Asynchronous Distributed Matrix Balancing and Application to Suppressing Epidemic

Van Sy Mai and Abdella Battou

Abstract— **This paper presents an efficient asynchronous distributed algorithm for the problem of balancing a nonnegative matrix using a network of processors, each of which has access to a portion of the global matrix. The goal of the algorithm is for the processors to collaborate through local information exchange so that each processor can determine its local weighting coefficients that balance the matrix. Our algorithm is of Gauss-Seidel type with strict relaxation and converges geometrically under mild assumptions on the communication model between neighboring processors. The analysis of our algorithm is based on a novel reformulation of matrix balancing as a network consensus problem, from which an upper bound on the convergence rate can be derived. Finally, we demonstrate the applicability of the algorithm to a problem of optimally allocating curing resources for suppressing epidemic spread in a directed weighted network, where the spreading dynamic is captured by a susceptible-infected-susceptible model.**

I. Introduction

Matrix balancing and scaling refer to problems of adjusting the entries of a matrix to meet some prior consistency requirements, e.g., to have a certain zero-nonzero structure, or equal row and column sums. Such problems appear frequently in various fields, including economics, statistics, demographics, numerical analysis, transportation planning and image reconstruction; see [1], [2] for a survey.

This paper revisits the problem of finding a positive diagonal matrix X such that, for a given nonnegative matrix A, the matrix XAX^{-1} is *balanced*, i.e., each row sum equals the corresponding column sum. Such an X is said to balance A and its diagonal elements are called balancing coefficients. This problem arises in economics, e.g., in adjusting social accounting matrices that represent the flow of funds between agents/accounts in a country's economy, where the direct estimate of such a matrix is never balanced due to inconsistent data–an inherent problem associated with statistical methods for estimating underlying economic models [3]. Osborne [4] was the first to study this problem in the context of preconditioning matrix A in order to increase the accuracy of the computation of its eigenvalues. The paper [4] proposed an iterative algorithm for computing X, also known as diagonal similarity scaling, where each step involves scaling each row and the corresponding column appropriately. Variations and generalizations to this classical algorithm have been subsequently developed and analyzed [5]–[7]. More efficient centralized algorithms for this problem exist (see, e.g., [1], [8]) but do not admit efficient parallelization, and thus become limited in dealing with very large-scale problems, where data could be dispersed over a network of machines and gathering/processing the whole matrix in a centralized unit may be impractical, not to mention possible privacy issues.

The authors are with the Information Technology Laboratory, NIST, USA. Emails: {vansy.mai,abdella.battou}@nist.gov

Our Contributions: This paper proposes an efficient distributed solution to the matrix balancing problem by relying on the presence of a network of processors, where each processor has access to part of the global matrix. In particular,

(i) We present an iterative algorithm to be implemented on the network of processors so that each processor, through *local and asynchronous computations and communications*, can obtain an estimate of its corresponding local balancing coefficients.

(ii) We derive a novel relation between our nonlinear algorithm and a linear time-varying consensus model, based on which *global exponential convergence* to an optimal solution is demonstrated under certain assumptions on bounded asynchronism. As an important consequence, the ϵ-approximate convergence time of our algorithm is linear in $\log(1/\epsilon)$, which is an improvement over known bounds for Osborne's type updates.

(iii) We demonstrate the applicability of the algorithm for solving a large-scale epidemic control problem, where the spreading dynamic in a weighted directed network is captured by a *susceptible-infected-susceptible* (SIS) model and the objective is to optimally allocate curing resources to prevent the epidemic spread.

The rest of the paper is organized as follows. We review related work in Section II. In Section III, we describe the matrix balancing problem and the setup of a network of processors on which our intended distributed algorithm is implemented. Our main results are given in Sections IV, where we present our algorithm and analyze its convergence. In Section V, we present a new application of matrix balancing to an epidemic control problem, where our algorithm is particularly suited. Finally, numerical examples are given in Section VI to illustrate the efficiency of our algorithm.

