Flow–Sinkhorn for Graph Wasserstein–1

Gabriel Peyré
CNRS and ENS, Université PSL
gabriel.peyre@ens.fr

February 3, 2026

Abstract

This note presents the *Sinkhorn–flow* algorithm for approximating the Wasserstein–1 distance on a graph. We start from the unregularized Beckmann (transshipment) linear program, introduce a lifting with two flow variables so that the constraints split into two simple affine blocks, add an entropic (KL) regularization, and derive the resulting cyclic Kullback–Leibler projections. We also provide the dual viewpoint as an alternating block coordinate maximization, and we state an implementation-ready version using stable scaling (and log-domain primitives) to avoid numerical overflow/underflow when the regularization is small. Further details, extensions, and theoretical guarantees can be found in the full article *Robust Sublinear Convergence Rates for Iterative Bregman Projections* (arXiv: https://arxiv.org/abs/2602.01372).

1 Unregularized W_1 on a graph via a flow linear program

Let an undirected connected graph be encoded by a vertex set $V = \{1, \dots, n\}$ and an edge–length matrix $W \in \overline{\mathbb{R}}_+^{n \times n}$ with $W = W^\top$. An entry $W_{i,j} < \infty$ means that (i, j) is an edge of length $W_{i,j}$. Define the edge set and its size

$$E \coloneqq \big\{(i, j) \in V^2 : W_{i,j} < \infty\big\}, \qquad p \coloneqq |E|.$$

Let $D \in \mathbb{R}_+^{n \times n}$ denote the shortest-path (geodesic) distance induced by W (finite since the graph is connected). Given two probability vectors $b_1, b_2 \in \mathbb{R}_+^n$ with $\langle b_1, \mathbb{1}\rangle = \langle b_2, \mathbb{1}\rangle = 1$, the Wasserstein–1 distance associated to D admits a sparse *flow* formulation.

Sparse flows. We consider nonnegative flows supported on E,

$$\mathbb{F} \coloneqq \big\{f \in \mathbb{R}_+^{n \times n} : \ \forall (i, j) \notin E, \ f_{i,j} = 0\big\},$$

where $f_{i,j}$ represents transported mass *from j to i*. Define the discrete divergence operator

$$\mathrm{div}(f) \coloneqq f^\top \mathbb{1} - f\mathbb{1} \in \mathbb{R}^n. \tag{1}$$

Beckmann (transshipment) program. The graph Wasserstein–1 distance equals the optimal value of the linear program

$$\mathrm{W}_D(b_1, b_2) \ = \ \min_{f \in \mathbb{F}} \Big\{\langle W, f\rangle \ : \ \mathrm{div}(f) = b_1 - b_2\Big\}. \tag{2}$$

Because f is supported on E, the effective number of variables is p rather than n^2.

2 Constraint splitting by lifting to two flows

A direct entropic regularization of (2) leads to constraints that are not as easily projected onto in KL. A simple remedy is to duplicate the flow variable and split the constraints into two affine blocks.

Lifted variable. Introduce a pair of flows

$$x \coloneqq (f, g) \in \mathbb{F}^2, \qquad d = 2p.$$

Using this lifting, the unregularized objective can be written as

$$\mathrm{W}_D(b_1, b_2) \;=\; \min_{(f,g)\in\mathcal{C}_1\cap\mathcal{C}_2} \langle W, f\rangle + \langle W, g\rangle, \tag{3}$$

with the two affine constraint sets

$$\mathcal{C}_1 \coloneqq \Big\{(f, g) \in \mathbb{F}^2 :\; -f\mathbb{1} + g^\top\mathbb{1} = b_1 - b_2\Big\}, \tag{4}$$

$$\mathcal{C}_2 \coloneqq \Big\{(f, g) \in \mathbb{F}^2 :\; f = g\Big\}. \tag{5}$$

At feasibility, $\mathcal{C}_2$ forces $f = g$, and $\mathcal{C}_1$ then reduces to $\mathrm{div}(f) = b_1 - b_2$.

3 Entropic regularization and Gibbs kernel on edges

Fix a reference flow $z \in \mathbb{F}$ with strictly positive values on E (and $z_{i,j} = 0$ off E). For $\gamma > 0$, consider the entropically regularized lifted program

$$\min_{(f,g)\in\mathcal{C}_1\cap\mathcal{C}_2} \Big\{\langle W, f\rangle + \langle W, g\rangle + \gamma\,\mathrm{KL}(f \,|\, z) + \gamma\,\mathrm{KL}(g \,|\, z)\Big\}. \tag{6}$$

Here, for a flow $h \in \mathbb{F}$,

$$\mathrm{KL}(h \,|\, z) \coloneqq \sum_{(i,j)\in E} \Big(h_{i,j}\log\frac{h_{i,j}}{z_{i,j}} - h_{i,j} + z_{i,j}\Big).$$

