average. Figure 2 (right) shows the delay distributions and O-BDRS arrives at a given tolerance earlier than BDRS in general. At $\tau = 10^{-2}$, BDRS typically requires roughly five to six times its oracle iteration count. At $\tau = 10^{-4}$, the difference widens again and the certification delay for BDRS at this tolerance is not provided as the tolerance was not achieved by 10^4 iterations. For practitioners, this means that running O-BDRS achieves the desired tolerance earlier than BDRS. Appendix G.3 gives the definitions, the experiment setup and classwise distributions.

Large-Scale Pixel-Level Color Transfer The experiments here run in `float32` and use a single L40S GPU. We compare with Mirror Descent Optimal Transport – Truncated Newton (MDOT-TNT) (Kemertas et al., 2025a), the current state-of-the-art in large-scale OT, at $\gamma = 2^{10}$ on a $10^6 \times 10^6$ OT problem. MDOT-TNT required 3.5 hours and BDRS finished in 24 minutes ($9\times$ speedup), while achieving a lower transport cost (0.0497 vs. 0.0507). In addition, we run BDRS on a 4238×2365



Figure 3: Large-scale pixel-level color transfer between two 4238×2365 images using Dual BDRS. Top: original images A (left) and B (right). Bottom: transfers from A to B (left) and B to A (right). Each direction runs for 1000 iterations in `float32` on a single NVIDIA L40S GPU and takes approximately 35 hours. The best relative duality gaps $(U - L)/U$ are 2.41% and 2.80% respectively. Transferred images are obtained by barycentric projection.

image, which translates to a $10^7 \times 10^7$ OT problem, increasing the scale by an order of magnitude. On this experiment, we ran BDRS for 1000 iterations with $\eta = 0.01$ over 35 hours and achieved a best duality gap of 2.41%. The outputs are shown in Figure 3 and additional details are in Appendix G.4.

5 Conclusion

We established that BDRS is identical to IPOT with a single inner Sinkhorn iteration and derived an equivalent dual formulation. This formulation reveals BDRS as annealed Sinkhorn with log scaling momentum and a cooler kernel, while eliminating the need to store the primal transport plan, enabling linear memory. We further introduced O-BDRS and a primal–dual certificate that can be evaluated in linear memory using the available scalings. The certificate provides a computable stopping rule for unregularized OT whenever the requested gap is reached. The linear memory requirement of both the algorithm and the stopping rule thus enables unregularized OT on dense matrices at scale.

The DOTmark experiments show that the two modifications of momentum and a cooler kernel underlying BDRS work together: neither modification alone reproduces the observed performance of the combined update. The matched-temperature experiments further indicate that much of O-BDRS's advantage at fixed η is explained by faster cooling, since temperature-adjusted BDRS attains closely matching objective values. Together, these results clarify the relationship of unregularized OT viewed through proximal point algorithms (Xie et al., 2020), operator splitting (Ma et al., 2025), and annealed (Chizat, 2024) and overrelaxed Sinkhorn (Thibault et al., 2021) and provide a way to both compute the solution and monitor solution quality in linear memory.

Limitations. Our algebraic equivalences and certificate do not establish global convergence or an a-priori convergence rate for BDRS or O-BDRS. In particular, a valid certificate does not guarantee that a prescribed tolerance will be reached, and restricting λ to $[1, 2)$ does not by itself establish convergence. The ablations use the schedule intrinsic to BDRS and O-BDRS and do not establish superiority over independently optimized schedules. Linear memory also does not remove the $O(NM)$ pairwise work per iteration for dense problems with N source and M target atoms. Finally, the certificate is proved in exact arithmetic and thus its floating-point evaluation and attainable tolerances require numerical validation.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods. We have not used generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results are not applicable to this work. Additionally, we used generative AI tools to discover research topics or identify gaps, brainstorming, edit a research paper to improve readability, identify relevant literature, suggest a structure for a research paper. We have reviewed all AI-assisted work. For example, LLM-generated code was verified with unit tests and both the code and unit tests were checked. For proof writing, LLMs assisted with the manipulation of equations and the authors ultimately checked the equations. We further tested the proofs with code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We do not foresee and ethics issues with our work.

REPRODUCIBILITY STATEMENT

All code used to support this work will be released upon publication. The assumptions underlying our theoretical results are stated in Sections 2 and 3, with proofs and supporting derivations provided in Appendices A–F. Algorithm 1 specifies the log-domain solver, while Algorithm 2 and Appendix F detail the computation of the primal–dual certificate. Appendix G describes the datasets, preprocessing, parameter choices, implementation details, numerical precision, hardware, and evaluation procedures used in our experiments. Additional classwise results for the momentum and kernel ablation are provided in Appendix H.

REFERENCES

Pankaj K. Agarwal, Sharath Raghvendra, Pouyan Shirzadian, and Keegan Yao. Fast and Accurate Approximations of the Optimal Transport in Semi-Discrete and Discrete Settings. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 4514–4529. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1 2024. doi: 10.1137/1.9781611977912.159.

Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. *Advances in neural information processing systems*, 30, 2017.

Matthew X. Burns and Jiaming Liang. Log-Averaged Mirror Prox for Fast, Large-Scale Optimal Transport in Linear Space. 5 2026.

Lénaïc Chizat. Annealed Sinkhorn for Optimal Transport: convergence, regularization path and debiasing. 8 2024.

Hong T. M. Chu, Ling Liang, Kim-Chuan Toh, and Lei Yang. An efficient implementable inexact entropic proximal point algorithm for a class of linear programming problems. *Computational Optimization and Applications*, 85(1):107–146, 5 2023. ISSN 0926-6003. doi: 10.1007/s10589-023-00459-2.

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

Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational Optimal Transport: Complexity by Accelerated Gradient Descent Is Better Than by Sinkhorn's Algorithm. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1367–1376. PMLR, 7 2018. URL https://proceedings.mlr.press/v80/dvurechensky18a.html.

Jonathan Eckstein. Nonlinear Proximal Point Algorithms Using Bregman Functions, with Applications to Convex Programming. *Mathematics of Operations Research*, 18(1):202–226, 2 1993. ISSN 0364-765X. doi: 10.1287/moor.18.1.202.

Jonathan Eckstein and Dimitri P. Bertsekas. On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3): 293–318, 4 1992. ISSN 0025-5610. doi: 10.1007/BF01581204.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T H Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22 (78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Leonid V Kantorovich. On the translocation of masses. In *Dokl. akad. nauk. ussr (ns)*, volume 37, pp. 199–201, 1942.

Mete Kemertas, Amir-massoud Farahmand, and Allan Jepson. A Truncated Newton Method for Optimal Transport. In Y Yue, A Garg, N Peng, F Sha, and R Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 79276–79309, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c4b652b7e228b18e1c65478da3a4a2cf-Paper-Conference.pdf.

Mete Kemertas, Allan Douglas Jepson, and Amir-massoud Farahmand. Efficient and Accurate Optimal Transport with Mirror Descent and Conjugate Gradients. *Transactions on Machine Learning Research*, 2025b. ISSN 2835-8856. URL https://openreview.net/forum?id=FVFqrxeF8e.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 424–433, 2014.

Jacob Lindbäck, Zesen Wang, and Mikael Johansson. Bringing regularized optimal transport to lightspeed: a splitting method adapted for GPUs. *Advances in Neural Information Processing Systems*, 36:26845–26871, 2023.

Haihao Lu and Jinwen Yang. PDOT: a Practical Primal-Dual Algorithm and a GPU-Based Solver for Optimal Transport. 7 2024.

Shiqian Ma. Convergence of BDRS as a Matrix Scaling Algorithm. *Optimization Online preprint. Available at*, 2026.

Shiqian Ma, Lin Xiao, and Renbo Zhao. Bregman Douglas-Rachford Splitting Method. 9 2025.

Vien V. Mai, Jacob Lindbäck, and Mikael Johansson. A Fast and Accurate Splitting Method for Optimal Transport: Analysis and Implementation. 10 2021.

Gaspard Monge. *Mémoire sur la théorie des déblais et des remblais*. Imprimerie royale, 1781. URL https://books.google.com/books?id=IG7CGwAACAAJ.

Jianting Pan, Ji'an Li, and Ming Yan. Inexact Bregman Sparse Newton Method for Efficient Optimal Transport. 3 2026.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3): 127–239, 2014.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th international conference on computer vision*, pp. 460–467, 2009.

Gabriel Peyré and Marco Cuturi. Computational Optimal Transport. 3 2020.

R. Tyrrell Rockafellar. Monotone Operators and the Proximal Point Algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 8 1976. ISSN 0363-0129. doi: 10.1137/0314056.

Bernhard Schmitzer. Stabilized Sparse Scaling Algorithms for Entropy Regularized Transport Problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 1 2019. ISSN 1064-8275. doi: 10.1137/16M1106018.

Jörn Schrieber, Dominic Schuhmacher, and Carsten Gottschlich. Dotmark–a benchmark for discrete optimal transport. *IEEE Access*, 5:271–282, 2016.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Zihao Tang and Yixuan Qiu. Safe and Sparse Newton Method for Entropic-Regularized Optimal Transport. In A Globerson, L Mackey, D Belgrave, A Fan, U Paquet, J Tomczak, and C Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 129914–129943. Curran Associates, Inc., 2024. doi: 10.52202/079017-4127. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ea8620683340facbd5f754dd169e0980-Paper-Conference.pdf.

Alexis Thibault, Lénaïc Chizat, Charles Dossal, and Nicolas Papadakis. Overrelaxed Sinkhorn-Knopp Algorithm for Regularized Optimal Transport. 3 2021.

Di Wu, Ling Liang, and Haizhao Yang. PINS: Proximal Iterations with Sparse Newton and Sinkhorn for Optimal Transport. 5 2026.

Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for computing exact wasserstein distance. In *Uncertainty in artificial intelligence*, pp. 433–453, 2020.

Lei Yang and Kim-Chuan Toh. Bregman Proximal Point Algorithm Revisited: A New Inexact Version and Its Inertial Variant. *SIAM Journal on Optimization*, 32(3):1523–1554, 9 2022. ISSN 1052-6234. doi: 10.1137/20M1360748.

A PROOF: NON-RECURSIVE PRIMAL UPDATE

Lemma 1 (Non-recursive Primal Update)**.** *The matrix X^k generated by the BDRS iterations at the start of any iteration $k \geq 0$ takes the following non-recursive form:*

$$X^{k+1} = \mathit{diag}(a^k) K^{\odot(k+1)} \mathit{diag}(b^k). \tag{4}$$

Equivalently,

$$X_{ij}^{k+1} = \exp\left(\log a_i^k + \log b_j^k - \frac{k+1}{\eta} C_{ij}\right). \tag{5}$$

where X^k no longer relies on a recursive update.

Proof. We prove by induction the equivalent index-shifted identity and assume the form holds for arbitrary k

$$X^k = \mathrm{diag}(a^{k-1}) K^{\odot k} \mathrm{diag}(b^{k-1}), \qquad k \geq 0.$$

For the base case $k = 0$, using $K^{\odot 0} = \mathbf{1}_{m \times n}$, we have

$$X^0 = \mathrm{diag}(a^{-1}) K^{\odot 0} \mathrm{diag}(b^{-1}) = \mathrm{diag}(\mathbf{1}) K^{\odot 0} \mathrm{diag}(\mathbf{1}) = \mathbf{1}_{m \times n}$$

We substitute this assumed form into the intermediate core matrix $(X^k \odot K)$ which appears in the BDRS state update:

$$X^k \odot K = \big(\mathrm{diag}(a^{k-1}) K^{\odot k} \mathrm{diag}(b^{k-1})\big) \odot K$$

Because pre- and post-multiplying by diagonal matrices only scales the rows and columns, an element-wise multiplication by K passes through the diagonals and directly multiplies the inner kernel. Thus, $K^{\odot k} \odot K$ becomes $K^{\odot(k+1)}$:

$$X^k \odot K = \mathrm{diag}(a^{k-1}) K^{\odot(k+1)} \mathrm{diag}(b^{k-1}).$$

We now substitute this resolved core matrix directly into the state update equation 3c for the next step:

$$\begin{aligned} X^{k+1} &= \mathrm{diag}(u^k)\big(X^k \odot K\big)\mathrm{diag}(v^k) \\ &= \mathrm{diag}(u^k)\Big[\mathrm{diag}(a^{k-1}) K^{\odot(k+1)} \mathrm{diag}(b^{k-1})\Big]\mathrm{diag}(v^k). \end{aligned}$$

Matrix multiplication is associative, and the product of diagonal matrices is exactly the diagonal matrix of their element-wise product (i.e., $\mathrm{diag}(a)\mathrm{diag}(b) = \mathrm{diag}(a \odot b)$). Regrouping the left and right terms yields:

$$X^{k+1} = \mathrm{diag}(u^k \odot a^{k-1}) K^{\odot(k+1)} \mathrm{diag}(b^{k-1} \odot v^k).$$

Applying the definition of the cumulative vectors, $a^k = a^{k-1} \odot u^k$ and $b^k = b^{k-1} \odot v^k$, we arrive at the exact unrolled form:

$$X^{k+1} = \mathrm{diag}(a^k) K^{\odot(k+1)} \mathrm{diag}(b^k).$$

This completes the induction. To show the equivalent representation, we can take the (i, j)th entry of equation 4:

$$X_{ij}^{k+1} = a_i^k \, (K_{ij})^{k+1} \, b_j^k.$$

Since $K_{ij} = \exp(-C_{ij}/\eta)$, we have

$$(K_{ij})^{k+1} = \exp\left(-\frac{k+1}{\eta} C_{ij}\right).$$

Therefore,

$$X_{ij}^{k+1} = \exp\left(\log a_i^k + \log b_j^k - \frac{k+1}{\eta} C_{ij}\right),$$

which establishes the equivalent entrywise form.

