ESAIM: M2AN 59 (2025) 1239–1270
https://doi.org/10.1051/m2an/2025022

ESAIM: Mathematical Modelling and Numerical Analysis
www.esaim-m2an.org

FLOW UPDATES FOR DOMAIN DECOMPOSITION OF ENTROPIC OPTIMAL TRANSPORT

Ismael Medina and Bernhard Schmitzer*

Abstract. Domain decomposition has been shown to be a computationally efficient distributed method for solving large scale entropic optimal transport problems. However, a naive implementation of the algorithm can freeze in the limit of very fine partition cells (*i.e.* it asymptotically becomes stationary and does not find the global minimizer), since information can only travel slowly between cells. In practice this can be avoided by a coarse-to-fine multiscale scheme. In this article we introduce flow updates as an alternative approach. Flow updates can be interpreted as a variant of the celebrated algorithm by Angenent, Haker, and Tannenbaum, and can be combined canonically with domain decomposition. We prove convergence to the global minimizer and provide a formal discussion of its continuity limit. We give a numerical comparison with naive and multiscale domain decomposition, and show that the flow updates prevent freezing in the regime of very many cells. While the multiscale scheme is observed to be faster than the hybrid approach in general, the latter could be a viable alternative in cases where a good initial coupling is available. Our numerical experiments are based on a novel GPU implementation of domain decomposition that we describe in the appendix.

Mathematics Subject Classification. 49M27, 49M29, 65B99.

Received May 16, 2024. Accepted March 25, 2025.

1. Introduction

1.1. Motivation

(Computational) optimal transport. Optimal transport is a fundamental optimization problem with profound connections to various branches of mathematics, and applications in image analysis and machine learning. Given two probability measures μ and ν on spaces X and Y and a measurable cost function $c : X \times Y \to \mathbb{R}$, the optimal transport problem consists in finding the optimal probability measure π in the set of transport plans $\Pi(\mu, \nu)$ – probability measures on $X \times Y$ with marginals μ and ν – that solves

$$\inf_{\pi \in \Pi(\mu,\nu)} \int_{X \times Y} c(x, y) \, \mathrm{d}\pi(x, y). \tag{1.1}$$

Thorough expositions can be found in [32, 36]. The field has seen immense progress in the development of numerical algorithms, such as solvers for the Monge–Ampère equation [4], semi-discrete methods [25, 28], and

Keywords and phrases. Optimal transport, domain decomposition, min-cost flow.

Göttingen University, Institute for Computer Science, Goldschmidtstraße 7, 37077 Göttingen, Germany.
*Corresponding author: schmitzer@cs.uni-goettingen.de

multiscale methods [29, 33]. An introduction to computational optimal transport, an overview on numerical methods, and applications can be found in [31].

An important variant of the above problem is to add entropic regularization of the transport plan, *i.e.* solving

$$\inf_{\pi\in\Pi(\mu,\nu)} \int_{X\times Y} c(x,y)\,\mathrm{d}\pi(x,y) + \varepsilon\,\mathrm{KL}(\pi|\mu\otimes\nu) \tag{1.2}$$

where KL is the Kullback–Leibler divergence (see (2.1) for a definition) and $\varepsilon > 0$ is a positive regularization parameter.

This problem has very convenient analytical and computational advantages with respect to (1.1): it is strictly convex, differentiable with respect to the marginal distributions, and comes with an efficient, relatively simple, and GPU friendly numerical solver: the Sinkhorn algorithm, see for instance [17, 19, 31] and references therein. A mature library for GPU implementation is presented in [20, 21]. Exemplary references for the analysis of the limit $\varepsilon \to 0$ are [14, 16, 27]. Entropic regularization is also useful for the analysis of stochastic processes and for stochastic optimal control, see for instance [2, 15, 26].

Domain decomposition for optimal transport. Large optimal transport problems can be addressed with domain decomposition. Following [3, 7], the strategy of the algorithm is the following: let $\mathcal{J}_A$ and $\mathcal{J}_B$ be two partitions of X, which need to overlap in a suitable sense. Then, given an initial feasible coupling $\pi^0 \in \Pi(\mu,\nu)$, optimize its restriction to each set of the form $X_J \times Y$, where X_J are the subdomains in the partition $\mathcal{J}_A$, leaving the marginals on each of the restrictions fixed. That is, defining π_J as the restriction of π^0 to $X_J \times Y$, and μ_J and ν_J respectively the X- and Y-marginals of π_J, the subplan π_J is replaced by the optimizer of

$$\min_{\pi\in\Pi(\mu_J,\nu_J)} \int_{X_J\times Y} c\,\mathrm{d}\pi + \varepsilon\ \mathrm{KL}(\pi|\mu_J\otimes\nu).$$

This can be done independently for each subdomain in the partition, so the algorithm can be easily parallelized. Then repeat this procedure on partition $\mathcal{J}_B$, and keep alternating between partitions until convergence.

Convergence of domain decomposition for unregularized optimal transport (1.1) with the squared Eulidean distance cost is shown in [3, 8], under different assumptions on the overlap between partitions. For entropy-regularized optimal transport, Bonafini and Schmitzer [7] shows linear convergence under minimal assumptions on the cost, marginal and partitions. Besides, Bonafini and Schmitzer [7] proposes an efficient, parallel, multiscale implementation for two-dimensional grids that outperforms a state-of-the-art sparse Sinkhorn solver even with a single worker. Indeed, while the concept of domain decomposition (and the flow updates introduced in this article) is in principle applicable to unstructured point clouds in any dimension, the method will be most practical on low-dimensional grids, where an efficient partition structure is readily available.

In [9] we studied the asymptotic behavior of domain decomposition in the limit of small subdomains, with the motivation of providing an analogous analysis to what [5] obtained for the Sinkhorn algorithm. Let $X = [0,1]^d$, let $(\pi^{n,k})_k$ be the domain decomposition iterates where the subdomains are hypercubes of edge length $1/n$, and define $\boldsymbol{\pi}^n(t) := \pi^{n,\lfloor nt\rfloor}$, that is, each iterate is assigned a time step of $1/n$. Then under suitable assumptions, the trajectories $(\boldsymbol{\pi}^n)_n$ converge to a limit trajectory, which solves the horizontal continuity equation

$$\partial_t \boldsymbol{\pi}_t + \mathrm{div}_X\, \boldsymbol{\omega}_t = 0 \qquad \text{for } t > 0 \text{ and} \qquad \boldsymbol{\pi}_{t=0} = \pi_{\mathrm{init}}$$

where the horizontal momentum field $\boldsymbol{\omega}_t$ describes how mass moves in X, while retaining the same position in Y and div_X is the corresponding divergence operator. $\boldsymbol{\omega}_t$ can be obtained from an asymptotic version of the domain decomposition subdomain problem. As opposed to [5], however, in the asymptotic limit not all initializations evolve to the optimal coupling but may become stationary in sub-optimal states. This is referred to as *freezing*.

Freezing in domain decomposition. Even though under the assumptions of [3], [7] or [8] every sequence of iterates $(\pi^{n,k})_k$ must converge to the optimal coupling, the rate may deteriorate with increasing number of

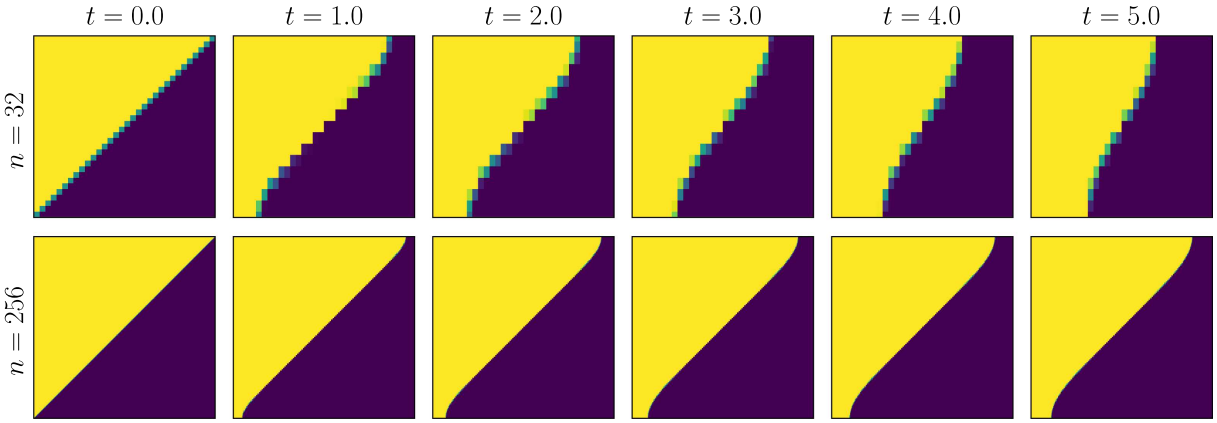

FIGURE 1. An example of the freezing behaviour in domain decomposition. For $\mu = \mathcal{L} \llcorner [-1/2, +1/2]^2$, $\nu = \frac{1}{2}[\delta_{(-1/2,0)} + \delta_{(+1/2,0)}]$, we visualize the couplings $\pi^{n,k}$ by colouring the domain decomposition cells that are mapped to $(-1/2, 0)$ in yellow and those mapped to $(+1/2, 0)$ in dark blue; non-deterministic assignments feature an intermediate color. We show two domain decomposition trajectories at different resolutions, for an initialization that features a global rotation with respect to the optimal configuration, in this case corresponding to a vertical interface. Both trajectories evolve towards the optimal coupling, but the convergence rate deteriorates drastically as the resolution increases.

subdomains even in smooth problem instances. A striking example of this (see Fig. 1 for an illustration) is when the initial coupling is given by $\pi^0 = (\mathrm{id}, T)_{\#}\mu$ for some non-optimal transport map T, such that T has non-zero curl, *e.g.* when T contains some *non-local rotation*. Indeed, the number of required iterations for domain decomposition to undo such a non-local rotation increases with the number of partition cells, the intuitive reason being that information can only travel by one cell per iteration. Based on the experiments in [7], this freezing behaviour does not seem to be an issue when domain decomposition is used in combination with a multiscale scheme, as with the latter, macroscopic curl is removed from the initial coupling efficiently at a coarse resolution scale and thus at the finer levels essentially only local updates to the coupling are necessary. However, as of now, there is no theoretical description of this behaviour.

Curl in optimal transport and the AHT scheme. The relation between curl and optimal transport is well known since the seminal work of Brenier [10], who showed that for μ absolutely continuous with respect to the Lebesgue measure $\mathcal{L}$ (in fact, a slightly weaker but more technical condition is shown to be sufficient in [10]), any mass rearrangement map T can be uniquely decomposed as $T = \nabla\phi \circ s$, for some convex function ϕ and a μ-measure preserving map s, *i.e.* $s_{\#}\mu = \mu$, where $s_{\#}\mu$ denotes the pushforward of measure μ by the map s. $\nabla\phi$ is then the optimal transport map from μ to $T_{\#}\mu$ with respect to the squared Euclidean distance.

Based on this insight, Angenent, Haker, and Tannenbaum proposed an algorithm [1] (also known as AHT scheme) for computing the optimal map $\nabla\phi$. Given μ and ν, starting from some initial feasible transport map T_0 with $T_{0\#}\mu = \nu$, one (formally) constructs a flow of maps

$$T_t = T_0 \circ s_t^{-1} \qquad \text{with} \qquad s_0 = \mathrm{id}, \qquad \partial_t s_t = v_t \circ s_t, \qquad \mathrm{div}(v_t \cdot \mu) = 0 \tag{1.3}$$

where v_t is a Eulerian velocity field and the divergence constraint ensures that $s_{t\#}\mu = \mu$ and therefore $T_{t\#}\mu = \nu$ at all times t. Among admissible velocity fields, v_t is chosen such that the transport cost

$$E(T) := \int_X c(x, T(x))\, \mathrm{d}\mu(x) \tag{1.4}$$

is decreasing in time. Formally one finds for T_t as in (1.3) that

$$\partial_t E(T_t) = \int_X \langle \nabla_X c(x, T_t(x)), v_t(x) \rangle \, \mathrm{d}\mu(x).$$

The notion of steepest descent then depends on the choice of a metric on the space of velocity fields. For instance, the steepest descent with respect to the $\mathrm{L}^2(\mu)$-metric would be given by setting v_t to the projection of the vector field $x \mapsto -\nabla_X c(x, T_t(x))$ onto the subspace satisfying $\mathrm{div}(v_t \cdot \mu) = 0$ in $\mathrm{L}^2(\mu)$. In [1] a slightly different choice is proposed: One reparametrizes $v_t(x) = u_t(x)/\mu(x)$ (where $\mu(x)$ denotes the Lebesgue density of μ at x). u_t is then set to be the $\mathrm{L}^2(\mathcal{L})$-projection of $x \mapsto -\nabla_X c(x, T_t(x))$ onto the subspace satisfying $\mathrm{div}(u_t \cdot \mathcal{L}) = 0$. This can be computed efficiently by solving a standard Poisson equation.

When c is the squared Euclidean distance, at a critical point of E the map T_t must be a gradient of some potential ϕ, and if the Jacobian of T_t happens to be positive semi-definite, ϕ will be convex, *i.e.* T_t must be the sought-after optimal transport map.

Despite its elegance, the AHT scheme is in general not guaranteed to be well-posed (*i.e.* the initial value problem (1.3) where v_t is some L^2-projection of $-\nabla_X c(\mathrm{id}, T_t)$ is not necessarily well-posed) or to converge to the optimal transport map, because for stationary $T_t = \nabla\phi$, ϕ will not necessarily be convex (except in particular cases and under strong assumptions, see [32], Prop. 6.3, [11], Sect. 1.3). Optimization based on re-arrangements s_t will also not be applicable to the case of general transport plans $\pi \in \Pi(\mu, \nu)$ that are not concentrated on graphs of maps, as for instance in entropic transport.

However, the AHT scheme provides a powerful intuition as well as appealing properties, such as the score being non-increasing, with the rate of decrement of the score related to the curl component of the current transport map.

1.2. Contributions and outline

A hybrid scheme. We observe that the strengths of domain decomposition and the AHT scheme are complementary. On the one hand, (entropy regularized) domain decomposition converges to the globally optimal plan but is slow at resolving curl. On the other hand, the AHT scheme removes curl quickly, but is not guaranteed to be well-posed or to achieve optimality. In this article we present a hybrid of the two methods that will always converge to the global minimizer and is efficient at removing curl.

Several issues must be resolved for this combination. In particular the AHT scheme needs to be discretized consistently at the level of the domain decomposition partitions X_J, and it must be adapted to work with entropic plans from domain decomposition, whereas the original AHT scheme was based on transport maps for unregularized transport.

Flow updates. In Section 3 we present a hybrid scheme for (entropic) optimal transport where domain decomposition updates are intertwined with *flow updates*, which can be interpreted as a min-cost flow adaptation of the AHT scheme. Flow updates and their basic properties (preserving the marginals and decreasing the objective) are given in Sections 3.1 and 3.2. Combination with the domain decomposition updates and proof of convergence to the optimal coupling are given in Section 3.3.

Interpretation of flow updates as an L^∞-version of the AHT scheme. In Section 4 we show that flow updates can formally be interpreted as a discretized L^∞-version of the AHT scheme, *i.e.* in (1.4) one chooses the admissible v_t that yields the largest decrement, subject to the constraint $|v_t(x)|_\infty \leq 1$ for all x where $|\cdot|_\infty$ denotes the supremum norm on $\mathbb{R}^d$. The domain decomposition iterates then deal with general non-deterministic transport plans π and with discretization artefacts. Our discussion remains purely formal to provide an intuitive interpretation. We anticipate that a rigorous derivation will be non-trivial and far beyond the scope of this article.

