SIAM J. SCI. COMPUT.
Vol. 43, No. 3, pp. A2295–A2319

FAST ITERATIVE SOLUTION OF THE OPTIMAL TRANSPORT PROBLEM ON GRAPHS*

ENRICO FACCA† AND MICHELE BENZI‡

Abstract. In this paper, we address the numerical solution of the optimal transport problem on undirected weighted graphs, taking the shortest path distance as transport cost. The optimal solution is obtained from the long-time limit of the gradient descent dynamics. Among different time stepping procedures for the discretization of this dynamics, a backward Euler time stepping scheme combined with the inexact Newton–Raphson method results in a robust and accurate approach for the solution of the optimal transport problem on graphs. It is found experimentally that the algorithm requires solving between $\mathcal{O}(1)$ and $\mathcal{O}(m^{0.36})$ linear systems involving weighted Laplacian matrices, where m is the number of edges. These linear systems are solved via algebraic multigrid methods, resulting in an efficient solver for the optimal transport problem on graphs.

Key words. optimal transport problem, gradient descent, implicit time stepping scheme, saddle point problem, algebraic multigrid methods

AMS subject classifications. 65K10, 90B06, 05C21, 65F10

DOI. 10.1137/20M137015X

1. Introduction.

The optimal transport problem (OTP) is a type of optimization problem where the goal is to determine the optimal reallocation of resources from one configuration to another. The OTP is also known as the Monge–Kantorovich problem, after Gaspard Monge, who proposed the first formulation of the problem in [33], and Leonid Kantorovich, who introduced in [24] the relaxed formulation studied nowadays.

In a rather general framework, the Monge–Kantorovich problem can be formulated as follows. Consider a measurable space $\mathcal{X}$, two nonnegative measures f^+ and f^- with equal mass, and a cost function $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $c(x, y)$ describes the cost paid for transporting one unit of mass from x to y. We want to find the optimal plan γ^* in the space of nonnegative measures on the product space $\mathcal{X} \times \mathcal{X}$ (which we denote with $\mathcal{M}_+(\mathcal{X} \times \mathcal{X})$) that solves the following minimization problem:

(1.1)
$$\inf_{\gamma \in \mathcal{M}_+(\mathcal{X}\times\mathcal{X})} \left\{ \int_{\mathcal{X}\times\mathcal{X}} c(x,y)\mathrm{d}\gamma(x,y) \ : \ \begin{array}{c} \text{for all } A, B \text{ measurable subsets of } \mathcal{X} \\ \gamma(A,\mathcal{X}) = f^+(A), \quad \gamma(\mathcal{X},B) = f^-(B) \end{array} \right\}.$$

In recent years, a number of authors have contributed to the development of the OTP theory (see, for example, [40, 3, 46]), and several reformulations of this problem have been found, together with many connections with apparently unrelated areas. More recently, the mathematical "tools" provided by the OTP started to be used in more applied settings. An example of application is the use of the Wasserstein distance (which is the value obtained in the minimization problem (1.1)) in inverse problems [49, 32], image processing [25, 23], and machine learning [21]. These applications have been enabled by novel numerical schemes based on the entropic regularization of

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 28, 2020; accepted for publication (in revised form) April 14, 2021; published electronically June 22, 2021.

https://doi.org/10.1137/20M137015X

†Centro di Ricerca Matematica Ennio De Giorgi, Scuola Normale Superiore, Piazza dei Cavalieri, 3, 56126 Pisa, Italy (enrico.facca@sns.it).

‡Scuola Normale Superiore, Piazza dei Cavalieri, 7, 56126 Pisa, Italy (michele.benzi@sns.it).

OTP (see [38] for an extensive review of the topic). In [2], the authors proved that, when n unknowns are used to discretized f^+ and f^- in problem (1.1), these methods have $\mathcal{O}(\frac{\log(n)}{\varepsilon^3})$ complexity in computing an ε-approximation of the Wasserstein distance and $\mathcal{O}(n^2\frac{\log(n)}{\varepsilon^3})$ complexity in finding an ε-approximation of the solution of problem (1.1). As a consequence, the use of optimal transport tools in applied science is still hindered by the high computation cost when accurate solutions of OTP are sought.

In this paper we focus on the case where the ambient space $\mathcal{X}$ of the OTP is a connected, weighted, undirected, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices and with m edges with strictly positive weights $\boldsymbol{w} \in \mathbb{R}^m$. In particular, we focus on sparse graphs; thus $m = \mathcal{O}(n)$. We consider the cost function c given by the shortest weighted path metric on the graph $\mathcal{G}$ which, given $v_i, v_j \in \mathcal{V}$, is defined as

$$c(v_i, v_j) = \mathrm{dist}_{\mathcal{G},\boldsymbol{w}}(v_i, v_j) = \inf_{\mathcal{P}(v_i,v_j)} \left\{ \sum_{e\in\mathcal{P}(v_i,v_j)} w_e \,:\, \mathcal{P}(v_i, v_j) = \begin{pmatrix} \text{edge-path} \\ \text{from } v_i \text{ to } v_j \end{pmatrix} \right\},$$

a problem often referred to as L^1-OTP. As an example of application, the Wasserstein distance on graphs has been used in [26] to introduce an analogue of Ricci curvature on graphs. This notion has been then used in [42, 36] in the study of community detection on graphs.

In the graph setting the transported measures f^+ and f^- of the OTP are simply two vectors $\boldsymbol{b}^+, \boldsymbol{b}^- \in \mathbb{R}^n_+ = [0, +\infty[^n$ such that $\sum_{i=1}^n b_i^+ = \sum_{i=1}^n b_i^-$. Then problem (1.1) becomes the following linear-programming problem:

$$\inf_{\boldsymbol{\gamma}\in\mathbb{R}^{n,n}_+} \left\{ \mathrm{tr}(\boldsymbol{D}^T\boldsymbol{\gamma}) \;:\; \begin{matrix} \boldsymbol{\gamma}\mathbf{1} = \boldsymbol{b}^+ \\ \mathbf{1}\boldsymbol{\gamma}^T = \boldsymbol{b}^- \end{matrix} \right\}, \tag{1.2}$$

where $\boldsymbol{D}$ is the distance matrix of the graph $\mathcal{G}$, i.e., the symmetric matrix whose (i, j) entry is equal to $\mathrm{dist}_{\mathcal{G},\boldsymbol{w}}(v_i, v_j)$. We are now looking for an optimal matrix $\boldsymbol{\gamma}^* \in \mathbb{R}^{n,n}_+$ (in general there may exist multiple solutions) in which entry $\gamma^*_{i,j}$ describes the portion of "mass" optimally transported from the node v_i to node v_j. Unfortunately, since the number of unknowns scales with $\mathcal{O}(n^2)$, problem (1.2) becomes intractable numerically for graphs with a large number of nodes. Moreover, in the graph setting, there is the additional cost of computing the distance matrix $\boldsymbol{D}$.

On the other hand, when $c = \mathrm{dist}_{\mathcal{G},\boldsymbol{w}}$, the OTP can be rewritten in the equivalent form of an ℓ^1-minimal flow problem, given by

$$\min_{\boldsymbol{q}\in\mathbb{R}^m} \left\{ \sum_{e=1}^m w_e|q_e| \text{ s.t. } \boldsymbol{E}\boldsymbol{q} = \boldsymbol{b}^+ - \boldsymbol{b}^- \right\}, \tag{1.3}$$

where $\boldsymbol{E}$ is the signed incidence matrix of the graph $\mathcal{G}$. This problem has long been known in the study of network flows as the minimum-cost flow problem with no edge capacities. Different approaches like cycle canceling, network simplex, and the Ford–Fulkerson algorithms have been proposed to solve it (see [1] for a complete overview). More recently, the numerical solution of problem (1.3) has been addressed via interior point methods in [31], where it is shown that the problem can be solved on bipartite graphs by solving $\mathcal{O}(m^{3/7\approx 0.428})$ weighted Laplacian systems. The numerical solution of problem (1.3) as OTP on a graph has been studied in [27] using the simplex algorithm. More recently, [14] considered a quadratically regularized version of problem (1.3) in order to deal with the nonuniqueness of the minimal flow.

In this paper, we consider a different approach where we obtain a solution of problem (1.3) by means of an equivalent formulation described in [16], as a minimization problem for an energy functional $\mathcal{L}(\boldsymbol{\mu}) : \mathbb{R}^m_+ \to \mathbb{R}$ with $\boldsymbol{\mu} \in \mathbb{R}^m_+$. Here $\boldsymbol{\mu}$ can be interpreted as a conductivity associated to the edges of the graph. In this paper, a minimum of $\mathcal{L}$ is sought via a gradient descent approach, applied not directly to the functional $\mathcal{L}$ but rather to its composition with the change of variable $\Psi : \mathbb{R}^m \to \mathbb{R}^m_+$ that, acting componentwise, gives $\boldsymbol{\mu} = \Psi(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^2/4$. While minimizing $\tilde{\mathcal{L}} = \mathcal{L} \circ \Psi$ is clearly equivalent to minimizing $\mathcal{L}$, the first shows a much nicer convergence behavior.

The gradient descent approach applied to minimization of $\tilde{\mathcal{L}}(\boldsymbol{\sigma})$ may be seen as finding the vector $\boldsymbol{\sigma}^* = \lim_{t\to+\infty} \boldsymbol{\sigma}(t)$, where $\boldsymbol{\sigma}(t)$ solves the initial value problem

$$\partial_t \boldsymbol{\sigma}(t) = -\nabla \tilde{\mathcal{L}}\left(\boldsymbol{\sigma}(t)\right), \quad \boldsymbol{\sigma}(0) = \boldsymbol{\sigma}_0.$$

We compare in terms of accuracy and efficiency three approaches for the discretization of this ODE: the forward Euler time stepping (the classical gradient descent algorithm), the accelerated gradient descent method described in [34], and the backward Euler time stepping. The last approach results in solving a sequence on nonlinear equations given by

$$\boldsymbol{\sigma}^{k+1} = \boldsymbol{\sigma}^k - \Delta t_{k+1} \nabla \tilde{\mathcal{L}}(\boldsymbol{\sigma}^{k+1}), \quad k = 0, 1, \ldots,$$

iterated until convergence to a steady-state configuration. The above nonlinear equation is solved via an inexact damped Newton–Raphson method that requires the solution of a sequence of saddle point linear systems in the form

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T \\ \boldsymbol{\mathcal{B}} & -\boldsymbol{\mathcal{C}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix}$$

with $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}$ sparse matrices. In contrast, the forward Euler and the accelerated gradient descent both require the solution of only one weighted Laplacian system per step. However, the additional cost is offset by the much smaller number of time steps and linear systems to be solved to reach a steady state hence the backward Euler method turns out to be the most efficient among those considered. We examine different approaches for the solution of the saddle point linear systems. The simplest approach, which exploits matrix $\boldsymbol{\mathcal{C}}$ being diagonal, turns out to be the most efficient, even if it may require using a smaller time step size Δt_k and thus more time steps. In fact, it only requires the solution of a sequence of linear systems involving weighted Laplacian matrices, which can be solved via algebraic multigrid methods. Moreover, the efficiency of this approach can be drastically improved along the time evolution by removing those edges (and the corresponding nodes, if isolated) with conductivity $\boldsymbol{\mu} = \Psi(\boldsymbol{\sigma})$ below a given small threshold.

The resulting method has been tested on different graphs and vectors $\boldsymbol{b}$. In the numerical experiments where exact solutions are available, the numerical method proposed shows a high level of accuracy. The total number of linear systems to be solved, which accounts for almost all of the computation effort and which coincides with the total number of Newton steps, has been found experimentally to range between being constant with respect to number of edges to scaling like $\mathcal{O}(m^{0.36})$ on graphs generated using the Erdös–Rényi, the Watts–Strogatz, and the Barabási–Albert models.

2. OTP on graphs with cost = $\mathbf{dist}_{\mathcal{G},\boldsymbol{w}}$. Before presenting the functional $\mathcal{L}$ we want to minimize, we need to introduce the equivalent formulations of the OTP

addressed in this paper. To this aim, let us introduce some definitions and fix some notations. First, we define the vector

$$\boldsymbol{b} = \boldsymbol{b}^+ - \boldsymbol{b}^-,$$

which will be referred to as the *forcing term*. Then, fixing an orientation on the graph $\mathcal{G}$ (that will have no influence in our formulation) we introduce matrices $\boldsymbol{E} \in \mathbb{R}^{m,n}$, $\boldsymbol{W} \in \mathbb{R}^{m,m}$, and $\boldsymbol{G} \in \mathbb{R}^{n,m}$ defined as

$$E_{e,i} = \begin{cases} 1 & \text{if } i = \text{"head" of edge } e, \\ -1 & \text{if } i = \text{"tail" of edge } e \\ 0 & \text{otherwise,} \end{cases}, \quad \boldsymbol{W} = \operatorname{diag}(\boldsymbol{w}), \quad \boldsymbol{G} = \boldsymbol{W}^{-1}\boldsymbol{E}^T,$$

where we recall that $\boldsymbol{w}$ is the vector having as components the strictly positive weights associated to the graph $\mathcal{G}$. Matrix $\boldsymbol{E}$, which is the signed incidence matrix of the graph $\mathcal{G}$, and matrix $\boldsymbol{G}$ play the role of the (negative) divergence and gradient operators in the weighted graphs setting. Moreover, for any *nonnegative* "conductivity" $\bar{\boldsymbol{\mu}} \in \mathbb{R}^m_+$, we will denote with

$$\boldsymbol{L}[\bar{\boldsymbol{\mu}}] := \boldsymbol{E}\operatorname{diag}(\bar{\boldsymbol{\mu}})\,\boldsymbol{G}$$

the $\bar{\boldsymbol{\mu}}$-weighted Laplacian matrix of the graph $\mathcal{G}$, and with

$$\boldsymbol{\mathcal{U}}[\bar{\boldsymbol{\mu}}] := \boldsymbol{L}[\bar{\boldsymbol{\mu}}]^\dagger \boldsymbol{b}$$

the "potential" induced by the "conductivity" $\bar{\boldsymbol{\mu}}$ and the forcing term $\boldsymbol{b}$. Here and in the following, the symbol † denotes the Moore–Penrose generalized inverse. The action of the pseudoinverse on a given vector is found computing the minimum norm solution of the corresponding (singular) linear system. Note that, since the graph $\mathcal{G}$ is connected and the entries of the forcing term vector $\boldsymbol{b}$ sum to zero, if $\bar{\boldsymbol{\mu}}$ is strictly positive, then $\boldsymbol{\mathcal{U}}[\bar{\boldsymbol{\mu}}]$ is always well defined. If some entries of $\bar{\boldsymbol{\mu}}$ are null, the forcing term $\boldsymbol{b}$ must belong to range of $\boldsymbol{L}[\bar{\boldsymbol{\mu}}]$ for $\boldsymbol{\mathcal{U}}[\bar{\boldsymbol{\mu}}]$ to be well defined.