□

Because $K = \exp(-C/\eta)$, we have $K^{\odot(k+1)} = \exp(-(k+1)C/\eta)$. Lemma 1 proves that the BDRS state matrix implicitly uses an effective kernel $K^{\odot(k+1)} = \exp(-(k+1)C/\eta)$ and the recursive form of equation 3c can be substituted with the non-recursive form.

B BDRS EQUIVALENCE WITH IPOT AND RELATION TO MDOT

The IPOT algorithm uses the generalized proximal point method, which traditionally aims to minimize a convex objective function $f(x)$ by iteratively evaluating a proximal operator structured around a regularization term. In the context of optimal transport, IPOT tackles the exact Wasserstein distance by using a Bregman divergence—specifically, the Kullback-Leibler divergence generated by the entropy function.

By framing the primal transportation matrix as a Bregman proximal point method with the transportation polytope constraints of unregularized OT, the subproblem at each iteration t mirrors an entropic OT problem and thus enables the use of the Sinkhorn algorithm. Solving this subproblem to convergence at every step is computationally expensive and IPOT circumvents this by introducing an *inexact* evaluation scheme that runs only a single Sinkhorn iteration ($L = 1$).

Counterintuitively, this inexact primal approximation yields an update sequence that is identical to the *exact* application of BDRS to the dual optimal transport problem (Ma et al., 2025). BDRS requires no relaxation or truncation and its exact alternating updates produce the same sequence as IPOT with $L = 1$.

B.1 EQUIVALENCE TO IPOT

To explicitly demonstrate this link, we map the BDRS variables to their IPOT counterparts (cf. Xie et al., 2020, Algorithm 1): the marginals r, c correspond to μ, ν; the cost kernel K corresponds to G; the scaling vectors u^k, v^k correspond to $a^{(t)}, b^{(t)}$; and the transport plan X^k corresponds to $\Gamma^{(t)}$. As shown below, the single-iteration inner loop of IPOT is equivalent to the alternating dual updates of BDRS.

BDRS (Operator Splitting)	**IPOT (Proximal Point Algorithm)**
$u^k := \dfrac{r}{(X^k \odot K)v^{k-1}}$	$a^{(t)} := \dfrac{\mu}{(\Gamma^{(t)} \odot G)b^{(t-1)}}$
$v^k := \dfrac{c}{(X^k \odot K)^\top u^k}$	$b^{(t)} := \dfrac{\nu}{(\Gamma^{(t)} \odot G)^\top a^{(t)}}$
$X^{k+1} := \mathrm{diag}(u^k)(X^k \odot K)\mathrm{diag}(v^k)$	$\Gamma^{(t+1)} := \mathrm{diag}(a^{(t)})(\Gamma^{(t)} \odot G)\mathrm{diag}(b^{(t)})$

This equivalence provides two complementary interpretations of the same matrix-scaling iteration. Specifically, what appears as a truncated inner Sinkhorn solve with a single iteration in IPOT is simultaneously a complete operator splitting iteration in BDRS. Though this equivalence was established, it is important not to immediately assume the convergence analysis can be directly applied. The convergence theory from Xie et al. (2020) requires conditions on the accumulated error of the inner solve with Sinkhorn and are not shown to hold for the practical choice of $L = 1$. Nonetheless, we are hopeful that by drawing this connection, a convergence proof can eventually emerge in future.

B.2 RELATIONSHIP WITH MDOT

Given the close connection between MDOT and IPOT (cf. Kemertas et al., 2025b, Appendix G), the equivalence of BDRS to IPOT at $L = 1$ connects BDRS and MDOT. In particular, the unregularized OT objective

$$f(X) = \langle C, X \rangle$$

is linear, so that $\nabla f(X^k) = C$. Consequently, the negative entropy mirror descent (MD) subproblem

$$X^{k+1} \in \arg\min_{X \in \Pi(r,c)} \left\{ \langle C, X \rangle + \frac{1}{\Delta\gamma_k} D_{\mathrm{KL}}(X|X^k) \right\}$$

coincides exactly with the KL-proximal subproblem underlying IPOT when $\Delta\gamma_k = 1/\beta_k$. Therefore, under matched initialization and exact subproblem solutions, exact MDOT and exact IPOT trace the same entropic-regularization path, with $\gamma_{k+1} = \gamma_k + \Delta\gamma_k$.

While these three methods are similar, their equivalences should not be conflated. BDRS interprets the single Sinkhorn iteration as a complete operator splitting iteration and thus does not solve the subproblem exactly. IPOT runs with a fixed β and each subproblem is solved by a prescribed number of Sinkhorn iterations, commonly $L = 1$. MDOT specifies an inverse temperature schedule $\{\gamma_k\}$ and approximately solves each subproblem to a chosen accuracy, potentially using Sinkhorn or any convex optimization solver. It is not surprising that these three methods are similar since it is well established that mirror descent, proximal point algorithms and operator splitting are all closely related. The focus here is to connect BDRS, IPOT and MDOT.

C Dual BDRS Extended Analysis

C.1 Proof: Dual Formulation of BDRS

In this section, we prove that BDRS can be represented in a dual formulation such that X^k is not necessary in the intermediate steps and thus need not be persisted in memory. We first prove Lemma 2, and then use the results from Lemma 1 and 2 to prove Proposition 1.

For convenience, we reproduce the BDRS update equations for discrete OT (Ma et al., 2025):

$$u^k := \frac{r}{(X^k \odot K)v^{k-1}}$$
$$v^k := \frac{c}{(X^k \odot K)^\top u^k}$$
$$X^{k+1} := \text{diag}(u^k)(X^k \odot K)\text{diag}(v^k)$$

where $K = \exp(-C/\eta)$ and $\odot$ denotes element-wise multiplication. In the following, we analyze the algorithm through the algebraic unrolling of the update equations.

Lemma 2 (Cumulative Dual Updates)**.** *Given the non-recursive form of X^k from Lemma 1, the row and column updates in BDRS can be rewritten as a sequence strictly over their cumulative counterparts:*

$$a^k = \frac{r}{K^{\odot(k+1)}(b^{k-1} \odot v^{k-1})} \tag{25a}$$
$$b^k = \frac{c}{(K^{\odot(k+1)})^\top a^k} \tag{25b}$$

Proof. From Lemma 1,

$$X^{k+1} = \text{diag}(a^k)K^{\odot k+1}\text{diag}(b^k),$$

therefore,

$$X^k \odot K = \text{diag}(a^{k-1})K^{\odot(k+1)}\text{diag}(b^{k-1})$$

and substituting into the row update u^k, we obtain:

$$u^k = \frac{r}{\left[\text{diag}(a^{k-1})K^{\odot(k+1)}\text{diag}(b^{k-1})\right] v^{k-1}}.$$

In the denominator, multiplying the diagonal matrix $\text{diag}(b^{k-1})$ by the column vector v^{k-1} simplifies to the element-wise product $b^{k-1} \odot v^{k-1}$. This yields:

$$u^k = \frac{r}{\text{diag}(a^{k-1})\left[K^{\odot(k+1)}(b^{k-1} \odot v^{k-1})\right]}.$$

Because pre-multiplying by the diagonal matrix $\text{diag}(a^{k-1})$ scales the rows of the resulting denominator vector, dividing the numerator r by this scaled vector is mathematically equivalent to multiplying both sides of the equation element-wise by a^{k-1}:

$$a^{k-1} \odot u^k = \frac{r}{K^{\odot(k+1)}(b^{k-1} \odot v^{k-1})}$$

Recognizing by definition that $a^{k-1} \odot u^k = a^k$ yields exactly equation 25a. Applying an identical substitution and factorization to the column update v^k yields equation 25b, completing the proof. □

By isolating the dynamics in Lemma 2, a profound structural reality of BDRS emerges. The intermediate matrix X^k is entirely redundant, and though not obvious now, tracking the step-wise vectors (v^{k-1}) alongside the cumulative vectors is unnecessary. We can now formalize the complete reduction of BDRS into a sequence of alternating projections analogous to the Sinkhorn algorithm, revealing its exact equivalence with annealed Sinkhorn.

Proposition 1 (Dual Formulation of BDRS)**.** *The BDRS iterations can be completely decoupled from the intermediate matrix X^k. The algorithm reduces exactly to a sequence of alternating projections over the cumulative scaling vectors a and b, with initial conditions $a^{-1} = b^{-1} = b^{-2} = \mathbf{1}$:*

$$a^k = \frac{r}{K^{\odot(k+1)}\left[\frac{(b^{k-1})^{\odot 2}}{b^{k-2}}\right]} \tag{6a}$$

$$b^k = \frac{c}{(K^{\odot(k+1)})^\top a^k} \tag{6b}$$

$$X^{k+1} = \mathit{diag}(a^k) K^{\odot(k+1)} \mathit{diag}(b^k) \tag{6c}$$

Proof. By definition, the step-wise dual vector v^{k-1} is the element-wise ratio between the current and previous cumulative vectors:

$$v^{k-1} = \frac{b^{k-1}}{b^{k-2}} \tag{26}$$

We substitute this ratio into the target denominator term from Lemma 2:

$$b^{k-1} \odot v^{k-1} = b^{k-1} \odot \left(\frac{b^{k-1}}{b^{k-2}}\right) = \frac{(b^{k-1})^{\odot 2}}{b^{k-2}} \tag{27}$$

Next, we substitute the resolved term from equation 27 directly into the cumulative row update equation 25a. This isolates the update rule so that it depends exclusively on the sequence of cumulative column vectors:

$$a^k = \frac{r}{K^{\odot(k+1)}\left[\frac{(b^{k-1})^{\odot 2}}{b^{k-2}}\right]} \tag{28}$$

The cumulative column update equation 25b already relies entirely on the cumulative vectors, remaining unchanged:

$$b^k = \frac{c}{(K^{\odot(k+1)})^\top a^k} \tag{29}$$

At this stage, the step-wise variables u and v have been eliminated from the system. Because the alternating dynamics of equation 28 and equation 29 are now entirely self-contained, the cumulative origin of a and b is no longer relevant to the forward progress of the algorithm, though it still inherently contains the property of being cumulative vectors. In other words, we can easily substitute a and b with u and v to obtain a form analogous to the Sinkhorn algorithm. Finally, we substitute the unrolled state equation from Lemma 1 and the proof is complete. □

We further verified dual BDRS computationally and the iterates match up to machine precision with primal-dual BDRS. This reformulation preserves the iterates of BDRS and warm-started IPOT with $L = 1$ and removes the need for the dense primal state. In Appendix F, when costs are evaluated implicitly, both the updates and the primal–dual certificate can be computed in linear memory, without materializing the transport plan or its feasible repair.

C.2 Annealed Sinkhorn Detailed Comparison

Annealed Sinkhorn. Chizat (2024) analyzes standard annealed Sinkhorn and show that a temperature sequence of $\eta_k = \Theta(1/\sqrt{k})$ balances the entropic error and relaxation error. Furthermore,

under specific conditions, annealed Sinkhorn converges to unregularized OT. The sequence of updates proceeds as follows:

$$u^k = \frac{r}{K_{k-1} v^{k-1}} \tag{30a}$$

$$v^k = \frac{c}{K_k^\top u^k} \tag{30b}$$

$$X^{k+1} = \text{diag}(u^k) K_k \text{diag}(v^k). \tag{30c}$$

Notably, if one were to prescribe an inverse linear temperature schedule where $\eta_k = \eta/(k+1)$, then the kernel at step k is given by:

$$K_k = \exp\left(-\frac{C}{\eta_k}\right) = \exp\left(-\frac{(k+1)C}{\eta}\right) = \left[\exp\left(-\frac{C}{\eta}\right)\right]^{\odot(k+1)} = K^{\odot(k+1)}, \tag{31}$$

with $k = 0$ as the initial step and the $\odot$ symbol to denote element-wise (Hadamard) operations. Because the exponential function maps C element-wise, scaling η by a factor of $\frac{1}{k+1}$ is equivalent to raising the base kernel K to the $(k+1)$-th Hadamard power. This notation reveals a crucial property: annealing the temperature inverse linearly is identical to repeatedly multiplying the kernel K by itself element-wise. Under this specific schedule, substituting K_k with $K^{\odot(k+1)}$ in equation 30c is equivalent to equation 6c. We now compare annealed Sinkhorn with dual BDRS side-by-side:

Annealed Sinkhorn	**Dual BDRS**
$u^k = \frac{r}{K^{\odot k} v^{k-1}}$	$u^k = \frac{r}{K^{\odot(k+1)} \left[\frac{(v^{k-1})^{\odot 2}}{v^{k-2}}\right]}$
$v^k = \frac{c}{(K^{\odot(k+1)})^\top u^k}$	$v^k = \frac{c}{(K^{\odot(k+1)})^\top u^k}$
$X^{k+1} = \text{diag}(u^k) K^{\odot(k+1)} \text{diag}(v^k)$	$X^{k+1} = \text{diag}(u^k) K^{\odot(k+1)} \text{diag}(v^k)$

Corollary 1 then immediately follows.