Numerical experiments. The ability of the hybrid scheme to overcome the freezing behaviour is illustrated in Figure 2 and examined experimentally in more detail in Section 5. We demonstrate that the hybrid scheme

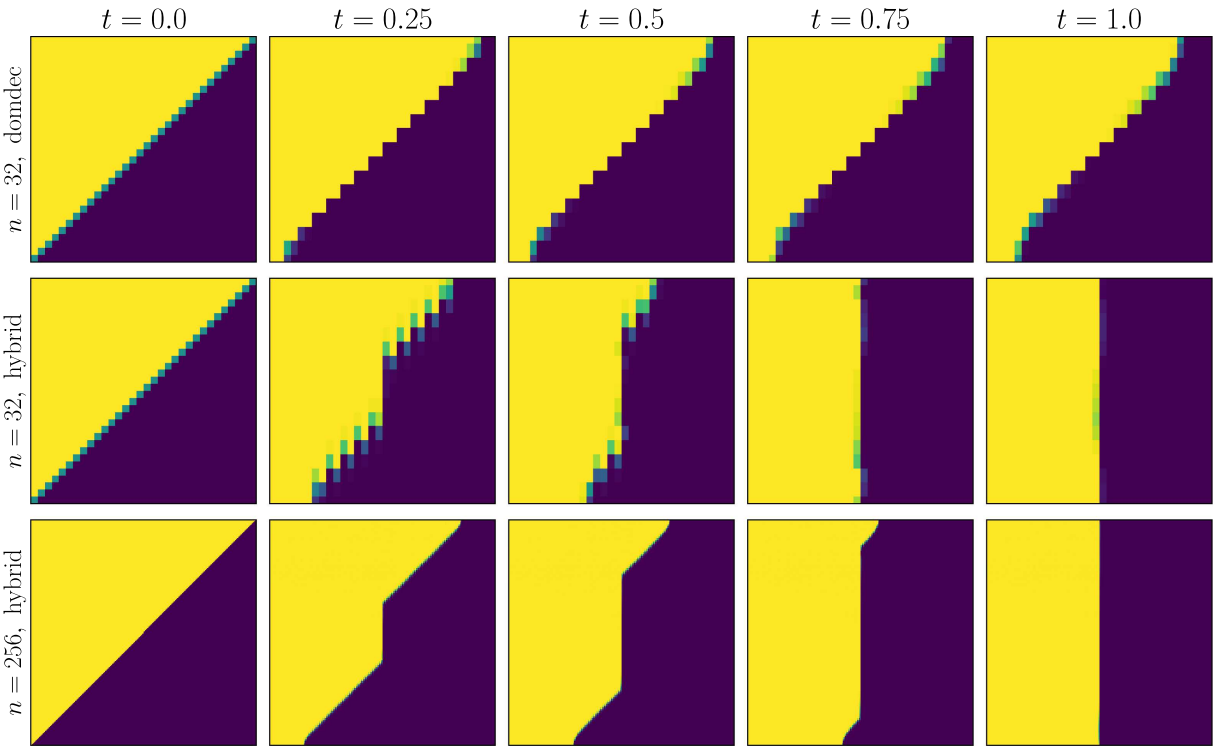

FIGURE 2. Hybrid scheme overcomes freezing: The top row shows the same sequence of iterates as Figure 1, now from $t = 0$ until $t = 1$, showcasing the freezing. The center row shows the corresponding trajectory for the hybrid scheme iterates: after the same number of iterations the iterates get much closer to the global optimizer, leaving only small local perturbations of the optimal assignment that can quickly be resolved by a few additional domain decomposition iterations. Finally, the bottom row shows a higher resolution example, converging in approximately the same time as the one with lower resolution, demonstrating the resilience to freezing.

clearly improves upon domain decomposition at a single scale. On the other hand we observe that multiscale domain decomposition does not suffer from freezing and is in general faster than the hybrid scheme. However, the hybrid scheme could be a viable alternative in cases where a good initial coupling is known.

A GPU implementation. Our numerical experiments are based on a novel GPU implementation of domain decomposition that we describe in Appendix A.

Background on entropic optimal transport and domain decomposition is briefly recalled in Section 2.

2. Background

2.1. Setting and notation

- Let $\mathbb{R}_+ = [0, \infty)$.
- Let X and Y be compact metric spaces. Typically, X and Y are convex, compact subsets of $\mathbb{R}^d$, but our scheme can be applied to more general spaces. We assume compactness to avoid overly technical arguments while covering the numerically relevant setting.

- For a compact metric space Z we denote by $\mathcal{C}(Z)$ the continuous function on Z and by $\mathcal{M}(Z)$ the set of signed Radon measures over Z (and finite by compactness of Z). We identify the latter with the topological dual of the former, and recall that integration against test functions in $\mathcal{C}(Z)$ induces the weak* topology on $\mathcal{M}(Z)$. The subset of non-negative and probability measures are $\mathcal{M}_+(Z)$ and $\mathcal{M}_1(Z)$. The Radon norm of $\mu \in \mathcal{M}(Z)$ is denoted by $\|\mu\|_{\mathcal{M}(Z)}$, and it holds $\|\mu\|_{\mathcal{M}(Z)} = \mu(Z)$ for $\mu \in \mathcal{M}_+(Z)$. If there is no ambiguity we will simply write $\|\mu\|$.
- For $\mu, \nu \in \mathcal{M}(Z)$ we write $\mu \ll \nu$ to indicate that μ is absolutely continuous with respect to ν.
- For μ in $\mathcal{M}_+(Z)$ and $S \subset Z$ measurable the restriction of μ to S is denoted by $\mu \llcorner S$.
- We denote by $\langle \phi, \rho \rangle$ the integration of a measurable function ϕ against a measure ρ.
- The maps P_X and P_Y denote the projections of measures on $X \times Y$ to its marginals, *i.e.*

$$(\mathrm{P}_X \pi)(S_X) := \pi(S_X \times Y) \qquad \text{and} \qquad (\mathrm{P}_Y \pi)(S_Y) := \pi(X \times S_Y)$$

  for $\pi \in \mathcal{M}_+(X \times Y)$, $S_X \subset X$, $S_Y \subset Y$ measurable.
- For a compact metric space Z and measures $\mu \in \mathcal{M}(Z)$, $\nu \in \mathcal{M}_+(Z)$, the *Kullback–Leibler divergence* of μ with respect to ν is given by

$$\mathrm{KL}(\mu|\nu) := \begin{cases} \int_Z \varphi\left(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\right) \mathrm{d}\nu & \text{if } \mu \ll \nu,\ \mu \geq 0, \\ +\infty & \text{else,} \end{cases} \quad \text{with } \varphi(s) := \begin{cases} s\log(s) - s + 1 & \text{if } s > 0, \\ 1 & \text{if } s = 0. \end{cases} \tag{2.1}$$

- For $p \in [1, \infty)$ we denote by $|v|_p$ the p-norm on $\mathbb{R}^d$ and $|v|_\infty := \max_{i=1,\dots,d} |v_i|$.

In Section 3.3 we will need the following lemma.

Lemma 2.1 (Continuity of the restriction operator, [6], Prop. 8.4.4)**.** *Let $(\rho_n)_n \subset \mathcal{M}_+(Z)$ converging weak* to ρ. Let $A \subset Z$ closed, and assume that $\rho(A) = \lim_{n\to\infty} \rho_n(A)$. Then $\rho \llcorner A = \lim_{n\to\infty} \rho_n \llcorner A$.*

2.2. Entropic optimal transport

For $\mu \in \mathcal{M}_+(X)$ and $\nu \in \mathcal{M}_+(Y)$ we define the set of *transport plans* or *couplings* between μ and ν as

$$\Pi(\mu, \nu) := \{\pi \in \mathcal{M}_+(X \times Y) \mid \mathrm{P}_X \pi = \mu, \quad \mathrm{P}_Y \pi = \nu\}. \tag{2.2}$$

Let c be a lower-semicontinuous, positive, bounded function on $X \times Y$ and $\varepsilon \in \mathbb{R}_+$. The entropic optimal transport problem between μ and ν, with cost c and regularization strength ε is given by:

$$\min_{\pi \in \Pi(\mu,\nu)} \langle c, \pi \rangle + \varepsilon\, \mathrm{KL}(\pi|\mu \otimes \nu). \tag{2.3}$$

We call the first term in (2.3) the *transport objective* and the second the *entropic objective*. For $\varepsilon = 0$ one recovers the Kantorovich optimal transport problem; existence of solutions is covered for example in Theorem 4.1 of [36]. Some properties of the minimizers of the entropic problem are given below:

Proposition 2.2 (Optimal entropic transport couplings)**.**

(i) *Equation (2.3) has a unique minimizer $\pi^* \in \Pi(\mu, \nu)$.*
(ii) *There exist measurable $\alpha^* : X \to \mathbb{R}$, $\beta^* : Y \to \mathbb{R}$ such that $\pi^* = e^{(\alpha^* \oplus \beta^* - c)/\varepsilon} \mu \otimes \nu$. The entropic potentials α^* and β^* are unique μ-a.e. and ν-a.e. up to a constant offsets, i.e. $(\alpha^*, \beta^*) \to (\alpha^* + C, \beta^* - C)$ for $C \in \mathbb{R}$.*
(iii) *A transport plan $\pi^* \in \Pi(\mu, \nu)$ of the form $\pi^* = e^{(\alpha^* \oplus \beta^* - c)/\varepsilon} \mu \otimes \nu$ with $\alpha^* \in \mathrm{L}^\infty(X, \mu)$, $\beta^* \in \mathrm{L}^\infty(Y, \nu)$ is optimal for (2.3).*

For proofs see Propositions 2.3 and 2.5 of [7] and references therein.

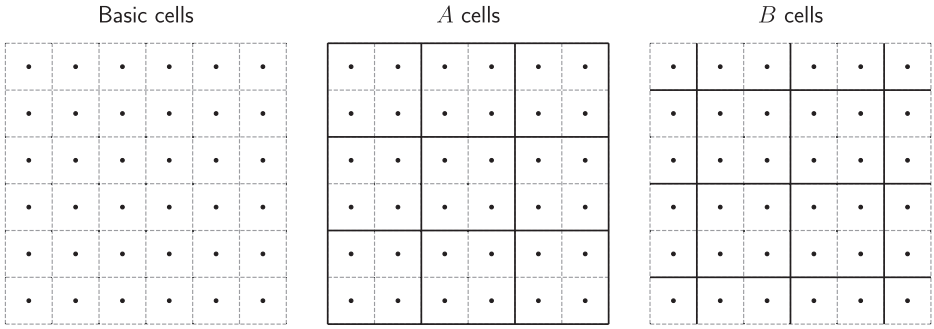

FIGURE 3. Prototypical choice of basic and composite partitions for $X = [0,1]^2$. For a resolution scale $n \in \mathbb{N}$, X is first partitioned into *basic cells* of size $1/n \times 1/n$. The restriction of μ to every basic cell is approximated by a Dirac delta in the center of the cell (indicated by the dots). The A and B partitions are constructed by 2×2 groups of basic cells, with an offset between A and B groups.

2.3. Domain decomposition for optimal transport

Domain decomposition for optimal transport was originally proposed in [3] and studied in [7] for entropic transport. We recall here the main definitions. From now on, μ, ν will be two probability measures, respectively on X and Y.

Definition 2.3 (Basic and composite partitions)**.**

(1) For some finite index set I, let $\{X_i\}_{i\in I}$ be partition of X into closed, disjoint subsets (up to a set of μ-zero measure), were $m_i := \mu(X_i)$ is positive for all $i \in I$. We call $\{X_i\}_{i\in I}$ the *basic partition*, and $\{m_i\}_{i\in I}$ the *basic cell masses*. The restriction of μ to basic cells, *i.e.* $\mu_i := \mu \llcorner X_i$ for $i \in I$, are called the *basic cell X-marginals*.

(2) A *composite partition* is a partition of I. For each J in a composite partition $\mathcal{J}$, we define the *composite cells*, and the *composite cell X-marginals* respectively by

$$X_J := \bigcup_{i\in J} X_i, \qquad \mu_J := \sum_{i\in J} \mu_i = \mu \llcorner X_J.$$

We will consider two composite partitions $\mathcal{J}_A$ and $\mathcal{J}_B$.

A prototypical choice for basic and composite partitions for $X = [0,1]^2$ is given by an interleaving grid of cubes as sketched in Figure 3. Convergence of domain decomposition to the optimal coupling hinges on some form of connectivity property of these partitions, where the precise notion of connectivity depends on the setting (*e.g.* without or with entropic regularization, see [3,7,8]). The key requirement is that optimality of the coupling on each of the composite cells implies global optimality of the coupling. For entropic optimal transport, connectedness of the *partition graph* (see Def. 3.13) is sufficient and necessary [7].

The domain decomposition algorithm is now stated in Algorithms 1 and 2, where we will reuse the sub-routine of Algorithm 1 in the hybrid scheme. One may notice that the subproblem in line 4 of Algorithm 1 is slightly different from (2.3), since the reference measure in the KL term is not the product of the problem marginals. This reference measure is inherited by restricting the global entropic optimal transport objective (2.3) to the cell $X_J \times Y$. Nevertheless, noting that $0 \le \nu_J \le \nu$, a direct computation shows that the two problems are equal up to a constant contribution, and therefore Proposition 2.2 also applies to the domain decomposition cell subproblem. Keeping $\mu_J \otimes \nu$ as reference measure in line 4 emphasizes the interpretation of domain decomposition as a block coordinate descent method that decreases the global objective in every step.

Algorithm 1. DomDecIter ([7], Algorithm 1).

Input: current coupling $\pi \in \Pi(\mu, \nu)$ and partition $\mathcal{J}$.
Output: new coupling $\pi' \in \Pi(\mu, \nu)$

1: **for all** $J \in \mathcal{J}$ **do** // iterate over each composite cell
2: $\quad \mu_J \leftarrow \mathrm{P}_X(\pi \llcorner (X_J \times Y))$ // retrieve X-marginal on cell
3: $\quad \nu_J \leftarrow \mathrm{P}_Y(\pi \llcorner (X_J \times Y))$ // compute Y-marginal on cell
4: $\quad \pi_J \leftarrow \arg\min \left\{ \int_{X_J \times Y} c \, \mathrm{d}\pi + \varepsilon \, \mathrm{KL}(\pi | \mu_J \otimes \nu) \,\middle|\, \pi \in \Pi(\mu_J, \nu_J) \right\}$
5: **end for**
6: $\pi' \leftarrow \sum_{J \in \mathcal{J}} \pi_J$

Algorithm 2. Domain decomposition for optimal transport ([7], Algorithm 1).

Input: initial coupling $\pi_{\text{init}} \in \Pi(\mu, \nu)$
Output: a sequence $(\pi^k)_k$ of feasible couplings in $\Pi(\mu, \nu)$

1: $\pi^0 \leftarrow \pi_{\text{init}}$
2: $k \leftarrow 0$
3: **loop**
4: $\quad k \leftarrow k + 1$
5: $\quad$ **if** (k is odd) **then** $\mathcal{J}_k \leftarrow \mathcal{J}_A$ **else** $\mathcal{J}_k \leftarrow \mathcal{J}_B$ // select the partition
6: $\quad \pi^k \leftarrow \text{DomDecIter}(\pi^{k-1}, \mathcal{J}_k)$ // solve all subproblems
7: **end loop**

Given the iterates $(\pi^k)_k$ computed in Algorithm 2, we call

$$\nu_i^k := \mathrm{P}_Y\big(\pi^k \llcorner (X_i \times Y)\big), \quad i \in I. \tag{2.4}$$

the *basic cell Y-marginals*. In fact, for a performant implementation one should merely store $(\nu_i^k)_{i \in I}$ instead of the full π^k, since they require less memory and are sufficient to compute ν_J in line 3 of Algorithm 1, and (parts of) the full iterates can be reconstructed efficiently from the basic cell Y-marginals wherever required. For more details on the implementation of the domain decomposition algorithm we refer to Section 6 of [7].

3. Hybrid scheme

As discussed in the introduction, domain decomposition performs poorly on initializations that contain a substantial nonlocal curl component. In this section we show how to enhance the domain decomposition iteration with an additional global update step designed to reduce this curl, which we call the *flow update*.

We start in Section 3.1 with a simplified setting to build intuition. Section 3.2 defines the flow update in full generality and establishes that it decreases the entropic optimal transport objective (2.3) while preserving the marginals. Finally, Section 3.3 introduces a scheme that interleaves entropic domain decomposition with flow updates, and shows convergence of the iterates to the optimal coupling. Numerical experiments demonstrating the efficiency of the new scheme are presented in Section 5.

3.1. Flow update for singleton basic cells

For now, assume that X is finite and the basic partition I divides X into singletons, *i.e.* each X_i consists of a single x_i for all $i \in I$. We assume that a directed graph with vertex set I and edge (or neighbourhood) set $\mathcal{N} \subset I \times I$ is given, satisfying the following definition.

Definition 3.1. A *neighborhood set* is a set $\mathcal{N} \subset I \times I$ with the properties:

(1) $(i, i) \in \mathcal{N}$ for all $i \in I$.
(2) For all $i, j \in I$, $(i, j) \in \mathcal{N}$ if and only if $(j, i) \in \mathcal{N}$.

We write $\mathcal{N}(i) := \{j \in I | (i,j) \in \mathcal{N}\}$.

That is, the directed graph $(I, \mathcal{N})$ is symmetric and each vertex has an edge to itself. On this graph we can introduce the set of mass preserving flows.

Definition 3.2. For a neighbourhood set $\mathcal{N}$, a tuple $w := (w_{ij})_{(i,j)\in\mathcal{N}}$, $w_{ij} \in \mathbb{R}_+$ for $(i,j) \in \mathcal{N}$, is a *(mass preserving) flow* if it satisfies the divergence constraint

$$\sum_{j\in\mathcal{N}(i)} w_{ij} = \sum_{j\in\mathcal{N}(i)} w_{ji} = \mu(X_i) =: m_i \quad \text{for all } i \in I. \tag{3.1}$$

For $(i,j) \in \mathcal{N}$ we interpret w_{ij} as the amount of mass flowing from cell i to j, in particular w_{ii} denotes the mass that remains in cell i. Denote by $\mathcal{F}$ the set of mass preserving flows (omitting the dependency on I, $\mathcal{N}$, and μ in the notation for simplicity).