II. Related Work

The work in [4] was specialized for matrix balancing in the ℓ_2 (vector) norm. Parlett and Reinsch [9] generalized Osborne's algorithm to the matrix balancing problem in any ℓ_p norm. Another algorithm for (approximate) ϵ-balancing in ℓ_1 norm was given in [5], which exploits the relation between matrix balancing and an unconstrained convex optimization problem whose optimal solution can be approximated by an ellipsoid algorithm. Bounds on the running time of Osborne's algorithm was provided in [7] for ℓ_∞ norm and in [6] for the

ℓ_p case–the latter also showed an upper bound of $O(1/\epsilon^2)$ on the convergence time. In a slightly different setting, [10] presented a distributed algorithm for constructing a balanced weight matrix for a given network structure.

Closely related to matrix balancing is the *Matrix Scaling* problem, where for given nonnegative vectors $\mathbf{r}$ and $\mathbf{c}$ the goal is to find diagonal positive matrices X and Y such that $\mathbf{r}$ and $\mathbf{c}$ are, respectively, the row and column sums of XAY. The literature on this problem is even much richer; see, e.g., [2], [11] and references therein. A well-known algorithm for matrix scaling is called RAS (see, e.g., [12], [13]), which is similar to Osborne's algorithm in that a pair of row and column is scaled at every iteration. Matrix scaling can also be reduced to a convex optimization problem–this is used in [14] to obtain an algorithm for ϵ-approximate solution. In [15], a distributed algorithm is proposed for a slightly different problem, namely, assigning weights to edges in a directed network (given structure) so that the weight matrix is doubly stochastic, i.e., row and column sums are all 1.

It is also well-known that both Osborne's and RAS algorithms can be viewed as coordinate descent iterations for a convex optimization problem, of which the dual is a maximum entropy problem with linear equality constraints; see, e.g., [2], [5], [14]. Using this duality, [16] studied local convergence and asymptotic rate of convergence of some methods of Gauss-Seidel and Jacobi types. More complicated methods, e.g., Newton and interior-point methods, have been employed to obtain algorithms that run in polynomial time [13], [17] as well as nearly linear time [8].

III. Preliminaries

A. Notation and terminology

$\mathbb{R}$ denotes the set of real numbers. For a finite set $\mathcal{A}$, $|\mathcal{A}|$ denotes its cardinality. For a matrix $A = [a_{ij}]$, let a_{ij} denote its (ij) element, A^T its transpose, and $\lambda_i(A)$ its i-th largest (in magnitude) eigenvalue. Vectors are boldfaced, e.g., $\mathbf{x} = [x_1, ..., x_n]^\mathsf{T}$, $\mathbf{1} = [1, ..., 1]^\mathsf{T}$. $\|\mathbf{x}\|_p$ denotes p-norm (or ℓ_p norm) of $\mathbf{x}$. If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, ∇f denotes its gradient and $\partial_i f$ the i-th partial derivative. A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of direct edges. A directed path is a sequence of edges in the form $(i_1, i_2), (i_2, i_3), ..., (i_{k-1}, i_k)$. $\mathcal{G}$ is strongly connected if there is a directed path from each node to any other node. For $i \in \mathcal{V}$, $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$ denote the sets of in-neighbors and out-neighbors of i, respectively, i.e., $\mathcal{N}_i^+ = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$ and $\mathcal{N}_i^- = \{j \in \mathcal{V} \mid (j,i) \in \mathcal{E}\}$. Moreover, $\mathcal{N}_i^\pm := \mathcal{N}_i^+ \cup \mathcal{N}_i^-$.

B. Problem Statement and Convex Reformulation

We are interested in the following problem:

Given a nonnegative matrix A, design an asynchronous distributed algorithm for balancing A.

The following result characterizes the existence of a balancing matrix and an equivalent optimization problem.

Lemma 1: (e.g., [1]) For any matrix $A \in \mathbb{R}_+^{N\times N}$, let $\mathcal{G}_A$ denotes the weighted directed graph associated with A. Then the following statements are equivalent:

(i) A positive diagonal matrix X that balances A exists.

(ii) The graph $\mathcal{G}_A$ is strongly connected.

(iii) There exists a positive vector $\mathbf{x}$ that solves the following

$$\min_{\mathbf{x}>0} \;\; G(\mathbf{x}) := \textstyle\sum_{1\le i,j\le N} a_{ij} x_i / x_j. \tag{1}$$

Moreover, matrix X and vector $\mathbf{x}$ are unique up to scalars.