Debiased Annealed Sinkhorn. Chizat (2024) further proposes *Debiased Annealed Sinkhorn*, which modifies the row-scaling update to reduce the relaxation error induced by annealing. Define

$$\theta_k := 1 - \frac{\eta_{k-1}}{\eta_{(k-2)\vee 0}} = \begin{cases} 0, & k = 1, \\ \dfrac{1}{k}, & k \geq 2. \end{cases}$$

Under this schedule, Debiased Annealed Sinkhorn becomes

$$u^k = \frac{\left(u^{k-1}\right)^{\odot \theta_k} \odot r}{K^{\odot k} v^{k-1}}, \qquad v^k = \frac{c}{\left(K^{\odot(k+1)}\right)^\top u^k}, \qquad X^{k+1} = \text{diag}(u^k) K^{\odot(k+1)} \,\text{diag}(v^k).$$

Here, $a \vee b := \max\{a, b\}$, so $(k-2) \vee 0$ prevents the temperature schedule from being evaluated at a negative index. The factor $\left(u^{k-1}\right)^{\odot \theta_k}$ is the debiasing term: it modifies r using the previous scaling vector, with a strength determined by the relative decrease in temperature. This modification is designed to reduce the leading relaxation error introduced by annealing. At $k = 1$, $\theta_1 = 0$, so the correction is inactive and the numerator reduces to r.

D Overrelaxed BDRS Update and relation to BPRS

In the following, we restructure the BDRS state update:

$$X^{k+1} = \text{diag}(u^k)(X^k \odot K)\text{diag}(v^k)$$

such that the multiplicative nature in some form of an update rule with respect to X^k is revealed. Because pre-multiplying a matrix by a diagonal matrix scales its rows, and post-multiplying scales

its columns, this operation is equivalent to taking the element-wise (Hadamard) product of the inner matrix with the outer product of the scaling vectors. Applying this identity yields:

$$X^{k+1} = (X^k \odot K) \odot \left(u^k (v^k)^\top\right)$$

Because the Hadamard product is both associative and commutative, we can regroup these terms to isolate the previous state:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^k)^\top\right)$$

In this form, the underlying mechanics of the algorithm become immediately transparent. The update simply takes the previous state X^k and applies an element-wise multiplication against a step-wise correction term, defined as $\left(K \odot u^k (v^k)^\top\right)$. Building directly on this geometric insight, we can introduce a parameter $\lambda \in (1, 2)$ to explicitly tune the magnitude of this correction. By element-wise exponentiating the correction term, we can produce an accelerated update:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^k)^\top\right)^{\odot \lambda}$$

Interestingly, this update equation has a relation to Peaceman–Rachford (PR). The PR equations with half steps, extracted from Theorem 4.2 in Ma et al. (2025) are given as follows:

$$u^k := \frac{r}{(X^k \odot K) v^{k-1}} \tag{32a}$$

$$X^{k+\frac{1}{2}} := \text{diag}(u^k)(X^k \odot K)\text{diag}(v^{k-1}) \tag{32b}$$

$$v^k := \frac{c}{(X^{k+\frac{1}{2}} \odot K)^\top u^k} \tag{32c}$$

$$X^{k+1} := \text{diag}(u^k)(X^{k+\frac{1}{2}} \odot K)\text{diag}(v^k) \tag{32d}$$

The half-step equations can be reduced to a single step equation as follows:

$$u^k := \frac{r}{(X^k \odot K) v^{k-1}} \tag{33a}$$

$$v^k := \frac{c}{v^{k-1} \odot ((X^k \odot K^{\odot 2})^\top (u^k)^{\odot 2})} \tag{33b}$$

$$X^{k+1} := \text{diag}((u^k)^{\odot 2}) \left(X^k \odot K^{\odot 2}\right) \text{diag}(v^{k-1} \odot v^k) \tag{33c}$$

It turns out that equation 33c can be rewritten as follows:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^{k-1})^\top\right)^{\odot(\lambda-1)} \odot \left(K \odot u^k (v^k)^\top\right)$$

such that $\lambda = 1$ yields the primal update rule for BDRS and $\lambda = 2$ yields the primal update rule for BPRS. However, the update rule that we propose, guided by the insights of faster annealing, is somewhat similar to BPRS, with the only exception of v^{k-1} replaced by v^k in the rewrite of equation 33c to obtain:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^k)^\top\right)^{\odot(\lambda-1)} \odot \left(K \odot u^k (v^k)^\top\right)$$

At $\lambda = 1$, the proposed update reduces to BDRS. At $\lambda = 2$, it differs from the BPRS state update as described above. This similarity motivates further investigation of methods that potentially interpolate between BDRS and BPRS.

E Proof: Dual Overrelaxed BDRS

Recall from Section 3.3 and Appendix D that the factored BDRS primal update isolates a step-wise correction term, $(K \odot u^k (v^k)^\top)$. We accelerate this sequence by introducing an overrelaxation parameter $\lambda \in (1, 2)$, which corresponds to the element-wise exponentiation of this correction term:

$$X^{k+1} = X^k \odot \left(K \odot u^k (v^k)^\top\right)^{\odot \lambda}$$

This is the only change for A-BDRS and the dual updates remain the same. In the following, we restate Proposition 2 and the proof follows.

Proposition 2 (O-BDRS as Overrelaxed Annealed Sinkhorn)**.** *Define cumulative row and column scaling vectors exponentiated by λ up to step k as $\alpha^k = \bigodot_{t=0}^{k}(u^t)^{\odot\lambda}$ and $\beta^k = \bigodot_{t=0}^{k}(v^t)^{\odot\lambda}$, $\lambda \in [1, 2)$, with initial conditions $\alpha^{-1} = \beta^{-2} = \beta^{-1} = \mathbf{1}$. For $k \geq 0$, define $K_k^{\mathrm{O}} := K^{\odot(\lambda k+1)}$ and*

$$\widetilde{\beta}^{k-1} := \beta^{k-1} \odot \left(\beta^{k-1} \oslash \beta^{k-2}\right)^{\odot 1/\lambda}.$$

Then O-BDRS admits the following equations

$$\widehat{\alpha}^k := r \oslash \left(K_k^{\mathrm{O}} \widetilde{\beta}^{k-1}\right), \qquad \widehat{\beta}^k := c \oslash \left((K_k^{\mathrm{O}})^\top \widehat{\alpha}^k\right), \tag{11}$$

$$\alpha^k := (\alpha^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\alpha}^k)^{\odot\lambda}, \qquad \beta^k := (\beta^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\beta}^k)^{\odot\lambda}. \tag{12}$$

The corresponding O-BDRS state is

$$X^{k+1} = \mathrm{diag}(\alpha^k) K^{\odot\lambda(k+1)} \mathrm{diag}(\beta^k). \tag{13}$$

Proof. We define the cumulative row and column scaling vectors exponentiated by λ up to step k as follows:

$$\alpha^k = \bigodot_{t=0}^{k}(u^t)^{\odot\lambda} \quad \text{and} \quad \beta^k = \bigodot_{t=0}^{k}(v^t)^{\odot\lambda}$$

Assuming the same initialization where the primal state is a matrix of ones ($X^0 = \mathbf{1}_{m\times n}$), the overrelaxed state matrix unrolls inductively to carry an accelerated kernel exponent:

$$X^k = \mathrm{diag}(\alpha^{k-1}) K^{\odot\lambda k} \mathrm{diag}(\beta^{k-1})$$

When evaluating the local row and column projections, the algorithm evaluates the core matrix $(X^k \odot K)$. Substituting the unrolled state matrix into this core evaluation yields:

$$X^k \odot K = \mathrm{diag}(\alpha^{k-1}) K^{\odot(\lambda k+1)} \mathrm{diag}(\beta^{k-1})$$

This intermediate result is critical: it proves that the λ-exponentiated primal update forces the algorithm to evaluate its projections against a cooling kernel accelerated by λ, merging overrelaxation with an accelerated annealing schedule.

To eliminate the dense matrix X^k from the algorithmic loop, we must express the step-wise projections strictly in terms of the cumulative vectors. From our definition of β^k, the previous cumulative vector is updated via $\beta^{k-1} = \beta^{k-2} \odot (v^{k-1})^{\odot\lambda}$. By isolating the step-wise multiplier v^{k-1}, we obtain:

$$v^{k-1} = \left(\frac{\beta^{k-1}}{\beta^{k-2}}\right)^{\odot\frac{1}{\lambda}}$$

We now substitute the unrolled core matrix and the isolated v^{k-1} multiplier into the denominator of the standard row projection, $u^k = r/((X^k \odot K)v^{k-1})$:

$$u^k = \frac{r}{\alpha^{k-1} \odot \left[K^{\odot(\lambda k+1)} \left(\beta^{k-1} \odot \left(\frac{\beta^{k-1}}{\beta^{k-2}}\right)^{\odot\frac{1}{\lambda}}\right)\right]}$$

To execute the cumulative update $\alpha^k = \alpha^{k-1} \odot (u^k)^{\odot\lambda}$, we must raise this entire projection to the power of λ and multiply it element-wise with the previous state α^{k-1}. When the denominator is exponentiated, the isolated α^{k-1} scaling term becomes $(\alpha^{k-1})^{\odot\lambda}$. Dividing the outer α^{k-1} multiplier by this exponentiated denominator naturally produces the geometric averaging term $(\alpha^{k-1})^{\odot(1-\lambda)}$. This algebraic simplification results in the final unrolled row update:

$$\alpha^k = (\alpha^{k-1})^{\odot(1-\lambda)} \odot \left(\frac{r}{K^{\odot(\lambda k+1)} \left[\beta^{k-1} \odot \left(\frac{\beta^{k-1}}{\beta^{k-2}}\right)^{\odot\frac{1}{\lambda}}\right]}\right)^{\odot\lambda}$$

By applying the symmetric algebraic substitutions to the standard column projection, $v^k = c/((X^k \odot K)^\top u^k)$, we obtain the complementary column update:

$$\beta^k = (\beta^{k-1})^{\odot(1-\lambda)} \odot \left(\frac{c}{(K^{\odot(\lambda k+1)})^\top \left[\alpha^{k-1} \odot \left(\frac{\alpha^k}{\alpha^{k-1}} \right)^{\odot \frac{1}{\lambda}} \right]} \right)^{\odot \lambda}$$

We note that substituting $\lambda = 1$ reduces the geometric averaging exponents $(1 - \lambda)$ to zero and the momentum exponents $1/\lambda$ to one, perfectly recovering the dual BDRS formulation. Again, the choice of variables α and β is arbitrary and u and v can be used to obtain a form that is analogous to overrelaxed and annealed Sinkhorn. At this point, the complete equations for dual O-BDRS are given as follows:

$$\alpha^k = (\alpha^{k-1})^{\odot(1-\lambda)} \odot \left(\frac{r}{K^{\odot(\lambda k+1)} \left[\beta^{k-1} \odot \left(\frac{\beta^{k-1}}{\beta^{k-2}} \right)^{\odot \frac{1}{\lambda}} \right]} \right)^{\odot \lambda} \tag{34a}$$

$$\beta^k = (\beta^{k-1})^{\odot(1-\lambda)} \odot \left(\frac{c}{(K^{\odot(\lambda k+1)})^\top \left[\alpha^{k-1} \odot \left(\frac{\alpha^k}{\alpha^{k-1}} \right)^{\odot \frac{1}{\lambda}} \right]} \right)^{\odot \lambda} \tag{34b}$$

$$X^{k+1} = \text{diag}(\alpha^k) K^{\odot \lambda(k+1)} \text{diag}(\beta^k) \tag{34c}$$

Next, we show how the recursion can be reduced further to the stated form. Recall the definitions

$$K_k^{\text{O}} := K^{\odot(\lambda k+1)}, \qquad \widetilde{\beta}^{k-1} := \beta^{k-1} \odot \left(\beta^{k-1} \oslash \beta^{k-2} \right)^{\odot 1/\lambda},$$
$$\widehat{\alpha}^k := r \oslash \left(K_k^{\text{O}} \widetilde{\beta}^{k-1} \right), \qquad \widehat{\beta}^k := c \oslash \left((K_k^{\text{O}})^\top \widehat{\alpha}^k \right).$$

Substituting $\widehat{\alpha}^k$ into equation 34a directly gives

$$\begin{aligned} \alpha^k &= (\alpha^{k-1})^{\odot(1-\lambda)} \odot \left[r \oslash \left(K_k^{\text{O}} \widetilde{\beta}^{k-1} \right) \right]^{\odot \lambda} \\ &= (\alpha^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\alpha}^k)^{\odot \lambda}, \end{aligned}$$

which immediately proves the first equality in equation 12. It remains to simplify the vector appearing inside the column update equation 34b. Define

$$s^k := \alpha^{k-1} \odot \left(\alpha^k \oslash \alpha^{k-1} \right)^{\odot 1/\lambda}.$$

We show that $s^k = \widehat{\alpha}^k$. From the overrelaxation derived above,

$$\alpha^k = (\alpha^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\alpha}^k)^{\odot \lambda}.$$

Therefore,

$$\begin{aligned} \alpha^k \oslash \alpha^{k-1} &= (\alpha^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\alpha}^k)^{\odot \lambda} \oslash \alpha^{k-1} \\ &= (\alpha^{k-1})^{\odot(-\lambda)} \odot (\widehat{\alpha}^k)^{\odot \lambda} \\ &= \left(\widehat{\alpha}^k \oslash \alpha^{k-1} \right)^{\odot \lambda}. \end{aligned}$$

All entries are strictly positive, so taking the elementwise $1/\lambda$ power yields

$$\left(\alpha^k \oslash \alpha^{k-1} \right)^{\odot 1/\lambda} = \widehat{\alpha}^k \oslash \alpha^{k-1}.$$

Consequently,

$$s^k = \alpha^{k-1} \odot \left(\widehat{\alpha}^k \oslash \alpha^{k-1} \right) = \widehat{\alpha}^k. \tag{35}$$

Substituting equation 35 into the original column update equation 34b gives

$$\begin{aligned}\beta^k &= (\beta^{k-1})^{\odot(1-\lambda)} \odot \left[c \oslash \left((K_k^{\mathrm{O}})^\top \widehat{\alpha}^k\right)\right]^{\odot\lambda} \\ &= (\beta^{k-1})^{\odot(1-\lambda)} \odot (\widehat{\beta}^k)^{\odot\lambda},\end{aligned}$$

which immediately proves the second equality in equation 12. Finally, when $\lambda = 1$, we have

$$K_k^{\mathrm{O}} = K^{\odot(k+1)}, \qquad \widetilde{b}^{k-1} = (b^{k-1})^{\odot 2} \oslash b^{k-2},$$

and equation 12 reduces to $\alpha^k = \widehat{\alpha}^k$ and $\beta^k = \widehat{\beta}^k$. These are exactly the dual BDRS updates. □

F Proof and Evaluation of the Primal–Dual Certificate

We first restate Proposition 3 here for convenience.