Equivalent formulations of OTP with cost $\text{dist}_{\mathcal{G},\boldsymbol{w}}$. The first equivalent formulation of the OTP with cost $\text{dist}_{\mathcal{G},\boldsymbol{w}}$ is given by the following minimal-cost flow problem:

$$\min_{\boldsymbol{q}\in\mathbb{R}^m} \left\{ \|\boldsymbol{W}\boldsymbol{q}\|_1 \text{ s.t. } \boldsymbol{E}\boldsymbol{q} = \boldsymbol{b} \right\}, \tag{2.1}$$

where $\|\cdot\|_1$ denotes the ℓ^1-norm. In an electrical flow analogy, we are looking for an optimal current $\boldsymbol{q}^*$ that minimizes the overall total flux on the graph, measured with a weighted ℓ^1-norm, under the constraint that the flux satisfies the Kirchhoff laws $\boldsymbol{E}\boldsymbol{q} = \boldsymbol{b}$.

The second equivalent problem, which is the dual of problem (2.1), is the following maximization problem:

$$\max_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ \boldsymbol{b}^T\boldsymbol{u} \text{ s.t. } \|\boldsymbol{G}\boldsymbol{u}\|_\infty \le 1 \right\}, \tag{2.2}$$

where $\|\cdot\|_\infty$ denotes the ℓ^∞-norm. This last problem is actually a simplified version of the dual problem of the primal problem (1.2) (see [40] for the detailed proof). Problems (2.1) and (2.2) may have multiple solutions.

A third equivalent formulation, which subsumes problems (2.1) and (2.2) and which will be the one we actually solve in this paper, can be introduced by rewriting problem (2.2) as follows:

$$\max_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ \boldsymbol{b}^T\boldsymbol{u} \text{ s.t. } |(\boldsymbol{G}\boldsymbol{u})_e| \le 1 \ \forall e = 1,\dots,m \right\}, \tag{2.3}$$

$$\equiv \max_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ \boldsymbol{b}^T\boldsymbol{u} \text{ s.t. } \frac{w_e}{2}\left(|(\boldsymbol{G}\boldsymbol{u})_e|^2 - 1\right) \le 0 \ \forall e = 1,\dots,m \right\}, \tag{2.4}$$

where we first square both sides of the inequality constraints in (2.3), and then we multiply by the positive factor $w_e/2$ (the reason for these transformations will be discussed in section 3). Now, we can introduce a nonnegative Lagrange multiplier $\boldsymbol{\mu} \in \mathbb{R}^m_+$ for the inequality constraints in (2.4), so that problems (2.2) and (2.4) become equivalent to the following inf-sup problem:

$$\inf_{\boldsymbol{\mu}\in\mathbb{R}^m_+} \sup_{\boldsymbol{u}\in\mathbb{R}^n} \Phi(\boldsymbol{u},\boldsymbol{\mu}) := \boldsymbol{b}^T\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^T\boldsymbol{L}[\boldsymbol{\mu}]\boldsymbol{u} + \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{w} \tag{2.5}$$

with Lagrangian Φ. If we now introduce the functional

$$\mathcal{E}(\boldsymbol{\mu}) = \sup_{\boldsymbol{u}\in\mathbb{R}^n} \left(\boldsymbol{b}^T\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^T\boldsymbol{L}[\boldsymbol{\mu}]\boldsymbol{u}) \right) = \frac{1}{2}\boldsymbol{b}^T\boldsymbol{\mathcal{U}}[\boldsymbol{\mu}] = \frac{1}{2}\boldsymbol{b}^T\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\boldsymbol{b},$$

which is the Joule dissipated energy associated to $\boldsymbol{\mu}$, we may look at problem (2.5) only in terms of the variable $\boldsymbol{\mu}$. Hence, we want to find the optimal $\boldsymbol{\mu}^*$ solving the following minimization problem:

$$\min_{\boldsymbol{\mu}\in\mathbb{R}^m_+} \left\{ \mathcal{L}(\boldsymbol{\mu}) = \mathcal{E}(\boldsymbol{\mu}) + \frac{1}{2}\boldsymbol{w}^T\boldsymbol{\mu} \right\}, \tag{2.6}$$

and then setting $\boldsymbol{u}^* = \boldsymbol{\mathcal{U}}[\boldsymbol{\mu}^*]$. Again, in the electrical network analogy where $\boldsymbol{b}^+$ and $\boldsymbol{b}^-$ describe the electrical charge distributions, the above problem amounts to finding the optimal conductivity $\boldsymbol{\mu}^*$ giving the best trade-off between the Joule dissipated energy $\mathcal{E}(\boldsymbol{\mu})$ and the "infrastructure cost" $\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{w}$. Problem (2.5) is actually a min-max problem since functional $\mathcal{L}(\boldsymbol{\mu})$ admits a minimum $\boldsymbol{\mu}^*$, being convex and since $\lim_{\|\boldsymbol{\mu}\|\to+\infty} \mathcal{L}(\boldsymbol{\mu}) = +\infty$. Then, given a minimizer $\boldsymbol{\mu}^*$, such maximization of the Lagrangian Φ of problem (2.5) is equivalent to finding $\boldsymbol{u}^* = \boldsymbol{\mathcal{U}}[\boldsymbol{\mu}^*]$. Any vector $\boldsymbol{u}^*$ also solves problem (2.2).

The relation among the different optimization problems presented in this section are described in the following proposition.

PROPOSITION 1. *Problems* (2.1), (2.2), *and* (2.6) *are equivalent. This means that the following equalities hold:*

$$\max_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ \boldsymbol{b}^T\boldsymbol{u} \ : \ \|\boldsymbol{G}\boldsymbol{u}\|_\infty \le 1 \right\} = \min_{\boldsymbol{q}\in\mathbb{R}^m} \left\{ \|\boldsymbol{W}\boldsymbol{q}\|_1 \ : \ \boldsymbol{E}\boldsymbol{q} = \boldsymbol{b} \right\} = \min_{\boldsymbol{\mu}\in\mathbb{R}^m \ge 0} \mathcal{L}(\boldsymbol{\mu}).$$

Moreover, given a solution $\boldsymbol{\mu}^*$ that minimizes the functional $\mathcal{L}$ we can recover a maximizer $\boldsymbol{u}^*$ of problem (2.2) *by solving*

$$\boldsymbol{L}[\boldsymbol{\mu}^*]\boldsymbol{u}^* = \boldsymbol{b}, \tag{2.7a}$$

$$|(\boldsymbol{G}\boldsymbol{u}^*)_e| \le 1 \quad \forall e = 1,\dots,m, \tag{2.7b}$$

$$|(\boldsymbol{G}\boldsymbol{u}^*)_e| = 1 \quad \mu^*_e > 0 \tag{2.7c}$$

(the KKT equations for problem (2.2)*). A minimizer* $\boldsymbol{q}^*$ *of problem* (2.1) *can be represented in the form*

$$\boldsymbol{q}^* = \operatorname{diag}(\boldsymbol{\mu}^*)\,\boldsymbol{G}\boldsymbol{u}^*. \tag{2.8}$$

Moreover, the following equality holds:

$$\boldsymbol{\mu}^* = |\boldsymbol{q}^*|.$$

The proof of the equivalence between problems (2.1) and (2.2) is a straightforward application of standard duality results (see [11, section 4, Chapter 3]). We refer the reader to [16] for the proof of the equivalence with the problem of minimizing $\mathcal{L}$. The above proposition states that any optimal flux $\boldsymbol{q}^*$ is induced by the gradient of the potential $\boldsymbol{u}^*$, multiplied by the optimal conductivity $\boldsymbol{\mu}^*$. Note that, for any pair $(\boldsymbol{q}, \boldsymbol{u}) \in \mathbb{R}^m \times \mathbb{R}^n$ satisfying the constraints $\boldsymbol{E}\boldsymbol{q} = \boldsymbol{b}$ and $\|\boldsymbol{G}\boldsymbol{u}\|_\infty \leq 1$, the following inequality holds:

$$\operatorname{DGap}(\boldsymbol{q}, \boldsymbol{u}) := \|\boldsymbol{W}\boldsymbol{q}\|_1 - \boldsymbol{b}^T\boldsymbol{u} \geq 0, \tag{2.9}$$

where equality guarantees the optimality of the pair $(\boldsymbol{q}, \boldsymbol{u}) \in \mathbb{R}^m \times \mathbb{R}^n$. The quantity $\operatorname{DGap}(\boldsymbol{q}, \boldsymbol{u})$ is called the *duality gap*.

It is important to note that any optimal solution $\boldsymbol{\mu}^*$ may be zero on several edges, and there may be nodes where the corresponding rows of the matrix $\boldsymbol{L}[\boldsymbol{\mu}^*]$ is null, leading to a singular but compatible system of equations. This means that there may exist a large null space of $\boldsymbol{L}[\boldsymbol{\mu}^*]$ that may be added to any admissible solution u^* (an optimal potential), as long as the constraints in (2.7b) are satisfied. However, it is easy to identify a base for this subspace, given by the "support" vectors of the node "surrounded" by edges with zero conductivity. Uniqueness of the solution of the primal problem may fail even in case when the dual admits a unique (up to a constant) solution (for example, see Test-case 1 in section 5).

All the discrete problems mentioned in this section find their counterpart in the OTP posed in a domain $\Omega \subset \mathbb{R}^d$ with cost equal to the Euclidean distance. We refer to [16] for a detailed description of these connections between discrete and continuous OTPs.

3. OTP solution via dynamical approaches. Among the different formulations of the OTP presented in the previous section, in this paper we will address the problem of finding a solution $\boldsymbol{\mu}^*$ of the minimization problem (2.6). The solution $\boldsymbol{u}^*$ of problem (2.2) will be derived from the equivalent min-max problem (2.5), while the $\boldsymbol{q}^*$ solution of problem (2.1) will be given by (2.8).

In this paper, the problem of minimizing $\mathcal{L}$ is solved via a gradient descent approach, where we introduced a one-to-one transformation $\Psi : \mathbb{R}^m \to \mathbb{R}^m_+$, acting componentwise, with

$$\mu_e = \Psi(\sigma_e).$$

It is clear that any minimum $\boldsymbol{\mu}^*$ of the functional $\mathcal{L}$ can be recovered from any minimum $\boldsymbol{\sigma}^*$ of the composed functional

$$\tilde{\mathcal{L}}(\boldsymbol{\sigma}) = \mathcal{L}(\Psi(\boldsymbol{\sigma})) = \frac{1}{2}\left(\boldsymbol{b}^T\boldsymbol{L}^\dagger[\Psi(\boldsymbol{\sigma})]\boldsymbol{b} + \boldsymbol{w}^T\Psi(\boldsymbol{\sigma})\right) \tag{3.1}$$

and vice versa. We now differentiate with respect to σ_e the functional in (3.1), obtaining the following expression:

$$\partial_{\sigma_e}\tilde{\mathcal{L}}\left(\boldsymbol{\sigma}\right)=\partial_{\mu_e}\left(\mathcal{L}\left(\boldsymbol{\mu}\right)\right)\Psi'(\sigma_e)=\frac{1}{2}\left(\boldsymbol{b}^T\partial_{\mu_e}\left(\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\right)\boldsymbol{b}+w_e\right)\Psi'(\sigma_e).$$

Using the following matrix identity (see [20, Theorem 4.3.]),

$$\begin{aligned}\partial_{\mu_e}\left(\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\right)=&-\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\partial_{\mu_e}\left(\boldsymbol{L}[\boldsymbol{\mu}]\right)\boldsymbol{L}^\dagger[\boldsymbol{\mu}]+\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\boldsymbol{L}^\dagger[\boldsymbol{\mu}]^T\partial_{\mu_e}\left(\boldsymbol{L}^T[\boldsymbol{\mu}]\right)\left(\boldsymbol{I}-\boldsymbol{L}[\boldsymbol{\mu}]\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\right)\\&+\left(\boldsymbol{I}-\boldsymbol{L}^\dagger[\boldsymbol{\mu}]\boldsymbol{L}[\boldsymbol{\mu}]\right)\partial_{\mu_e}\left(\boldsymbol{L}^T[\boldsymbol{\mu}]\right)\boldsymbol{L}^\dagger[\boldsymbol{\mu}]^T\boldsymbol{L}^\dagger[\boldsymbol{\mu}],\end{aligned}$$

and the fact that $(\boldsymbol{I}-\boldsymbol{L}[\boldsymbol{\mu}]\boldsymbol{L}^\dagger[\boldsymbol{\mu}])\boldsymbol{b}=\boldsymbol{0}$, we obtain the following equation for the gradient of the functional $\tilde{\mathcal{L}}(\boldsymbol{\sigma})$:

$$\nabla\tilde{\mathcal{L}}\left(\boldsymbol{\sigma}\right)=\boldsymbol{w}\odot\Psi'\left(\boldsymbol{\sigma}\right)\odot\frac{1}{2}\left(-\left(\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\Psi(\boldsymbol{\sigma})\right]\right)^2+\boldsymbol{1}\right). \tag{3.2}$$

Similar computations in the evaluation of the gradient of $\mathcal{L}(\boldsymbol{\mu})$ can be found in [16, 9].