Thus, the existence of a solution can be checked by verifying condition (ii) of this lemma. Moreover, the diagonal elements of matrix A do not play any role in the problem and thus can be assumed to be zeros. Formally, we assume the following throughout the rest of the paper.

Assumption 1: Matrix A has zero diagonal entries and the graph $\mathcal{G}_A$ is strongly connected.

We now briefly describe a class of algorithms for finding a balancing A based on Osborne's classical algorithm, which iteratively balances row-column pairs in a fixed round-robin order, starting from matrix A. Specifically,

- *At $t = 0$, $A(0) = A$.*
- *For each $t > 0$, balance an index i of $A(t)$:*

$$A(t+1) = D(t)A(t)D^{-1}(t), \tag{2}$$

where $D(t)$ is a diagonal matrix with $d_{jj}(t) = 1$ for $j \neq i$ and $d_{ii}(t) = \sqrt{\sum_{1\le k\le N} a_{ki}(t)/\sum_{1\le j\le N} a_{ij}(t)}$.

In each step, the algorithm scales the i-th row by $d_{ii}(t)$ and the i-th column by $d_{ii}^{-1}(t)$. Recently, Ostrovsky [6] obtained bounds on the convergence rate of this algorithm under various choices of balancing index order (namely, greedy, round-robin, and randomized balancing). The algorithm is based on sequential coordinate updates, and thus is amenable for distributed implementation. But, there has been no analysis on the version of this algorithm with delays. Moreover, it cannot be implemented fully in parallel–the synchronous version may not converge while the asynchronous update requires certain ordering of the balancing index selection that is not distributed; see also Remark 2 below.

In this paper, we approach the problem from the geometric program in (1), where the optimal value and an optimal solution are denoted by G^* and $\mathbf{x}^*$, respectively. By a change of variables $y_i = \log x_i$, (1) is equivalent to the following:

$$\min_{\mathbf{y}\in\mathbb{R}^N} \;\; F(\mathbf{y}) := \textstyle\sum_{1\le i,j\le N} a_{ij} e^{y_i - y_j}. \tag{3}$$

Let Y^* be the set of optimal solutions to this problem. Then

$$\mathbf{y}^* \in Y^* \quad \Leftrightarrow \quad \nabla F(\mathbf{y}^*) = \mathbf{0}, \tag{4}$$

where each partial derivative is given by

$$\partial_i F(\mathbf{y}) = \textstyle\sum_{1\le j\le N} a_{ij} e^{y_i - y_j} - \sum_{1\le k\le N} a_{ki} e^{y_k - y_i}. \tag{5}$$

Remark 1: Note that F is not strictly convex, its gradient ∇F is not Lipschitz continuous and the level set $\{\mathbf{y} \mid F(\mathbf{y}) \le c\}$ is either empty or unbounded. Thus, most (linearized) descent algorithms are not readily guaranteed to converge as their step size design depends on either the Lipschitz parameter of ∇F or boundedness of level sets (see, e.g., [18]). Moreover, it can be shown that the nonlinear Jacobi algorithm may not converge, while the nonlinear Gauss-Seidel can be used to solve the problem but has some caveats described in Remark 2 below. As suggested in [18], these algorithms do converge if one coordinate is fixed. However, the convergence rate of such modified algorithms has not been analyzed and distributed election of such a fixed index requires further communications.

C. Problem Decomposition

Suppose $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the directed graph generated by A, where $\mathcal{V} = \{1, ..., N\}$ and $\mathcal{E}$ is the set of edges such that a_{ij} is the weight of link $(i, j) \in \mathcal{E}$. Since our goal is to design a distributed algorithm for solving the matrix balancing problem, we will assume the presence of a network of computing entities, hereafter called *processors*. This network is denoted by $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$, where $\tilde{\mathcal{V}} = \{1, ..., m\}$ and each processor $c \in \tilde{\mathcal{V}}$ is assumed to have access to a subpart $\mathcal{C}_c$ of $\mathcal{G}$ (i.e., c has access to both row i and column i of A, $\forall i \in \mathcal{C}_c$). Moreover, $\{\mathcal{C}_c\}_{c=1}^m$ forms a partition of $\mathcal{G}$. We also denote by $\mathcal{C}_c$ the processor c. $\mathcal{C}_c$ and $\mathcal{C}_d$ are neighbors if $\exists(i, j) \in \mathcal{E}$ with $i \in \mathcal{C}_c$ and $j \in \mathcal{C}_d$; see Fig. 1. We assume in this paper that two neighboring processors can exchange information without any communication noise. Moreover, the following is also a blanket assumption.