Proposition 3 (Primal–Dual Certificate)**.** *Let $r \in \mathbb{R}^m_{++}$ and $c \in \mathbb{R}^n_{++}$ have total mass one, let $C \in \mathbb{R}^{m\times n}$ have finite entries, and let $A, B, t > 0$ and $\varepsilon > 0$ satisfy equation 15. Let Ω be any known bound satisfying $\Omega \geq \operatorname{osc}(C)$. Define*

$$f := \varepsilon \log A, \qquad g := \varepsilon \log(B \oslash c), \tag{18}$$

$$\delta := \frac{1}{2}\|s - c\|_1, \qquad w := (C \odot K_\varepsilon)^\top A, \tag{19}$$

and

$$L := \langle r, f\rangle + \langle c, g\rangle, \qquad U := \langle t, w\rangle + \Omega\delta. \tag{20}$$

Then (f, g) is feasible for the unregularized Kantorovich dual. Furthermore, the intermediate plan Z admits a feasible repair $\widehat{Z} \in \Pi(r, c)$ such that

$$L \leq p^\star \leq \langle C, \widehat{Z}\rangle \leq U. \tag{21}$$

Consequently,

$$0 \leq \langle C, \widehat{Z}\rangle - p^\star \leq U - L. \tag{22}$$

We prove Proposition 3 using the normalization identities in equation 15, reproduced here for convenience:

$$A = r \oslash (K_\varepsilon t), \qquad B = c \oslash (K_\varepsilon^\top A).$$

These identities hold for both choices in equation 14, so the same argument applies to BDRS and O-BDRS, reproduced here for convenience:

Method	A	B	t	ε
BDRS	a^k	b^k	$\widetilde{b}^{k-1}$	$\eta/(k+1)$
O-BDRS	$\widehat{\alpha}^k$	$\widehat{\beta}^k$	$\widetilde{\beta}^{k-1}$	$\eta/(\lambda k+1)$

Proof of Proposition 3. We first identify the marginals of the intermediate plan, then establish the dual lower bound and construct a feasible plan for the primal upper bound.

Intermediate marginals. From $A = r \oslash (K_\varepsilon t)$, the plan $Z = \operatorname{diag}(A) K_\varepsilon \operatorname{diag}(t)$ satisfies

$$Z\mathbf{1}_n = A \odot (K_\varepsilon t) = r. \tag{36}$$

Its column marginal is

$$Z^\top \mathbf{1}_m = t \odot (K_\varepsilon^\top A) = c \odot t \oslash B = s, \tag{37}$$

where the final equality uses the column update for B. All entries of Z and s are positive. Since the row marginal is r, Z has total mass one, and hence

$$\mathbf{1}_n^\top s = \mathbf{1}_m^\top r = \mathbf{1}_n^\top c = 1. \tag{38}$$

Dual lower bound. Taking the (i, j)th entry of Z gives

$$Z_{ij} = A_i \exp(-C_{ij}/\varepsilon) t_j.$$

Because $s_j = c_j t_j / B_j$, we obtain

$$\begin{aligned} f_i + g_j - C_{ij} &= \varepsilon \left(\log A_i + \log \frac{B_j}{c_j} - \frac{C_{ij}}{\varepsilon} \right) \\ &= \varepsilon \log \frac{Z_{ij}}{s_j} \leq 0. \end{aligned} \tag{39}$$

The inequality follows because $s_j = \sum_i Z_{ij}$, so every entry in column j is at most its column sum. Thus, (f, g) is dual feasible, and weak duality gives $L \leq p^\star$.

Feasible primal repair. We note that this is simply a one-sided repair that can be adapted from Altschuler et al. (2017) and state this in full for convenience and for the analysis later. Since the row marginal of Z already equals r, we now restore the column marginal to c. Since s and c have equal total mass,

$$\sum_j (s_j - c_j)_+ = \sum_j (c_j - s_j)_+ = \frac{1}{2} \|s - c\|_1 = \delta, \tag{40}$$

where $(x)_+ := \max\{x, 0\}$. If $\delta = 0$, then $s = c$ and Z is already feasible; set $\widehat{Z} = Z$. Suppose now that $\delta > 0$.

First, clip each column whose mass exceeds its target. Define

$$e_j := \min\left\{1, \frac{c_j}{s_j}\right\}, \qquad \widetilde{Z} := Z \operatorname{diag}(e). \tag{41}$$

Then $\widetilde{Z} \leq Z$ entrywise and the mass in column j becomes $\min\{s_j, c_j\}$. This removes exactly δ units of mass. Define the remaining row and column deficits by

$$p := r - \widetilde{Z}\mathbf{1}_n, \qquad q := c - \widetilde{Z}^\top \mathbf{1}_m. \tag{42}$$

Both vectors are nonnegative: column clipping only decreases the row masses, and every clipped column has mass at most its target. Moreover, $q_j = (c_j - s_j)_+$ and

$$\mathbf{1}_m^\top p = \mathbf{1}_n^\top q = 1 - \sum_j \min\{s_j, c_j\} = \delta. \tag{43}$$

To fill these deficits, distribute the missing mass q_j in each column among the rows in proportions p_i/δ. This gives

$$\widehat{Z} := \widetilde{Z} + \frac{pq^\top}{\delta}. \tag{44}$$

Indeed, the correction has row marginal p and column marginal q:

$$\frac{pq^\top}{\delta}\mathbf{1}_n = p, \qquad \left(\frac{pq^\top}{\delta}\right)^\top \mathbf{1}_m = q.$$

Consequently, $\widehat{Z} \geq 0$, $\widehat{Z}\mathbf{1}_n = r$, and $\widehat{Z}^\top \mathbf{1}_m = c$, so $\widehat{Z} \in \Pi(r, c)$.

Primal upper bound. Let $R := Z - \widetilde{Z}$ denote the mass removed by clipping and let $T := pq^\top/\delta$ denote the correction. Both matrices are nonnegative and have total mass δ. Writing $C_{\min} := \min_{i,j} C_{ij}$ and $C_{\max} := \max_{i,j} C_{ij}$ therefore gives

$$\begin{aligned} \langle C, \widehat{Z} - Z \rangle &= \langle C, T \rangle - \langle C, R \rangle \\ &\leq C_{\max}\delta - C_{\min}\delta \\ &= \operatorname{osc}(C)\delta \leq \Omega\delta. \end{aligned} \tag{45}$$

In words, the repair relocates δ units of mass, and relocating one unit increases the cost by at most the range of C. For $\delta = 0$, the same inequality holds because $\widehat{Z} = Z$.

Finally, the cost of the intermediate plan is

$$\begin{aligned}\langle C, Z\rangle &= \sum_j t_j \sum_i C_{ij} A_i (K_\varepsilon)_{ij} \\ &= \langle t, (C \odot K_\varepsilon)^\top A\rangle = \langle t, w\rangle. \end{aligned} \tag{46}$$

Since $\widehat{Z}$ is feasible, we conclude that

$$L \le p^\star \le \langle C, \widehat{Z}\rangle \le \langle C, Z\rangle + \Omega\delta = U.$$

Subtracting $p^\star$ and using $L \le p^\star$ proves equation 22. □

Evaluating the cost in the log domain. Key to the primal upper bound is computing the variable w. While it is simple to state mathematically

$$w := (C \odot K_\varepsilon)^\top A,$$

and with the indexed form

$$\begin{aligned} w_j &= \sum_i C_{ij} A_i \exp(-C_{ij}/\varepsilon) \\ &= \sum_i C_{ij} \exp\left(\log A_i - \frac{C_{ij}}{\varepsilon}\right). \end{aligned}$$

Our implementation is already in the log domain, and thus $\log A_i$ is readily available. However, directly exponentiating this expression can still cause underflow, even when the entries of A sum to one and the costs lie in $[0, 1]$. For example, consider two rows with $A_1 = A_2 = 1/2$ and a column j with $C_{1j} = C_{2j} = 1$. At $\varepsilon = 10^{-3}$, both exponential weights are

$$\exp\left(\log A_i - \frac{C_{ij}}{\varepsilon}\right) = \frac{1}{2} e^{-1000}, \qquad i \in \{1, 2\}.$$

These values underflow to zero in both single and double precision, so normalizing them by their sum would produce $0/0$. Yet their relative contributions are equal: each should receive half the weight when computing the mean cost within this column. We can preserve these relative contributions by subtracting the largest log weight before exponentiating. In this example, both shifted logarithms become zero, giving weights $(1, 1)$, which normalize to $(1/2, 1/2)$. This common rescaling leaves the weighted mean cost unchanged and motivates the reformulation of the computation of interest, $\langle t, w\rangle$, which we discuss next.

Recall that $s_j = c_j t_j / B_j$. Using the column update $B_j = c_j/(K_\varepsilon^\top A)_j$, we obtain

$$s_j = \frac{c_j t_j}{B_j} = t_j (K_\varepsilon^\top A)_j = \sum_i Z_{ij}.$$

Thus s_j is the total mass in column j of the intermediate plan Z. Define μ_j as the mean transport cost within this column. Substituting $Z_{ij} = A_i (K_\varepsilon)_{ij} t_j$ gives

$$\begin{aligned} \mu_j &:= \frac{\sum_i C_{ij} Z_{ij}}{s_j} \\ &= \frac{\sum_i C_{ij} A_i (K_\varepsilon)_{ij} t_j}{t_j (K_\varepsilon^\top A)_j} \\ &= \frac{\sum_i C_{ij} A_i (K_\varepsilon)_{ij}}{(K_\varepsilon^\top A)_j} \\ &= \frac{w_j}{(K_\varepsilon^\top A)_j}. \end{aligned}$$

Multiplying the mean cost by the column mass therefore gives $s_j \mu_j = t_j w_j$. Summing over columns, we obtain

$$\langle t, w\rangle = \langle s, \mu\rangle = \langle C, Z\rangle.$$

The column masses s are readily computed from the available log scalings. We now turn to μ. We first rewrite its numerator w_j and define

$$\ell_{ij} := \log A_i - \frac{C_{ij}}{\varepsilon}, \qquad M_j := \max_i \ell_{ij}.$$

Then, for each column j, we subtract M_j from every log weight. Since M_j does not depend on the row index i, we can factor out e^{M_j} to obtain

$$w_j = e^{M_j} \sum_i C_{ij} \exp(\ell_{ij} - M_j),$$
$$(K_\varepsilon^\top A)_j = \sum_i \exp(\ell_{ij}) = e^{M_j} \sum_i \exp(\ell_{ij} - M_j),$$

and thus achieve the necessary rescaling effect. However, the common factor e^{M_j} still carries the original scale and can underflow to zero. In the example above, $M_j = \log(1/2) - 1000$, so $e^{M_j} = \frac{1}{2}e^{-1000}$ is numerically zero. Multiplying the stabilized sums by this factor would therefore reintroduce the same $0/0$ problem when computing μ_j. We avoid this by canceling e^{M_j} algebraically between the numerator and denominator, so this factor is never evaluated.