Consider now some coupling $\pi \in \Pi(\mu,\nu)$ and recall the basic cell Y-marginals $\nu_i := \mathrm{P}_Y(\pi \llcorner \{x_i\} \times Y)$ for each $i \in I$, (2.4). Note that $\pi = \sum_{i\in I} \delta_{x_i} \otimes \nu_i$. A mass preserving flow $w \in \mathcal{F}$ induces a new coupling *via*

$$\hat{\nu}_i := \sum_{j\in\mathcal{N}(i)} w_{ji} \frac{\nu_j}{m_j} \qquad \text{for } i \in I, \qquad\qquad \hat{\pi} := \sum_{i\in I} \delta_{x_i} \otimes \hat{\nu}_i. \tag{3.2}$$

Using (3.1) it is easy to verify that $\hat{\pi} \in \Pi(\mu,\nu)$ (see Sect. 3.2 for a proof in a more general setting). For the change in the transport objective we find

$$\begin{aligned}\langle c, \hat{\pi} - \pi\rangle &= \sum_{i\in I} \langle c(x_i,\cdot), \hat{\nu}_i - \nu_i\rangle = \sum_{i\in I} \left\langle c(x_i,\cdot), \sum_{j\in\mathcal{N}(i)} w_{ji}\frac{\nu_j}{m_j} - \sum_{j\in\mathcal{N}(i)} w_{ij}\frac{\nu_i}{m_i}\right\rangle \\ &= \sum_{(i,j)\in\mathcal{N}} w_{ij} \left\langle c(x_j,\cdot) - c(x_i,\cdot), \frac{\nu_i}{m_i}\right\rangle.\end{aligned} \tag{3.3}$$

Minimizing this over all flows $w \in \mathcal{F}$ (*i.e.* finding the best new coupling $\hat{\pi}$ that can be reached by a flow from π) corresponds to solving the min-cost flow problem

$$\min_{w\in\mathcal{F}} \sum_{(i,j)\in\mathcal{N}} w_{ij} \cdot \bar{c}_{ij} \tag{3.4}$$

for cost coefficients

$$\bar{c}_{ij} := \left\langle c(x_j,\cdot) - c(x_i,\cdot), \frac{\nu_i}{m_i}\right\rangle \qquad \text{for } (i,j) \in \mathcal{N}. \tag{3.5}$$

Note that (3.4) can be interpreted as an optimal transport problem on the graph $(I, \mathcal{N})$. We call the procedure of replacing π by $\hat{\pi}$ as given by (3.2) with a minimizing flow w of (3.4), a *flow update*. Key properties of the flow update are:

- **The problem (3.4) is global**, *i.e.* it will be able to detect and remove curl that may be barely detectable at the level of individual composite cells.
- **The optimal flow w can be computed efficiently** for moderate sizes of I and sparse $\mathcal{N}$, since there are highly efficient solvers for the min-cost flow problem.
- **The flow update does not modify the marginals.** This is a consequence of the divergence constraints (3.1).
- **The flow update does not increase the transport objective nor the entropy objective.** For the former this is rather easy to see, since $\bar{c}_{ii} = 0$ and an admissible flow is given by keeping all mass in its original location, *i.e.* $w_{ij} = 0$ for $i \neq j$. The latter is not as obvious, but it hinges on convexity of the entropy and the fact that the $\hat{\nu}_i$ are (up to normalization) convex combinations of the ν_i. Proofs for both are given in Section 3.2.

– **The flow update can formally be interpreted as a discretization of an L^∞-version of the AHT scheme.** To anticipate this, note that in the limit of very fine partitions, $c(x_j,\cdot)-c(x_i,\cdot)$ formally becomes the partial derivative of c in X at x_i in direction $x_j - x_i$ (here $X \subset \mathbb{R}^d$), ν_i/m_i becomes the disintegration of π with respect to its X-marginal at x_i, so that $\bar{c}_{ij}$ becomes the partial derivative at x_i, averaged over Y with respect to the disintegration. Finally, w_{ij} formally becomes a momentum measure, or w_{ij}/m_i becomes a velocity field, that describes the horizontal movement of mass and the divergence constraints (3.1) enforce that the velocity field has its components bounded by 1 almost everywhere. We revisit this intuition in Section 4.

On the other hand, **for large X the above strategy will not be computationally practical**, since solving problem (3.4) will become inefficient. As a compromise, in Section 3.2, we will adapt the flow update to the setting where the basic cells X_i are not singletons. This will require a more complex interpretation of flows w as induced plans $\hat{\pi}$, (3.2), and a corresponding adjusted definition of the cost coefficients $\bar{c}_{ij}$, (3.5). Intuitively, the flow update will then implement a global optimization of π at the resolution level of the basic cells $(X_i)_{i\in I}$ (which can be kept at a moderate size), whereas domain decomposition will locally optimize on the scale of individual cells.

3.2. Flow update for general basic cells

If basic cells X_i are no longer singletons, construction (3.2) is no longer applicable to translate a flow into an update of π. Instead, we will need a more general instruction on how to arrange mass from cell i in cell j, while preserving the marginals of the coupling. This role is played by *edge candidates*.

Definition 3.3 (Edge candidates)**.** A candidate for edge $(i,j)\in\mathcal{N}$ is a coupling $\hat{\pi}_{ij} \in \Pi(\mu_j/m_j, \nu_i/m_i)$.

We will specify in Definition 3.9 and below how edge candidates can be constructed in a practical way. Once a set of such candidates is fixed, we can define the general flow update.

Definition 3.4 (Flow update)**.** Let $\pi \in \mathcal{M}_+(X\times Y)$, $(\pi_i)_{i\in I}$ its decomposition into basic cells, with cell masses $(m_i)_{i\in I}$, cell Y-marginals $(\nu_i)_{i\in I}$ and cell X-marginals $(\mu_i)_{i\in I}$. Let $\mathcal{N}$ be a neighborhood set and $(\hat{\pi}_{ij})_{(i,j)\in\mathcal{N}}$ be edge candidates, with $\hat{\pi}_{ii} = \pi_i/m_i$ for each $i\in I$. Define the edge cost coefficients

$$\bar{c}_{ij} := \langle c, \hat{\pi}_{ij} - \hat{\pi}_{ii}\rangle. \tag{3.6}$$

A *flow update* is a coupling of the form

$$\hat{\pi} := \sum_{j\in I} \hat{\pi}_j, \quad \text{with } \hat{\pi}_j := \sum_{i\in\mathcal{N}(j)} w_{ij}\cdot\hat{\pi}_{ij} \tag{3.7}$$

where $(w_{ij})_{ij}$ is a solution to the min-cost flow problem (3.4) with cost coefficients $(\bar{c}_{ij})_{(i,j)\in\mathcal{N}}$.

Remark 3.5. If the basic cells are singletons, *i.e.* $X_i = \{x_i\}$ for $i\in I$, as above, then the only feasible candidate for each edge is $\hat{\pi}_{ij} := \delta_{x_j}\otimes\nu_i/m_i$. In this case (3.6) becomes (3.5) and (3.7) becomes (3.2).

Remark 3.6. For a given π, the flow update might not be unique, since solutions to the min-cost flow problem (3.4) might not be unique. This is however no issue for the algorithm.

The following two lemmas show that a flow update does not change the marginals nor does it increase the transport score.

Lemma 3.7. *A coupling $\pi\in\mathcal{M}_+(X\times Y)$ preserves its global marginals under a flow update. The new cell Y-marginals are given by*

$$\hat{\nu}_j := \sum_{i\in\mathcal{N}(j)} w_{ij}\cdot\frac{\nu_i}{m_i}. \tag{3.8}$$

Proof. Let π be the initial plan, and let $\hat{\pi}$ be a flow update as given in (3.7). Then, on each basic cell $j \in I$,

$$\mathrm{P}_X \hat{\pi}_j = \sum_{i \in \mathcal{N}(j)} w_{ij} \cdot \mathrm{P}_X \hat{\pi}_{ij} = \sum_{i \in \mathcal{N}(j)} \frac{w_{ij}}{m_j} \cdot \mu_j = \mathrm{P}_X \pi_j.$$

The new cell Y-marginals is given by

$$\hat{\nu}_j = \mathrm{P}_Y \hat{\pi}_j = \sum_{i \in \mathcal{N}(j)} w_{ij} \cdot \mathrm{P}_Y \hat{\pi}_{ij} = \sum_{i \in \mathcal{N}(j)} w_{ij} \cdot \frac{\nu_i}{m_i},$$

and so the global Y-marginal is

$$\mathrm{P}_Y \hat{\pi} = \sum_{j \in I} \hat{\nu}_j = \sum_{j \in I} \sum_{i \in \mathcal{N}(j)} \frac{w_{ij}}{m_i} \nu_i = \sum_{i \in I} \nu_i \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{m_i} = \sum_{i \in I} \nu_i = \mathrm{P}_Y \pi.$$

□

Lemma 3.8. *The transport objective is non-increasing under a flow update.*

Proof. Let π be an initial plan, and $\hat{\pi}$ a flow update of π where w is the corresponding optimal flow of (3.4). By Definition 3.4 one has that

$$\hat{\pi} = \sum_{(i,j) \in \mathcal{N}} w_{ij} \cdot \hat{\pi}_{ij} \qquad \text{and} \qquad \pi = \sum_{(i,j) \in \mathcal{N}} w_{ij} \cdot \hat{\pi}_{ii}$$

and therefore one obtains for the change in the transport objective

$$\langle c, \hat{\pi} - \pi \rangle = \sum_{(i,j) \in \mathcal{N}} w_{ij} \cdot \langle c, \hat{\pi}_{ij} - \hat{\pi}_{ii} \rangle = \sum_{(i,j) \in \mathcal{N}} w_{ij} \cdot \bar{c}_{ij} \tag{3.9}$$

which is precisely the objective of (3.4). A feasible candidate for w is the "stationary flow" that leaves all mass in its original cells, *i.e.* $w_{ij} = m_i$ if $i = j$ and $w_{ij} = 0$ otherwise. For this admissible choice one has that (3.9) equals zero, and thus the minimal value of (3.4) must be non-positive, and therefore, so must be the change in the transport objective. □

The entropic score is not necessarily decreasing under a flow update for general edge candidates $\hat{\pi}_{ij}$. The following definition gives a practical way to craft edge candidates for which the entropic score is also non-increasing.

Definition 3.9 (Gluing edge candidate)**.** Let $i \in I$, $j \in \mathcal{N}(i) \setminus \{i\}$ and $\gamma_{ij} \in \Pi(\mu_i/m_i, \mu_j/m_j)$. We refer to the plan $\hat{\pi}_{ij} \in \Pi(\mu_j/m_j, \nu_i/m_i)$ that is obtained by combining γ_{ij} with $\pi_i/m_i \in \Pi(\mu_i/m_i, \nu_i/m_i)$ with the gluing lemma ([35], Lem. 7.6) along their common μ_i/m_i-marginal as *gluing edge candidate* for edge (i, j). $\hat{\pi}_{ij}$ acts on measurable sets $A \times B \subset X_j \times Y$ as follows:

$$\hat{\pi}_{ij}(A \times B) := \frac{1}{m_i} \int_{X_i} (\gamma_{ij})_x(A) \cdot (\pi_i)_x(B) \, \mathrm{d}\mu_i(x) \tag{3.10}$$

where $(\gamma_{ij})_x$ and $(\pi_i)_x$ denote the disintegrations of both measures against their μ_i/m_i or μ_i marginal at x respectively. (Disintegrations are by convention probability measures, so the normalization of π_i/m_i can be neglected at this point.)

Remark 3.10. Two prototypical choices for γ_{ij} are the product coupling $\mu_i/m_i \otimes \mu_j/m_j$, and an optimal transport plan (*e.g.* for the squared distance on X).

For the former one has $(\gamma_{ij})_x = \mu_j/m_j$ μ_i-almost everywhere and thus (3.10) simplifies to $\hat{\pi}_{ij} = \mu_j/m_j \otimes \nu_i/m_i$. This is straightforward to implement. However, the induced coupling candidates $\hat{\pi}$ in (3.7) may be somewhat "bulky" and thus the associated cost coefficients $\bar{c}_{ij}$ might not be favourable for moving mass, slowing down the optimization scheme.

The latter choice requires to pre-compute and store the plans γ_{ij}. This is feasible since basic cells are usually small and the set $\mathcal{N}$ is sparse. Equation (3.10) can then be implemented by a simple matrix multiplication.

In numerical experiments (Sect. 5) we find that both options perform well as long as π is far from the optimum. Close to the optimum, the latter choice performs slightly better, at the cost of some additional computational effort. See Section 5.2 for more details.

One may also consider the optimal entropic transport plan between μ_i/m_i and μ_j/m_j. The above choices then correspond to the limits of $\varepsilon = \infty$ and $\varepsilon = 0$, respectively.

Lemma 3.11 (Properties of the gluing edge candidates)**.** *For every $i \in I$, $j \in \mathcal{N}(i) \setminus \{i\}$ let γ_{ij} be a transport plan between μ_i/m_i and μ_j/m_j, and let $\hat{\pi}_{ij}$ be a gluing edge candidate following (3.10). Then:*

(1) *$\hat{\pi}_{ij}$ is a candidate for edge (i,j), i.e. $\hat{\pi}_{ij} \in \Pi(\mu_j/m_j, \nu_i/m_i)$.*
(2) *If $\pi_i \ll \mu_i \otimes \nu$, then $\hat{\pi}_{ij} \ll \mu_j \otimes \nu$, and its density is given by*

$$\frac{\mathrm{d}\hat{\pi}_{ij}}{\mathrm{d}(\mu_j \otimes \nu)}(x,y) = \frac{1}{m_j} \int_{X_i} \frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x',y) \mathrm{d}(\gamma_{ij})_x(x') \tag{3.11}$$

$\mu_j \otimes \nu$-almost everywhere, where now $(\gamma_{ij})_x$ denotes the disintegration of γ_{ij} at x against the μ_j/m_j-marginal.
(3) *A flow update with edge candidates $(\hat{\pi}_{ij})_{ij}$ does not increase the entropic objective.*

Proof. Throughout this proof, variables x, x' and y run respectively over X_j X_i, and Y.

Item 1. This is a direct consequence of the gluing lemma and can also quickly be verified explicitly from (3.10), *e.g.* for the X-marginal for measurable $A \subset X_j$ *via*

$$\begin{aligned}(\mathrm{P}_X \hat{\pi}_{ij})(A) = \hat{\pi}_{ij}(A \times Y) &= \frac{1}{m_i} \int_{X_i} (\gamma_{ij})_x(A) \cdot (\pi_i)_x(Y) \,\mathrm{d}\mu_i(x) = \frac{1}{m_i} \int_{X_i} (\gamma_{ij})_x(A) \,\mathrm{d}\mu_i(x) \\ &= \gamma_{ij}(A \times X_i) = \mu_j(A)/m_j.\end{aligned}$$

Item 2. Since $\frac{\mathrm{d}\pi_i}{\mathrm{d}\mu_i \otimes \nu}$ is non-negative, the quantity in (3.11) is unambiguously defined and we just need to check that it yields the same measure as (3.10). Let us integrate the right-hand side of (3.11) with respect to $\mu_j \otimes \nu$ on a set of the form $A \times B \subset X_j \times Y$: (Note that in the following computation we switch from the disintegration of γ_{ij} against its μ_j/m_j marginal to the one against its μ_i/m_i marginal, denoted by $(\gamma_{ij})_x$ and $(\gamma_{ij})_{x'}$ respectively.)

$$\begin{aligned}\int_{A\times B} \left[\frac{1}{m_j} \int_{X_i} \frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x',y) \,\mathrm{d}(\gamma_{ij})_x(x')\right] \mathrm{d}\mu_j(x) \,\mathrm{d}\nu(y) &= \int_{X_i \times A \times B} \frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x',y) \,\mathrm{d}\gamma_{ij}(x',x) \,\mathrm{d}\nu(y) \\ = \frac{1}{m_i} \int_{A \times X_i \times B} \frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x',y) \,\mathrm{d}(\gamma_{ij})_{x'}(x) \,\mathrm{d}\mu_i(x') \,\mathrm{d}\nu(y) &= \frac{1}{m_i} \int_{A \times X_i \times B} \mathrm{d}(\gamma_{ij})_{x'}(x) \,\mathrm{d}\pi_i(x',y) \\ = \frac{1}{m_i} \int_{A \times X_i \times B} \mathrm{d}(\gamma_{ij})_{x'}(x) \,\mathrm{d}(\pi_i)_{x'}(y) \,\mathrm{d}\mu_i(x') &= \hat{\pi}_{ij}(A \times B).\end{aligned}$$

Item 3. If π_i is singular with respect to $\mu_i \otimes \nu$ for some $i \in I$, the entropic score cannot increase. Otherwise, by the previous point $\hat{\pi}_{ij}$ has a density with respect to $\mu_i \otimes \nu$ given by (3.11). Let us first compute how the relative entropy $\mathrm{KL}(m_j\, \hat{\pi}_{ij} | \mu_j \otimes \nu)$ relates to $\mathrm{KL}(\pi_i | \mu_i \otimes \nu)$:

$$\mathrm{KL}(m_j\, \hat{\pi}_{ij} | \mu_j \otimes \nu) = \int_{X_j \times Y} \varphi\left(m_j \frac{\mathrm{d}\hat{\pi}_{ij}}{\mathrm{d}(\mu_j \otimes \nu)}(x,y)\right) \mathrm{d}\mu_j(x) \,\mathrm{d}\nu(y)$$

$$= \int_{X_j \times Y} \varphi\left(\int_{X_i} \frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x', y) \mathrm{d}(\gamma_{ij})_x(x') \right) \mathrm{d}\mu_j(x)\, \mathrm{d}\nu(y).$$

Since $(\gamma_{ij})_x$ is a probability measure and φ is a convex function, we can use Jensen's inequality to bound this quantity:

$$\begin{aligned} &\leq \int_{X_j \times X_i \times Y} \varphi\left(\frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x', y) \right) \mathrm{d}(\gamma_{ij})_x(x') \mathrm{d}\mu_j(x) \mathrm{d}\nu(y) \\ &= m_j \int_{X_j \times X_i \times Y} \varphi\left(\frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x', y) \right) \mathrm{d}\gamma_{ij}(x', x)\, \mathrm{d}\nu(y) \\ &= \frac{m_j}{m_i} \int_{X_i \times Y} \varphi\left(\frac{\mathrm{d}\pi_i}{\mathrm{d}(\mu_i \otimes \nu)}(x', y) \right) \mathrm{d}\mu_i(x') \mathrm{d}\nu(y) = \frac{m_j}{m_i} \mathrm{KL}(\pi_i | \mu_i \otimes \nu). \end{aligned} \tag{3.12}$$

With this estimate, let us check that the global entropy does not increase after a flow update:

$$\mathrm{KL}(\hat{\pi} | \mu \otimes \nu) = \sum_{j \in I} \mathrm{KL}(\hat{\pi}_j | \mu_j \otimes \nu) = \sum_{j \in I} \mathrm{KL}\left(\sum_{i \in \mathcal{N}(j)} w_{ij}\, \hat{\pi}_{ij} | \mu_j \otimes \nu \right) \leq \sum_{(i,j) \in \mathcal{N}} \frac{w_{ij}}{m_j} \mathrm{KL}(m_j\, \hat{\pi}_{ij} | \mu_j \otimes \nu),$$

where in the last inequality we used Jensen's inequality in the first argument of the KL. Now we use the estimate from (3.12):

$$\leq \sum_{(i,j) \in \mathcal{N}} \frac{w_{ij}}{m_j} \frac{m_j}{m_i} \mathrm{KL}(\pi_i | \mu_i \otimes \nu) = \sum_{i \in I} \left(\sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{m_i} \right) \mathrm{KL}(\pi_i | \mu_i \otimes \nu) = \mathrm{KL}(\pi | \mu \otimes \nu).$$

□

Finally, we summarize the previous results to collect the desired properties of the flow update.