In this paper, we use the transformation

$$\Psi(\boldsymbol{\sigma})=\frac{\boldsymbol{\sigma}^2}{4}, \tag{3.3}$$

and we set the gradient descent dynamics in the space $\mathbb{R}^m$ endowed with the scalar product $\langle\boldsymbol{x},\boldsymbol{y}\rangle_{\boldsymbol{w}}:=\boldsymbol{x}^T\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{y}$ and the norm $\|x\|_{\boldsymbol{w}}:=\sqrt{\langle\boldsymbol{x},\boldsymbol{x}\rangle_{\boldsymbol{w}}}$, which is introduced to remove the vector $\boldsymbol{w}$ from the right-hand side of (3.2). Given $\boldsymbol{\sigma}_0>0$, the corresponding initial value problem is given by

$$\partial_t\boldsymbol{\sigma}(t)=-\nabla_{\boldsymbol{\sigma}}\tilde{\mathcal{L}}(\boldsymbol{\sigma})=\boldsymbol{\sigma}(t)\odot\frac{1}{4}\left(\left(\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\Psi(\boldsymbol{\sigma})\right]\right)^2-\boldsymbol{1}\right), \tag{3.4a}$$

$$\boldsymbol{\sigma}(0)=\boldsymbol{\sigma}_0. \tag{3.4b}$$

A minimizer $\boldsymbol{\sigma}^*$ of functional $\tilde{\mathcal{L}}(\boldsymbol{\sigma})$ is sought as the long-time limit of the solution $\boldsymbol{\sigma}(t)$ of (3.4). Solutions of problems (2.1) and (2.6) and (2.7) are then given by

$$\boldsymbol{\mu}^*=\Psi(\boldsymbol{\sigma}^*),\quad \boldsymbol{u}^*=\boldsymbol{\mathcal{U}}\left[\boldsymbol{\mu}^*\right]\quad \boldsymbol{q}^*=\boldsymbol{\mu}^*\odot\boldsymbol{G}\boldsymbol{u}^*.$$

The change of variable Ψ in (3.3) is introduced in order to recover the following dynamical system:

$$\partial_t\boldsymbol{\mu}(t)=\boldsymbol{\mu}(t)\odot\left(\left(\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\boldsymbol{\mu}(t)\right]\right)^2-\boldsymbol{1}\right)=2\boldsymbol{w}^{-1}\odot\boldsymbol{\mu}(t)\odot\left(-\nabla_{\boldsymbol{\mu}}\mathcal{L}(\boldsymbol{\mu}(t))\right), \tag{3.5a}$$

$$\boldsymbol{\mu}(0)=\boldsymbol{\mu}_0>\boldsymbol{0}. \tag{3.5b}$$

The steady-state solution $\boldsymbol{\mu}^*=\lim_{t\to+\infty}\boldsymbol{\mu}(t)$ of (3.5) and its variants have been related to different types of optimization problems, like the shortest path problem [45, 10], the basis pursuit problem [16, 9], and different transport problems in a continuous setting [17, 15]. In [9] it has been shown that the dynamical system in (3.5) can be interpreted as mirror descent dynamics for the functional $\mathcal{L}$, where the descent direction is given by the product of the vector $-\nabla_{\boldsymbol{\mu}}\mathcal{L}(\boldsymbol{\mu}(t))$ and the vector $\boldsymbol{\mu}(t)$ (the factor $2\boldsymbol{w}^{-1}$ has no particular influence on the dynamics). Due to the presence of the multiplicative term $\boldsymbol{\mu}(t)$, the right-hand side of (3.5a) tends to zero on those entries where $\boldsymbol{\mu}(t)$ tends to zero. The numerical results presented in

the above-mentioned papers suggest that this scaling plays a crucial role in the good behavior of the numerical methods based on the dynamical system (3.5). However, using the transformation in (3.3) we can reinterpret (3.5) as a "classical" gradient descent dynamics. In fact, if we multiply componentwise (3.4a) by $\frac{1}{2}\boldsymbol{\sigma}(t)$, we obtain

$$\partial_t \boldsymbol{\mu}(t) = \frac{1}{2}\boldsymbol{\sigma}(t) \odot \partial_t \boldsymbol{\sigma}(t) = \frac{1}{2}\boldsymbol{\mu}(t) \odot \left((\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\Psi(\boldsymbol{\sigma})\right])^2 - \mathbf{1} \right),$$

which is equal to (3.5a). Nevertheless, in (3.4a), we keep scaling the gradient direction $(\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\boldsymbol{\mu}(t)\right])^2 - \mathbf{1}$ by a multiplicative factor $\boldsymbol{\sigma}(t)$ which tends to zero on the same entries of $\boldsymbol{\mu}(t)$, since $\boldsymbol{\sigma}(t) = \sqrt{4\boldsymbol{\mu}(t)}$.

Remark 3.1. Other transformations, like

$$\Psi(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^p, \quad p \geq 1, \quad \text{and} \quad \Psi(\boldsymbol{\sigma}) = e^{\boldsymbol{\sigma}},$$

have been considered in our experiments. Using the identity transformation $\boldsymbol{\mu} = \Psi(\boldsymbol{\sigma}) = \boldsymbol{\sigma}$, we recover problem (2.6), where we face the problem of ensuring the positivity of $\boldsymbol{\mu}$ along the gradient descent dynamics. The positivity is automatically guaranteed taking $\boldsymbol{\mu} = \Psi(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^p$ for $p > 1$ and $\Psi(\boldsymbol{\sigma}) = e^{\boldsymbol{\sigma}}$. Using the exponential transformation $\Psi(\boldsymbol{\sigma}) = e^{\boldsymbol{\sigma}}$, the gradient flow equation in the variable $\boldsymbol{\sigma}$ and the corresponding $\boldsymbol{\mu}$ dynamics are given by

$$\begin{aligned} \partial_t \boldsymbol{\sigma}(t) = &= \exp\left(\boldsymbol{\sigma}(t)\right) \odot \left((\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\boldsymbol{\Psi}\boldsymbol{\sigma}(t)\right])^2 - \mathbf{1} \right) = -\nabla_{\boldsymbol{\sigma}} \tilde{\mathcal{L}}(\boldsymbol{\sigma}), \\ \partial_t \boldsymbol{\mu}(t) &= (\boldsymbol{\mu}(t))^2 \odot \left((\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\boldsymbol{\mu}(t)\right])^2 - \mathbf{1} \right). \end{aligned}$$

However, the term $(\boldsymbol{\mu}(t))^2$ on the right-hand side of the previous equation becomes small for those entries converging to zero along the dynamics. This squared term slows the dynamics so much that only after several large time steps we converge to the optimal solution, compromising the efficiency of the algorithm. We mention that, among the transformations $\Psi(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^p$ for $p > 1$, the squared transformation in (3.3) is the one giving the best results in terms of global efficiency and robustness.

4. Numerical solution.

4.1. Time discretization. In this section, we describe three approaches for the time discretization of (3.4) and to get a steady-state solution, i.e., we show how to build an approximate sequence $(\boldsymbol{\sigma}^k)_{k=1,\dots,k_{\max}}$ such that

$$\left\| \nabla_{\boldsymbol{\sigma}} \tilde{\mathcal{L}}(\boldsymbol{\sigma}^k) \right\|_{\boldsymbol{w}} = \left\| \sqrt{\boldsymbol{w}} \odot \boldsymbol{\sigma}^k \odot \frac{1}{2}(\left|\boldsymbol{G}\boldsymbol{u}^k\right|^2 - \mathbf{1}) \right\|_2 \leq \hat{\epsilon}, \tag{4.1}$$

where $\hat{\epsilon} > 0$ is a fixed tolerance. We denote with $\hat{k}$ the step at which (4.1) is satisfied, writing $(\hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{u}}) = (\boldsymbol{\sigma}^{\hat{k}}, \boldsymbol{u}^{\hat{k}})$. Our approximate solutions of problems (2.1), (2.2), and (2.6) will be given, respectively, by

$$(\hat{\boldsymbol{q}} = \hat{\boldsymbol{\mu}} \odot \boldsymbol{G}\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\mu}}) \quad \text{with} \quad \hat{\boldsymbol{\mu}} = \Psi(\hat{\boldsymbol{\sigma}}).$$

Forward Euler/gradient descent. The first approach is the forward Euler time-stepping, resulting in the classical *gradient descent* (GD) algorithm, where, given a sequence $\Delta t_k > 0$, the approximation sequence $(\boldsymbol{\sigma}^k)_{k=1,\dots,k_{\max}}$ is given by the following equation:

$$\boldsymbol{\sigma}^{k+1} = \boldsymbol{\sigma}^k + \Delta t_k \frac{\boldsymbol{\sigma}^k}{4} \odot \left((\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\Psi(\boldsymbol{\sigma}^k)\right])^2 - \mathbf{1} \right), \quad k = 1, \dots, k_{\max}.$$

At each time step, we only need to solve the linear system

$$\boldsymbol{L}[\boldsymbol{\mu}^k]\boldsymbol{u}^k = \boldsymbol{b}.$$

A similar discretization scheme, applied to (3.5), has been successfully adopted in [9, 17, 16, 18, 19].

Accelerated gradient descent. The second approach considered in this paper is the *accelerated gradient descent (AGD)*. In the AGD approach, first introduced in [34], the approximating sequence is generated as follows:

$$\begin{aligned}\boldsymbol{\sigma}^{k+1} &= \boldsymbol{y}^k - \Delta t_k \nabla_{\boldsymbol{\sigma}} \tilde{\mathcal{L}}(\boldsymbol{y}^k),\\ \boldsymbol{y}^k &= \boldsymbol{\sigma}^k + \frac{k-1}{k+2}(\boldsymbol{\sigma}^k - \boldsymbol{\sigma}^{k-1}).\end{aligned}$$

Typically, the AGD performs better than GD, reaching steady state with fewer time steps (we refer to [35] for a complete overview on the AGD approach and on its applications).

In our problem, GD and AGD have practically the same computational cost, requiring the solution of only a weighted Laplacian matrix per time step. However, both the GD and the AGD are explicit time stepping schemes as shown in [41], and thus, in both approaches, the time step size Δt_k cannot be taken too large, due to well-known stability issues. Hence, a very large number of time steps may be required to reach the optimum, affecting the performance of the algorithm. On the other hand, implicit schemes do not suffer from this time step size limitation, at the cost of solving a non-linear equation. In this paper, we will see that, in our formulation, the additional computational cost required at each time step is offset by a drastic reduction of the total number of time steps required to reach the steady-state solution. Among implicit time stepping schemes we adopt backward Euler, since it provides the largest stability region [44]. In fact, we are not interested in approximating the solution curve $\boldsymbol{\sigma}(t)$ but rather in converging toward the steady state $\boldsymbol{\sigma}^* = \lim_{t\mapsto+\infty}\boldsymbol{\sigma}(t)$ in fewer time steps.

Backward Euler-gradient flow. We present the backward Euler time stepping of (3.4), in the form of finding a sequence $(\boldsymbol{\sigma}^k)_{k=1,\dots,k_{\max}}$ that, given a sequence of Δt_k and fixing $\boldsymbol{\sigma}^0 = \boldsymbol{\sigma}_0$, solves the following minimization problem:

$$\boldsymbol{\sigma}^{k+1} = \underset{\boldsymbol{\sigma}\in\mathbb{R}^m}{\operatorname{argmin}} \left\{ \tilde{\mathcal{L}}(\boldsymbol{\sigma}) + \frac{1}{2\Delta t_k}\left\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^k\right\|_{\boldsymbol{w}}^2 \right\}. \tag{4.2}$$

Readers familiar with optimization techniques will recognize in (4.2) the *proximal mapping*, while readers familiar with the discretization of ODEs will recognize the classical backward Euler discretization by casting the Euler–Lagrange equations of (4.2) in the form

$$\frac{(\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^k)}{\Delta t_k} = -\boldsymbol{W}^{-1}\nabla_{\boldsymbol{\sigma}}\tilde{\mathcal{L}}(\boldsymbol{\sigma}^{k+1}) = \frac{1}{4}\boldsymbol{\sigma}^{k+1} \odot \left((\boldsymbol{G}\boldsymbol{\mathcal{U}}\left[\Psi(\boldsymbol{\sigma}^{k+1})\right])^2 - \mathbf{1}\right).$$

However, it is better to rewrite (4.2) in the form of an inf-sup problem, using the definition of $\mathcal{L}$ in problems (2.5) and (2.6). In fact, the Lagrangian functional Φ in problem (2.5) in the $\boldsymbol{\sigma}$-variable becomes

$$\tilde{\Phi}(\boldsymbol{u},\boldsymbol{\sigma}) = \Phi(\boldsymbol{u},\Psi(\boldsymbol{\sigma})) = \boldsymbol{b}^T\boldsymbol{u} - \frac{1}{2}\boldsymbol{u}^T\boldsymbol{L}[\Psi(\boldsymbol{\sigma})]\boldsymbol{u} + \frac{1}{2}\Psi(\boldsymbol{\sigma})^T\boldsymbol{w}; \tag{4.3}$$

hence we look for the solution of the following problem:

(4.4)
$$(\boldsymbol{u}^{k+1}, \boldsymbol{\sigma}^{k+1}) = \underset{\boldsymbol{u}\in\mathbb{R}^n}{\operatorname{argmax}} \underset{\boldsymbol{\sigma}\in\mathbb{R}^m}{\operatorname{argmin}} \left\{ \tilde{\Gamma}(\boldsymbol{u}, \boldsymbol{\sigma}, \boldsymbol{\sigma}^k, \Delta t_k) := \tilde{\Phi}(\boldsymbol{u}, \boldsymbol{\sigma}) + \frac{1}{2\Delta t_k} \left\| \boldsymbol{\sigma} - \boldsymbol{\sigma}^k \right\|_{2,\boldsymbol{w}}^2 \right\}.$$

The solution pair $(\boldsymbol{u}^{k+1}, \boldsymbol{\sigma}^{k+1})$ can be found as the stationary point of the functional $\tilde{\Gamma}$ in (4.4), described by the following nonlinear system of equations:

(4.5)
$$0 = \begin{pmatrix} -\nabla_{\boldsymbol{u}}\tilde{\Gamma} \\ -\nabla_{\boldsymbol{\sigma}}\tilde{\Gamma} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{\sigma}, \boldsymbol{\sigma}^k, \Delta t_k) \\ \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{\sigma}, \boldsymbol{\sigma}^k, \Delta t_k) \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}[\Psi(\boldsymbol{\sigma})]\boldsymbol{u} - \boldsymbol{b} \\ \boldsymbol{w} \odot \left(\frac{1}{4}\boldsymbol{\sigma} \odot \left((\boldsymbol{G}\boldsymbol{u})^2 - \mathbf{1} \right) - \frac{1}{\Delta t_k} \left(\boldsymbol{\sigma} - \boldsymbol{\sigma}^k \right) \right) \end{pmatrix}.$$

At each time step k, we apply a damped version of the inexact Newton–Raphson method to (4.5) that, given $0 < \alpha \leq 1$, amounts to finding the approximating sequence $(\boldsymbol{u}^{k,r}, \boldsymbol{\sigma}^{k,r})_{r=1,\ldots,r_{\max}}$ with $(\boldsymbol{u}^{k,1}, \boldsymbol{\sigma}^{k,1}) = (\boldsymbol{u}^k, \boldsymbol{\sigma}^k)$ and defined as follows:

$$\boldsymbol{\mathcal{J}} \left[\boldsymbol{u}^{k,r}, \boldsymbol{\sigma}^{k,r}, \Delta t_k \right] \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{f}(\boldsymbol{u}^{k,r}, \boldsymbol{\sigma}^{k,r}, \boldsymbol{\sigma}^k, \Delta t_k) \\ \boldsymbol{g}(\boldsymbol{u}^{k,r}, \boldsymbol{\sigma}^{k,r}, \boldsymbol{\sigma}^k, \Delta t_k) \end{pmatrix}, \tag{4.6}$$

$$\begin{pmatrix} \boldsymbol{u}^{k,r+1} \\ \boldsymbol{\sigma}^{k,r+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}^{k,r} \\ \boldsymbol{\sigma}^{k,r} \end{pmatrix} + \alpha \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix}, \tag{4.7}$$

where matrix $\boldsymbol{\mathcal{J}}$ is the Jacobian matrix of the function $(\boldsymbol{f}, \boldsymbol{g})$. At each Newton step we find an approximate solution of the linear system in (4.6) such that

$$\left\| \boldsymbol{\mathcal{J}} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} + \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix} \right\|_2 \leq \epsilon \left\| \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix} \right\|_2 \tag{4.8}$$

with $\epsilon > 0$, iterated until

$$\left\| \begin{pmatrix} \boldsymbol{f}/\|b\| \\ \boldsymbol{g} \end{pmatrix} \right\|_2 \leq \epsilon_N, \tag{4.9}$$

with $\epsilon_N > 0$ (we scale by $\|b\|$ in order to get a relative residual in the solution of the linear system that defines the function $\boldsymbol{f}$). The damping parameter should be kept ideally $\alpha = 1$ in order to get the quadratic convergence of the Newton method. However, smaller α is used in order to prevent failures of the linear and the nonlinear solvers.

The Jacobian matrix $\boldsymbol{\mathcal{J}}$ in (4.6) can be calculated as follows:

$$\boldsymbol{\mathcal{J}} \left[\boldsymbol{u}, \boldsymbol{\sigma}, \Delta t \right] = \begin{pmatrix} \boldsymbol{\mathcal{A}} \left[\boldsymbol{\sigma} \right] & \boldsymbol{\mathcal{B}}^T \left[\boldsymbol{u}, \boldsymbol{\sigma} \right] \\ \boldsymbol{\mathcal{B}} \left[\boldsymbol{u}, \boldsymbol{\sigma} \right] & -\boldsymbol{\mathcal{C}} \left[\Delta t, \boldsymbol{u}, \boldsymbol{\sigma} \right] \end{pmatrix}, \tag{4.10}$$

where

$$\begin{aligned} \boldsymbol{\mathcal{A}} \left[\boldsymbol{\sigma} \right] &= \boldsymbol{L}[\Psi(\boldsymbol{\sigma})] = \boldsymbol{E} \operatorname{diag} \left(\Psi(\boldsymbol{\sigma}) \right) \boldsymbol{G}; \\ \boldsymbol{\mathcal{B}} \left[\boldsymbol{u}, \boldsymbol{\sigma} \right] &= \frac{1}{2} \operatorname{diag} \left((\boldsymbol{G}\boldsymbol{u}) \odot \boldsymbol{\sigma} \right) \boldsymbol{E}^T, \\ \boldsymbol{\mathcal{C}} \left[\Delta t, \boldsymbol{u}, \boldsymbol{\sigma} \right] &= \operatorname{diag} \left(\boldsymbol{w} \odot \left(\frac{1}{4} \odot \left((\boldsymbol{G}\boldsymbol{u})^2 - \mathbf{1} \right) + \frac{1}{\Delta t} \right) \right). \end{aligned}$$

We will refer to the procedure described in this section as the *gradient flow* (GF) approach, as in [4].

Remark 4.1. Here, it is worth highlighting some similarities between the GF approach proposed in this paper and the interior point methods for the solution of minimization problems with nonlinear constraints [48]. In fact, the functional $\mathcal{L}$ is nothing but the Lagrangian functional of the dual problem of problem (2.2), where the constraints are rewritten as in (2.3) and (2.4). In interior point methods, this Lagrangian functional is relaxed with a penalization functional of the constraints (typically a logarithmic barrier function), scaled by a relaxation factor $\nu > 0$, that is progressively reduced along the optimization process. In our approach the relaxation functional is the distance square term scaled by the factor $\frac{1}{2\Delta t}$, which plays the role of the factor ν. In both schemes, the stationary equations for the scaled functional are solved via inexact Newton. Moreover the sequence of approximated solutions is following in both cases an underlying "curve," the integral solution of the ODE (3.4) for the GD equation, and the central path of the interior point methods.

4.2. Linear system solution. All the discretization schemes described in subsection 4.1 require the solution of a sequence of sparse linear systems. In the GD and the AGD approaches, these linear systems only involve weighted Laplacian matrices; thus algebraic multigrid solvers can efficiently handle these problems, often with computational time that scales linearly with the size of the matrix. In particular, in our experiments we adopt the aggregation-based algebraic multigrid (AGMG) software described in [37] and the lean algebraic multigrid (LAMG) described in [28], which has been developed specifically for the solution of linear systems involving weighted Laplacian matrices. The GF approach instead requires the solution of the saddle point linear system in (4.8). We refer to [6] for a complete overview on this topic. We propose two approaches to tackle this linear algebra problem.

Reduced-multigrid (reduced-MG) approach. The first approach proposed in this paper exploits the fact that the matrix $\boldsymbol{\mathcal{C}}$ is diagonal; thus it can be trivially inverted under some restriction on the time step size Δt. The resulting reduced linear system reads as

$$\overbrace{\left(\boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}^T \boldsymbol{\mathcal{C}}^{-1} \boldsymbol{\mathcal{B}}\right)}^{\boldsymbol{S_{\mathcal{C}}}} \boldsymbol{x} = \boldsymbol{f} + \boldsymbol{\mathcal{B}}^T \boldsymbol{g} = \bar{\boldsymbol{f}},$$
$$\boldsymbol{y} = \boldsymbol{\mathcal{C}}^{-1} \left(\boldsymbol{\mathcal{B}} \boldsymbol{x} - \boldsymbol{g}\right),$$

where matrix $\boldsymbol{S_{\mathcal{C}}}$ is nothing but the Schur complement of the block $\boldsymbol{\mathcal{C}}$ of the matrix $\boldsymbol{\mathcal{J}}$, which can be rewritten in the form

$$\boldsymbol{S_{\mathcal{C}}} = \boldsymbol{E} \mathrm{diag}\left(\bar{\boldsymbol{\mu}}\right) \boldsymbol{W}^{-1} \boldsymbol{E}^T = \boldsymbol{L}[\bar{\boldsymbol{\mu}}] \tag{4.11}$$

with $\bar{\boldsymbol{\mu}}$ given by

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu} + \Delta t \frac{\left(\frac{1}{2}\boldsymbol{\sigma} \odot (\boldsymbol{G}\boldsymbol{u})\right)^2}{\mathbf{1} - \frac{\Delta t}{4}((\boldsymbol{G}\boldsymbol{u})^2 - \mathbf{1})}.$$

As long as $\bar{\boldsymbol{\mu}}$ is positive, matrix $\boldsymbol{S_{\mathcal{C}}}$ in (4.11) takes the form of a weighted Laplacian matrix; thus it can be solved efficiently via algebraic multigrid solvers. We denote this approach with *reduced-MG*. The downside of the reduced-MG approach is that we are forced to impose a restriction on the time step size Δt_k; thus more time steps are required to reach steady state. In fact, in order to preserve positivity of $\bar{\boldsymbol{\mu}}$ and to

avoid division by numbers close to zero, at each time step, before starting the Newton process, we choose Δt_k to satisfy the constraint

$$\max_{e=1,\dots,m} \left\{ \mathcal{C}_{e,e} = \frac{1}{\Delta t_k} - (\boldsymbol{G}\boldsymbol{u})_e^2 + 1 \right\} \geq \varepsilon_{\boldsymbol{\mathcal{C}}} > 0. \tag{4.12}$$

Then, the damping parameter α in (4.7) is tuned so that (4.12) holds at each step of the Newton process. We start from $\alpha = 1$ (which corresponds to the classical Newton method), and we progressively reduce α until condition (4.12) is met. If α becomes smaller than a prefixed lower bound α_0 (which in our experiments is fixed at $\alpha_0 = 5 \times 10^{-2}$) we abort the Newton procedure, restarting it after halving the time step size Δt_k. Although having $\alpha < 1$ destroys the quadratic convergence of the Newton–Raphson approach, our numerical experiments suggest that it prevents in most cases the failure of the Newton method. After a few damped Newton iterations, α remains equal to 1, and the quadratic convergence is restored.

Full least square commutator approach. A possible strategy for avoiding the time step size limitation of the reduced-MG approach is to solve the entire saddle point system instead of the reduced system. However, we want to avoid forming and then approximating the inverse of the Schur complement of the block $\boldsymbol{\mathcal{A}}$ of the matrix $\boldsymbol{\mathcal{J}}$, which is given by

$$\boldsymbol{S}_{\boldsymbol{\mathcal{A}}} = -(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^{-1}\boldsymbol{\mathcal{B}}^T).$$

To this aim, we first note that the matrix $\boldsymbol{\mathcal{J}}$ in (4.10) can be factored as follows:

$$\boldsymbol{\mathcal{J}} = \boldsymbol{\mathcal{Z}}\boldsymbol{\mathcal{W}}\left[\Delta t, \boldsymbol{u}, \boldsymbol{\sigma}\right]\boldsymbol{\mathcal{Z}}^T, \tag{4.13}$$

where matrices $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{Z}}$ are given by

$$\boldsymbol{\mathcal{W}}\left[\Delta t, \boldsymbol{u}, \boldsymbol{\sigma}\right] = \begin{pmatrix} \operatorname{diag}\left(\Psi(\boldsymbol{\sigma})\right) & \frac{1}{2}\operatorname{diag}\left(\boldsymbol{\sigma} \odot (\boldsymbol{G}\boldsymbol{u})\right) \\ \frac{1}{2}\operatorname{diag}\left(\boldsymbol{\sigma} \odot (\boldsymbol{G}\boldsymbol{u})\right) & -\operatorname{diag}\left(\frac{1}{\Delta t} - \frac{1}{4} \odot \left((\boldsymbol{G}\boldsymbol{u})^2 - \mathbf{1}\right)\right) \end{pmatrix},$$

$$\boldsymbol{\mathcal{Z}} = \begin{pmatrix} \boldsymbol{E}\boldsymbol{W}^{-1/2} & 0 \\ 0 & \boldsymbol{W}^{1/2} \end{pmatrix}.$$

The factorization of $\boldsymbol{\mathcal{J}}$ in (4.13) and the fact that matrices $\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Z}}^T$ resemble the divergence and gradient operators in the graph setting make us consider the *least square commutator* (LSC) preconditioner, introduced in [12], as a preconditioner for solving the saddle point linear systems in our problem. The LSC preconditioner, applied to the solution of the linear system in (4.6), can be written as

$$\begin{aligned} \boldsymbol{\mathcal{P}}_{\boldsymbol{\mathcal{W}}} &= \left(\boldsymbol{\mathcal{Z}}\boldsymbol{\mathcal{Z}}^T\right)^{\dagger}\boldsymbol{\mathcal{Z}}\boldsymbol{\mathcal{W}}^{-1}\boldsymbol{\mathcal{Z}}^T\left(\boldsymbol{\mathcal{Z}}\boldsymbol{\mathcal{Z}}^T\right)^{\dagger}, \\ &= \begin{pmatrix} \boldsymbol{L}^{\dagger}[\mathbf{1}] & 0 \\ 0 & \boldsymbol{W}^{-1} \end{pmatrix} \boldsymbol{\mathcal{Z}}\boldsymbol{\mathcal{W}}^{-1}\boldsymbol{\mathcal{Z}}^T \begin{pmatrix} \boldsymbol{L}^{\dagger}[\mathbf{1}] & 0 \\ 0 & \boldsymbol{W}^{-1} \end{pmatrix}. \end{aligned}$$