Starting from the definition of μ_j, we obtain

$$\begin{aligned}
\mu_j &= \frac{w_j}{(K_\varepsilon^\top A)_j} \\
&= \frac{\sum_i C_{ij} A_i (K_\varepsilon)_{ij}}{\sum_i A_i (K_\varepsilon)_{ij}} \\
&= \frac{\sum_i C_{ij} \exp(\ell_{ij})}{\sum_i \exp(\ell_{ij})} \\
&= \frac{\sum_i C_{ij} e^{M_j} \exp(\ell_{ij} - M_j)}{\sum_i e^{M_j} \exp(\ell_{ij} - M_j)} \\
&= \frac{e^{M_j} \sum_i C_{ij} \exp(\ell_{ij} - M_j)}{e^{M_j} \sum_i \exp(\ell_{ij} - M_j)} \\
&= \frac{\sum_i C_{ij} \exp(\ell_{ij} - M_j)}{\sum_i \exp(\ell_{ij} - M_j)}.
\end{aligned} \tag{47}$$

Here we first expand w_j and the column sum, then use $A_i(K_\varepsilon)_{ij} = \exp(\ell_{ij})$. Since M_j is constant within column j, the factor e^{M_j} can be taken outside both sums and canceled. We evaluate the final expression directly, so neither e^{M_j} nor w_j is formed numerically.

Define the shifted column sum, which is essentially the denominator of μ_j as

$$D_j := \sum_i \exp(\ell_{ij} - M_j).$$

Observe that $1 \le D_j \le m$, since at least one shifted exponential equals 1 and every term lies in $[0, 1]$. Thus, when the log inputs are finite, the denominator cannot underflow to zero. Moreover, μ_j is a weighted average of the costs in column j, and therefore

$$\min_i C_{ij} \le \mu_j \le \max_i C_{ij}.$$

The computation of μ is thus stable.

Now, we address the computation of s. Recall

$$(K_\varepsilon^\top A)_j = e^{M_j} D_j.$$

Taking logarithms, we obtain

$$\begin{aligned}
d_j &:= \log\big((K_\varepsilon^\top A)_j\big) \\
&= \log\left(e^{M_j} D_j\right) \\
&= \log\left(e^{M_j}\right) + \log D_j \\
&= M_j + \log D_j.
\end{aligned} \tag{48}$$

Algorithm 2 Log-domain evaluation of the primal upper bound

Require: Log scalings $\log A, \log t$, marginal c, temperature $\varepsilon > 0$, access to costs C_{ij}, and $\Omega \geq \mathrm{osc}(C)$.

for $j = 1, \ldots, n$ **do**
  $\ell_i \leftarrow \log A_i - C_{ij}/\varepsilon$ for $i = 1, \ldots, m$
  $M_j \leftarrow \max_i \ell_i$
  $E_i \leftarrow \exp(\ell_i - M_j)$ for $i = 1, \ldots, m$
  $D_j \leftarrow \sum_i E_i, \quad N_j \leftarrow \sum_i C_{ij} E_i$
  $\mu_j \leftarrow N_j / D_j$
  $d_j \leftarrow M_j + \log D_j$
  $s_j \leftarrow \exp(\log t_j + d_j)$
end for
$\delta \leftarrow \frac{1}{2} \sum_j |s_j - c_j|$
$U \leftarrow \sum_j s_j \mu_j + \Omega \delta$
return U

Finally, we have

$$\begin{aligned} s_j &= t_j (K_\varepsilon^\top A)_j \\ &= \exp\left(\log t_j + \log((K_\varepsilon^\top A)_j)\right) \\ &= \exp(\log t_j + d_j). \end{aligned}$$

The transport cost is then obtained as

$$\langle C, Z \rangle = \sum_j s_j \mu_j.$$

To summarize, the primal upper bound can be evaluated using Algorithm 2. Here A and t are the scalings defining the row feasible intermediate plan Z; for O-BDRS, A is the unrelaxed row scaling.

G Experiment Details

Data and ground cost. We use the DOTmark benchmark (Schrieber et al., 2016), which contains ten classes of ten images at each resolution. For every class, we evaluate all $\binom{10}{2} = 45$ unordered image pairs, yielding 450 instances per resolution. Each image is normalized to have total mass one and we do not remove zero mass pixels. For an $s \times s$ image, let $z_i \in \{0, \ldots, s-1\}^2$ denote the coordinate of pixel i. Following DOTmark, we use the squared Euclidean ground cost on the pixel grid. We additionally normalize the ground cost by the maximum cost as follows:

$$C_{ij} = \frac{\|z_i - z_j\|_2^2}{2(s-1)^2}.$$

Consequently, $C_{ij} \in [0, 1]$ and $\|C\|_\infty = 1$. The main experiments use 64×64 instances, for which we compute the optimal $p^\star_{\mathrm{LP}}$ with the network simplex implementation in Python Optimal Transport (POT) (Flamary et al., 2021) with 10^7 maximum iterations, and verify successful termination on every instance.

Methods. Our main comparison is with AS, reflecting the paper's focus on understanding BDRS through its connection to annealed Sinkhorn. We include standard AS, AS with a cooler kernel, AS with momentum, and the debiased variant (Chizat, 2024). For the large-scale color-transfer experiment, we additionally compare with MDOT-TNT (Kemertas et al., 2025a), a state-of-the-art second-order method with demonstrated scalability to approximately one million pixels per image.

Implementation. All methods use the log domain implementation for numerical stability. Lazy pairwise reductions are only used for the large-scale experiments. Computations use float64 and a single H200 GPU for all experiments except for the large-scale color transfer task, where we opt to use float32 and a single L40S GPU to show the broad applicability of our method. We observed that

Table 1: Controlled ablation of kernel timing and log-scaling momentum. All variants share the same column update, annealing schedule, initialization, and transport plan reconstruction formula.

Row-update kernel	No momentum: b^{k-1}	Momentum: $\widetilde{b}^{k-1}$
Old: $K^{\odot k}$	Annealed Sinkhorn	Annealed Sinkhorn with momentum
Current: $K^{\odot(k+1)}$	Current-kernel Annealed Sinkhorn	Dual BDRS

the first run of any method normally has a longer runtime of a few milliseconds, and thus warm start the GPUs. Following the spirit of BDRS, where η does not have to be tuned, we simply set $\eta = 1$ for all experiments except for the color transfer task and maximum iterations to 10^5. For the color transfer task, we tuned η as we wanted to obtain lower duality gaps and the details are provided in Section G.4. The goal is to analyze full scale OT and thus zero mass values in the DOTmark benchmark are not removed, and we give those pixels a small mass of $1e^{-15}$.

G.1 Momentum and Kernel Ablation

We isolate the two modifications that distinguish BDRS from standard AS. We reproduce the equation in Corollary 1 for convenience:

$$\underbrace{\frac{r}{K^{\odot k} b^{k-1}}}_{\text{Annealed Sinkhorn}} \quad \longrightarrow \quad \underbrace{\frac{r}{K^{\odot(k+1)}\,[(b^{k-1})^{\odot 2} \oslash b^{k-2}]}}_{\text{Dual BDRS}}.$$

The first modification replaces the old kernel $K^{\odot k}$ in the row update with the current kernel $K^{\odot(k+1)}$. The second replaces b^{k-1} with the unit log-scaling extrapolation

$$\widetilde{b}^{k-1} := (b^{k-1})^{\odot 2} \oslash b^{k-2}.$$

The resulting controlled ablations form the 2×2 design in Table 1. We additionally compare against DAS, whose row update under the same inverse-linear temperature schedule is

$$u^k = \frac{(u^{k-1})^{\odot \theta_k} \odot r}{K^{\odot k} v^{k-1}}, \qquad \theta_k = \begin{cases} 0, & k = 1, \\ 1/k, & k \geq 2. \end{cases}$$

This method retains the old kernel and does not extrapolate the incoming column scaling; instead, it modifies the row scaling target using the previous row scaling. Note that in this ablation, we deliberately use the inverse-linear temperature schedule that is exactly induced by dual BDRS, rather than a schedule selected to optimize AS or DAS. In particular, the inverse temperature increment does not vanish so the convergence conditions established by Chizat (2024) do not apply to this linear schedule. Accordingly, the experiment here should be interpreted as a controlled comparison of Table 1 and DAS under schedule intrinsic to BDRS, and not a claim that BDRS dominates annealed methods under their individually optimized schedules. Replacing this schedule would define a different method that is no longer BDRS or O-BDRS studied here. We leave the design and analysis of such schedule modified variants to future work.

In this ablation, we ran a total of eight methods: four methods in Table 1, DAS, and O-BDRS with $\lambda = [1.2, 1.5, 1.99]$ in float64 precision. Each method was run for 100K iterations and rounding from Altschuler et al. (2017) was used at every iteration to compute f, the cost of the feasible primal transport plan and f^* was obtained from POT. For wall-clock timing, the time it took to round the result was excluded. Figure 1 shows that replacing the cooler kernel alone has little effect under this schedule. Adding momentum alone reduces the gap, but combining momentum with the cooler kernel, as in BDRS, produces substantially smaller errors. In addition, DAS (Chizat, 2024) does not outperform BDRS. O-BDRS further lowers the aggregate rounded primal gaps at the fixed η. The corresponding classwise results are shown in Figures 7–11 in Appendix H.

G.2 Matched Temperature Ablation

The fixed-η comparison in Appendix G.1 combines two effects: O-BDRS changes the scaling dynamics and reaches lower effective temperatures sooner. We now examine how much of its observed

objective improvement remains when the kernel temperatures are matched. We use the same 450 DOTmark pairs, numerical precision, initialization, and feasibility rounding as in Appendix G.1, and consider $\lambda \in \{1.2, 1.5, 1.99\}$.

For BDRS and O-BDRS, respectively, the temperatures of the kernels used in the scaling updates are

$$\varepsilon_k^{\mathrm{B}} = \frac{\eta_{\mathrm{B}}}{k+1}, \qquad \varepsilon_k^{\mathrm{O}} = \frac{\eta_{\mathrm{O}}}{\lambda k + 1}.$$

We compare the rounded feasible-plan costs using

$$\Delta_i = |f_{\mathrm{O},i} - f_{\mathrm{B},i}|,$$

and report the mean and standard deviation of Δ_i across instances in Table 2. This measures agreement between the two objective values, rather than either method's error relative to the optimum.

Regime 1: common initial temperature and approximately matched effective temperatures. We keep $\eta_{\mathrm{B}} = \eta_{\mathrm{O}} = 1$ and compare BDRS at iteration k with O-BDRS at $k_{\mathrm{O}} = \lfloor k/\lambda \rfloor$. This asks whether O-BDRS can attain a comparable objective value using approximately $1/\lambda$ as many iterations. The update-kernel temperatures are close, but are exactly equal only when $\lambda k_{\mathrm{O}} = k$. In particular,

$$\frac{\varepsilon_{k_{\mathrm{O}}}^{\mathrm{O}}}{\varepsilon_k^{\mathrm{B}}} - 1 = \frac{k - \lambda k_{\mathrm{O}}}{\lambda k_{\mathrm{O}} + 1}, \qquad 0 \le k - \lambda k_{\mathrm{O}} < \lambda.$$

Thus, the relative temperature mismatch decreases with the iteration budget.

Table 2(a) shows that the rounded objective values become closely aligned at the later checkpoints. At $k = 10^3$, the mean discrepancies are of order 10^{-4}, whereas at $k = 10^5$ they range from 2.3×10^{-11} to 9.0×10^{-10}. For $\lambda = 1.99$, the final comparison uses $50{,}251$ O-BDRS iterations against $100{,}000$ BDRS iterations. This supports an approximately twofold reduction in iteration count for comparable late-stage objective values under the common initial η; it does not by itself measure wall-clock or certified-stopping speedup.

Regime 2: matched terminal temperature at a common iteration budget. We fix $\eta_{\mathrm{O}} = 1$ and lower BDRS's initial parameter to

$$\eta_{\mathrm{B}} = \frac{T+1}{\lambda T + 1}, \qquad T = 10^5.$$

Both methods are evaluated at the same iteration k, and their kernel temperatures agree exactly at the terminal checkpoint T. Their earlier temperature schedules are not identical. This comparison asks whether adjusting BDRS's initial temperature reproduces the objective improvement seen with overrelaxation.

Table 2(b) shows very small objective discrepancies, with terminal means ranging from 1.8×10^{-11} to 1.8×10^{-9}. The agreement is already close at the earlier checkpoints, although the discrepancy and its variation are larger for $\lambda = 1.99$ than for the smaller relaxation parameters at several checkpoints. These results indicate that much of O-BDRS's fixed-η objective advantage can be reproduced by choosing a smaller initial temperature for BDRS.

Interpretation. The two regimes support a common explanation: O-BDRS reaches lower effective temperatures sooner, which accounts for much of its observed improvement at a fixed initial η. When η is held common across methods, overrelaxation provides a practical reduction in iteration count. When η is adjusted to approximately align the cooling schedules, the rounded objective values become very similar. This is an empirical observation about the tested instances and checkpoints, rather than an equivalence of transport plans, numerical stability, or certification behavior. It motivates separating temperature selection from the effect of the update dynamics in future comparisons and in the tuned color-transfer experiment below.

G.3 Effectiveness of Anytime Certificate

Experiment Setup. We evaluate the practical usefulness of the anytime certificate described in Section 3.4 and use an iteration budget of 10^4 steps. Suppose first that the optimal value $p_i^\star$ of

Table 2: Rounded transport-cost discrepancies under temperature matching on 450 DOTmark pairs at resolution 64×64. Entries report the mean and standard deviation of $\Delta_i = |f_{\mathrm{O},i} - f_{\mathrm{B},i}|$, where both costs are evaluated after feasibility rounding. (a) A common initial $\eta = 1$ with approximately matched kernel temperatures at k and $\lfloor k/\lambda \rfloor$. (b) A smaller initial η for BDRS, chosen to match O-BDRS's update-kernel temperature exactly at the terminal checkpoint $T = 10^5$, with both methods evaluated at the same iteration.