Proposition 3.12. *A flow update with gluing edge candidates preserves the marginals and has a non-increasing entropic optimal transport objective* (2.3).

Proof. Let π be the initial plan, $\hat{\pi}$ a flow update obtained with gluing edge candidates. Then Lemma 3.7 guarantees the marginals of $\hat{\pi}$ are those of π, Lemma 3.8 proves that $\langle c, \hat{\pi} \rangle \leq \langle c, \pi \rangle$ and Lemma 3.11 establishes that $\mathrm{KL}(\hat{\pi} | \mu \otimes \nu) \leq \mathrm{KL}(\pi | \mu \otimes \nu)$. □

3.3. Hybrid scheme

The flow updates described in the previous section are in general unable to generate a minimizing sequence on their own. For example, if $\pi = \mu \otimes \nu$, any flow update would yield again $\hat{\pi} = \mu \otimes \nu$, since π already minimizes the relative entropy and by Lemma 3.11 a flow update cannot increase it (at least when using gluing edge candidates). Conversely, if π is the optimal unregularized plan between μ and ν, a flow update cannot bring it to the optimal entropic plan, since doing so would typically involve introducing a bit of blur, which will inevitably increase the transport score.

However, flow updates can be intertwined with domain decomposition steps to provide fast removal of global curl, on which domain decomposition performs poorly. The domain decomposition steps will in turn make progress where the flow updates would be stuck and guarantee convergence to a global minimizer.

We call the combination of domain decomposition and flow updates the *hybrid scheme*. Algorithm 3 outlines a possible implementation, where two domain decomposition iterations are followed by a flow update. Of course different schedules can be considered (*e.g.* running the flow update after only every n-th domain decomposition iteration), and they do not change significantly the properties of the algorithm.

Algorithm 3. Hybrid scheme for optimal transport.

Input: initial coupling $\pi_{\text{init}} \in \Pi(\mu, \nu)$
Output: a sequence $(\pi^k)_k$ of feasible couplings in $\Pi(\mu, \nu)$

$\pi^0 \leftarrow \pi_{\text{init}}$
$k \leftarrow 0$
loop
  $k \leftarrow k + 1$
  if $(k\%3 == 1)$ **then** $\pi^k \leftarrow \textsc{DomDecIter}(\pi^{k-1}, \mathcal{J}_A)$
  elseif $(k\%3 == 2)$ **then** $\pi^k \leftarrow \textsc{DomDecIter}(\pi^{k-1}, \mathcal{J}_B)$
  elseif $(k\%3 == 0)$ **then** $\pi^k \leftarrow \textsc{FlowUpdate}(\pi^{k-1})$
end loop

Due to the monotonicity properties of the flow update, it is natural to expect the hybrid scheme to converge under the same assumptions as regular domain decomposition. In the case of entropic domain decomposition, apart from boundedness of the cost, the only other crucial assumption was connectedness of the *partition graph* (see [7], Sect. 4.3). We give below a simplified version of the concept that is sufficient for our purposes:

Definition 3.13 (Partition graph, adapted from Definition 7 of [7])**.** The partition graph is given by the vertex set $V := \mathcal{J}_A \cup \mathcal{J}_B$ and the edge set

$$E := \Big\{ (J, \hat{J}) \subset \mathcal{J}_A \cup \mathcal{J}_B \,\Big|\, J \cap \hat{J} \neq \varnothing \Big\},$$

i.e. there is an edge between two composite cells if their intersection is non-empty. (Recall that the composite partitions $\mathcal{J}_A$ and $\mathcal{J}_B$ are partitions of the discrete index set I for the basic partition cells of X.) In this way, connectedness of the partition graph intuitively translates to the ability of mass or information to travel between arbitrary subdomains.

Proposition 3.14 (Convergence of the hybrid scheme)**.** *Let c be continuous, $\varepsilon > 0$, consider partitions in the sense of Definition 2.3 with strictly positive basic cell masses, and assume that the partition graph is connected. Let $(\pi^k)_k$ be a sequence obtained with Algorithm 3, where the flow update uses gluing edge candidates (cf. Defs. 3.4, 3.9). Then $(\pi^k)_k$ converges weak* to the unique minimizer of (2.3).*

For convenience we assume c to be continuous here. See Remark 3.16 for a discussion of more general costs.

Proof. Define by S_A the map that applies an A iteration to π, *i.e.* $\pi \mapsto \textsc{DomDecIter}(\pi, \mathcal{J}_A)$, analogously for S_B. These maps are continuous with respect to the weak* topology, since they are a composition of restriction of measures, projection and solution of entropic problems, which in our setting are all continuous (restriction by Lem. 2.1, solution by stability of entropic OT when the cost is continuous [22], projection is obvious). Besides, these maps do not increase the score of a given coupling – since domain decomposition is a coordinate descent algorithm –, and keep it feasible.

The iterates of Algorithm 3 thus verify:

$$\begin{aligned} \pi^{3\ell+1} = S_A\big(\pi^{3\ell}\big), \qquad \pi^{3\ell+2} = S_B\big(\pi^{3\ell+1}\big) \qquad & \text{for all } \ell \in \mathbb{N}, \\ \big\langle c, \pi^{k+1} \big\rangle + \varepsilon\, \mathrm{KL}\big(\pi^{k+1} | \mu \otimes \nu\big) \leq \big\langle c, \pi^{k} \big\rangle + \varepsilon\, \mathrm{KL}\big(\pi^{k} | \mu \otimes \nu\big) \qquad & \text{for all } k \in \mathbb{N}, \end{aligned} \tag{3.13}$$

the decrement for the flow updates being granted by (3.12), because we are using gluing candidates.

Since the couplings are supported on the compact space $X \times Y$, the sequence $(\pi^{3\ell})_\ell$ is tight, so it has weak* cluster points. Besides, by monotonicity of the score, all such cluster points share the same score, and they all lie in $\Pi(\mu, \nu)$ by continuity of the projection operator.

Let $(\pi^{3\ell})_\ell$ be a subsequence (not relabeled) converging to one such cluster point π^*. Then, by continuity of the solve maps, $(\pi^{3\ell+2})_\ell = (S_B(S_A(\pi^{3\ell})))_\ell$ must converge to $S_B(S_A(\pi^*))$. This means that π^* and $S_B(S_A(\pi^*))$

share the same score, since they are both cluster points of the original sequence. In turn, this implies that π^* is a fixed point for both S_A and S_B; otherwise applying S_A or S_B would have decreased its score. This follows from uniqueness of minimizers in Algorithm 1, line 4, see Proposition 2.2, item (i). Therefore we conclude that π^* is locally optimal on each subdomain of $\mathcal{J}_A$ and $\mathcal{J}_B$, which by Proposition 2.2, item (iii) implies that there exist functions α_J on X_J and β_J on Y such that

$$\pi^*_J = \exp\left(\frac{\alpha_J(x) + \beta_J(y) - c(x,y)}{\varepsilon}\right)\mu_J \otimes \nu, \qquad \text{for all } J \in \mathcal{J}_A \cup \mathcal{J}_B. \tag{3.14}$$

Now we will build global entropic potentials from the local potentials in (3.14). Take an edge $(J, \hat{J})$ of the partition graph. Since the intersection of J and $\hat{J}$ is non-empty, there is some basic cell i they share, which has positive mass by assumption. For $\mu \otimes \nu$ a.e. (x, y) in $X_i \otimes Y$, the corresponding expressions (3.14) for J and $\hat{J}$ must coincide, which means that the difference $\beta_J - \beta_{\hat{J}}$ is a constant ν-a.e. By connectedness of the partition graph, this property extends to every pair of composite cells, so we conclude that for every cell $J \in \mathcal{J}_A \cup \mathcal{J}_B$, $\beta_J = \beta^* - s_J$ ν-a.e. where $\beta^* := \beta_{J_0}$ for some arbitrary cell J_0, and $(s_J)_{J \in \mathcal{J}_A \cup \mathcal{J}_B}$ are suitable real offsets.

This means by Proposition 2.2 (ii) that the potentials $(\alpha_J - s_J, \beta^*)$ also yield the same π^*_J in (3.14). We can then obtain a global potential α^* in the following way: for each $x \in X$ find the cell $J \in \mathcal{J}_A$ such that $x \in X_J$ (which is μ-a.e. unique) and define

$$\alpha^*(x) = \alpha_J(x) - s_J. \tag{3.15}$$

Then $\pi^* = e^{(\alpha^* \oplus \beta^* - c)/\varepsilon}\mu \otimes \nu$, since this new expression coincides with (3.14) for every composite cell in $\mathcal{J}_A$, and these constitute a partition of X. Finally, again by Proposition 2.2 (iii) this is the optimal coupling, this time for the global problem. □

Remark 3.15. Note that in the proof of Proposition 3.14 we only leverage the fact that the flow updates do not increase the score. The task of leveraging the structure of the flow updates to improve the convergence rate of domain decomposition seems daunting, since they can be interpreted as discretization of a gradient flow in a space where the objective is not geodesically convex (see end of Sect. 4 for a detailed discussion). Of course, intuitively, we expect that flow updates and domain decomposition updates have complementary strengths and weaknesses (cf. Sect. 1.2). This intuition is supported by the numerical experiments in Section 5.

Remark 3.16 (Convergence of the hybrid scheme for more general costs)**.** As shown in Corollary 4.12 of [7] entropic domain decomposition converges to the optimal coupling for any bounded measurable cost function c. We believe this to be the case also for the hybrid scheme, but to show it we should use a different stability result for the solve operator, in particular ([13], Thm. 4.3), which assumes that the density of the marginals with respect to some fixed reference measure is bounded from above and away from zero. This is the case for domain decomposition after sufficiently many iterations, since the coupling will eventually have fully support due to the entropic regularization, and a lower bound on the density can be given (see [7], Lem. 4.2, (ii)). However, a proof that this "diffusion" cannot be reverted by the flow updates would be overly technical and not adequate for the simple presentation we intend to give. At any rate, if the initialization π^0 has a density $\frac{\mathrm{d}\pi^0}{\mathrm{d}\mu\otimes\nu}$ bounded away from zero, or if enough iterations are performed before the first flow update, all the subsequent iterations will satisfy the necessary density bounds. The reason is that flow updates mix the marginals, and as such cannot worsen a uniform lower bound on the density.

4. Formal continuity limit and comparison with AHT scheme

We now return to the setting of Section 3.1. Let $n \in \mathbb{N}$ and let X be a regular Cartesian grid in $[0,1]^d$ with n grid points along each axis, let $\mathcal{N}$ be given by the standard $2d$-neighbourhood (*e.g.* the 4-neighbourhood in $d = 2$). We now formally discuss the limit $n \to \infty$. We can identify a flow $w \in \mathcal{F}$ with a vector measure $W := \sum_{(i,j)\in\mathcal{N}} w_{ij} \cdot n \cdot (x_j - x_i) \cdot \delta_{x_i} \in \mathcal{M}(X)^d$. Note that $|x_j - x_i|_\infty = 1/n$ for all $(i,j) \in \mathcal{N}$ with $i \neq j$.

Therefore, one has that $W \ll \mu = \sum_{i\in I} m_i\,\delta_{x_i}$ and $|\frac{\mathrm{d}W}{\mathrm{d}\mu}(x)|_\infty \leq 1$ for μ-almost all x. For $\phi \in \mathrm{C}^1(X;\mathbb{R}^d)$ one then obtains

$$\int_X \nabla\phi\,\mathrm{d}W = \sum_{(i,j)\in\mathcal{N}} \langle \nabla\phi(x_i), x_j - x_i\rangle \cdot n \cdot w_{ij} = \sum_{(i,j)\in\mathcal{N}} [n\cdot\phi(x_j) - n\cdot\phi(x_i) + o(1)]w_{ij} = o(1).$$

Here we used the divergence constraints (3.1), which also imply $\sum_{(i,j)\in\mathcal{N}} w_{ij} = 1$, and $o(1)$ is to be understood with respect to the limit $n\to\infty$ and is obtained from the uniform continuity of $\nabla\phi$ on X. So as $n\to\infty$, any weak* cluster point of a sequence of measures W will be divergence free in a distributional sense. We interpret $\mathcal{F}$ therefore as a discrete approximation of the weak* compact set

$$\big\{v\cdot\mu \big| v : [0,1]^d \to [-1,1]^d \text{ measurable}, \operatorname{div}(v\cdot\mu) = 0\big\} \tag{4.1}$$

where div denotes distributional divergence. We do not expect that this discrete approximation will be dense in (4.1) as $n\to\infty$, due to anisotropy artefacts of the grid graph. Fortunately, the combination with domain decomposition ensures that this is not a problem. By similar arguments, for $c\in C^1(X\times Y)$, the min-cost flow objective (3.4) can be written as

$$\sum_{(i,j)\in\mathcal{N}} w_{ij}\cdot\left\langle c(x_j,\cdot) - c(x_i,\cdot), \frac{\nu_i}{m_i}\right\rangle = \int_X \left\langle v(x), \int_Y \nabla_X c(x,y)\,\mathrm{d}\pi_x(y)\right\rangle \mathrm{d}\mu(x) + o(1)$$

where v is, as above, the density of W with respect to μ, and we used that $\nu_i/m_i \in \mathcal{M}_1(Y)$ is the disintegration of $\pi = \sum_{i\in I}\delta_{x_i}\otimes\nu_i$ with respect to its X-marginal at x_i. Constructing a new coupling $\hat\pi$ *via* (3.2) then corresponds to a time-step of size $1/n$.

Consider now the following minimization problem over measurable velocity fields:

$$\inf_{v:\operatorname{div}(v\cdot\mu)=0} \int_X \left\langle v(x), \int_Y \nabla_X c(x,y)\,\mathrm{d}\pi_x(y)\right\rangle \mathrm{d}\mu(x) + \frac{1}{p}\int_X |v(x)|_p^p\,\mathrm{d}\mu(x). \tag{4.2}$$

For $p=2$ and $\pi = (\mathrm{id}, T_t)_\#\mu$ (*i.e.* $\pi_x = \delta_{T_t(x)}$) the minimizing velocity field will be the $\mathrm{L}^2(\mu)$-projection of $-\nabla_X c(\mathrm{id}, T_t)$ onto the constraint $\operatorname{div}(v\cdot\mu)=0$, as discussed in (1.4) and below. Conversely, the formal limit $p\to\infty$ will correspond to the minimization of the first term over the set (4.1), which is the formal continuity limit of our flow update.

Note that one also has $\Phi_{t\#}(\mu\otimes\nu) = \mu\otimes\nu$ and that formally Φ_t is invertible (if the flow of v_t were well-posed) therefore $\mathrm{KL}(\pi_t|\mu\otimes\nu) = \mathrm{KL}(\Phi_{t\#}\pi_0|\Phi_{t\#}(\mu\otimes\nu)) = \mathrm{KL}(\pi_0|\mu\otimes\nu)$. So formally the continuum limit of the flow updates keeps the entropy unchanged and the possible decrease that we observe at finite resolution is a discretization artefact.