We use the preconditioner $\boldsymbol{\mathcal{P}}_{\boldsymbol{\mathcal{W}}}$ within the flexible GMRES algorithm [39]. This preconditioner would be extremely appealing from the computational point of view, since, besides a few diagonal scalings and sparse matrix-vector products, it only relies on solving twice a linear system with an unweighted Laplacian matrix. Unfortunately, the performance of the preconditioner degrades after a few time steps starting from $\boldsymbol{\mu}_0 = \mathbf{1}$. This suggests considering a second factorization of the Jacobian matrix, namely,

$$\boldsymbol{\mathcal{J}} = \boldsymbol{\mathcal{Z}}_{\boldsymbol{\mu}}\boldsymbol{\mathcal{V}}\left[\Delta t, \boldsymbol{u}, \boldsymbol{\sigma}\right]\boldsymbol{\mathcal{Z}}_{\boldsymbol{\mu}}{}^T,$$

where matrices $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{Z}_\mu}$ are given by

$$\boldsymbol{\mathcal{V}}[\Delta t, \boldsymbol{u}, \boldsymbol{\sigma}] = \begin{pmatrix} \boldsymbol{I} & \operatorname{diag}(\boldsymbol{Gu}) \\ \operatorname{diag}(\boldsymbol{Gu}) & -\operatorname{diag}\left(\frac{1}{\Delta t} - \frac{1}{4} \odot \left((\boldsymbol{Gu})^2 - \mathbf{1}\right)\right) \end{pmatrix},$$
$$\boldsymbol{\mathcal{Z}_\mu} = \begin{pmatrix} \boldsymbol{E}\boldsymbol{W}^{-1/2}\operatorname{diag}(\boldsymbol{\mu})^{1/2} & 0 \\ 0 & \boldsymbol{W}^{1/2} \end{pmatrix}.$$

Note that this factorization strongly relies on the transformation chosen, $\Psi(\boldsymbol{\sigma}) = (\boldsymbol{\sigma})^2/4$. The resulting LSC preconditioner reads as follows:

$$\begin{aligned} \boldsymbol{\mathcal{P}_\mathcal{V}} &= (\boldsymbol{\mathcal{Z}_\mu}\boldsymbol{\mathcal{Z}_\mu}^T)^\dagger \boldsymbol{\mathcal{Z}_\mu}\boldsymbol{\mathcal{V}}^{-1}\boldsymbol{\mathcal{Z}_\mu}^T(\boldsymbol{\mathcal{Z}_\mu}\boldsymbol{\mathcal{Z}_\mu}^T)^\dagger \\ &= \begin{pmatrix} \boldsymbol{L}[\boldsymbol{\mu}]^\dagger & 0 \\ 0 & \boldsymbol{W}^{-1} \end{pmatrix} \boldsymbol{\mathcal{Z}_\mu}\boldsymbol{\mathcal{V}}^{-1}\boldsymbol{\mathcal{Z}_\mu}^T \begin{pmatrix} \boldsymbol{L}[\boldsymbol{\mu}]^\dagger & 0 \\ 0 & \boldsymbol{W}^{-1} \end{pmatrix}, \end{aligned}$$

again used within the flexible GMRES algorithm. Similar to the reduced system approach, at each time step and at each Newton step we restrict the time step size Δt_k and the damping parameter α so that the following condition holds:

$$\max\left\{\frac{1}{\Delta t} - \frac{1}{4}\left((\boldsymbol{Gu})^2 - \mathbf{1}\right) + (\boldsymbol{Gu})^2\right\} \geq \varepsilon_{\boldsymbol{\mathcal{C}}}, \tag{4.14}$$

in order to avoid inversion of matrices close to being singular. In this case, the main computational effort is spent in the solution of two linear systems with $\boldsymbol{\mu}$-weighted Laplacian matrices, which can be achieved with the algebraic multigrid approach. We denote this preconditioner approach by *full-LSC*.

5. Numerical experiments. In this section, we describe three test cases we devised in order to test the accuracy and efficiency of the different methods proposed for the solution of the OTP on graphs.

Test-case 1. The first test-case problem is the single source shortest path (SSSP), which can be described as follows. Fix a node $\bar{v}$ in the graph $\mathcal{G}$, say, $\bar{v} = v_1$. We want to find the shortest path distance from the node v_1 to all other nodes in $\mathcal{G}$. Thus we want to determine

$$\boldsymbol{d^*} \in \mathbb{R}^n, \quad d_i^* := \operatorname{dist}_{\mathcal{G}}(v_1, v_i).$$

The solution of the SSSP is naturally related to the problem of finding a shortest path tree rooted at v_1. This is a spanning tree T of $\mathcal{G}$ such that, given any node v in $\mathcal{V} \setminus v_1$, the path from the root v_1 to v (unique since T is a tree) is a shortest path from v_1 to v in $\mathcal{G}$. Note that there may exist multiple shortest path trees. The SSSP and the OTP on graphs are related by the following proposition.

PROPOSITION 2. *The solution of the SSSP problem rooted at v_1 is equivalent to solving the OTP on graphs with the following source and sink vectors:*

$$b_i^+ = \begin{cases} 0 & \text{if } i = 1, \\ 1/(n-1) & \text{otherwise,} \end{cases} \qquad b_i^- = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Given any shortest path tree T rooted at v_1, the vectors

$$\boldsymbol{u}^* = \boldsymbol{d}^*, \tag{5.1}$$

$$\mu_e^*(T) := \begin{cases} \dfrac{1}{n-1} \left\{ \begin{matrix} \textit{Number of shortest paths in } T \textit{ between any} \\ \textit{node } v_i \textit{ and } v_1 \textit{ passing through edge } e \end{matrix} \right\} & \textit{if } e \in T, \\ 0 & \textit{otherwise,} \end{cases} \tag{5.2}$$

$$\boldsymbol{q}^* = \boldsymbol{\mu}^* \odot \boldsymbol{G}\boldsymbol{u}^*, \tag{5.3}$$

solve problems (2.2), (2.6), *and* (2.1), *respectively.*

Proof. The proof of the above proposition is based on the fact that vectors $\boldsymbol{q}^*$ and $\boldsymbol{u}^*$ in (5.1) and (5.3) are admissible solutions of problems (2.1) and (2.2), and the duality gap in (2.9) is zero, ensuring optimality of both solutions. Note that this holds true taking any convex combination in $\{\boldsymbol{\mu}^*(T) : T = \text{shortest path tree rooted at } v_1\}$. ∎

This test case is particularly relevant in our discussion, being an example where the solution of problem (2.2) is unique (up to a constant), while the solutions of problems (2.1) and (2.6) are not. In fact, for any $\boldsymbol{\mu}^*$, the kernel of the matrix $\boldsymbol{L}[\boldsymbol{\mu}^*]$ consists only of the constant vectors, since each node of the graph is "surrounded" by at least one edge with nonzero conductivity. In contrast, the attractor of the dynamical equation (3.4) is the image through the map Ψ^{-1} of any convex combination of $\{\boldsymbol{\mu}^*(T) : T = \text{shortest path tree}\}$.

In this test case we consider graphs describing the node connection of a regular grid triangulation of the square $[0,1] \times [0,1]$; thus two coordinates (x_i, y_i) are assigned to each node i. The vector $\boldsymbol{w}$ is the Euclidean distance between the nodes connected by two edges. Node v_1 has coordinates $(0.5, 0)$. We generate a sequence of graphs $(\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j))_{j=0,\dots,5}$ starting from an initial triangulation $\mathcal{T}_0$ (reported in Figure 1), conformally refined 5 times.

Test-case 2. The second test-case problem is the graph counterpart of Test-case 1 considered in [18], where the OTP is posed in the continuous setting $\mathcal{X} = [0,1] \times [0,1]$. We consider the same sequence of graphs $(\mathcal{G}_j)_{j=0,\dots,5}$ described in Test-case 1 and a forcing term $\boldsymbol{b}$ given by

$$b_i = \begin{cases} C & \text{if } x_i, y_i \in [0.125, 0.375] \times [0.25, 0.75], \\ -C & \text{if } x_i, y_i \in [0.625, 0.875] \times [0.25, 0.75], \\ 0 & \text{otherwise,} \end{cases}$$

where $C = (\sqrt{n} - 1)$. For this test case there exist explicit solutions for problems (2.1), (2.2), and (2.6). In fact, taking the barycenters of the edges of $\mathcal{G}$

$$(\bar{x}, \bar{y})_e = 0.5(x_i + x_j, y_i + y_j)\ ,\ e = (v_i, v_j),$$

a solution $\boldsymbol{\mu}^*$ of problem (2.6) is given by

$$\mu_e^* = \begin{cases} C(\bar{x}_e - 0.125) & \text{if } \bar{x}_e \in [0.125, 0.375], \quad y_{v_1} = y_{v_2} \in [0.25, 0.75], \\ 2 & \text{if } \bar{x}_e \in [0.375, 0.625], \quad y_{v_1} = y_{v_2} \in [0.25, 0.75], \\ C(0.875 - \bar{x}_e) & \text{if } \bar{x}_e \in [0.625, 0.875], \quad y_{v_1} = y_{v_2} \in [0.25, 0.75], \\ 0 & \text{otherwise,} \end{cases}$$

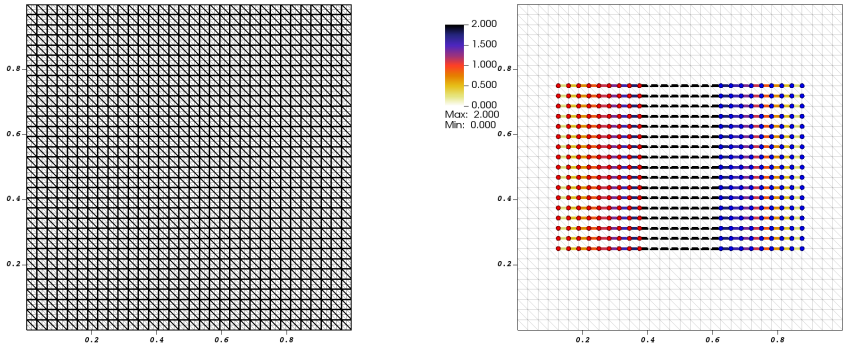

FIG. 1. *On the left panel we report the spatial distribution of the graphs $\mathcal{G}_0$ and $\mathcal{G}_1$, used in Test-cases* 1 *and* 2. *On the right panel we report the spatial distribution on graph $\mathcal{G}_0$ of the source term $\boldsymbol{b}^+$ (red dots) and the sink term $\boldsymbol{b}^-$ (blue dots) in Test-case* 2, *together with the corresponding solution $\boldsymbol{\mu}^*$ of problem* (2.6).

while a solution of problem (2.2) is given by

$$u_i^* = x_i. \tag{5.4}$$

Then, it is easy to check that $\boldsymbol{q}^* = \boldsymbol{\mu}^* \odot \boldsymbol{G}\boldsymbol{u}^*$ and $\boldsymbol{u}^*$ satisfy the constraints of problem (2.1) and (2.2) with zero duality gap, proving that they are optimal solutions. Note that there exist infinitely many solutions of problem (2.2). In fact, it is possible to "perturb" the values of the solution $\boldsymbol{u}^*$ in (5.4) on those nodes "outside" the rectangle $[0.125, 0.875] \times [0.25, 0.75]$, still satisfying the constraints of problem (2.2), while the value of the cost functional $\boldsymbol{b}^T\boldsymbol{u}^*$ remains the same.

In Figure 1, we report the graphs $\mathcal{G}_0$ and $\mathcal{G}_1$, together with the spatial distribution of the forcing term $\boldsymbol{b}$ and optimal solution $\boldsymbol{\mu}^*$.

Test-case 3. In the third test case, we consider graphs less structured than those in Test-case 1 and Test-case 2. In particular we consider three well-known categories of synthetic random graphs: the Erdös–Rényi [13], the Watts–Strogatz [47], and the Barabási–Albert graphs [5]. We generate graphs with increasing number of nodes/edges using the Python package Networkx (see [22]). We repeat this procedure 10 times, getting groups of 10 graphs with similar characteristics (number of nodes, number of edges, construction procedure) but different topology structure. We assign to each graph a different forcing term $\boldsymbol{b}$ and weight $\boldsymbol{w}$. The forcing term vector $\boldsymbol{b}$ is generated randomly within a uniform distribution in $[-1, 1]$ on a fraction of the nodes. We consider two scenarios, one where all entries of the forcing term $\boldsymbol{b}$ are nonzero (similar to Test-case 1) and another where this holds only for 10% of the entries (similar to Test-case 2). The negative entries of the resulting vector $\boldsymbol{b}$ are then scaled so that $\boldsymbol{b}^T\mathbf{1} = 0$. The weight vector $\boldsymbol{w}$ is generated randomly with a uniform distribution in $[0.5, 1.5]$.

This procedure is done in order to average the metrics of our simulations (total number of time steps, Newton steps, CPU time, etc.) when the results are presented.