Assumption 2: $\tilde{\mathcal{G}}$ is undirected and connected.

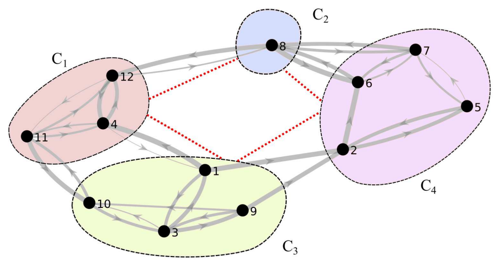

Fig. 1. An example of an overlay network of processors, where dotted lines represent bidirectional asynchronous communication links.

Note that the assumption on the bidirectional communication between neighboring processors in $\tilde{\mathcal{G}}$ will be critical to the implementation of our algorithm below; the main reason is to enable local evaluation of partial derivatives given in (5). It is also important to remark that we do not require instantaneous bidirectional information exchanges; instead, messages between $\mathcal{C}_i$ and $\mathcal{C}_j$ can arrive at different times.

IV. Algorithm and Convergence Analysis

A. Algorithm Description

Let time be discretized into slots and denoted by $t \in \mathbb{N}$, and let each processor $\mathcal{C}_c$ maintain and update a set of estimates $\{y_i\}_{i\in\mathcal{C}_c}$. At $t=0$, $\mathcal{C}_c$ freely initializes $\{y_i(0)\}_{i\in\mathcal{C}_c}$. At any $t>0$, $\mathcal{C}_c$ independently decides whether to update some (or all) of its estimates $\{y_i\}_{i\in\mathcal{C}_c}$. Let $\mathcal{T}_i$ denote the set of times at which y_i is updated. Then, $y_i(t+1)=y_i(t)$ if $t \notin \mathcal{T}_i$ and

$$y_i(t+1)=h_i\big(y_i(t), \{y_v(t-\tau_v^{(i)}(t))\}_{v\in\mathcal{N}_i^\pm}\big), \quad \forall t \in \mathcal{T}_i, \quad (6)$$

where h_i is a local update function (specified below) and the delay $\tau_v^{(i)}(t)$ represents the amount by which the information about y_v available to processor $\mathcal{C}_c$ is outdated. Subsequently, $\mathcal{C}_c$ sends $y_i(t+1)$ to those neighbors that require it (e.g., for the network in Fig. 1, $\mathcal{C}_3$ needs to send y_9 to $\mathcal{C}_4$ but not $\mathcal{C}_1$).

We now specify the choice of the update function h_i. Let

$$y_v^{(i)}(t)=y_v(t-\tau_v^{(i)}(t)), \quad v \in \mathcal{N}_i^\pm \quad (7)$$

$$\tilde{y}_i(t)=\tfrac{1}{2}\log\Big[\textstyle\sum_{k\in\mathcal{N}_i^+} a_{ki}e^{y_k^{(i)}(t)}/\sum_{j\in\mathcal{N}_i^-} a_{ij}e^{-y_j^{(i)}(t)}\Big] \quad (8)$$

$$h_i=y_i(t)+\alpha_i(\tilde{y}_i(t)-y_i(t)), \quad (9)$$

where $\alpha_i \in (0, 1)$ is a step size. Here, $y_v^{(i)}(t)$ represents a buffer with the most recently received y_v that is available to the processor associated with node $i \in \mathcal{V}$. The update for $\tilde{y}_i(t)$ is obtained from the optimality condition (4), namely, setting $\partial_{y_i}F(\mathbf{y})=0, \forall i \in \mathcal{V}$, and taking communication delays into account. Intuitively, processor $\mathcal{C}_c$ uses (possibly outdated) information from its neighbors to find $\tilde{y}_i(t)$ satisfying

$$\partial_{y_i}F(\tilde{y}_i(t), \{y_v^{(i)}(t)\}_{v\in\mathcal{V}}) = 0, \quad \forall i \in \mathcal{C}_c, \quad (10)$$

and then marches towards $\tilde{y}_i(t)$ with a step length equal to $\alpha_i(\tilde{y}_i(t) - y_i(t))$. Detailed implementation is shown in Algorithm 1 below, where we use the variables $\mathbf{x} = e^{\mathbf{y}}$ to avoid multiple $\exp()$ and $\log()$ operations.