In this sense, the flow update can formally be interpreted as a discretized L^∞-version of the AHT scheme; *i.e.* as a gradient flow for the transport cost on the space of differentiable domain rearrangements. As we have just seen, the L^∞-version allows for an exact discrete version of the divergence constraint as mass preserving flows on a discrete graph. The flow updates alone will in general not reach a global minimizer, due to anisotropy of the graph discretization and since it can only move fibers ν_i as a whole (recall the discussion at the beginning of Sect. 3.3). The combination with domain decomposition iterations resolves both of these issues.

Note that the suboptimal stationary points of the AHT scheme remain stationary for (4.2), including the $p\to\infty$ limit. Indeed, the constraint $\operatorname{div}(v\cdot\mu)=0$ for v implies:

$$\int_X \langle v(x), \nabla\varphi(x)\rangle\,\mathrm{d}\mu(x) = 0 \quad \text{for all } \varphi\in C^1(X). \tag{4.3}$$

This means that whenever the vector field $x\mapsto\int_Y \nabla_X c(x,y)\,\mathrm{d}\pi_x(y)$ can be written as the gradient of a function φ, the optimal velocity field in (4.2) is identically zero. This has several implications. First, it shows that the

L^∞-variant would not improve the convergence behavior of the AHT scheme; the suboptimal stationary points remain the same (our motivation to consider it is the compatibility with the min-cost flow discretization). Second, we expect that the discrete version will suffer from related non-optimal points where the updates will be zero or at least very small. Consequently, it might be impossible to obtain a convergence rate for the hybrid scheme that is faster than pure domain decomposition. This difficulty is caused by the fact that the transport cost is not geodesically convex within the space of admissible deformations (a challenge shared with the AHT scheme). A different relaxed version of the AHT scheme with entropic regularization is studied in [12]. It is shown that this relaxation has no sub-optimal stationary points and that it converges to the global minimizer. However, due to the aforementioned non-convexity, still no rate of convergence can be established.

5. Numerical experiments

In this section we provide numerical experiments that show how flow updates can resolve the freezing behaviour of the domain decomposition algorithm, we study how the choice of the edge candidates $\hat{\pi}_{ij}$ influences the behaviour of the hybrid algorithm, and how flow updates relate to the multiscale scheme.

5.1. Implementation notes

A CPU multiscale implementation of domain decomposition, parallelized *via* MPI was presented in [7]. In this article we apply a new GPU version, described in more detail in Appendix A[1]. The second component of our hybrid algorithm are the flow updates introduced in Sections 3.1 and 3.2. Their implementation can be divided into the following major steps:

(1) **Compute the edge costs $(\bar{c}_{ij})_{ij}$, (3.6).** This requires edge candidates $(\hat{\pi}_{ij})_{ij}$ as constructed in (3.10) which are constructed from transport plans $(\gamma_{ij})_{ij}$ from μ_i/m_i to μ_j/m_j. For $(\gamma_{ij})_{ij}$ we choose either an optimal entropic transport plan for some regularization strength ε_γ or the independent product plan $\mu_i/m_i \otimes \mu_j/m_j$ (which would correspond to the choice $\varepsilon_\gamma = \infty$), see Remark 3.10. Constructing $\hat{\pi}_{ij}$ from γ_{ij} also requires to temporarily instantiate the measure π_i, which can lead to memory bottlenecks if this is done fully in parallel.
(2) **Solve the min-cost flow problem (3.4).** Many efficient solvers are available for this problem. In our experiments we use `OR-tools` [30]. Popular alternatives are `LEMON` [18] or `CPLEX` [24].
(3) **Assemble the new basic cell Y-marginals** *via* (3.8)**.** This is enough to perform a subsequent domain decomposition iteration. The new full coupling π is not necessary.

5.2. Single scale experiments

As illustrated in Figure 1, domain decomposition can asymptotically exhibit *freezing* in non-optimal states in the limit of very fine cells and we propose flow updates as a remedy, as illustrated in Figure 2. In this section we discuss a more complex numerical experiment to examine freezing, its resolution *via* flow updates, and the influence of the choice of edge candidates $(\hat{\pi}_{ij})_{ij}$, see Section 5.1 and Remark 3.10.

Test problem and discretization. For our test problem we choose $\mu = \nu$ where μ is a measure on the square $[-1,1]^2$ with four symmetrical bumps, as pictured in Figure 4, left. In particular, μ has a 4-fold rotational symmetry, *i.e.* for the counterclockwise rotation by $\pi/2$, given by

$$T_0 : X \to Y, \qquad [-1,1] \ni (x_1, x_2) \mapsto (-x_2, x_1) \tag{5.1}$$

one has $T_{0\#}\mu = \mu$ and so $\pi_0 := (\mathrm{id}, T_0)_{\#}\mu$ is a non-optimal transport plan between $\mu = \nu$ and itself and we expect domain decomposition to freeze asymptotically when initialized with π_0, since the deviation from the optimal coupling $(\mathrm{id}, \mathrm{id})_{\#}\mu$ is essentially "pure curl", making it an interesting test case for the hybrid scheme.

[1]Code for both CPU and GPU implementation available at https://github.com/OTGroupGoe/DomainDecomposition.

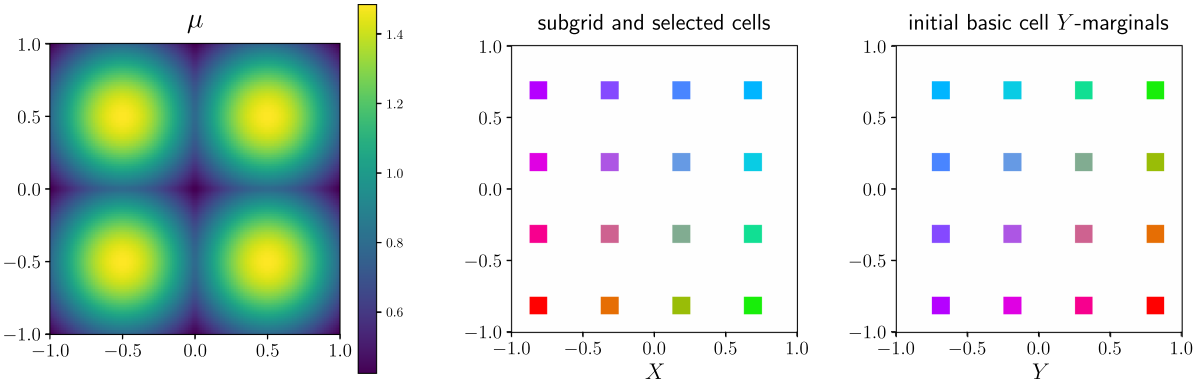

FIGURE 4. *Left*: marginal $\mu = \nu$ for the experiments of Section 5.2. *Center*: partition of X into basic cells with $n = 16$ cells along each axis. Some basic cells are highlighted in color. *Right*: to visualize a transport plan $\pi \in \Pi(\mu, \nu)$, for each basic cell X_i highlighted in color in the center panel, we show the ν-density of the corresponding Y-marginal $\nu_i = \mathrm{P}_Y(\pi \llcorner (X_i \times Y))$, cf. (2.4), in the same color on Y. Here this is exemplarily shown for the initialization $\pi^0 = (\mathrm{id}, T_0)_\# \mu$ where T_0 is a rotation by angle $\pi/2$, (5.1).

For numerical experiments the domain $[-1, 1]^2$ and the measure μ are then discretized and represented by a uniform Cartesian grid with N points along each axis. Basic cells are then formed by squares of $s \times s$ pixels, corresponding to $n = N/s$ basic cells along each axis, each basic cell being a square with edge length $1/n = s/N$. We choose $N \in \{32, 128\}$ and $s \in \{2, 8\}$ in this section and the concrete values will always be stated in the respective figures. Finally, for comparing iterations across different resolutions we follow [9], where we assign to each domain decomposition iterate a timestep $\Delta t = 1/n = s/N$ and define the measure trajectory $\mathbb{R}_+ \ni t \mapsto \pi_t := \pi^{\lfloor t/\Delta t \rfloor}$ where π^k refers to the iterates of Algorithm 2, initialized with $\pi^0 := \pi_0$. To keep the comparison consistent, for the hybrid scheme we will only track the number of domain decomposition iterations, or alternatively, we consider the flow update to be integrated at the end of the B-partition iteration. Hence we construct trajectories from the iterates of Algorithm 3 as follows:

$$\pi_t := \tilde{\pi}^{\lfloor t/\Delta t \rfloor}, \quad \text{with } \tilde{\pi}^{2\ell+m} = \pi^{3\ell+m}, \quad \text{for all } \ell \in \mathbb{N},\ m \in \{0, 1\}. \tag{5.2}$$

For the global entropic regularization parameter in (2.3) we choose $\varepsilon = (\Delta x/2)^2$, where $\Delta x = 2/N$ is the distance between adjacent pixels. With this regularization strength the entropic blurring is of the scale of the discretization, so we operate close to the unregularized regime. With larger ε, the number of required domain decomposition iterations would decrease (since the freezing effect is less severe with more blur), as would the number of Sinkhorn iterations to solve each subproblem. On the other hand, the approximate support of π_t (where mass is above the threshold of numerical precision) would increase and require more memory for storage. The curl that has to be removed by the flow updates is by definition non-local and thus on a length scale above the entropic blur scale. The flow updates are therefore not affected as much by a change in ε.

Visualization of trajectories. Visualizing the evolution of a coupling between 2-dimensional measures is challenging. We approach this issue as follows: We first color selected regions A_i of X as in Figure 4, center. Then for each A_i we compute the density of the Y-marginal of $\pi \llcorner (A_i \times Y)$ with respect to ν, *i.e.* the function

$$\frac{\mathrm{d}(\mathrm{P}_Y \pi \llcorner (A_i \times Y))}{\mathrm{d}\nu}$$

which takes values in $[0, 1]$ on Y and we show this density in the same color as A_i. For the counterclockwise rotation T_0 and the induced plan π_0 this is done in Figure 4, right. When the sets A_i are (unions of) basic cells, this visualization can conveniently be combined with the domain decomposition algorithm.

The hybrid scheme resolves the freezing behavior consistently over discretization scales. The evolution of π_0 under the pure domain decomposition algorithm and the hybrid scheme is visualized in Figure 5 for fixed $s = 2$ and different image sizes $N \in \{32, 128\}$, *i.e.* for cell resolutions $n \in \{16, 64\}$. As expected, due to the presence of a global rotation, domain decomposition is already quite slow for $N = 32$ and slows down further for $N = 128$ (even when accounting for the time steps associated with each iteration). To appreciate this in detail observe the slow movement of the colored blobs close to the boundary.

In contrast, the hybrid scheme evolves quickly towards the optimal coupling, with approximately equal speed for both $N \in \{32, 128\}$, indicating that problems can consistently be solved also at fine resolutions. Close to the optimal coupling the flow updates stop, in the sense that the optimal flows in (3.4) are zero. This means that at the flow updates are no longer able to reach a better configuration and the residual optimization is done purely by the domain decomposition iterations.

Influence of the edge candidates and relation to cellsize. The initial coupling π_0 features a strong global suboptimal rotation and thus we expect that edge candidates for all values of ε_γ will yield similar or even identical flow updates. The intuitive reason is that, in this regime, the edge costs $(\bar{c}_{ij})_{ij}$ are dominated by the large rotation whereas the fine substructure within the basic cells has little influence. Closer to optimality, on the other hand, the remaining curl is smaller and a more accurate estimation of the potential benefit of a flow update becomes beneficial.

Figure 6 shows an example of residual rotation after the flow updates have stopped. We plot the hybrid trajectories for $N = 128$, with two different choices of cellsize ($s = 2$ and $s = 8$) and for ε_γ being "small" ($(\Delta x/2)^2$) and "large" (∞) at $t = 4$ (as we saw in Fig. 5 the flow updates stop approximately at $t = 2$). For comparison we show the optimal plan. Our visualization choice for this plot consists in showing the aggregate cell Y-marginals for the same basic cells as in previous examples, so that one can focus on the displacement with respect to the optimal coupling. The residual displacements are on the order of one or two pixels.

Cell size has the largest influence on the residual displacement. Smaller cell sizes allow for a finer rearrangement of mass, which is specially noticeable close to optimality. This must be weighed against an increased size and computational complexity of the associated min-cost flow problems. At large cell sizes, iterates with a smaller value of ε_γ reach a smaller residual displacement, as the cost coefficients in the flow update can capture the benefit of a rearrangement more accurately.

To sum up, the hybrid scheme iterates converge fast towards the global optimum for a range of initializations and resolutions. When flow updates stop to make progress, there remains a residual displacement on the scale of one or two pixels, that depends on the cell size and the level of resolution used in the edge candidates.

5.3. Flow updates and multiscale domain decomposition

In [7] a multiscale version of domain decomposition was presented and it is straight-forward to combine this with flow updates. However we observed that in this combined scheme flow updates were almost always zero. The rare non-zero updates were usually local and would also have been obtained by the next domain decomposition iteration.

While this observation is somewhat disappointing from the perspective of the flow updates, it once more underlines the efficiency of the multiscale scheme, as observed in [7] where only a fixed number of multiscale iterations was required for solving large problems with high accuracy, where the number of iterations only increases logarithmically with the image size. Freezing does not seem to be an issue in combination with the multiscale scheme. Intuitively it appears plausible that the multiscale scheme does not introduce substantial curl as it moves from coarse to fine scales. A theoretical confirmation of this fact would be an interesting and challenging problem for future work.

5.4. Quantitative analysis and comparison with multiscale domain decomposition

In Section 5.2 a qualitative comparison between basic domain decomposition and the hybrid scheme was given. It was demonstrated that domain decomposition becomes arbitrarily slow on large problems where the

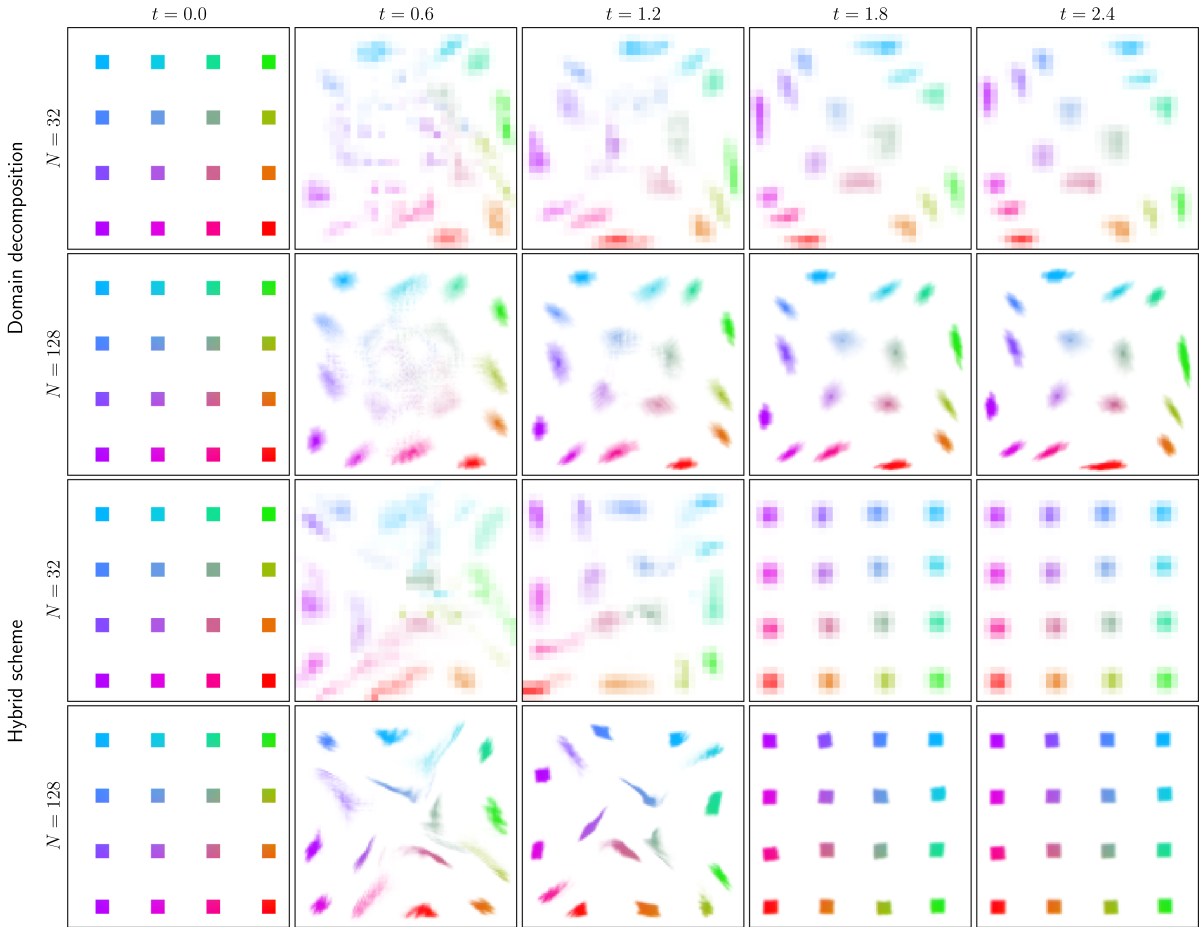

FIGURE 5. Domain decomposition and hybrid trajectories for $s = 2$, $\varepsilon_\gamma = (\Delta x/2)^2$. Domain decomposition alone clearly exhibits freezing. On the other hand, the hybrid scheme achieves n near-optimal configuration at approximately the speed for $N = 32$ and $N = 128$.

initialization contains curl and that the hybrid scheme resolves this issue consistently. In this Section we perform a quantitative comparison between multiscale domain decomposition, single-scale domain decomposition, and the hybrid scheme. We focus on a modified version of the experiment proposed in Figure 4 where we obtain the Y-marginal ν as a rotation of the X marginal μ by an angle θ and a contraction, such that the rotated image fits into $[-1, 1]^2$. This transformation map also serves as non-optimal initialization for the transport plan in the single-scale solvers (domain decomposition and hybrid). An example is given in Figure 7. We monitor the primal score of the algorithms, compute the relative primal-dual gap (using for the dual score the one obtained from multiscale domain decomposition) and plot its evolution with respect to both the iteration number and the running time. The results are summarized in Figure 8.