6. Numerical results. All the experiments presented in this section are done on a quad-core 1.8 GHz 64-bit Intel Core I7 machine with 16 GBs memory. The

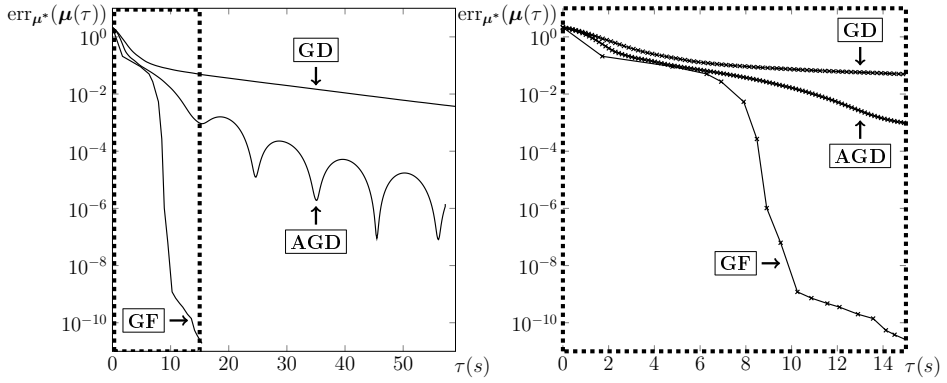

FIG. 2. *Plot of the quantity* $\mathrm{err}_{\boldsymbol{\mu}^*}$ *defined in* (6.1) *versus CPU time for the GD, AGD, and GF approaches. We report the results for Test-case* 2 *using* $\mathcal{G}_3$ *(*66049 *nodes and* 197120 *edges). The left panel reports the error evolution for the whole simulation. The right panel, with dashed border, magnifies the portion of the left panel where the GF approach converged to the optimal solution.*

software is written in MATLAB, with the linear solvers AGMG and LAMG working via MEX interface. All simulations were performed on a single core. We summarize in subsection 6.6 a discussion on the values of the different parameters that control our algorithm.

6.1. GD—AGD—GF. In this section we compare in terms of accuracy and computational efficiency the three time discretization schemes for (3.4) described in subsection 4.1, GD, AGD, and GF. To this aim, we consider only Test-case 2, where we have an explicit formula for the optimal solution $\boldsymbol{\mu}^*$ of problem (2.6), and we report in Figure 2 the evolution of the $\boldsymbol{\mu}$-error

$$\mathrm{err}_{\boldsymbol{\mu}^*}(\boldsymbol{\mu}) = \|\boldsymbol{\mu} - \boldsymbol{\mu}^*\|_{\boldsymbol{w}} / \|\boldsymbol{\mu}^*\|_{\boldsymbol{w}} \tag{6.1}$$

with respect to CPU time for the three approaches. We present only the results for $\mathcal{G}_3$ (66049 nodes and 197120 edges) since they are representative of what we observed using smaller and larger graphs. The initial data used is $\boldsymbol{\sigma}^0 = \Psi^{-1}(\mathbf{1})$. In the GD and the AGD approaches, we start from an initial time step size $\Delta t_k = 0.1$ progressively increased by a factor 1.05 until a maximum of $\Delta t_k = 2$, which was found experimentally to guarantee stability of the schemes. In the GF scheme we adopt the reduced-MG approach for the solution of the linear systems arising from the Newton method; thus the time step size is tuned according to condition (4.12). For these experiments, the AGMG software turns out to be the most efficient among the multigrid solvers for the arising linear systems.

The left panel in Figure 2 shows that the GF approach outperforms the other two methods in terms of overall efficiency. On the right panel we magnify the portion of the left panel where the GF approach converges toward the optimal solution. Here, we can see that each time step of GD and AGD requires much less computational effort with respect to the GF approach, but more time steps are required. The error-versus-time plot using the GD approach is monotonically decreasing, but the convergence is extremely slow. Using the ADG scheme, the convergence toward the optimum is faster, although oscillations occur as the optimum is approached. These are both well-known phenomena using this method (see, for example, [43]). During the initial

steps of the GF approach (from the first to the fourth) the error reduction per CPU time is practically the same for all approaches. But then, the convergence of the GF becomes much faster since we are getting closer to the steady state and thus larger time step sizes Δt_k are used. In the intermediate steps (5th–9th), the GF approach reduces the $\boldsymbol{\mu}$-error by at least one order of magnitude per step. In the final steps (9th–17th), the convergence toward the optimal solution slows down due to some issues with the linear solver, discussed in subsection 6.4. When these problems arise, the AGD approach may be a valid alternative to the GF approach that guarantees a good trade-off between accuracy and efficiency.

6.2. Reduced-MG versus full-LSC. In this section we compare the different approaches presented in subsection 4.2 for the solution of the saddle point linear system. We take the graphs $\mathcal{G}_0, \mathcal{G}_1$, and $\mathcal{G}_2$ of Test-case 1. We consider $\boldsymbol{\sigma}^0 = \Psi^{-1}(\mathbf{1})$, and we fix $\hat{\epsilon} = 10^{-6}$. The corresponding results are reported in Table 1. In this experiment AGMG turns out to be, again, the most efficient multigrid solver for the linear systems with weighted Laplacian matrices we need to solve in both approaches. The full-LSC approach requires less time steps, since the restriction in (4.14) is less strict than that in (4.12). This results in fewer linear systems to be solved and in fewer failures of the linear and nonlinear solvers, but the overall performance of the preconditioner $\boldsymbol{\mathcal{P}_\mathcal{V}}$ is worse. In fact, not only does it require the approximate solution of two weighted Laplacian systems via multigrid solver for each flexible GMRES iteration, but the total number of iterations increases rapidly with the size of the graph.

The reduced-MG approach may require a greater number of time steps and nonlinear iterations, resulting in a greater number of linear systems to be solved. However, these linear systems involve weighted Laplacian matrices that can be efficiently handled via algebraic multigrid methods.

6.3. Test-case 1. In this section we present the results obtained for Test-case 1, taking the graphs $(\mathcal{G}_j)_{j=0,\dots,5}$. In this test case, we adopt the GF scheme combined with the reduced-MG approach, using the AGMG algorithm as algebraic multigrid solver, since it turns out to be the most efficient in terms of CPU time. In all experiments we used $\boldsymbol{\sigma}^0 = \Psi^{-1}(\mathbf{1})$ as initial data. We use a tight tolerance $\hat{\epsilon} = 10^{-14}$ to

TABLE 1

Comparison between preconditioning approaches, the reduced-MG (R-MG) and the full-LSC (F-LSC), described in subsection 4.2. The first two columns describe the dimensions of the graph considered. The remaining columns, from left to right, report the following results: the total number of time steps required to achieve convergence $\hat{\epsilon} \leq 10^{-6}$, the total number of Newton steps (which is equal to the number of saddle point linear system solved), the total number of multigrid iterations performed (using AGMG). The last column reports the CPU time in seconds required to achieve convergence and the percentage of CPU time spent in solving linear systems (L), building the matrices (B), and wasted due to failures of linear and nonlinear solvers (W).

Graph		App.	Time	New.	AGMG	CPU	
$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$		steps	steps	iters.	(s)	%(L,B,W)
1,089	3,136	F-LSC	4	20	568	0.7	(83.3-16.7-00.0)
		R-MG	8	36	423	0.14	(83.6-12.2-04.2)
4,225	12,416	F-LSC	4	27	1,092	4.58	(87.4-12.6-00.0)
		R-MG	8	40	589	0.41	(88.5-06.6-05.0)
16,641	49,408	F-LSC	4	37	2,148	36.5	(89.1-10.9-00.0)
		R-MG	9	52	868	2.07	(92.4-04.8-02.8)

TABLE 2

Results for Test-case 1. *From left to right, we report the graph considered with the number of nodes $|\mathcal{V}|$ and edges $|\mathcal{E}|$, the number of time steps required to achieve steady state, the total number of Newton steps (equal to the number of linear systems to be solved), the total number of multigrid iterations performed, the CPU time in seconds, and the error in the steady-state solution $\hat{\boldsymbol{u}}$ with respect to the optimal potential $\hat{\boldsymbol{u}}$ defined in* (6.2).

$\mathcal{G}$	Graphs $\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Time steps	Newton steps	AGMG iters.	CPU (s)	Errors $\mathrm{err}_{\boldsymbol{u}^*}(\hat{\boldsymbol{u}})$
$\mathcal{G}_0$	$1.1 \cdot 10^3$	$3.1 \cdot 10^3$	9	29	335	0.0963	3.3e-15
$\mathcal{G}_1$	$4.2 \cdot 10^3$	$1.2 \cdot 10^4$	6	25	359	0.226	2.7e-13
$\mathcal{G}_2$	$1.7 \cdot 10^4$	$4.9 \cdot 10^4$	7	26	399	0.902	9.0e-14
$\mathcal{G}_3$	$6.6 \cdot 10^4$	$2.0 \cdot 10^5$	7	28	456	4.43	3.3e-15
$\mathcal{G}_4$	$2.6 \cdot 10^5$	$7.9 \cdot 10^5$	7	31	545	24.2	6.7e-16
$\mathcal{G}_5$	$1.1 \cdot 10^6$	$3.1 \cdot 10^6$	8	34	646	136	5.0e-14

declare the convergence toward the steady state. For this test case, we measure the accuracy of the method using the relative error with respect to the optimal solution $\boldsymbol{u}^*$ of problem (2.2) given in (5.1), i.e.,

$$\mathrm{err}_{\boldsymbol{u}^*}(\boldsymbol{u}) := \|\boldsymbol{u} - \boldsymbol{u}^*\|_2 / \|\boldsymbol{u}^*\|_2. \tag{6.2}$$

The results in Table 2 show that the number of time steps required to reach steady state is practically constant with respect to the size of graph. The same holds with the total number of Newton steps, which corresponds to the number of linear systems we need to solve. We attribute this phenomenon to the fact that, for this test case, the graph considered is "grid-like" and the kernel of matrix $\boldsymbol{L}[\boldsymbol{\mu}^*]$ consists only of the constant vectors. The experiments in the next sections show that when one of these two conditions fails, the total number of linear systems to be solved grows with the size of the graphs. We observed a modest increase in the average number of multigrid iterations per linear system, which goes from 12 for the smallest ($\approx 10^3$ nodes/edges) to 19 for the largest graph ($\approx 10^6$ nodes/edges). This results in a CPU time that grows slightly more than linearly with respect to the graph size. While in term of CPU time the proposed method is still not comparable with algorithms dedicated to the solution of the SSSP, such as those presented in [30], we are pleasantly surprised by its accuracy and efficiency.

In Figure 3 we show the evolution of the relative error with respect to the optimal potential $\boldsymbol{u}^*$ and norm of the gradient of $\tilde{\mathcal{L}}(\boldsymbol{\sigma})$ against the CPU time, measured in seconds. Only the results for the smallest and largest graphs ($\mathcal{G}_0$ and $\mathcal{G}_5$) are reported. The plots show how most of the computational effort is spent during the initial steps, and then, when $\|\nabla \tilde{\mathcal{L}}(\boldsymbol{\sigma})\|_{\boldsymbol{w}}$ reaches the values $10^{-5} - 10^{-7}$, the convergence accelerates since larger time size steps are used; thus the time relaxation in (4.2) is so small that the GF approach becomes practically the pure Newton method applied to (2.7).

Summarizing the results obtained for Test-case 1, the reduced-MG approach requires, approximately, a fixed number of time steps and the solution of $\mathcal{O}(m^{0.05})$ linear systems to achieve the optimal solution. Solving these linear systems requires in total $\mathcal{O}(m^{0.11})$ multigrid iterations of the AGMG solver, while the overall CPU time required scales, approximately, as $\mathcal{O}(m^{1.21})$.

6.4. Test-case 2. Selection of the active subgraph. The second test case turns out to be a more challenging problem for the GF approach. In fact, using the

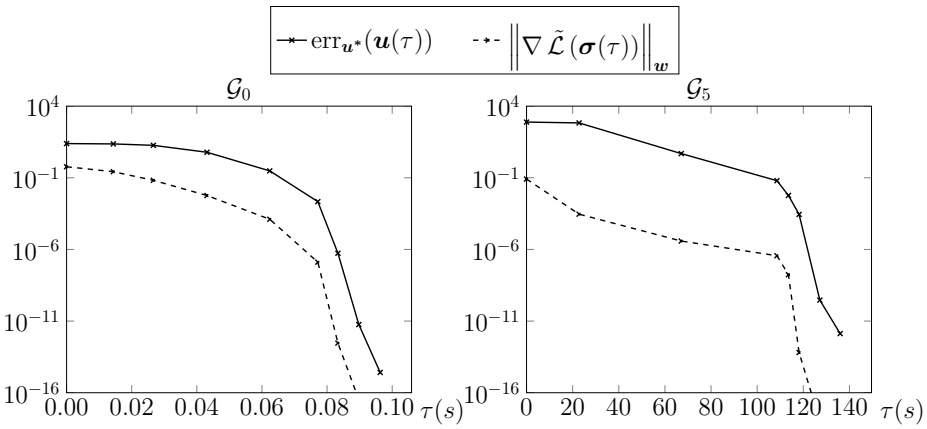

FIG. 3. *Plot of the $\boldsymbol{u}^*$-error (given in (6.2)) and the norm of $\nabla \tilde{\mathcal{L}}(\boldsymbol{\sigma})$ (used to estimate convergence toward steady state) against the CPU time for Test-case 1, for $\mathcal{G}_0$ (left) and $\mathcal{G}_5$ (right).*

reduced-MG approach, the efficiency is strongly influenced by the evolution of the vector

$$(\boldsymbol{G}\boldsymbol{u})^2 - \mathbf{1},$$

appearing in the definition of the matrix $\boldsymbol{\mathcal{C}} = \mathrm{diag}\left(\mathbf{1}/\Delta t - ((\boldsymbol{G}\boldsymbol{u})^2 - \mathbf{1})\right)$ in (4.10). When $\max((\boldsymbol{G}\boldsymbol{u})^2) < 1$ or $\max((\boldsymbol{G}\boldsymbol{u})^2) \approx 1$ a larger time step size Δt_k and $\alpha = 1$ (the undamped Newton algorithm) can be used. On the other hand, if $\max((\boldsymbol{G}\boldsymbol{u})^2) > 1$, more time and Newton steps are required to reach steady state. However, we note that most of the edges where $\max((\boldsymbol{G}\boldsymbol{u})^2) >> 1$ are those with conductivity $\boldsymbol{\mu}$ going to zero. Thus, the rows of the matrix $\boldsymbol{L}[\boldsymbol{\mu}]$ corresponding to the nodes that belong to these edges became almost null. Oscillation of the potential $\boldsymbol{u}$ and thus in the gradient vector $\boldsymbol{G}\boldsymbol{u}$ may appear when we get closer to the optimal solution $\boldsymbol{\sigma}^* = \Psi^{-1}(\boldsymbol{\mu}^*)$, which contains several zero entries.