Algorithm 1: Asynchronous Distributed Balancing

```
1  init: $\mathcal{C}_c$ sets $\alpha_i \in (0,1)$, $x_i(0)$, $\{x_v^{(i)}(0)\}_{v\in\mathcal{N}_i^\pm}$, $\forall i \in \mathcal{C}_c$
2  for $t = 1, 2, \ldots$ do
3      forall $i \in \mathcal{C}_c, c \in \tilde{\mathcal{V}}$ do
4          $x_v^{(i)}(t) \leftarrow x_v(t-\tau_v^{(i)}(t)), \quad v \in \mathcal{N}_i^\pm$
5          if $t \in \mathcal{T}_i$ then
6              $x_i(t+1) \leftarrow x_i(t)\left[\frac{\sum_{k\in\mathcal{N}_i^+} a_{ki}x_k^{(i)}(t)}{x_i^2(t)\sum_{j\in\mathcal{N}_i^-} a_{ij}/x_j^{(i)}(t)}\right]^{\frac{\alpha_i}{2}}$
7              Send $x_i(t+1)$ to neighboring processors
8          else
9              $x_i(t+1) \leftarrow x_i(t)$
10 $\mathcal{C}_c \leftarrow \{x_i^*\}_{i\in\mathcal{C}_c}$
```

Our algorithm belongs to the class of asynchronous iterative algorithms ([18]). The main novelty here is the requirement of strict relaxation $\alpha_i \in (0, 1)$ when applied to matrix balancing. This also enables us to develop a new tailored analysis for the convergence rate of the algorithm. Although we assume that α_i is fixed, as will be seen later in our analysis, time-varying step sizes are also possible.

Remark 2: (Gauss-Seidel Algorithm as a Limit Case) Consider a particular scenario corresponding to the nonlinear Gauss-Seidel algorithm, where $\tau_{ij}(t) = 0$, $\mathcal{T}_i = i + tN$ and $\alpha_i = 1$ for all $t \geq 0, i \in \mathcal{V}$. Then, line 6 of Algorithm 1 becomes $x_i(t+1) = \big(\sum_{k\in\mathcal{V}} a_{ki}x_k(t)/\sum_{j\in\mathcal{V}} a_{ij}x_j^{-1}(t)\big)^{1/2}$. It can be verified that this is indeed equivalent to Osborne's classical algorithm (2), which iteratively balances row-column pairs in a fixed cyclic order. To implement this algorithm in a distributed fashion, its update, however, needs to follow a coloring scheme on the graph $\mathcal{G}$, the design of which usually necessitates a centralized computing entity (see, e.g. [18]). In the next section, we show that such a requirement is not needed for the convergence of our algorithm; in fact, geometric convergence can be ensured whether the algorithm is implemented in full parallel or in an asynchronous distributed fashion.

B. *Convergence Analysis*

We will study convergence of Algorithm 1 under the following further mild assumptions (see, e.g., [18]):

Assumption 3: There exist constants $T_0, T_1 \in \mathbb{N}$ such that:

(a) Each y_i is updated as in (6) at least once during any iteration interval of length T_0.

(b) All delay functions $\tau_v^{(i)}$ are bounded by T_1 and $\tau_i^{(i)} \equiv 0, \forall i \in \mathcal{V}, \forall v \in \mathcal{N}_i^{\pm}$.

Note that T_0 and T_1 are not needed for the algorithm implementation. These assumptions simply mean that all nodes contribute sufficiently in optimizing the global cost F. We now establish the convergence of the algorithm to an optimal solution $\mathbf{y}^*$ of (3) as well as a linear convergence rate in terms of nodal variables $\{y_i\}$. An upper bound on the convergence rate will be given, which depends on both T_0 and T_1, i.e., the degree of asynchronism of the algorithm.