As before, basic domain decomposition at a single scale decreases the score slowly, the effect becoming more pronounced as N increases (while keeping s fixed). The hybrid scheme decreases the score much more quickly in the fist stage of the algorithm. We then observe the stop of the flow updates and a slow decrease in score as the domain decomposition iterations remove the residual deformations at a similar slope as pure domain

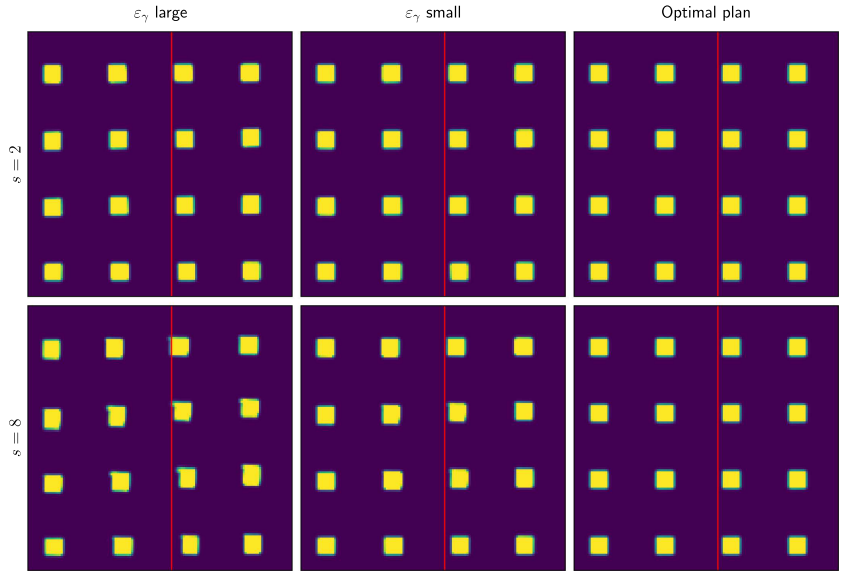

FIGURE 6. Y-marginals of selected basic cells at $t = 4$, after the flow updates have stopped, for $N = 128$ and two different cellsizes s. The red line is drawn in the same location for all the pictures for reference. Note the difference in misalignment for ε_γ large and small (recall that ε_γ refers to the regularization strength used to compute the edge candidates, see Sect. 5.1 for more details).

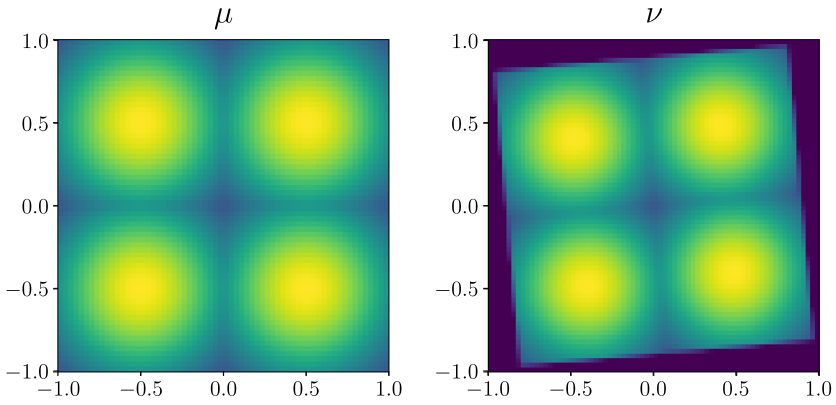

FIGURE 7. Initialization for the experiments in Section 5.4. The Y-marginal is obtained by applying a rotation with angle θ (above, $\theta = 5°$) to the marginal μ. This rotation provides us with an initialization of the transport plan for the single-scale experiments.

decomposition. The multiscale scheme is clearly the strongest algorithm in this comparison. It consistently reaches the smallest PD gap in a short time. This is particularly true for $\theta = 20°$.

For $\theta = 5°$ the initialization π^0 is presumably relatively close to being optimal. In this case we see that the flow updates in the hybrid scheme stop at a time that is comparable with the runtime of the multiscale scheme. The flow updates stop at a higher score, but the relative difference in these scores decreases as the resolution increases. This suggests that flow updates could be a relevant algorithmic alternative when a good approximate

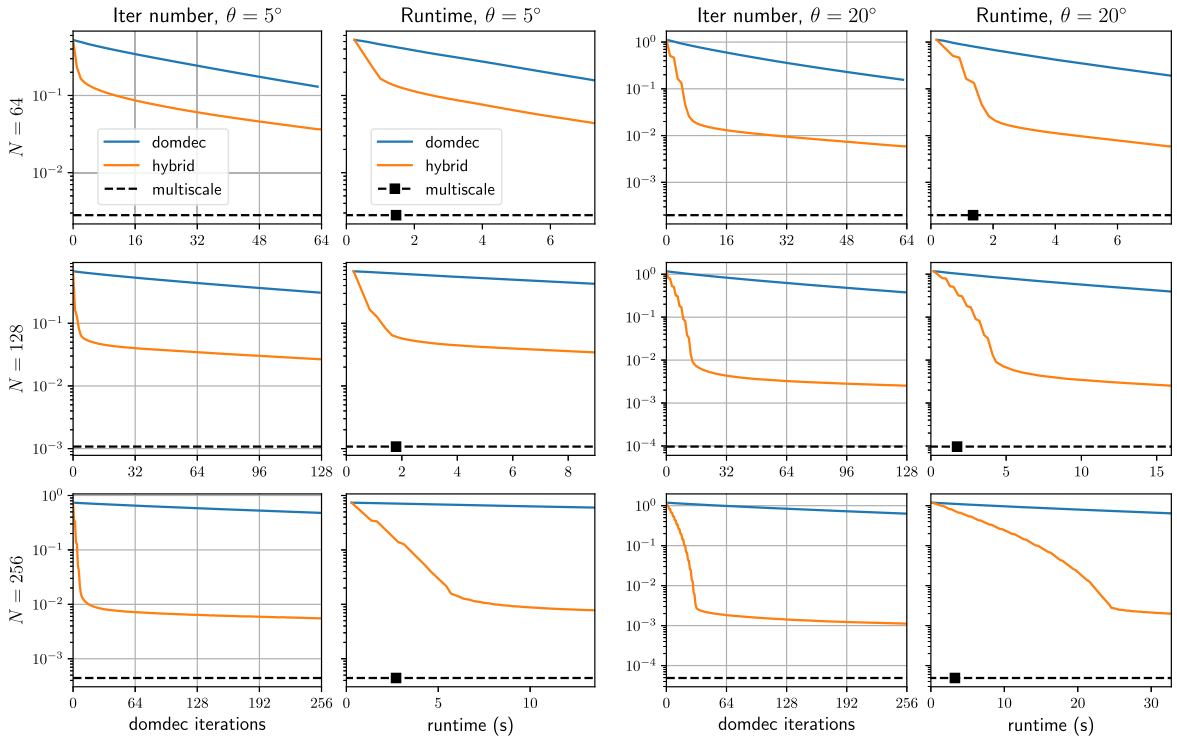

FIGURE 8. Relative primal-dual gap as a function of the iteration count and runtime, for $\theta \in \{5°, 20°\}$ (see Fig. 7), $N \in \{64, 128, 256\}$ and $s = 2$. The black square in the runtime column indicates the runtime of the multiscale scheme.

initial coupling is known and thus improving this initialization could be preferable to re-solving from scratch. However, for this to become fully viable it seems that additional work is required to make flow updates more efficient and more accurate.

6. Conclusion

In this article we introduced flow updates as a remedy for the freezing of singlescale domain decomposition for entropic optimal transport. These updates can be interpreted as an L^∞-variant of the famous Angenent–Haker–Tannenbaum scheme. This specific variant has the advantage that it can be discretized as a min-cost flow problem that exactly preserves the marginals of a discrete transport plan. It is always well-posed (provided the set of basic cells I is finite) and can also operate on transport plans that are not supported on the graph of a map, *e.g.* as in entropic optimal transport. In addition it can be combined in a natural way with domain decomposition, leading to an algorithm that converges to the globally optimal solution.

One limitation is that a global min-cost flow problem on the set of basic cells needs to be solved at each update, implying a trade-off between the size of this problem and its accuracy in predicting favourable flows. We find that far from the optimal solution, flow updates work well, even on coarse grids, whereas closer to the optimal solution some residual suboptimality remains that cannot be resolved by flow updates.

We observe that the hybrid scheme works much faster than single scale domain decomposition and consistently resolves the issue of freezing. We also observe that the multiscale domain decomposition avoids the issue of freezing and in general is faster than the hybrid scheme. The latter could be a viable alternative in cases where a good approximate initial coupling is available.

Methods that solve the optimal transport problem by gradually improving a given initial coupling are conceptually appealing and it would be interesting to study whether flow updates can be improved to become truly competitive with multiscale domain decomposition. Studying variants of the AHT scheme that are well-posed gradient flows on the space of transport plans and reliably reach the globally optimal solution also seems an interesting theoretical question in its own right. Other open questions are whether it can be established theoretically that multiscale domain decomposition avoids freezing, or whether domain decomposition can be applied to multi-marginal transport.

Acknowledgments

IM and BS were supported by the Emmy Noether programme of the DFG. We thank the anonymous referees for their valuable suggestions and comments.

References

[1] S. Angenent, S. Haker and A. Tannenbaum, Minimizing flows for the Monge–Kantorovich problem. *SIAM J. Math. Anal.* **35** (2003) 61–97.
[2] A. Baradat and H. Lavenant, Regularized unbalanced optimal transport as entropy minimization with respect to branching Brownian motion. Preprint arXiv:2111.01666 (2021).
[3] J.-D. Benamou, A domain decomposition method for the polar factorization of vector-valued mappings. *SIAM J. Numer. Anal.* **32** (1994) 1808–1838.
[4] J.-D. Benamou, B.D. Froese and A.M. Oberman, Numerical solution of the optimal transportation problem using the Monge–Ampère equation. *J. Comput. Phys.* **260** (2014) 107–126.
[5] R.J. Berman, The Sinkhorn algorithm, parabolic optimal transport and geometric Monge–Ampère equations. *Numer. Math.* **145** (2020) 771–836.
[6] V.I. Bogachev, Measure Theory. Mathematics and Statistics. Springer Berlin, Heidelberg (2007).
[7] M. Bonafini and B. Schmitzer, Domain decomposition for entropy regularized optimal transport. *Numer. Math.* **149** (2021) 819–870.
[8] M. Bonafini, I. Medina and B. Schmitzer, Asymptotic analysis of domain decomposition for optimal transport (arXiv v1 version). Preprint arXiv:2106.08084v1 (2021).
[9] M. Bonafini, I. Medina and B. Schmitzer, Asymptotic analysis of domain decomposition for optimal transport. *Numer. Math.* **153** (2023) 451–492.
[10] Y. Brenier, Polar factorization and monotone rearrangement of vector-valued functions. *Comm. Pure Appl. Math.* **44** (1991) 375–417.
[11] Y. Brenier, Optimal transport, convection, magnetic relaxation and generalized boussinesq equations. *J. Nonlinear Sci.* **19** (2009) 547–570.
[12] C. Cancès, D. Matthes, I. Medina and B. Schmitzer, Continuum of coupled wasserstein gradient flows. Preprint arXiv:2411.13969 (2024).
[13] G. Carlier and M. Laborde, A differential approach to the multi-marginal Schrödinger system. *SIAM J. Math. Anal.* **52** (2020) 709–717.
[14] G. Carlier, V. Duval, G. Peyré and B. Schmitzer, Convergence of entropic schemes for optimal transport and gradient flows. *SIAM J. Math. Anal.* **49** (2017) 1385–1418.
[15] Y. Chen, T.T. Georgiou and M. Pavon, On the relation between optimal transport and Schrödinger bridges: a stochastic control viewpoint. *J. Optim. Theory App.* **169** (2016) 671–691.
[16] R. Cominetti and J.S. Martin, Asymptotic analysis of the exponential penalty trajectory in linear programming. *Math. Program.* **67** (1992) 169–187.
[17] M. Cuturi, Sinkhorn distances: lightspeed computation of optimal transportation distances, in Advances in Neural Information Processing Systems 26 (NIPS 2013) (2013) 2292–2300.
[18] B. Dezsõ, A. Jüttner and P. Kovács, Lemon – an open source C++ graph template library. *Electron. Theor. Comput. Sci.* **264** (2011) 23–45.
[19] J. Feydy, *Geometric data analysis, beyond convolutions*. Ph.D. thesis, Universit'e Paris-Saclay (2020).
[20] J. Feydy, T. Séjourné, F.-X. Vialard, S.-I. Amari, A. Trouve and G. Peyré, Interpolating between optimal transport and MMD using Sinkhorn divergences, in The 22nd International Conference on Artificial Intelligence and Statistics (2019) 2681–2690.

[21] J. Feydy, A. Glaunès, B. Charlier and M. Bronstein, Fast geometric learning with symbolic matrices, in Advances in Neural Information Processing Systems, edited by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin. Vol. 33. Curran Associates, Inc. (2020) 14448–14462..
[22] P. Ghosal, M. Nutz and E. Bernton, Stability of entropic optimal transport and Schrödinger bridges. *J. Funct. Anal.* **283** (2022) 109622.
[23] M. Harris, Optimizing parallel reduction in cuda. *Nvidia Dev. Technol. Web.* **2** (2007) 70.
[24] IBM ILOG CPLEX, V12.1: User's manual for CPLEX. *Int. Bus. Mach. Corp.* **46** (2009) 157.
[25] J. Kitagawa, Q. Mérigot and B. Thibert, Convergence of a Newton algorithm for semi-discrete optimal transport. *J. Eur. Math. Soc.* **21** (2019) 2603–2651.
[26] H. Lavenant, S. Zhang, Y.-H. Kim and G. Schiebinger, Towards a mathematical theory of trajectory inference. Preprint arXiv:2102.09204 (2021).
[27] C. Léonard, From the Schrödinger problem to the Monge–Kantorovich problem. *J. Funct. Anal.* **262** (2012) 1879–1920.
[28] B. Lévy, A numerical algorithm for L2 semi-discrete optimal transport in 3D. *ESAIM Math. Model. Numer. Anal.* **49** (2015) 1693–1715.
[29] Q. Mérigot, A multiscale approach to optimal transport. *Comput. Graphics Forum* **30** (2011) 1583–1592.
[30] L. Perron and V. Furnon, OR-Tools. https://developers.google.com/optimization/ (2023).
[31] G. Peyré and M. Cuturi, Computational optimal transport. *Found. Trends Mach. Learn.* **11** (2019) 355–607.
[32] F. Santambrogio, Optimal Transport for Applied Mathematicians. Vol. 87 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser Boston (2015).
[33] B. Schmitzer and C. Schnörr, A hierarchical approach to optimal transport, in Scale Space and Variational Methods (SSVM 2013) (2013) 452–464.
[34] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du and L. Guibas, Convolutional Wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graphics (Proc. of SIGGRAPH 2015)* **34** (2015) 66:1–66:11.
[35] C. Villani, Topics in Optimal Transportation. Vol. 58 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence (2003).
[36] C. Villani, Optimal Transport: Old and New. Vol. 338 of *Grundlehren der mathematischen Wissenschaften*. Springer (2009).

Appendix A. A GPU implementation of domain decomposition for entropic optimal transport

An important factor in the adoption of entropic optimal transport has been the availability of fast implementations of the Sinkhorn algorithm on GPUs such as `keops` and `geomloss` [20,21]. It is thus natural to consider using these solvers for the cell subproblems in domain decomposition. However, in each domain decomposition iteration one must solve many small problems with irregular shapes on different domains, which in principle is not a GPU-friendly task.