A simple strategy to cope with this problem is to divide the graph $\mathcal{G}$ in two subgraphs, $\mathcal{G}_{\mathrm{on}} = (\mathcal{V}_{\mathrm{on}}, \mathcal{E}_{\mathrm{on}})$ and $\mathcal{G}_{\mathrm{off}} = (\mathcal{V}_{\mathrm{off}}, \mathcal{E}_{\mathrm{off}})$. The first, $\mathcal{G}_{\mathrm{on}}$, contains only those edges with conductivity $\boldsymbol{\mu}$ above a fixed threshold $\delta > 0$ and the nodes connected by those edges. The second, $\mathcal{G}_{\mathrm{off}}$, contains the remaining edges and nodes. The time evolution is restricted to the graph $\mathcal{G}_{\mathrm{on}}$. On the graph $\mathcal{G}_{\mathrm{off}}$ the conductivity $\boldsymbol{\mu}$ is set to zero on all edges in $\mathcal{E}_{\mathrm{off}}$, while the potential $\boldsymbol{u}$ restricted to node in $\mathcal{V}_{\mathrm{off}}$ is fixed at the value given at the time of the selection. This strategy closely resembles the heuristic optimality check approach considered in [29] for the solution of the basis pursuit problem. It may also be seen as an application of the active set method in optimization [50]. On the other hand, from an algebraic perspective, we are trying to identify and remove the near-null vectors of matrix $\boldsymbol{L}[\boldsymbol{\mu}]$.

We summarize in Table 3 the numerical results for Test-case 2, with and without the edge selection. While for the smallest graphs the differences are relatively small, for the largest one, not using the selection procedure leads to failure of the convergence of the scheme, because edges outside the "support" of the optimal solution $\boldsymbol{\mu}^*$ start having $\left|(\boldsymbol{G}\boldsymbol{u})^2\right| >> 1$. By the restriction imposed in (4.12), this forces us to shrink the time step size Δt_k below the value 2, where we can use the GD and the AGD approaches at a lower computational cost. As already mentioned in section 6, the

TABLE 3

Numerical results for Test-case 2. *Each row corresponds to a different graph. Number of nodes/edges approximately scales by a factor of* 4 *between different graphs (exact numbers are reported in Table* 2*). For each graph, the first row reports the data where the edge selection is not active, the second when the selection is active. Starting from the second column we report the number of time steps required to reach steady state with* $\hat{\epsilon} = 10^{-14}$ *(T); the total number of Newton steps, which coincides with the linear system solved (N); the total number of AGMG iterations (AGMG); the CPU time in seconds, with the percentages of time spent in solving linear systems (L), building the matrices (B), and wasted by linear and nonlinear solvers (W); and the CPU time in seconds required to solve the problem via the MATLAB linear programming solver (LP). The last columns report the error of constraints of the dual problem and the relative* $\boldsymbol{\mu}$*-error. Convergence is not achieved for graph* $\mathcal{G}_5$ *when no selection is active (case marked with* †*). In this case the MATLAB linear programming solver did not find a solution within the* 8 *hours prescribed limit (again denoted by* †*).*

	Iterations			CPU			Errors	
$\mathcal{G}$	T	N	AGMG	(s)	L-B-W(%)	LP(s)	$\lvert\lVert \boldsymbol{G}\hat{\boldsymbol{u}}\rVert_\infty - \mathbf{1}\rvert$	$\mathrm{err}_{\boldsymbol{\mu}^*}(\hat{\boldsymbol{\mu}})$
$\mathcal{G}_0$	10	29	415	0.11	(80-09-11)	0.06	2.3e-13	3.5e-11
	10	31	361	0.12	(74-10-16)		4.0e-14	8.4e-12
$\mathcal{G}_1$	12	36	539	0.39	(83-06-11)	0.54	1.9e-10	1.8e-11
	10	38	523	0.39	(80-08-12)		1.0e-10	4.8e-13
$\mathcal{G}_2$	11	45	825	1.79	(90-04-06)	8.77	2.1e-13	4.6e-12
	11	56	961	2.78	(87-07-06)		1.3e-11	2.5e-11
$\mathcal{G}_3$	11	56	1,166	14.4	(93-03-04)	189.74	1.4e-10	2.0e-11
	11	65	1,323	15.3	(89-06-05)		1.3e-16	1.9e-12
$\mathcal{G}_4$	22	121	2,530	144	(75-03-22)	6,287	3.3e-11	1.8e-11
	21	131	2,568	141	(71-05-24)		2.3e-16	9.2e-13
$\mathcal{G}_5$	**101**	329	5,312	1,230	(83-04-13)	†	†	†
	37	242	4,765	888	(76-06-18)		1.0e-15	8.9e-14

GD and the AGD approaches can represent robust alternatives to the GF approach, since in both cases the gradient direction is given by the product of the $\boldsymbol{\sigma}^k$ and $\left|(\boldsymbol{G}\boldsymbol{u}^k)\right|^2 - \mathbf{1}$; thus on the edges where $\boldsymbol{\sigma}^k$ tends to zero, the first factor annihilates oscillation and errors in the second. However, the selection procedure removes the edges with strong gradient oscillations; thus a larger time step size can be used; hence convergence toward the approximate steady state is achieved in fewer time steps. Moreover, it reduces the overall computational cost since $\mathcal{G}_{\mathrm{on}}$ contains fewer edges and nodes. The drawback of this approach is that it requires us to tune the parameter δ. We found experimentally that $\delta = 10^{-9}$ is a suitable threshold for the test cases considered. In all simulations we obtain highly accurate approximations of the optimal solutions and the constraints on the vector $\boldsymbol{G}\boldsymbol{u}$.

In order to show the accuracy and the efficiency of the GF approach along the whole optimization process, we present in Figure 4 the evolution of three different errors against the CPU time. In particular, we report the relative error with respect to the optimal solution $\boldsymbol{q}^*$, the error of the cost functional in problem (2.1), which estimates the error in computation of the Wasserstein distance, and the error of the constraint of problem (2.2). We report only the results for the smallest and the largest graph, $\mathcal{G}_0$ and $\mathcal{G}_5$, using the reduced-MG approach with selection of the active edges with $\delta = 10^{-9}$. Figure 4 shows how having a good approximation of the value of problem (2.1) does not guarantee having a good solution of the problem, since problem (2.1) is convex but not strictly. Similar to Test-case 1, most of the computational effort is spent during the initial phase where the errors are reduced

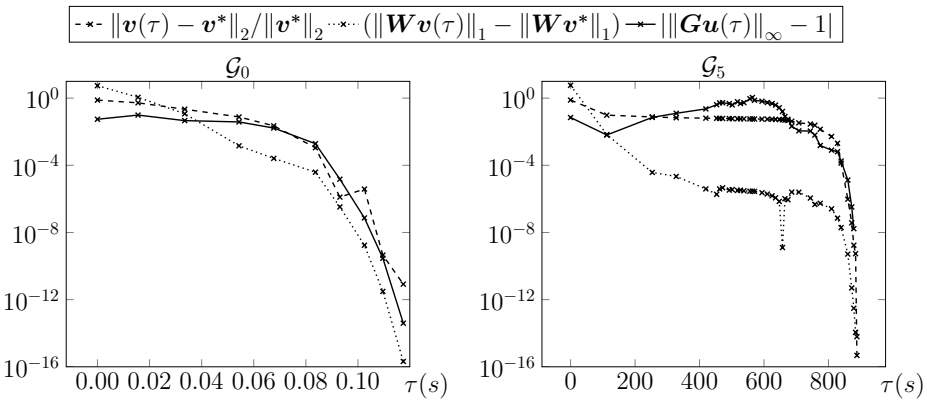

FIG. 4. *Plot of the relative error with respect to the optimal flux* $\boldsymbol{q}^*$*, the error in the value of the minimization problem, and the error of the constraints of problem* (2.2)*. We report the data against the CPU time* τ *only for Test-case* 2 *with* $\mathcal{G}_0$ *(left) and* $\mathcal{G}_5$ *(right).*

approximately to 10^{-4}. Then the convergence accelerates, and the optimal solutions of problems (2.1) and (2.2) are found in few time steps. On the left panel of Figure 4, which shows the results for $\mathcal{G}_0$, we can see that the ℓ^∞-norm of $\boldsymbol{G}\boldsymbol{u}(\tau)$ converges rapidly to 1; thus large time step sizes are used, and thus few time steps are required to reach steady state. On the right panel, which shows the results for $\mathcal{G}_5$, we can see that when the ℓ^∞-norm of the gradient vector $\boldsymbol{G}\boldsymbol{u}(\tau)$ exceeds 1 we are forced to shrink the time step size, resulting in more but less costly time steps.

Summarizing the results presented in this section for Test-case 2, the reduced-MG approach combined with the edge selection procedure requires $\mathcal{O}(m^{0.24})$ time steps and $\mathcal{O}(m^{0.33})$ linear systems to be solved to reach the optimal solution. Using the AGMG solver, the corresponding CPU time required by the proposed approach is, approximately, $\mathcal{O}(m^{1.4})$ seconds.

6.5. Test-case 3. In this section we summarize the numerical results obtained for the graphs of Test-case 3. In this case, the LAMG solver turns out to be more efficient and robust than the AGMG solver, which does not converge within the maximum number of multigrid iterations (fixed at 100) when we consider the largest graphs.We attribute this behavior to the different coarsening/interpolation approach used by LAMG and AGMG solvers. In fact, while the first is specifically designed to address to the solution of weighted Laplacian linear systems, the second is a purely algebraic aggregation scheme that may suffer working on matrices with large bandwidth, like those considered in this test case.

In all simulations, we started from $\boldsymbol{\sigma}^0=\Psi^{-1}(\mathbf{1})$, with the edge selection strategy active. We summarize in Table 4 the averaged results for the different groups of problems of Test-case 3. We observed an increased number of time steps and Newton steps with the size of the graph, with the latter scaling, approximately, between $\mathcal{O}(m^{0.31})$ and $\mathcal{O}(m^{0.36})$. This result compares well with the theoretical upper bound presented in [31], which estimates the cost of solving problem (2.1) on bipartite graphs with the solution of $\mathcal{O}(m^{3/7\approx 0.428})$ Laplacian systems. Unfortunately, as we already observed in Test-cases 1 and 2, the number of multigrid iterations per linear system increases with the size of the graphs. Using the LAMG solver, the resulting total CPU

TABLE 4

Results for Test-case 3. *The first column describes the test case (TC) considered, distinguishing among the Erdös–Rényi (ER), Watts–Strogatz (WS), and Barabási–Albert (BA) graphs. It also indicates the percentage of nonzero elements of* $\boldsymbol{b}$, 10% *or* 100%. *The second and third columns describe the dimensions of the graphs considered. From left to right, we report the total number of time steps required to achieve convergence (T), the total number of Newton steps (N), the total number of LAMG iterations performed (LAMG), and the CPU time in seconds required to achieve convergence. The results reported are the average of* 10 *different graphs with similar dimensions. The last columns report the minimum and the maximum of the error in the constraint of problems* (2.2) *and* (2.1). *In all simulations the duality gap in* (2.9) *is close to machine precision.*

TC	Graph		Iterations			CPU	$\lvert\lVert G\hat{\boldsymbol{u}}\rVert_\infty - \mathbf{1}\rvert$		$\lVert E\hat{\boldsymbol{q}} - \boldsymbol{b}\rVert_2/\lVert\boldsymbol{b}\rVert_2$	
	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	T	N	LAMG	(s)	min	max	min	max
WS	1e3	2e3	11	58	4.9e2	3.6e0	1e-13	4e-9	3e-11	1e-9
10%	1e4	2e4	23	147	1.7e3	2.4e1	2e-12	1e-8	3e-11	4e-9
	1e5	2e5	46	324	4.8e3	3.9e2	4e-11	8e-7	2e-14	3e-9
	1e6	2e6	86	670	1.3e4	8.8e3	2e-10	2e-6	1e-10	7e-9
WS	1e3	2e3	14	70	5.4e2	4.0e0	6e-12	3e-8	4e-12	8e-9
100%	1e4	2e4	34	202	2.0e3	3.2e1	6e-11	6e-8	7e-12	1e-9
	1e5	2e5	73	441	5.6e3	4.9e2	3e-12	7e-8	3e-11	9e-10
	1e6	2e6	140	882	1.3e4	1.3e4	4e-9	2e-6	4e-10	3e-9
ER	1e2	1e3	8	35	7.1e1	6.0e0	6e-15	3e-7	9e-16	8e-9
10%	1e3	1e4	17	99	8.6e2	9.1e0	1e-11	4e-8	5e-13	8e-10
	1e4	1e5	41	240	2.7e3	1.0e2	3e-11	5e-7	2e-11	9e-9
	1e5	1e6	51	325	4.6e3	1.8e3	7e-10	1e-6	2e-12	5e-9
ER	1e2	1e3	9	43	8.5e1	2.4e0	5e-13	4e-9	4e-11	3e-9
100%	1e3	1e4	19	107	9.0e2	8.5e0	1e-10	4e-8	1e-11	5e-9
	1e4	1e5	37	227	2.6e3	1.0e2	2e-11	6e-8	2e-12	5e-9
	1e5	1e6	81	498	7.3e3	2.2e3	6e-10	2e-5	6e-12	7e-9
BA	1e3	4e3	13	69	5.0e2	5.0e0	2e-14	4e-8	8e-12	8e-9
10%	1e4	4e4	26	149	1.5e3	4.5e1	1e-13	3e-9	7e-14	4e-9
	1e5	4e5	58	371	4.9e3	7.7e2	2e-11	8e-8	1e-11	4e-9
	1e6	4e6	108	724	1.2e4	2.1e4	2e-10	5e-7	3e-11	3e-9
BA	1e3	4e3	18	98	7.2e2	6.5e0	4e-15	6e-9	2e-14	6e-9
100%	1e4	4e4	36	214	2.1e3	5.7e1	7e-12	1e-8	5e-13	3e-9
	1e5	4e5	68	425	5.5e3	9.1e2	6e-9	2e-7	1e-11	8e-9
	1e6	4e6	133	860	1.4e4	3.0e4	1e-8	3e-7	5e-12	6e-9

time scales approximately between $\mathcal{O}(m^{1.14})$ and $\mathcal{O}(m^{1.51})$ seconds. Nevertheless, it scales certainly better than the $\mathcal{O}(n^2)$-time for solving the L^1-OTP described in [27].