Edgewise Gibbs kernel. As usual, it is convenient to absorb the linear cost into a tilted reference (Gibbs kernel). Define

$$z^C_{i,j} \coloneqq z_{i,j}\exp\Big(-\frac{W_{i,j}}{\gamma}\Big) \quad \text{for } (i,j) \in E, \qquad z^C_{i,j} = 0 \text{ for } (i,j) \notin E. \tag{7}$$

Then (6) is equivalent (up to an additive constant) to the KL projection problem

$$\min_{(f,g)\in\mathcal{C}_1\cap\mathcal{C}_2} \mathrm{KL}(f \,|\, z^C) + \mathrm{KL}(g \,|\, z^C). \tag{8}$$

4 Cyclic KL projections and dual block maximization

4.1 Primal viewpoint: alternating KL projections

Starting from a positive initialization, the regularized problem (8) can be solved by cyclic KL projections:

$$(f^{(k+\frac{1}{2})}, g^{(k+\frac{1}{2})}) = \mathrm{Proj}_{\mathcal{C}_1}\big(f^{(k)}, g^{(k)}\big), \qquad (f^{(k+1)}, g^{(k+1)}) = \mathrm{Proj}_{\mathcal{C}_2}\big(f^{(k+\frac{1}{2})}, g^{(k+\frac{1}{2})}\big),$$

where $\mathrm{Proj}_{\mathcal{C}}$ denotes the minimizer of $\mathrm{KL}(\cdot \,|\, \cdot)$ over $\mathcal{C}$.

A key feature of the present splitting is that both projections admit simple closed forms.

Proposition 1 (Closed-form KL projections for $\mathcal{C}_1$ and $\mathcal{C}_2$)**.** *Let $(h,h) \in \mathbb{F}^2$ with $h_{i,j} > 0$ on E. Then*

$$\mathrm{Proj}_{\mathcal{C}_1}(h,h) = \big(\mathrm{diag}(s)\, h,\ h\ \mathrm{diag}(s)^{-1}\big), \qquad \mathrm{Proj}_{\mathcal{C}_2}(f,g) = \big(\sqrt{f \odot g},\ \sqrt{f \odot g}\big),$$

where $\odot$ denotes the entrywise product and the scaling $s \in \mathbb{R}^n_{++}$ is given componentwise by

$$s = \phi\left(\frac{b_1 - b_2}{h\mathbb{1}}, \frac{h^\top \mathbb{1}}{h\mathbb{1}}\right), \qquad \phi(t,u) := \frac{\sqrt{t^2 + 4u} - t}{2}. \tag{9}$$

One-flow reduction. Because $\mathrm{Proj}_{\mathcal{C}_1}$ maps pairs of equal flows to a pair (f,g) with a simple diagonal left/right scaling, and $\mathrm{Proj}_{\mathcal{C}_2}$ then returns a pair of equal flows, it is natural to track a single flow variable:

$$(f^{(k+1)}, f^{(k+1)}) = \mathrm{Proj}_{\mathcal{C}_2} \circ \mathrm{Proj}_{\mathcal{C}_1}\big(f^{(k)}, f^{(k)}\big). \tag{10}$$

4.2 Dual viewpoint: alternating block coordinate maximization

The same iterations can be seen as alternating maximization of a smooth concave dual objective. Introduce a dual variable $u = (u_1, u_2)$ associated to the two affine blocks $(\mathcal{C}_1, \mathcal{C}_2)$. At the level of general entropic programs, the primal minimizer has an exponential form

$$x(u)_i = z_i^C \exp\Big(\frac{(A^\top u)_i}{\gamma}\Big),$$

and cyclic KL projections correspond exactly to alternating maximization over u_1 and u_2 (one block per step). Specializing to the flow splitting, one convenient parametrization of the resulting one-flow iterate is through a node potential $v^{(k)} \in \mathbb{R}^n$ and its scaling form

$$s^{(k)} := \exp\Big(\frac{v^{(k)}}{\gamma}\Big) \in \mathbb{R}^n_{++},$$

so that

$$f^{(k)} = \mathrm{diag}\big(s^{(k)}\big)\, z^C\ \mathrm{diag}\big(1/s^{(k)}\big), \qquad f^{(k)}_{i,j} = z^C_{i,j} \exp\Big(\frac{v_i^{(k)} - v_j^{(k)}}{2\gamma}\Big). \tag{11}$$

5 Sinkhorn–flow updates and stable implementation

5.1 Scaling-variable update

Define

$$r := \frac{b_1 - b_2}{2} \in \mathbb{R}^n.$$

Combining (10), Proposition 1, and the parametrization (11) yields an explicit update directly on $s^{(k)}$.