(a) Common initial temperature and approximately matched effective temperatures

BDRS (k)	$\boldsymbol{\lambda = 1.2}$ ($k_{\mathrm{O}} = \lfloor k/1.2 \rfloor$)	$\boldsymbol{\lambda = 1.5}$ ($k_{\mathrm{O}} = \lfloor k/1.5 \rfloor$)	$\boldsymbol{\lambda = 1.99}$ ($k_{\mathrm{O}} = \lfloor k/1.99 \rfloor$)
1,000	$(1.0 \pm 0.0) \times 10^{-4}$	$(1.0 \pm 0.0) \times 10^{-4}$	$(7.3 \pm 9.2) \times 10^{-5}$
10,000	$(8.3 \pm 0.4) \times 10^{-8}$	$(1.3 \pm 0.1) \times 10^{-7}$	$(4.5 \pm 0.2) \times 10^{-8}$
50,000	$(2.2 \pm 0.5) \times 10^{-10}$	$(3.4 \pm 0.7) \times 10^{-10}$	$(9.3 \pm 6.5) \times 10^{-10}$
100,000	$(2.3 \pm 1.0) \times 10^{-11}$	$(5.3 \pm 2.4) \times 10^{-11}$	$(9.0 \pm 14.1) \times 10^{-10}$

(b) Matched terminal temperature and equal iteration counts

Iteration (k)	**Pair 1:** $\boldsymbol{\lambda = 1.2}$ vs BDRS($\eta = \frac{100,001}{120,001}$) ($\epsilon_K = 8.33 \times 10^{-6}$)	**Pair 2:** $\boldsymbol{\lambda = 1.5}$ vs BDRS($\eta = \frac{100,001}{150,001}$) ($\epsilon_K = 6.67 \times 10^{-6}$)	**Pair 3:** $\boldsymbol{\lambda = 1.99}$ vs BDRS($\eta = \frac{100,001}{199,001}$) ($\epsilon_K = 5.03 \times 10^{-6}$)
1,000	$(2.2 \pm 0.3) \times 10^{-9}$	$(3.5 \pm 0.8) \times 10^{-9}$	$(5.9 \pm 11.0) \times 10^{-7}$
10,000	$(2.1 \pm 0.4) \times 10^{-10}$	$(5.1 \pm 1.7) \times 10^{-10}$	$(9.5 \pm 7.8) \times 10^{-10}$
50,000	$(3.5 \pm 1.3) \times 10^{-11}$	$(9.1 \pm 4.0) \times 10^{-11}$	$(7.8 \pm 13.3) \times 10^{-10}$
100,000	$(1.8 \pm 1.3) \times 10^{-11}$	$(4.3 \pm 5.6) \times 10^{-11}$	$(1.8 \pm 3.1) \times 10^{-9}$

instance i is available from an exact linear program (LP) solve. We can then evaluate the *certification delay*, which measures how many iterations are required before the duality gap achieves a prescribed tolerance τ compared with an oracle stopping rule that knows $p_i^\star$ and terminates when the optimality gap is within τ. To reflect how the certificate would be used in practice, we retain the best bounds observed up to iteration k:

$$U_{i,k}^{\mathrm{best}} := \min_{t \le k} U_{i,t}, \qquad L_{i,k}^{\mathrm{best}} := \max_{t \le k} L_{i,t}. \tag{49}$$

By weak duality,

$$L_{i,k}^{\mathrm{best}} \le p_i^\star \le U_{i,k}^{\mathrm{best}}. \tag{50}$$

If we indeed have access to $p_i^\star$, one can evaluate the optimality gap

$$E_{i,k}^{\mathrm{oracle}} := U_{i,k}^{\mathrm{best}} - p_i^\star. \tag{51}$$

In practice, however, $p_i^\star$ is unavailable, and termination must instead rely on the duality gap

$$G_{i,k}^{\mathrm{cert}} := U_{i,k}^{\mathrm{best}} - L_{i,k}^{\mathrm{best}}. \tag{52}$$

The two quantities satisfy

$$G_{i,k}^{\mathrm{cert}} = E_{i,k}^{\mathrm{oracle}} + \left(p_i^\star - L_{i,k}^{\mathrm{best}}\right) \ge E_{i,k}^{\mathrm{oracle}}. \tag{53}$$

Consequently, $G_{i,k}^{\mathrm{cert}} \le \tau$ guarantees that the returned plan is within τ of the optimal transport cost, although its true error may be smaller.

For a prescribed additive tolerance τ, define the oracle and certified stopping iterations by

$$k_i^{\mathrm{oracle}}(\tau) := \min\left\{k : E_{i,k}^{\mathrm{oracle}} \le \tau\right\}, \tag{54}$$

$$k_i^{\mathrm{cert}}(\tau) := \min\left\{k : G_{i,k}^{\mathrm{cert}} \le \tau\right\}. \tag{55}$$

Because the certified gap upper bounds the true optimality gap,

$$k_i^{\mathrm{cert}}(\tau) \ge k_i^{\mathrm{oracle}}(\tau). \tag{56}$$

We measure the conservatism of the certificate through the *certification delay*

$$D_i(\tau) := \frac{k_i^{\mathrm{cert}}(\tau)}{k_i^{\mathrm{oracle}}(\tau)} \ge 1. \tag{57}$$

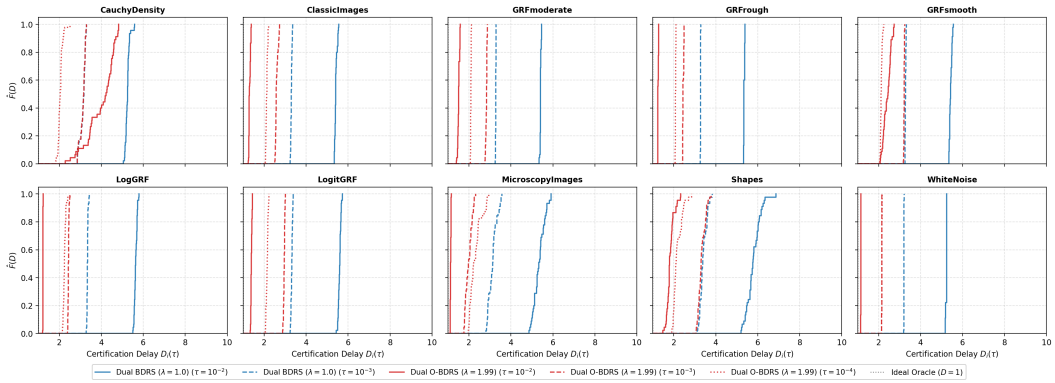


Figure 4: Classwise certification-delay distributions for Dual BDRS ($\lambda = 1$) and Dual O-BDRS ($\lambda = 1.99$) on DOTmark. Each panel contains 45 image pairs at resolution 64×64. Solid and dashed curves correspond to $\tau = 10^{-2}$ and $\tau = 10^{-3}$, respectively; dotted red curves additionally show O-BDRS at $\tau = 10^{-4}$, with BDRS omitted as certification was not obtained within the iteration budget. The horizontal axis is $D_i(\tau) = k_i^{\mathrm{cert}}(\tau)/k_i^{\mathrm{oracle}}(\tau)$ and the vertical axis is its empirical cumulative distribution. The reference $D_i = 1$ denotes agreement with the oracle stopping iteration. Curves further to the left indicate less additional iteration cost relative to each method's own oracle.

A value of $D_i(\tau) = 1$ means that the certificate identifies the first iterate satisfying the requested accuracy, while $D_i(\tau) = 2$ means that certification requires twice as many iterations as an oracle stopping rule. We report the empirical cumulative distribution of $D_i(\tau)$ over the DOTmark instances for $\tau \in \{10^{-2}, 10^{-3}, 10^{-4}\}$. The iteration budget used here is 10^5. If the oracle reaches the requested tolerance but the certificate does not do so within the terminal budget, we set $D_i(\tau) = +\infty$.

Results. We supplement the analysis in the main text with Figure 4, where it shows how the behavior of the delay certification varies across image classes. The reduction in delay at $\tau = 10^{-2}$ is visible throughout the benchmark, whereas the $\tau = 10^{-3}$ distributions are much closer for some classes, including GRF Smooth and Shapes. These results show that the delay depends on the method, tolerance, and problem class.

G.4 Large-scale color transfer

The preceding experiments examine the update structure and certification behavior on problems for which a reference optimum is available. We now consider a large scale problem in which storing the transport plan becomes the main obstacle. To this end, pixel-level color transfer allows us to assess two practical consequences of the dual formulation: the time required to reach a prescribed certified accuracy and the problem sizes that become accessible with a linear memory implementation.

The matched temperature results in Appendix G.2 motivate using Dual BDRS with $\lambda = 1$ as the primary solver while tuning its initial temperature for this application. This evaluates the performance of the tuned solver, whereas the fixed η DOTmark study isolates the effect of overrelaxation and targets the general case where tuning is unnecessary. Our initial experiments compares against MDOT-TNT (Kemertas et al., 2025a) at a common approximate transport cost. Following that, we run large-scale OT on 4238×2365 resolution images, comprising a total of 10^7 pixels, establishing a new state-of-the-art for large-scale OT.

Pixel-level OT formulation. Let I_s denote the source image whose spatial arrangement is retained and I_p the palette image whose colors are transferred to I_s. After flattening the images, let $x_i = \mathrm{RGB}(I_s, i) \in [0, 255]^3$ for $i = 1, \ldots, N$, and $y_j = \mathrm{RGB}(I_p, j) \in [0, 255]^3$ for $j = 1, \ldots, M$. Each pixel has uniform mass, so

$$r_i = \frac{1}{N}, \qquad c_j = \frac{1}{M}, \qquad C_{ij} = \frac{\|x_i - y_j\|_2^2}{\max_{p,q} \|x_p - y_q\|_2^2},$$

and C_{ij} is normalized by the maximum cost, thus $C_{ij} \in [0, 1]$. Every pixel in each input image is retained as a separate atom, including pixels with repeated RGB values. We do not aggregate equal colors or replace pixels with quantized color representatives as we want to evaluate the solver without compression. At each tested resolution, the solver treats the complete source and palette images as one global OT problem with NM implicit transport entries.

Implicit implementation and output. We use log-domain Dual BDRS and evaluate pairwise reductions lazily with PyKeOps. The cost matrix, kernel, and transport plan are never materialized, including during certificate evaluation and image reconstruction. The large-scale implementation uses `float32` on a single NVIDIA L40S GPU. We log all primal upper and dual lower bounds to show the trajectory of the algorithm but retain the state with the lowest relative duality gap for barycentric projection. The transferred image is obtained by barycentric projection of the chosen implicit output plan X:

$$\widehat{y}_i = \frac{\sum_{j=1}^{M} X_{ij} y_j}{\sum_{j=1}^{M} X_{ij}}.$$

The numerator and denominator are evaluated using lazy pairwise reductions. We then clip the RGB values to $[0, 255]$ and reshape them to the source image dimensions. In our experiments, both row and column residual have extremely low errors and thus we do not round the transport plan before barycentric projection. Therefore, the visual output is *not* the transformation from a feasible transport plan, but a means for us to visualize the optimality of the transport plan. The primal-dual guarantee applies to the transport cost of the repaired plan, if feasible solutions are required.

η tuning. We first use 512×512 instances to select the initial temperature. Specifically, we started with $\eta = 1$, and gradually reduced the temperature until we observe oscillations in the primal-dual gap. We found that $\eta = 0.1$ is approximately the limit where oscillations start to occur. In the spirit of implicit annealing in BDRS, we did not further tune η and $\eta = 1$ gives a duality gap curve that resembles a monotonically decreasing curve. At $\eta < 0.1$, we start to observe some oscillations in the duality gap. Despite that, lower η values can still arrive at a given optimality gap with fewer iterations.

Comparison with existing state-of-the-art. We compare BDRS with the released low-memory implementation of MDOT-TNT (Kemertas et al., 2025a) on the same pair of 1024×1024 images, using all pixels and the same normalized squared Euclidean cost in RGB space. Kemertas et al. (2025a) report approximately 10 hours for their `float64` PyKeOps implementation at a final inverse temperature of $\gamma = 2^{10}$, without reporting the corresponding linear transport cost. We ran the provided code in `float64` with $\gamma = 2^{10}$ and obtained a solution within 7 hours on a single L40S GPU. We also ran the provided code in `float32` with $\gamma = 2^{10}$ and obtained a solution within 3.5 hours on a single L40S GPU. The final costs obtained on both runs was identical. In the following experiments, we use `float32` to reflect operational realities where speed is critical.

To this end, we evaluate the baseline directly, with $\gamma = 2^5$, followed by $\gamma = 2^{10}$. We choose these two settings to represent different computational budgets: $\gamma = 2^5$ targets scenarios requiring a solution quickly, while $\gamma = 2^{10}$ targets those permitting longer computations, such as overnight runs. With $\gamma = 2^5$, MDOT-TNT takes 1685.76s of solver time and yields an unrepaired transport cost of 0.072147. Our `float32` BDRS implementation, with $\eta = 0.1$ and $\lambda = 1$, completes 10 iterations in 13.55s of solver time, or 15.71s including setup and certificate evaluation. The primal upper bound at iteration 10 is 0.070403, which is less than the unrepaired transport cost of MDOT-TNT. We recognize that the baseline's unreported marginal residuals limit the interpretation of this cost comparison, and therefore focus on the visual results in Figure 5, which clearly show BDRS with 10 iterations yielding a sharper result than MDOT-TNT.