Section A.1 describes the necessary adaptations to make domain decomposition amenable to GPU computing when the marginals μ and ν live on low-dimensional, regular grids. Section A.2 compares its performance against CPU-based parallel domain decomposition, and against a performant GPU implementation of the global Sinkhorn algorithm.

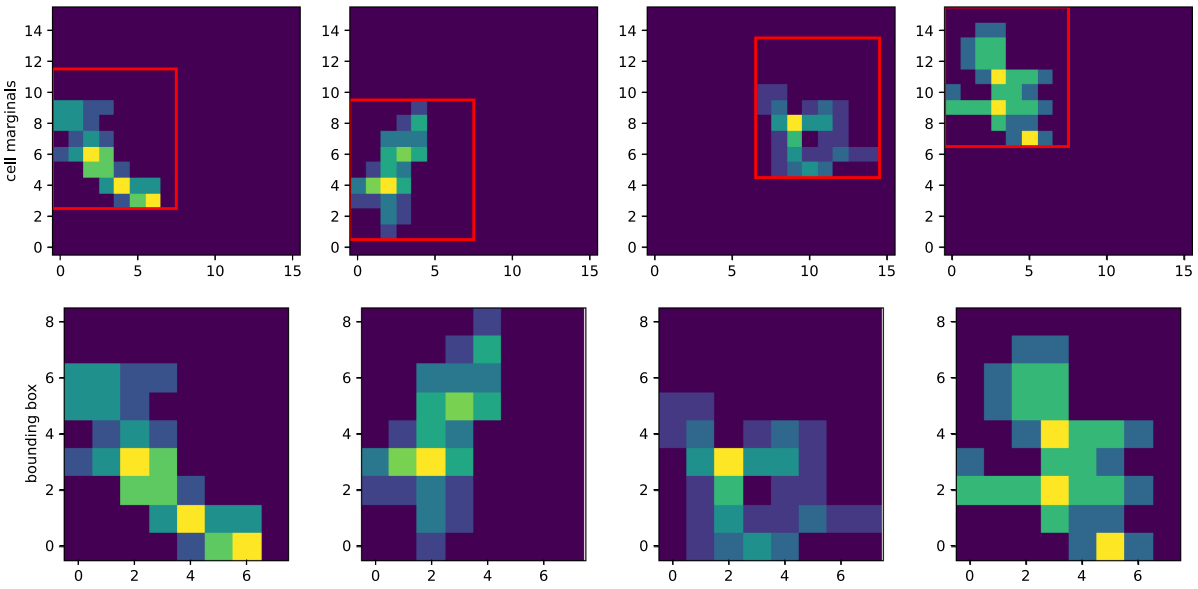

FIGURE A.1. *Top row*: cell marginals $(\nu_i)_{i\in I}$ for $I = \{0, 1, 2, 3\}$ and $Y = \{0, \dots, 15\}^2$. The maximum width and height of their supports is computed, and a red bounding box with these dimensions is pictured. *Bottom row*: using these maximum dimensions we can store the marginals in a compressed and contiguous fashion. The original coordinates of the bottom-left corner of each marginal are saved in the tensor `offsets`.

A.1. Implementation details

The main obstacles for a performant GPU implementation of domain decomposition are the following:

- The subproblems in domain decomposition are numerous, small, and with diverse shapes. In contrast, state-of-the-art implementations of the Sinkhorn algorithm, such as [20, 21], perform best when working with monolithic tensors of large size.
- In previous implementations of domain decomposition, the current cell Y-marginals $(\nu_i)_{i\in I}$ are stored in a sparse structure. However, GPUs stream their operations best on contiguous data.

These issues are addressed by the following adaptations.

Bounding box representation. The basic cell Y-marginals $(\nu_i)_{i\in I}$ are now stored in a *bounding box* structure. This consists of an array `data` of shape $(|I|, s_1, \dots, s_d)$ and an array `offsets` of shape $(|I|, d)$. For each basic cell $i \in I$ the slice `data[`i`]` contains a translated, truncated version of ν_i on a small regular grid of size $s_1 \times \dots \times s_d$, and the offset of this translation is saved in `offsets[`i`]`. This translation is chosen such that the dimensions of the bounding box $(s_1, \dots, s_d)$ are minimal, while containing all points of ν_i above a truncation threshold. Figure A.1 exemplifies this bounding box representation. We refer to the set of basic cell Y-marginals in bounding box structure as ν_I.

Basic cell manipulation. For solving the cell problems in partition $\mathcal{J}$ one must aggregate basic cell marginals into composite cell marginals as follows:

$$\nu_J := \sum_{i\in J} \nu_i \qquad \text{for each } J \text{ in } \mathcal{J}. \tag{A.1}$$

For $(\nu_i)_{i\in I}$ given in the bounding box representation ν_I, one must take the original offsets and widths of each basic cell marginal into account to compute the corresponding composite cell marginals. We implement this operation in a function BASICTOCOMPOSITE, which returns the bounding box representation of the composite cell marginals, denoted by $\nu_{\mathcal{J}}$. In order to obtain a performant implementation we devised a custom CUDA kernel for this task.

The composite cell X-marginals stay constant for any given partition $\mathcal{J}$, and can also be cast into a bounding box structure $\mu_{\mathcal{J}}$.

Custom CUDA Sinkhorn. To achieve the best performance for our particular type of problems, we implement the Sinkhorn reduction directly in CUDA. Our implementation has the following features:

- It is a batched implementation, *i.e.* several problems can be solved in parallel in the same kernel call. As input it takes a set of composite X- and Y-cell marginals $\mu_{\mathcal{J}}$ and $\nu_{\mathcal{J}}$ (in bounding box format), as well as an initialization for the dual potentials on the X-side, $\alpha_{\mathcal{J}}^{\text{init}}$, and returns the optimal dual potentials $\alpha_{\mathcal{J}}$ and $\beta_{\mathcal{J}}$. It can however also be used to solve a single problem.
- Following [20, 21], the Sinkhorn iterations are performed in the logarithmic domain, to avoid issues with numerical instability.
- Again following [20, 21], it is *online*, that is, it computes the transport cost matrix on the fly and does not store it. This allows us to keep the memory use linear in the problem size.
- For the squared cost we provide a *separable* implementation of the Sinkhorn loop, as proposed in [34], which reduces its computational complexity.
- We design it to be particularly performant at solving many small problems in parallel. Since in our case every individual marginal is relatively small (especially in the separable implementation), one can load each dual variable to shared memory and compute each whole line of the Sinkhorn's logsumexp reduction in a single CUDA thread. Since each given cell dual potential is used for multiple Sinkhorn reductions (corresponding to computing each entry of the conjugate dual potential), the GPU's shared memory is used efficiently and communication time is reduced significantly. This contrasts with usual strategies for computing reductions on GPUs [23], which are optimized for large tensors sizes and typically employ several threads processing synchronously different parts of the data, with a subsequent additional synchronization step.

In the following we refer to this implementation as SinkhornGPU.

Online reduction for basic cell marginals. Upon convergence of the subproblems on the composite cells, one must compute the new basic cell Y-marginals ν_I, see (2.4). A naive way to obtain these would be to first instantiate the optimal composite cell plans $\pi_{\mathcal{J}}$ and then perform a reduction on basic cells. However, this would be very memory-intensive. Instead, the partial marginals can be computed directly *via* a partial logsumexp reduction, which corresponds to a Sinkhorn half-iteration on the composite cells, restricted to the basic cells, as follows (for simplicity stated on a discrete problem):

$$\begin{aligned}
\nu_i(\{y\}) &= \sum_{x\in X_i} \pi_J(\{(x,y)\}) = \sum_{x\in X_i} \exp\left(\frac{\alpha_J(x)+\beta_J(y)-c(x,y)}{\varepsilon}\right)\mu(\{x\})\nu(\{y\}) \\
&= \exp\left(\frac{\beta_J(y)}{\varepsilon}\right)\nu(\{y\}) \sum_{x\in X_i} \exp\left(\frac{\alpha_J(x)-c(x,y)}{\varepsilon}\right)\mu(\{x\}) \\
&= \exp\left(\frac{\beta_J(y)-\tilde{\beta}_{J,i}(y)}{\varepsilon}\right)\nu(\{y\}), \\
\text{where } \tilde{\beta}_{J,i}(y) &:= -\varepsilon \log \sum_{x\in X_i} \exp\left(\frac{\alpha_J(x)-c(x,y)}{\varepsilon}\right)\mu(\{x\}).
\end{aligned}$$

This can be done efficiently thanks to the enhancements outlined above. We refer to the function that computes the new basic Y-marginals from the composite ones and the entropic potentials as CompositeToBasic.

Batching and clustering. While the bounding box format for the cell marginals provides a drastic reduction in memory compared to a dense format, it is not as memory efficient as more sophisticated (but less GPU-affine) sparse structures. This effect becomes more severe if a very large number of marginals has to be represented by a common bounding box size: a single large problem can increase the memory demand for all other problems in an unnecessary way. For these two reasons we solve the composite cell problems in batches, and these batches are *clustered* according to bounding box size. This means that fewer composite cell problems need to be held in memory at any given time, and memory bandwidth is used more efficiently due to more appropriate bounding box sizes. An illustration of bounding box clustering and batching is shown in Figure A.5 and a brief discussion of its efficiency is given in Section A.2. Without this batching we would run of of memory for the largest problems considered in our experiments.

We will refer to the function that constructs the batches as CreateBatches. It divides the current partition $\mathcal{J}$ into a set of subpartitions given by $(\tilde{\mathcal{J}}_b)_{b=1}^B$. Note that every batch $\tilde{\mathcal{J}}_b$ of composite cells is associated to a batch of basic cells given by $\tilde{I}_b := \cup_{J\in\tilde{\mathcal{J}}_b} J$. In addition, we implement a function CombineBatches that combines the solutions of the separate batches.

Other components. The features outlined above are specific to adapting domain decomposition to GPU computing. There are other implementation details that are necessary for domain decomposition to be efficient on any architecture, such as coarse-to-fine solving, epsilon scaling, truncation and balancing. All these are described in Section 6 of [7] and can be adapted to the GPU without difficulties, since most steps are parallelizable operations on the basic cell marginals.

Algorithm 4 summarizes a practical implementation of the domain decomposition iteration on the GPU. We refer to this as DomDecGPU.

Algorithm 4. Practical implementation of Algorithm 1 on the GPU.

Input:
– feasible basic Y-marginals ν_I, in bounding box format.
– partition $\mathcal{J}$
– initialization for scaling factors $\alpha_{\mathcal{J}}^{\text{init}}$
Output: new ν_I and $\alpha_{\mathcal{J}}$.

for all $\tilde{\mathcal{J}} \in$ CreateBatches($\mathcal{J}$) **do**
  $\tilde{I} \leftarrow \cup_{J\in\tilde{\mathcal{J}}} J$
  $\nu_{\tilde{\mathcal{J}}} \leftarrow$ BasicToComposite($\nu_I, \tilde{\mathcal{J}}$) // combine basic into composite cells
  $\alpha_{\tilde{\mathcal{J}}}, \beta_{\tilde{\mathcal{J}}} \leftarrow$ SinkhornGPU($\mu_{\tilde{\mathcal{J}}}, \nu_{\tilde{\mathcal{J}}}, \alpha_{\tilde{\mathcal{J}}}^{\text{init}}$) // solve entropic problems with Sinkhorn
  $\nu_{\tilde{I}} \leftarrow$ CompositeToBasic($\mu_{\tilde{I}}, \nu_{\tilde{\mathcal{J}}}, \alpha_{\tilde{\mathcal{J}}}, \beta_{\tilde{\mathcal{J}}}$) // get basic cell Y-marginals
  BalanceGPU($\mu_{\tilde{I}}, \nu_{\tilde{I}}, \tilde{\mathcal{J}}$) // balance measures in each composite cell
  TruncateGPU($\nu_{\tilde{I}}$) // remove negligible mass points
end for
$\nu_I \leftarrow$ CombineBatches($\nu_{\tilde{I}_1}, \ldots, \nu_{\tilde{I}_B}$)
$\alpha_{\mathcal{J}} \leftarrow$ CombineBatches($\alpha_{\tilde{\mathcal{J}}_1}, \ldots, \alpha_{\tilde{\mathcal{J}}_B}$)

A.2. Numerical experiments

A.2.1. Setup

Compared algorithms and implementations. In this Section we compare our GPU domain decomposition (DomDecGPU) to two related algorithms: parallel domain decomposition on the CPU with parallelization *via* MPI (DomDecMPI, using the implementation of [7]) and the Sinkhorn algorithm on the GPU without domain decomposition (SinkhornGPU). The former uses a dense matrix-multiplication Sinkhorn with absorption and automatic ε-fixing, see Section 7.4 of [7] for more details. For the latter we use the implementation SinkhornGPU discussed above, which compares favorably to an implementation based on `keops/geomloss` in all the range of problem sizes considered (see Fig. A.2).

Hardware. We ran our experiments at GWDG's[2] Scientific Compute Cluster, which uses a Linux operating system. We run DomDecMPI on a single compute node with an Intel Xeon Platinum 9242 Processor containing 48 cores (we use 1 master core and 16 working cores). For the GPU algorithms we use a node equipped with an Intel Xeon Gold 6252 Processor with 24 cores (we use 1) and an NVIDIA V100 with 32 GB of memory.

Stopping criterion and measure truncation. We use the L^1 marginal error of the X-marginal after the Y-iteration as stopping criterion for the Sinkhorn iterations. This error is required to be smaller than $\|\mu_{\mathcal{J}}\| \cdot \text{Err}$, where $\mu_{\mathcal{J}}$ gathers the X-marginals of the problems the Sinkhorn solver is handling: an individual problem for DomDecMPI, the global problem for SinkhornGPU, or a collection of problems (possibly all of them) for DomDecGPU. We use $\text{Err} = 10^{-4}$. For comparison, for SinkhornGPU we test in addition the value $\text{Err} = 10^{-3}$. Partial marginals $(\nu_i)_{i\in I}$ are truncated at 10^{-15} and stored as sparse vectors in DomDecMPI, or in bounding box structures in DomDecGPU.

Test data. We use the same problem data as in [7]: Wasserstein-2 optimal transport on 2D images with dimensions $2^n \times 2^n$, with n ranging from $n = 6$ to $n = 11$, *i.e.* images between the sizes 64×64 to 2048×2048. The images are Gaussian mixtures with random variances, means and magnitudes. This represents challenging problem data, featuring

[2] https://gwdg.de/.

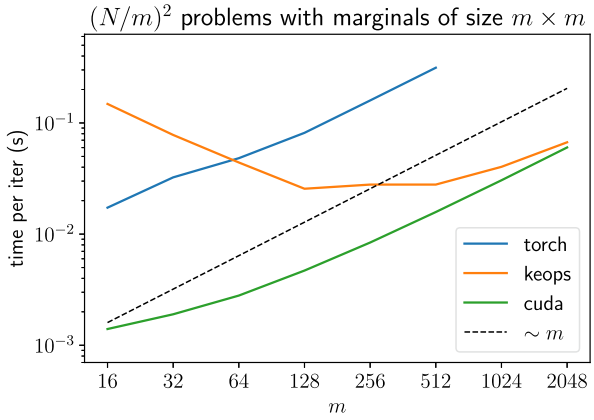

FIGURE A.2. Comparison of Sinkhorn GPU implementations with different backends: Time per Sinkhorn iteration on a batch of problems of the same size. All solvers use log-domain and separable kernels. The `torch` solver is dense, `keops` uses the homonymous library, and `cuda` stands for our custom CUDA implementation. To observe dependence both in batch dimension and in problem size, we solve simultaneously $(N/m)^2$ problems with size m^2, for $N = 2048$. The theoretical complexity grows linearly in m, which is captured by the `torch` solver (with a large, constant overhead). The `keops` solver shows additional overhead for small problem sizes but performs faster on large problems. Our implementation is efficient on large batches of small problems, without compromising its efficiency for large problems.

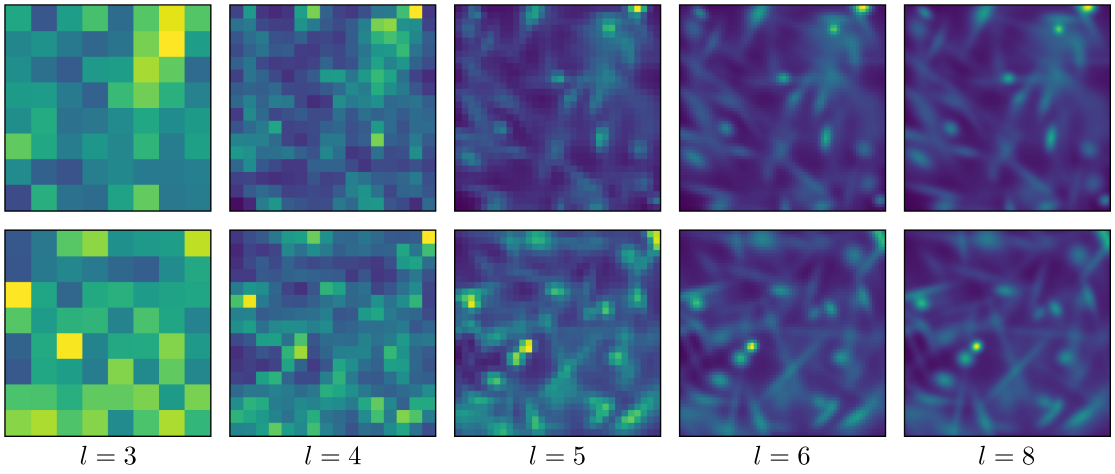

FIGURE A.3. Two example images for size 256×256, at different layers, $l = 8$ being the original.

strong compression, expansion and anisotropies. For each problem size we generated 10 test images, *i.e.* 45 pairwise non-trivial transport problems, and average the results. Examples of the images are shown in Figure A.3.