Proposition 2 (Flow–Sinkhorn update in scaling variables)**.** *For every $k \geq 0$, the scaling update can be written as*

$$s^{(k+1)} = \sqrt{\frac{s^{(k)}}{z^C\,(1/s^{(k)})} \odot \Big(\sqrt{r^2 + (z^C s^{(k)}) \odot (z^C(1/s^{(k)}))} - r\Big)}. \tag{12}$$

All products and square-roots are entrywise, and matrix-vector products such as $z^C s$ are understood as sparse sums along edges.

5.2 A stable update in log-domain primitives

When γ is small, exponentials may underflow/overflow if one updates $s^{(k)}$ directly. A robust approach is to update $v^{(k)} = \gamma \log s^{(k)}$ using a stable log-sum-exp operator.

Define the log-sum-exp operator (with temperature γ)

$$\mathcal{L}_\gamma(q) \coloneqq \gamma \log \sum_j \exp(q_j/\gamma), \qquad \text{implemented stably as } \ \mathcal{L}_\gamma(q - \max q) + \max q. \tag{13}$$

For each node i, define

$$\alpha_i^{\pm}(v) \coloneqq \mathcal{L}_\gamma(-w_{i,\cdot} \pm v/2), \qquad \beta_i \coloneqq \frac{b_{1,i} - b_{2,i}}{2} \exp\Big(-\frac{\alpha_i^+(v) + \alpha_i^-(v)}{2\gamma}\Big),$$

where $w_{i,\cdot}$ denotes the vector of incident edge lengths from i (restricted to neighbors, consistent with z^C sparsity), and $\operatorname{arsinh}(m) \coloneqq \log(\sqrt{1+m^2} + m)$.

Proposition 3 (Stable node-potential update)**.** *Let $v^{(k)} = \gamma \log s^{(k)}$. Then one can write $v^{(k+1)} = \Psi(v^{(k)})$ with*

$$\Psi(v)_i = -\frac{1}{2} v_i + \frac{1}{2}\big(\alpha_i^+(v) - \alpha_i^-(v)\big) + \gamma \operatorname{arsinh}(\beta_i), \qquad i = 1, \dots, n. \tag{14}$$

5.3 Algorithmic steps (implementation-ready)

We summarize the Sinkhorn–flow iteration using the stable update (14), and recover either $s^{(k)}$ or the flow $f^{(k)}$ when needed.

Algorithm 1: Sinkhorn–flow for graph W_1 (stable form)

Input: Graph (V, E) with lengths $(W_{i,j})_{(i,j)\in E}$; distributions $b_1, b_2 \in \mathbb{R}_+^n$; reference $z \in \mathbb{F}$; regularization $\gamma > 0$; iterations K.

Output: A regularized flow $f^{(K)} \in \mathbb{F}$ and (optionally) an estimate of $\mathrm{W}_D(b_1, b_2)$ via $\langle W, f^{(K)} \rangle$.

Compute z^C on edges by $z^C_{i,j} \leftarrow z_{i,j} \exp(-W_{i,j}/\gamma)$ for $(i,j) \in E$

Set $r \leftarrow (b_1 - b_2)/2$

Initialize a potential $v^{(0)} \in \mathbb{R}^n$ (e.g. $v^{(0)} = 0$)

for $k = 0, 1, \dots, K-1$ **do**

  For each node i, compute $\alpha_i^+(v^{(k)})$ and $\alpha_i^-(v^{(k)})$ using stable $\mathcal{L}_\gamma$ on its neighbors

  For each node i, compute β_i and update

  $v_i^{(k+1)} \leftarrow -\frac{1}{2} v_i^{(k)} + \frac{1}{2}(\alpha_i^+(v^{(k)}) - \alpha_i^-(v^{(k)})) + \gamma \operatorname{arsinh}(\beta_i)$

Set $s^{(K)} \leftarrow \exp(v^{(K)}/\gamma)$ (entrywise)

Recover the flow on edges by $f^{(K)} \leftarrow \operatorname{diag}(s^{(K)})\, z^C \operatorname{diag}(1/s^{(K)})$

return $f^{(K)}$

Computational note. All operations involving z^C and the neighborhood vectors $w_{i,\cdot}$ are sparse: for each node i, sums only run over neighbors $(i,j) \in E$. The update (14) avoids forming extremely small/large exponentials directly and relies on stable log-sum-exp.

Acknowledgements

This work was supported by the European Research Council (ERC project WOLF) and the French government under the management of Agence Nationale de la Recherche as part of the "France 2030" program, reference ANR-23-IACL-0008 (PRAIRIE-PSAI). I would like to thank Giulia Luise, Marco Cuturi, and Bernhard Schmitzer for fruitful discussions that led to the development of the Sinkhorn–flow algorithm.