We then compare BDRS with MDOT-TNT at $\gamma = 2^{10}$. We first run BDRS for 1,000 iterations to assess both optimization accuracy and marginal feasibility. We ran both the forward transfer and the backward transfer and Table 3 summarizes the bounds and marginal errors. For the unrepaired implicit plan used in barycentric reconstruction, the row and column marginal ℓ_1 errors are both below 7.34×10^{-5}. Each source and target pixel has a prescribed mass of $1/2^{20} \approx 9.54 \times 10^{-7}$. The maximum absolute row and column marginal errors are 2.06×10^{-10} and 7.87×10^{-10}, respectively. Dividing these errors by the prescribed pixel mass gives maximum relative errors of 0.0216% and



(a) BDRS: 10 iterations **(b) MDOT-TNT** **(c) BDRS reference**

Figure 5: Visual comparison on the same 1024×1024 color-transfer problem. All panels show barycentric reconstructions using all source and target pixels, `float32`, and the same normalized squared Euclidean cost in RGB space. **(a)** BDRS after 10 iterations, requiring 13.55 s of solver time (15.71 s including setup and certificate evaluation). **(b)** MDOT-TNT at final inverse temperature $\gamma = 2^5$, requiring 1685.76 s of solver time. **(c)** BDRS after 1,000 iterations, used as a reference solution with a reported relative duality gap of 1.59%. BDRS uses $\eta = 0.1$ and $\lambda = 1$. Solver times exclude image reconstruction. Observe how BDRS with 10 iterations is closer to the reference, where the brown parts of the image are more pronounced, whereas MDOT-TNT is still generally green, an indication of a diffuse transport plan. On the 10 iterations comparison, BDRS provides a $107\times$ speedup over MDOT-TNT. Best viewed digitally.

Table 3: Accuracy, runtime, and memory of BDRS after 1,000 iterations for both 1024×1024 color transfer directions in `float32`. Marginal errors refer to the unrepaired plan; paired entries are ordered as (row, column). Relative marginal errors are measured against the prescribed pixel mass 2^{-20}. Runtime includes setup and certificate evaluation but excludes image reconstruction. Memory is the peak GPU allocation tracked by PyTorch, including reconstruction and validation, and excludes allocations outside PyTorch.

Statistic	Forward ($1 \to 2$)	Backward ($2 \to 1$)
Primal upper bound U	0.0496885	0.0496843
Dual lower bound L	0.0488973	0.0488973
Relative duality gap $(U - L)/U$	1.59%	1.58%
Marginal ℓ_1 error	$(2.94 \times 10^{-5}, 6.35 \times 10^{-5})$	$(2.54 \times 10^{-5}, 7.34 \times 10^{-5})$
Maximum absolute marginal error	$(2.06 \times 10^{-10}, 7.87 \times 10^{-10})$	$(2.06 \times 10^{-10}, 7.87 \times 10^{-10})$
Maximum relative marginal error	$(0.0216\%, 0.0826\%)$	$(0.0216\%, 0.0826\%)$
Runtime including setup and certification	1433.06 s	1421.04 s
Peak allocated GPU memory	156.13 MiB	156.13 MiB

0.0826%. Thus, every row and column mass deviates from its prescribed value by less than 0.083%, indicating that the unrepaired transport plan is approximately feasible in practice. BDRS requires only 156MB and takes approximately 24 minutes to run to achieve a visual similarity, shown in Figure 6, that is indistinguishable to the naked eye from what was reported by Kemertas et al. (2025a). The matched run on the same hardware of MDOT-TNT also confirms this result, where it ran for 3.5 hours and returned a repaired cost of 0.0507. BDRS thus achieves a $9\times$ speed up, while achieving a lower cost of 0.0497 than MDOT-TNT.

Log-Averaged Mirror Prox (LAMP) (Burns & Liang, 2026) also provides a linear-memory approach to OT. We attempted to run the authors' Julia implementation but encountered a GPU kernel compilation error (`InvalidIRError`) in our environment. Our best-effort PyTorch reimplementation was tractable for color transfer between 512×512 images, but not between 1024×1024 images in our setup. We therefore omit a quantitative comparison with LAMP. These implementation difficulties do not establish a scalability limit for LAMP. The absence of a direct comparison with a validated LAMP implementation remains a limitation of our evaluation.



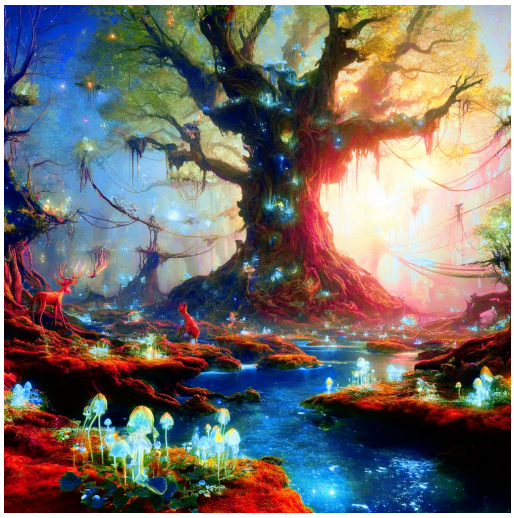
(a) Original Image A



(b) Original Image B



(c) Forward transfer (A to B).



(d) Backward transfer (B to A).

Figure 6: Bidirectional color transfer between two 1024×1024 images using BDRS. **(a, b)** Original images. **(c)** Image 1 transferred to the color distribution of image 2. **(d)** Image 2 transferred to the color distribution of image 1. Both runs use all source and target pixels and perform 1,000 iterations in `float32`, with $\eta = 0.1$ and $\lambda = 1$. The reported relative duality gaps are 1.59% and 1.58% for the forward and backward directions, respectively. Each direction ran in approximately 24 minutes with 156MB of memory. Transferred images are obtained by barycentric projection of the unrepaired transport plans.

Scaling pixel-level OT to ten million atoms per image. We further apply Dual BDRS to bidirectional color transfer between two 4238×2365 images. Each image contains $10{,}022{,}870$ pixels, yielding a single global OT problem with approximately 10^{14} implicit transport entries. We use $\eta = 0.01$, $\lambda = 1$, and `float32` pairwise computations on a single NVIDIA L40S GPU, evaluating the certificate every 20 iterations. Both directions complete 1,000 iterations in approximately 35 hours each, with a peak allocated GPU memory of $1.58\,\mathrm{GiB}$. The selected iterates, both at iteration 840, attain relative primal–dual gaps of 2.41% and 2.80% for the forward and backward transfers, respectively. Figure 3 in the main text shows the barycentric reconstructions from the unrepaired implicit plans, while Table 4 reports the bounds, marginal errors, runtimes, and memory usage. The cost matrix, kernel, and transport plan are never materialized.

Table 4: Accuracy, runtime, and memory of BDRS for both 4238×2365 color transfer directions, with $\eta = 0.01$, $\lambda = 1$, and `float32` pairwise computations. Images 1 and 2 are the canyon and outdoor images, respectively. Each run completes 1,000 iterations; reported accuracy corresponds to the selected iterate at iteration 840. Marginal errors refer to the unrepaired intermediate plan; paired entries are ordered as (row, column). Relative marginal errors are measured against the prescribed pixel mass $1/10{,}022{,}870$. Runtime covers the full run, including setup, certificate evaluation, and final verification, but excludes image reconstruction. Memory is the peak GPU allocation tracked by PyTorch, including reconstruction and validation, and excludes allocations outside PyTorch.

Statistic	Forward ($1 \to 2$)	Backward ($2 \to 1$)
Primal upper bound U	0.0101706	0.0102118
Dual lower bound L	0.00992558	0.00992557
Relative duality gap $(U - L)/U$	2.41%	2.80%
Marginal ℓ_1 error	$(1.16 \times 10^{-4},\ 3.29 \times 10^{-4})$	$(1.17 \times 10^{-4},\ 4.28 \times 10^{-4})$
Maximum absolute marginal error	$(4.68 \times 10^{-11},\ 1.50 \times 10^{-9})$	$(4.68 \times 10^{-11},\ 1.34 \times 10^{-9})$
Maximum relative marginal error	(0.0469%, 1.5023%)	(0.0469%, 1.3414%)
Runtime including setup and certification	125582.11 s	124820.21 s
Peak allocated GPU memory	1613.97 MiB	1613.97 MiB

Floating-point evaluation. The reported primal and dual bounds are floating-point evaluations of the exact arithmetic certificate. In the largest experiments, `float32` computations may introduce errors in normalization and certificate evaluation. The reported marginal residuals quantify mass-conservation errors, but do not bound the numerical errors in the certificate itself.

H Figures for Momentum and Kernel Ablation

In this section, we show all ten graphs for each class in the DOTmark dataset. We put all figures for the momentum and kernel ablation in a single section to improve readability.

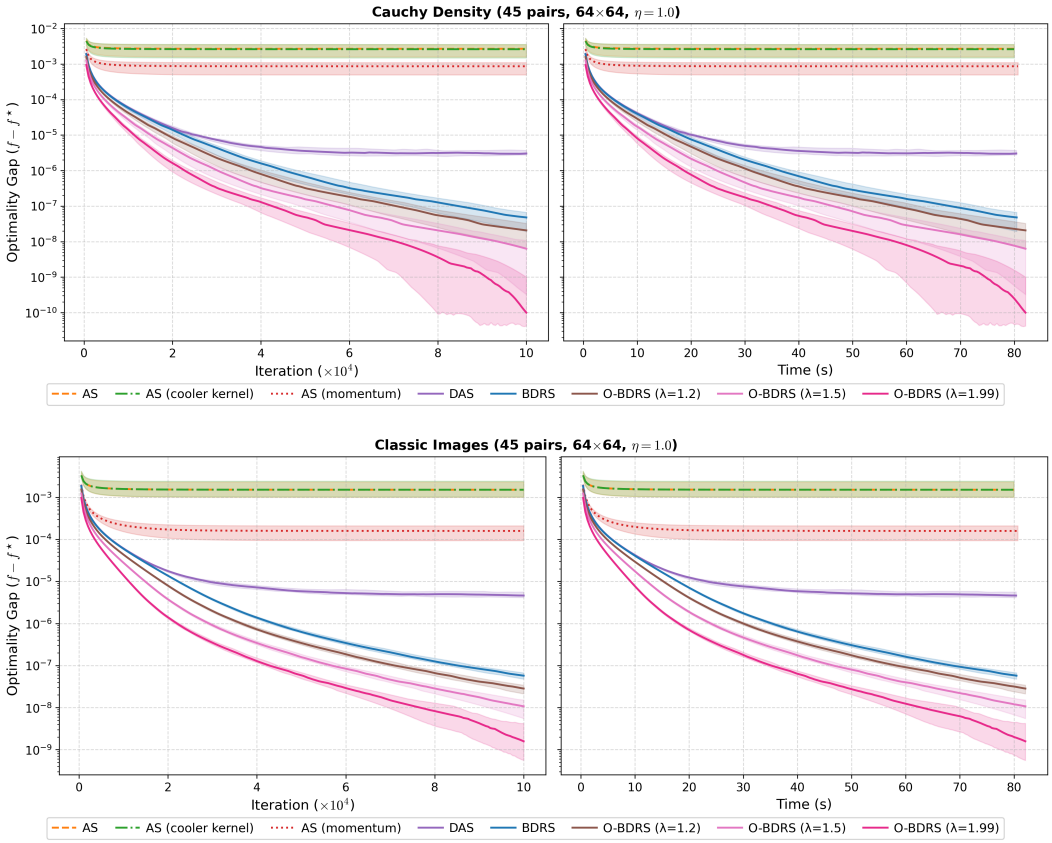


Figure 7: Momentum and kernel ablations on the Cauchy Density (top) and Classic images (bottom) classes of DOTmark. Each class contains 45 image pairs at resolution 64×64, with $\eta = 1$. The left panels show the transport cost difference $f - f^\star$ against iteration count, and the right panels show the same quantity against median solver time, excluding rounding. Here f is the evaluated transport cost and $f^\star$ is the reference value computed using POT's network simplex solver. Lines show medians across image pairs; shaded bands indicate the 25th–75th percentiles.