Multi-scale and ε-scaling. All algorithms implement the same strategy for multi-scale and ε-scaling as in Section 6.4 of [7]. At every multiscale layer, the initial regularization strength is $\varepsilon = 2(\Delta x)^2$, where Δx stands for the pixel size, and the final value is $\varepsilon = (\Delta x)^2/2$. In this way, the final regularization strength on a given multiscale layer coincides with the initial one in the next layer. Finally, on the finest layer ε is decreased further to the value $(\Delta x)^2/4$ implying a relatively small residual entropic blur.

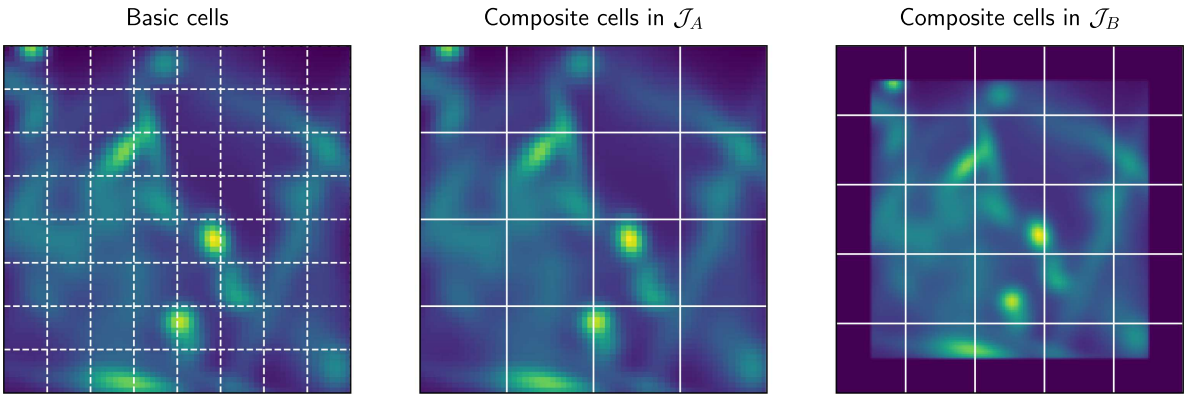

FIGURE A.4. Basic and composite cells for $N = 64$, $s = 8$. For B cells we pad the image with a margin of width s.

Double precision. All algorithms use double floating-point precision. For single precision we observed a degradation in the accuracy for problems with $N \gtrsim 256$. The cause for this behavior is that, for the chosen objective error Err and regularization strengths ε, the update to the dual potentials according to the Sinkhorn iterations eventually has a relative magnitude below the machine precision of single floating-point precision (2^{-24}). Consequently, the Sinkhorn iterations using single precision may stall before achieving the required solution quality.

Basic and composite partitions. We generate basic and composite partitions as in [7]: for the basic partition we divide each image into blocks of $s \times s$ pixels (where s is a divisor of the image size), while composite cells are obtained by grouping cells of the same size, as shown in Figure A.4. In the GPU version, for the B cells we pad the image boundary with a region of very low constant mass, such that all composite cells have the same size, to simplify compatibility. The choice of s entails a performance trade-off. Larger s implies fewer problems and fewer domain decomposition iterations until convergence, but increased complexity in the individual sub-problems. As in [7], the choice of cell size $s = 4$ yields the best results.

Batching and clustering. As anticipated in Section A.1, solving all composite cell problems simultaneously becomes challenging for big resolutions; in fact without further adjustments we run out of memory for $N = 2048$. Therefore we cluster subproblems according to bounding box size and solve these batches sequentially. In Figure A.5 we show an example of this clustering, as well as the dependence of runtime with the number of clusters for several resolutions (computed on a subset of our problem data). We obtained the best results using a single cluster for $N \leq 256$ and then scaling the number of clusters linearly in N. We will use this heuristic criterion for our subsequent experiments.

A.2.2. Results

Runtime. Results of the numerical experiments are summarized in Table A.1 and the total runtimes are visualized in Figure A.6. As can be seen, on large SINKHORNGPU struggles to solve large problems. Increasing the error tolerance for the marginal in the stopping criterion for SINKHORNGPU error shifts the runtime curve but does not change the scaling with respect to N. In contrast, both domain decomposition implementations exhibit a better scaling of the runtime with respect to N. The runtime of DOMDECMPI (with 16 workers) is approximately linear in the number of pixels N^2, as reported [7]. The new implementation DOMDECGPU is consistently faster and by far the fastest algorithm on the largest test problems by approximately one order of magnitude. The optimal transport between two megapixel images is computed in approximately 11 seconds. In the observed range the runtime scales sublinearly in N^2. It seems possible that this is due to the overhead of parallelization (similar to DOMDECMPI on small problems) and that it will transition to approximately linear scaling on even larger problems.

As reported for DOMDECMPI in [7], DOMDECGPU spends most time on solving the cell problems (around 70–80%). The remaining time is mainly spent mostly on handling the cell marginals in the methods BASICTOCOMPOSITE, COMPOSITETOBASIC, and BALANCEGPU. The refinement between multiscale layers and the clustering of the cell problems are relatively cheap steps, using up respectively only 1% and 2% of the total runtime for the finest resolutions.

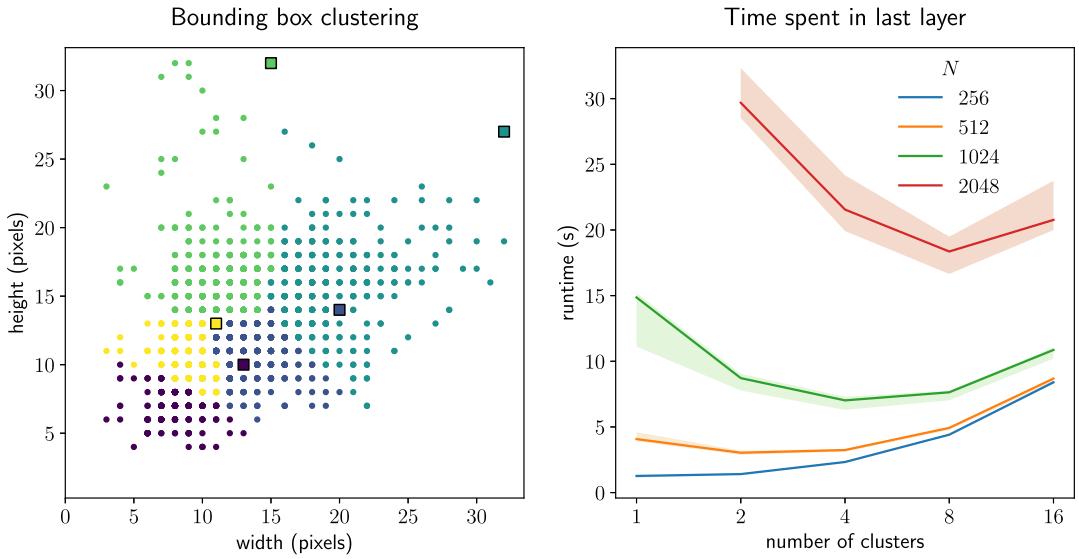

FIGURE A.5. *Left*: every point represents the width and height (in numbers of pixels) of one composite cell Y-marginal, in a problem with $N = 1024$, $s = 4$. The coloring represents the label after performing a K-means clustering. The squared markers indicate the dimensions of the resulting bounding boxes. *Right*: time spent in the last DOMDECGPU multiscale layer, depending of the number of clusters. The solid line represents the median time, while the shading extent represents the range between the 0.25 and 0.75 quantiles. The optimal number of clusters approximately seems to be given by $N/256$.

TABLE A.1. Summary of the average performance for the examined solvers. For each image size results are averaged over 45 problem instances (Sect. A.1).

Image size	64×64	128×128	256×256	512×512	1024×1024	2048×2048
Total runtime (s)						
DOMDECGPU	1.68	2.05	3.08	5.9	11.3	29.3
DOMDECMPI	1.51	3.05	8.52	30.5	130	561
SINKHORNGPU	1.73	2.77	7.08	53.7	483	–
SINKHORNGPU (Err = 10^{-3})	0.407	0.569	1.29	8.43	69.4	–
Time spent in DOMDECGPU's subroutines (s)						
Sinkhorn subsolver	1.21	1.38	2.21	4.56	9.06	22.5
Bounding box processing	0.173	0.275	0.341	0.523	0.992	3.98
Balancing	0.044	0.0752	0.15	0.351	0.644	1.38
Minibatching and clustering	0.105	0.127	0.151	0.183	0.24	0.417
Multiscale refining	2.2e−02	2.9e−02	0.0379	0.0561	0.0985	0.296
DOMDECGPU solution quality						
L^1 marginal error Y	7.5e−12	3.5e−11	1.5e−10	6.1e−10	2.5e−09	1.0e−08
L^1 marginal error X	9.2e−05	9.3e−05	9.3e−05	9.6e−05	9.3e−05	1.1e−04
Relative primal-dual gap	4.2e−05	2.6e−05	3.9e−05	2.0e−05	1.4e−05	1.2e−05
SINKHORNGPU solution quality						
L^1 marginal error X	1.0e−04	1.0e−04	1.0e−04	1.0e−04	1.0e−04	–
Relative dual score	3.3e−05	2.1e−05	3.0e−05	1.5e−05	1.0e−05	–

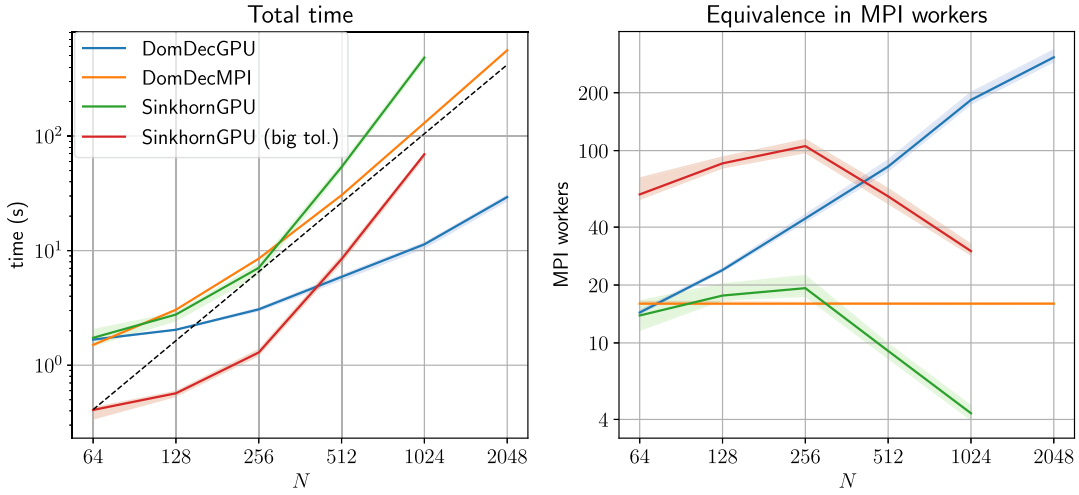

FIGURE A.6. *Left*: runtime comparison between different solvers. The solid line and shading represent the median and the 0.25 and 0.75 quantiles. The SINKHORNGPU solver with higher error tolerance has a looser X-marginal error objective of Err $= 10^{-3}$. The dashed line is proportional to the marginal size, *i.e.* N^2. *Right*: runtimes normalized to the DOMDECMPI runtime to show a rough equivalence of the performance in MPI workers.

Sparsity. In DOMDECMPI the basic cell marginals are stored in a sparse data structure. As outlined above, as GPUs work most efficiently on contiguous data, DOMDECGPU uses a bounding box data structure. While this can still benefit from sparsity, it is not as memory efficient as in DOMDECMPI. As shown in Figure A.1, the bounding boxes inflict some memory overhead that is increasing with the problem size. While the number of non-zero entries increases approximately as the marginal size N^2 (as observed for DOMDECMPI in [7]), the number of stored entries grows faster approximately by a factor $\sqrt{N}$ (Fig. A.7). This is currently the limiting factor for further increasing the problem size for DOMDECGPU. This can presumably be improved by more sophisticated clustering and batching of the cell problems.

Solution quality. We evaluate the solution quality of DOMDECGPU by measuring the violation of the transport plan marginal constraints and with the relative primal-dual gap

$$\frac{E(\pi) - D(\alpha^\dagger, \beta^\dagger)}{D(\alpha^\dagger, \beta^\dagger)}, \quad \text{with} \begin{cases} E(\pi) := \langle c, \pi \rangle + \varepsilon \,\mathrm{KL}(\pi | \mu \otimes \nu), \\ D(\alpha, \beta) := \langle \alpha, \mu \rangle + \langle \beta, \nu \rangle + \varepsilon \big\langle 1 - \exp\big(\frac{\alpha \oplus \beta - c}{\varepsilon}\big), \mu \otimes \nu \big\rangle \end{cases} \tag{A.2}$$

where $\alpha^\dagger$ and $\beta^\dagger$ are the dual domain decomposition potentials, which are obtained by applying the stitching procedure described in Section 6.3 of [7].

For comparison with the SINKHORNGPU solver we use the relative dual score

$$\frac{D(\alpha^*, \beta^*) - D(\alpha^\dagger, \beta^\dagger)}{D(\alpha^\dagger, \beta^\dagger)}, \tag{A.3}$$

where α^* and β^* the final dual potentials obtained by SINKHORNGPU.

We observe that DOMDECGPU generates solutions of high quality. The relative PD gap is always smaller than 10^{-5}. The idealized algorithm would by design perfectly satisfy the Y-marginal constraints. Deviations are purely due to finite floating point precision. Indeed, numerically the violation is very small, growing with the problem size, and is still below 10^{-8} on the largest test problems. The X marginal error is approximately 10^{-4}, consistent with the stopping criterion for the Sinkhorn algorithm on the cell problems.

Upon convergence with Err $= 10^{-4}$, the relative primal dual gap is slightly positive, indicating that the global Sinkhorn solver produced somewhat more precise dual variables. But this deviation is on the order of the PD gap of DOMDECGPU

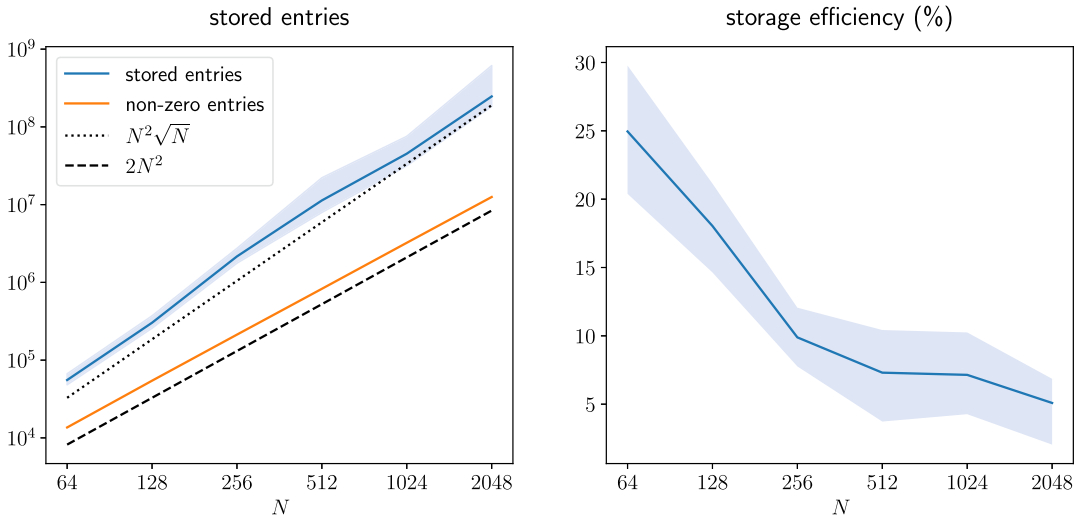

FIGURE A.7. *Left*: number of stored and non-zero entries in the final plan. The number of non-zero entries grows linearly with the problem size N^2, while the dimensions of the bounding box structure grow slightly faster. *Right*: storage efficiency (proportion of stored entries that are positive) over resolution.

and thus not substantial. However, as shown above this small increase in precision comes a a substantial cost in terms of runtime.

Conclusion. Previous work has shown that global Sinkhorn solvers struggle to achieve a good precision in large optimal transport problems, and that domain decomposition is an efficient alternative. Our new GPU implementation of domain decomposition outperforms the previous CPU implementation and a monolithic GPU Sinkhorn implementation. On megapixel-sized images, DOMDECGPU achieves a performance comparable to more than 200 MPI workers using a single GPU. An important task for future work is an improved memory efficiency for the bounding box data structure and subproblem clustering.