6.6. Parameter tuning. In this section we want to summarize the crucial role played by the different parameters introduced above, in particular, the nonlinear tolerance ϵ_N, the linear solver precision ϵ, the bound $\varepsilon_{\mathcal{C}}$ in (4.12), and the lower bound α_0 for the damping parameter α. These parameters play competing roles in the accuracy and the performance of the algorithm. For example, the value of $\varepsilon_{\mathcal{C}}$ influences the time step size Δt_k. Typically, taking smaller time steps results in a higher number of time steps but fewer nonlinear iterations, each one requiring the solution of the linear system in (4.6). Smaller time steps size also result in fewer failures of the linear and nonlinear solvers. A second example is given by the lower bound α_0 imposed to the damping parameter α. Smaller values of α_0 results in fewer failures of the nonlinear solver but an increased number of damped Newton steps. In certain cases, it may be more convenient to perform two time steps with Δt rather than one single time step with $2\Delta t$.

TABLE 5
Parameters used in most of the numerical experiments.

Parameter	Value
ϵ_N (nonlinear tolerance in (4.9))	10^{-8}
ϵ (linear solver tolerance in (4.8))	10^{-4}
$\varepsilon_{\boldsymbol{\mathcal{C}}}$ (Δt tuning in (4.12))	10^{-8}
α_0 (lower bound for α)	5×10^{-2}
$\hat{\epsilon}$ (convergence criterion in (4.1))	10^{-12}
δ (selection threshold)	10^{-9}
$r_{\max}$ (max. Newton steps)	30

Finding a good combination among all these parameters, in order to obtain fast and accurate solution of the OTP on graphs, is not a trivial task. In Table 5 we summarize a combination of parameters that we found experimentally to give a good trade-off between accuracy and efficiency in finding the solution of the OTP on graphs.

7. Conclusions. In this paper we have presented a novel GD approach for the numerical solution of the L^1-OTP on graphs. We have also presented different numerical approaches based on this GD dynamics. The GF method with the reduced-MG approach is found to be an accurate, efficient, and robust solver for the L^1-OTP on graphs. Different types of graphs have been used, with different forcing terms. The numerical experiments show that the number of time steps with the GF-reduced-MG approach ranges between being constant and scaling proportionally to $\mathcal{O}(m^{0.36})$ depending on the type of graph $\mathcal{G}$ and the forcing term $\boldsymbol{b}$. We observed the same scaling in the total number of Newton steps, which coincides with the number of linear systems with weighted Laplacian matrix we need to solve. The selection of active edges/nodes described in subsection 6.4 drastically affects the efficiency of the GF-reduced-MG approach, even if it requires tuning a parameter that is problem-dependent. The overall CPU time required to obtain accurate solution of the OTP scales between $\mathcal{O}(m^{1.2})$ and $\mathcal{O}(m^{1.51})$ seconds. These results improve on the quadratic complexity required by entropic regularization-based methods, opening the extensive application of OTP tools to those problems where accurate solutions are required, for example, the inverse problem studied in [32].

The proposed approach is iterative; thus any approximate solution can be used as initial data for the approximation process. Thus it may be integrated with a fast but inaccurate algorithm for problem (2.1), since most of the computation effort is spent in the initial steps, while closer to the optimum the convergence of the GF accelerates. This can be accomplished by introducing a multilevel version of the algorithm, like in [7], where an initial solution is obtained by interpolating an approximate solution computed on a coarser graph. The coarsening/prolongation strategy used by the AGMG and the LAMG softwares should provide an excellent tool for this aim. Parallelization of the multigrid linear solver is also a future improvement that we want to introduce in our method in order to tackle problems on large-scale graphs. Depending on the graph structure, optimal scalability of the multigrid solver (which accounts for almost all the computation effort) can be achieved.

A second line of research should be aimed in making more robust and less problem-dependent the algorithm. In particular, the selection of the active-inactive portion of the graph described in subsection 6.4 may be replaced by the identification, via algebraic methods, of the near-null space of the weighted Laplacian matrices, similar to the approach proposed in [8]. Last, a deeper theoretical investigation of the proposed

time stepping scheme may help in defining new and more robust strategies to tune the parameters, summarized in Table 5, that govern our algorithm.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] J. Altschuler, J. Niles-Weed, and P. Rigollet, *Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration*, in Advances in Neural Information Processing Systems 30 (NIPs 2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Viswanathan, and R. Garnett, eds., Curran Associates, Red Hook, NY, 2017, pp. 1964–1974.

[3] L. Ambrosio, *Lecture Notes on Optimal Transport Problems*, Lecture Notes in Math., Springer, Cham, 2003, pp. 1–52.

[4] L. Ambrosio, N. Gigli, and G. Savaré, *Gradient Flows in Metric Spaces and in the Space of Probability Measures*, 2nd ed., Birkhäuser, Basel, 2005.

[5] A. L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[6] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[7] M. Benzi, E. Haber, and L. Taralli, *Multilevel algorithms for large-scale interior point methods*, SIAM J. Sci. Comput., 31 (2009), pp. 4152–4175.

[8] L. Bergamaschi, E. Facca, Á. Martínez Calomardo, and M. Putti, *Spectral preconditioners for the efficient numerical solution of a continuous branched transport model*, J. Comput. Appl. Math., 354 (2019), pp. 259 – 270.

[9] V. Bonifaci, *A Laplacian approach to ℓ^1-norm minimization*, Comput. Optim. Appl. 79 (2021), pp. 441–469.

[10] V. Bonifaci, K. Mehlhorn, and G. Varma, *Physarum can compute shortest paths*, J. Theor. Biol., 309 (2012), pp. 121–133.

[11] I. Ekeland and R. Témam, *Convex Analysis and Variational Problems*, Classics Appl. Math., SIAM, Philadelphia, PA, 1999.

[12] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *Block preconditioners based on approximate commutators*, SIAM J. Sci. Comput., 27 (2006), pp. 1651–1668.

[13] P. Erdös and A. Rényi, *On random graphs*, Publ. Math. Debrecen, 6 (1959), pp. 290–297.

[14] M. Essid and J. Solomon, *Quadratically regularized optimal transport on graphs*, SIAM J. Sci. Comput., 40 (2018), pp. A1961–A1986.

[15] E. Facca, F. Cardin, and M. Putti, *Branching Structures Emerging from a Continuous Optimal Transport Model*, preprint, arXiv:1811.12691 [math.NA], 2018.

[16] E. Facca, F. Cardin, and M. Putti, *Physarum Dynamics and Optimal Transport for Basis Pursuit*, preprint, arXiv:1812.11782 [math.NA] 2018, https://arxiv.org/abs/1812.11782.

[17] E. Facca, F. Cardin, and M. Putti, *Towards a stationary Monge-Kantorovich dynamics: The Physarum polycephalum experience*, SIAM J. Appl. Math., 78 (2018), pp. 651–676.

[18] E. Facca, S. Daneri, F. Cardin, and M. Putti, *Numerical solution of Monge–Kantorovich equations via a dynamic formulation*, J. Scient. Comput., 82 (2020), pp. 1–26.

[19] E. Facca, A. Karrenbauer, P. Kolev, and K. Mehlhorn, *Convergence of the non-uniform directed Physarum model*, Theoret. Comput. Sci., 816 (2020), pp. 184–194.

[20] G. H. Golub and V. Pereyra, *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, SIAM J. Num. Anal., 10 (1973), pp. 413–432.

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, *Improved training of Wasserstein GANs*, in Advances in Neural Information Processing Systems 30 (NIPs 2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Viswanathan, and R. Garnett, eds., Curran Associates, Red Hook, NY, 2017, pp. 5767–5777.

[22] A. A. Hagberg, D. A. Schult, and P. J. Swart, *Exploring network structure, dynamics, and function using networkx*, in Proceedings of the 7th Python in Science Conference, G. Varoquaux, T. Vaught, and J. Millman, eds., Pasadena, CA, 2008, pp. 11–15.

[23] S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent, *Optimal mass transport for registration and warping*, Int. J. Comput. Vis., 60 (2004), pp. 225–240.

[24] L. V. Kantorovitch, *On the translocation of masses*, Manag. Sci., 5 (1958), pp. 1–4.

[25] J. Lellmann, D. A. Lorenz, C. Schonlieb, and T. Valkonen, *Imaging with Kantorovich–Rubinstein discrepancy*, SIAM J. Imaging Sci., 7 (2014), pp. 2833–2859.

[26] Y. Lin, L. Lu, and S. T. Yau, *Ricci curvature of graphs*, Tohoku Math. J., 63 (2011), pp. 605–627.
[27] H. Ling and K. Okada, *An efficient earth mover's distance algorithm for robust histogram comparison*, IEEE Trans. Pattern Anal. Mach. Intell., 29 (2007), pp. 840–853.
[28] O. E. Livne and A. Brandt, *Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver*, SIAM J. Sci. Comput., 34 (2012), pp. B499–B22.
[29] D. A. Lorenz, M. E. Pfetsch, and A. M. Tillmann, *Solving basis pursuit: Heuristic optimality check and solver comparison*, ACM Trans. Math. Software, 41 (2015), pp. 1–28.
[30] K. Madduri, D. A. Bader, W. B. Jonathan, and J. R. Crobak, *An experimental study of a parallel shortest path algorithm for solving large-scale graph instances*, in Proceedings of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics, 2007, pp. 23–35.
[31] A. Madry, *Navigating central path with electrical flows: From flows to matchings, and back*, in Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, 2013, pp. 253–262.
[32] L. Métivier, R. Brossier, Q. Merigot, E. Oudet, and J. Virieux, *An optimal transport approach for seismic tomography: Application to* $3d$ *full waveform inversion*, Inverse Problems, 32 (2016), 115008.
[33] G. Monge, *Mémoire sur la théorie des déblais et des remblais*, Mem. Acad. Roy. Sci. Paris, 1781, pp. 666–704.
[34] Y. Nesterov, *A method for solving the convex programming problem with convergence rate* $O(1/k^2)$, Dokl. Akad. Nauk. SSSR, 269 (1983), pp. 543–547.
[35] Y. E. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Springer, Cham, 2014.
[36] C.-C. Ni, Y.-Y. Lin, F. Luo, and J. Gao, *Community detection on networks with Ricci flow*, Sci. Rep., 9 (2019), pp. 1–12.
[37] Y. Notay, *Aggregation-based algebraic multigrid for convection-diffusion equations*, SIAM J. Sci. Comput., 34 (2012), pp. A2288–A2316.
[38] G. Peyré and M. Cuturi, *Computational optimal transport*, Found. Trends Mach. Learn., 11 (2019), pp. 355–607.
[39] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Stat. Comput., 14 (1993), pp. 461–469.
[40] F. Santambrogio, *Optimal Transport for Applied Mathematicians*, Birkäuser, Basel, 2015.
[41] D. Scieur, V. Roulet, F. Bach, and A. d'Aspremont, *Integration methods and optimization algorithms*, in Advances in Neural Information Processing Systems 30 (NIPs 2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Viswanathan, and R. Garnett, eds., Curran Associates, Red Hook, NY, 2017, pp. 1109–1118.
[42] J. Sia, E. Jonckheere, and P. Bogdan, *Ollivier-Ricci curvature-based method to community detection in complex networks*, Sci. Rep., 9 (2019), pp. 1–12.
[43] W. Su, S. Boyd, and E. Candes, *A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights*, in Advances in Neural Information Processing Systems 27 (NIPs 2014), Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Red Hook, NY, 2014, pp. 2510–2518.
[44] E. Süli and D. F. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, UK, 2003.
[45] A. Tero, R. Kobayashi, and T. Nakagaki, *A mathematical model for adaptive transport network in path finding by true slime mold*, J. Theor. Biol., 244 (2007), pp. 553–564.
[46] C. Villani, *Optimal Transport: Old and New*, Springer Science & Business Media, New York, 2008.
[47] D. J. Watts and S. H. Strogatz, *Collective dynamics of "small-world" networks*, Nature, 393 (1998), pp. 440–442.
[48] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, PA, 1997.
[49] Y. Yang, B. Engquist, J. Sun, and B. F. Hamfeldt, *Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion*, Geophysics, 83 (2018), pp. R43–R62.
[50] C. Zhang and R. Ordóñez, *Numerical Optimization*, Springer, Cham, 2012.