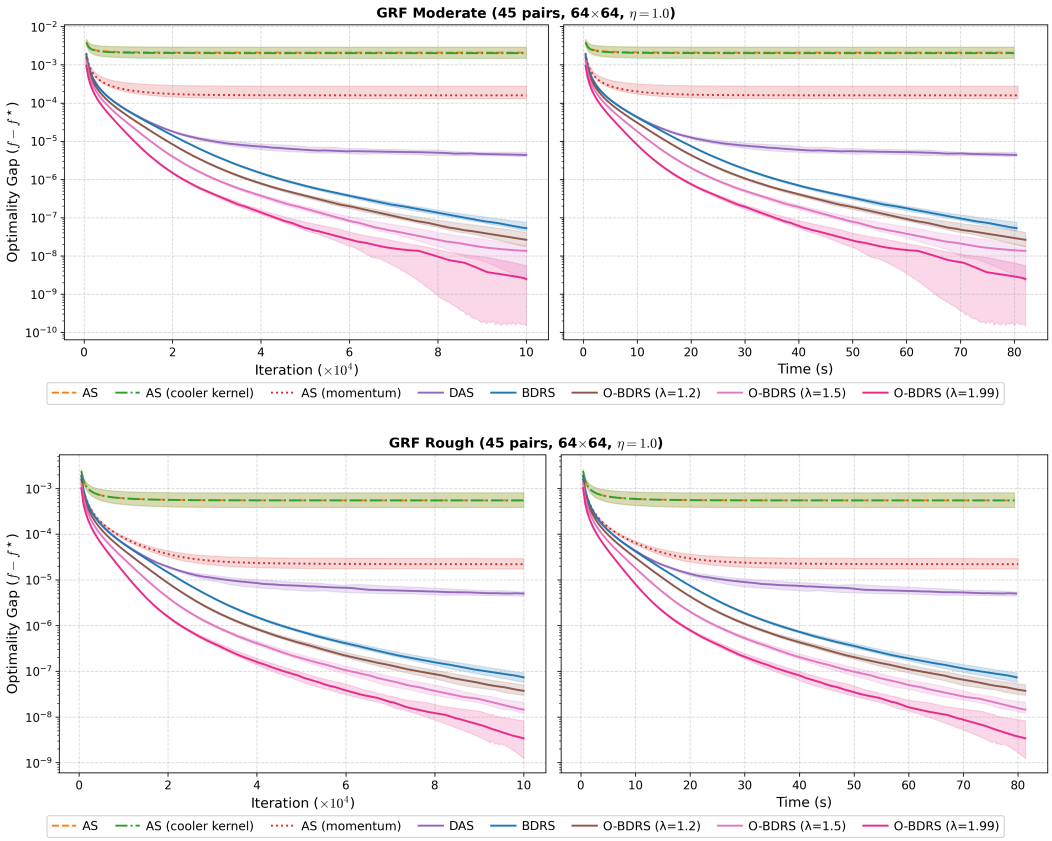


Figure 8: Momentum and kernel ablations on the GRF Moderate (top) and GRF Rough (bottom) classes of DOTmark. Each class contains 45 image pairs at resolution 64×64, with $\eta = 1$. The left panels show the transport cost difference $f - f^\star$ against iteration count, and the right panels show the same quantity against median solver time, excluding rounding. Here f is the evaluated transport cost and $f^\star$ is the reference value computed using POT's network simplex solver. Lines show medians across image pairs; shaded bands indicate the 25th–75th percentiles.

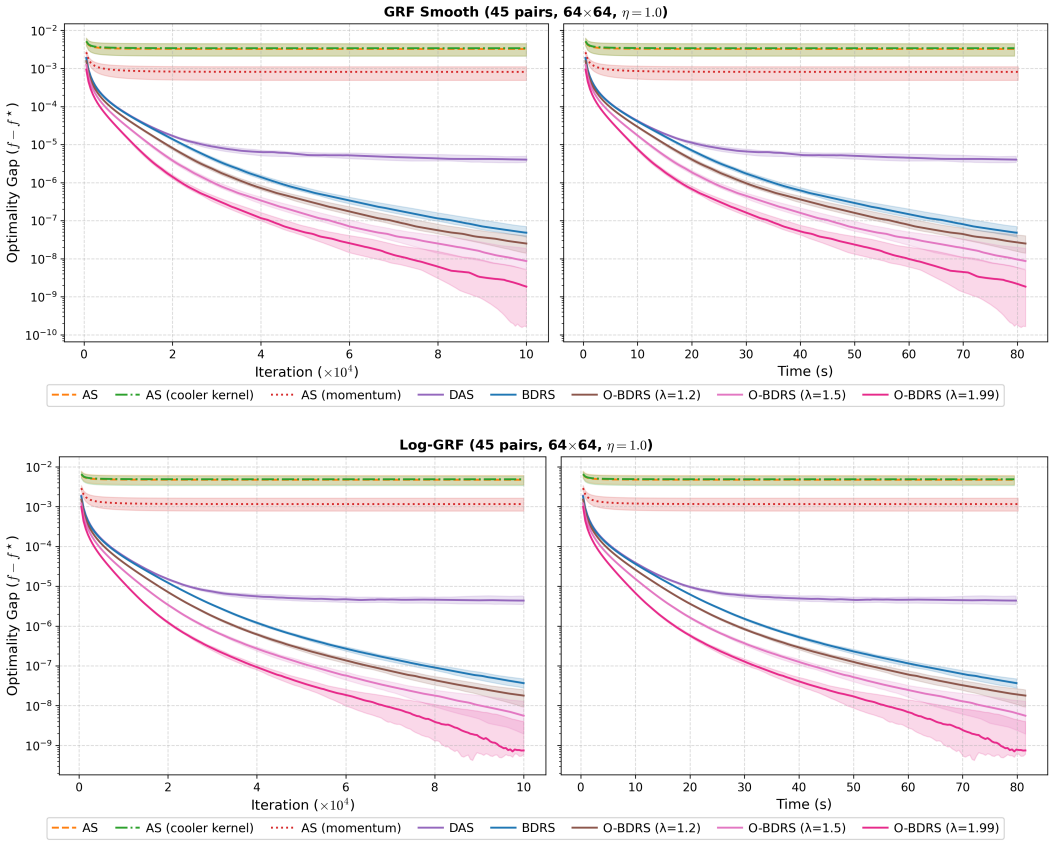


Figure 9: Momentum and kernel ablations on the GRF Smooth (top) and Log-GRF (bottom) classes of DOTmark. Each class contains 45 image pairs at resolution 64×64, with $\eta = 1$. The left panels show the transport cost difference $f - f^\star$ against iteration count, and the right panels show the same quantity against median solver time, excluding rounding. Here f is the evaluated transport cost and $f^\star$ is the reference value computed using POT's network simplex solver. Lines show medians across image pairs; shaded bands indicate the 25th–75th percentiles.

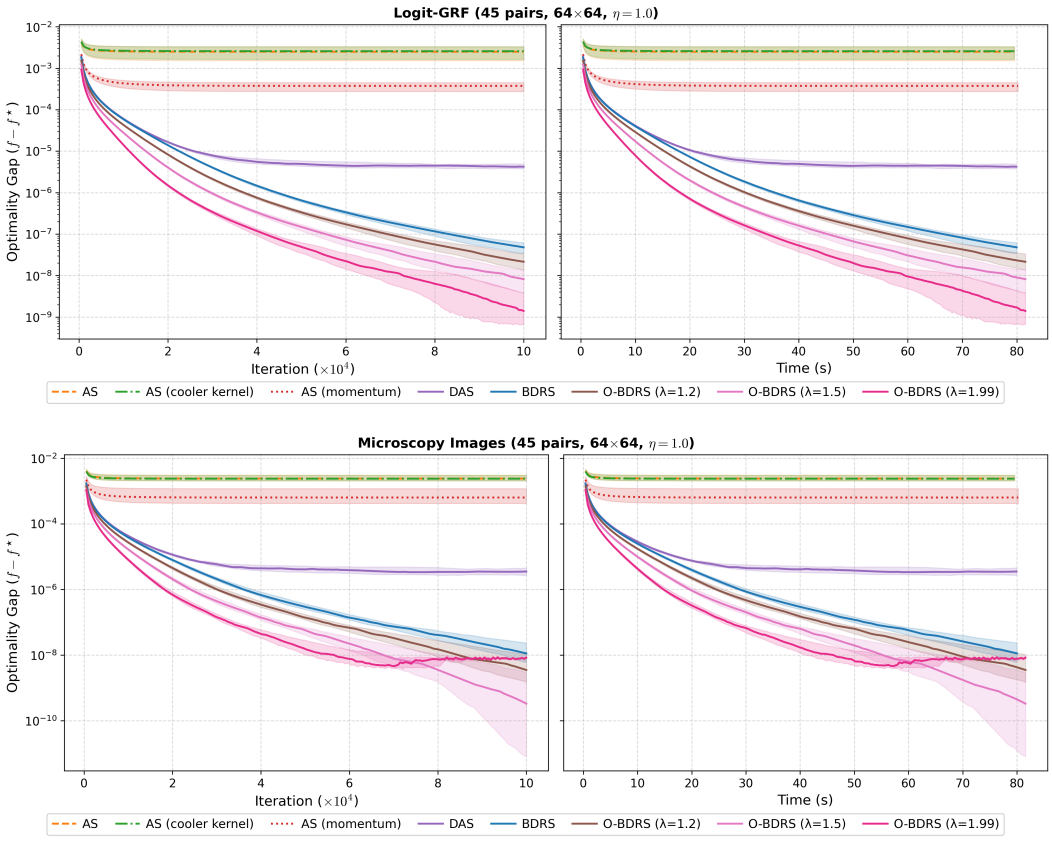


Figure 10: Momentum and kernel ablations on the Logit-GRF (top) and Microscopy Images (bottom) classes of DOTmark. Each class contains 45 image pairs at resolution 64×64, with $\eta = 1$. The left panels show the transport cost difference $f - f^\star$ against iteration count, and the right panels show the same quantity against median solver time, excluding rounding. Here f is the evaluated transport cost and $f^\star$ is the reference value computed using POT's network simplex solver. Lines show medians across image pairs; shaded bands indicate the 25th–75th percentiles. Here, we observe that O-BDRS has a performance degradation on Microscopy Images as the number of iterations increase. We suspect that this late-iteration degradation reflects numerical sensitivity at low effective temperatures, potentially exacerbated by the 10^{-15} mass assigned to zero-mass pixels, which are common in Microscopy Images and Shapes, but not the other classes.

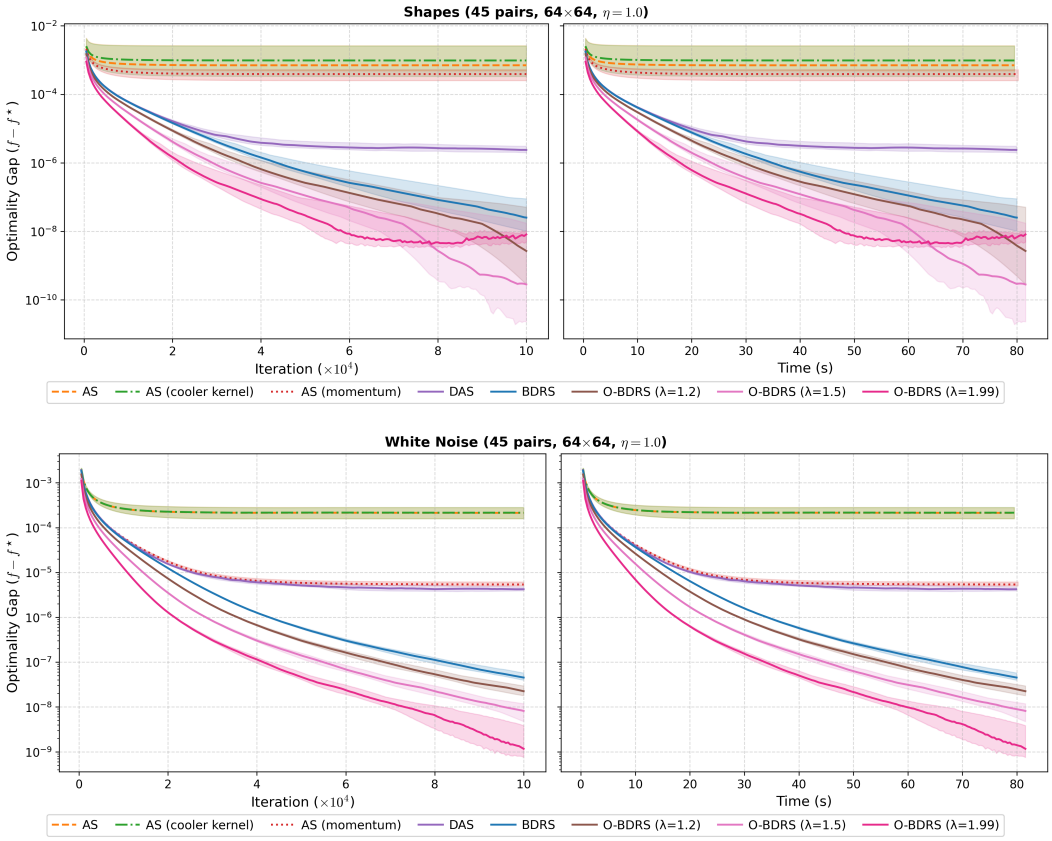


Figure 11: Momentum and kernel ablations on the Shapes (top) and White Noise (bottom) classes of DOTmark. Each class contains 45 image pairs at resolution 64×64, with $\eta = 1$. The left panels show the transport cost difference $f - f^\star$ against iteration count, and the right panels show the same quantity against median solver time, excluding rounding. Here f is the evaluated transport cost and $f^\star$ is the reference value computed using POT's network simplex solver. Lines show medians across image pairs; shaded bands indicate the 25th–75th percentiles. Here, we observe that O-BDRS has a performance degradation on Shapes as the number of iterations increase. We suspect that this late-iteration degradation reflects numerical sensitivity at low effective temperatures, potentially exacerbated by the 10^{-15} mass assigned to zero-mass pixels, which are common in Microscopy Images and Shapes, but not the other classes.