Theorem 1: Consider Algorithm 1 through $\mathbf{y} = \log(\mathbf{x})$ with any $\mathbf{y}(0) \in \mathbb{R}^N$ and $\boldsymbol{\alpha} \in (0,1)^N$. Let $\{\mathbf{y}(t)\}_{t \geq 0}$ be a sequence generated by the algorithm. Then $\{\mathbf{y}(t)\}_{t\geq 0}$ is bounded. Moreover, $\exists \mathbf{y}^* \in Y^*$ such that

$$|y_i(t+1) - y_i^*| \leq \Gamma \mu^t, \quad \forall i \in \mathcal{V}, \forall t \geq 0 \tag{11}$$

with $\Gamma = 2\frac{1+\gamma^{-T}}{1-\gamma^T}\|\mathbf{y}(0) - \mathbf{y}^*\|_1, \mu = (1-\gamma^T)^{\frac{1}{T}}, T = NT_1 + (N-1)T_0, \gamma = \min_{i\in\mathcal{V}} \min\{1-\alpha_i, \alpha_i\eta_i/2\}$ and

$$\eta_i = \min_{l \in \mathcal{N}_i^{\pm}} \min_{\mathbf{z}\in\bar{Y}} \Big(\frac{a_{li}e^{z_l}}{\sum_{k\in\mathcal{V}} a_{ki}e^{z_k}} + \frac{a_{il}e^{-z_l}}{\sum_{j\in\mathcal{V}} a_{ij}e^{-z_j}}\Big),$$

where $\bar{Y} \subset \mathbb{R}^N$ is any closed box containing $\{\mathbf{y}(t)\}_{t\geq 0}$.

Before proving this result, we note that the boundedness of $\{\mathbf{y}(t)\}_{t\geq 0}$ is necessary for the compactness of $\bar{Y}$, which in turn assures strict positivity of η_i and γ, and hence $\mu < 1$. We will prove the theorem by showing the equivalence of our algorithm and a network consensus problem, the convergence of which is demonstrated in the following result:

Theorem 2 ([19]): Consider $\varepsilon_i(t) \in \mathbb{R}$ that satisfies the following iteration for $i = 1, \ldots, N$ and $t = 0, 1, 2, \ldots$

$$\varepsilon_i(t+1) = w_{ii}(t)\varepsilon_i(t) + \textstyle\sum_{j\in\mathcal{V}\setminus\{i\}} w_{ij}(t)\varepsilon_j(t - \tau_{ij}(t)), \tag{12}$$

where $\tau_{ij}(t) \in [0, T_1 - 1], \forall i, j, t$ for some $T_1 > 0$, and $W(t) := [w_{ij}(t)] \in \mathbb{R}_+^{N\times N}$ satisfies the following for all $t \geq 0$:

(a) $W(t)\mathbf{1} = \mathbf{1}$ and $\exists \tilde{\gamma} \in (0,1)$ such that $w_{ii}(t) \geq \tilde{\gamma}$ and $w_{ij}(t) \in \{0\} \cup [\tilde{\gamma}, 1)$ for any $i, j = 1, \ldots, N$

(b) $\exists T_0 > 0$ such that the union of all graphs associated with $W(t), \ldots, W(t+T_0)$ is strongly connected.

Let $T = (N-1)T_0 + NT_1$. Then, there exists an $\bar{\varepsilon} \in \mathbb{R}$ such that

$$|\varepsilon_i(t+1) - \bar{\varepsilon}| \leq \Gamma\mu^t, \quad \forall i \in \mathcal{V}, \forall t \geq 0, \tag{13}$$

where $\Gamma = 2\frac{1+\tilde{\gamma}^{-T}}{1-\tilde{\gamma}^T}\sum_{j\in\mathcal{V}}|\varepsilon_j(0) - \bar{\varepsilon}|$ and $\mu = (1-\tilde{\gamma}^T)^{\frac{1}{T}}$.

Proof of Theorem 1: The proof is structured into the following steps:

(i) We first show that any sequence $\{\mathbf{y}(t)\}_{t\geq 0}$ generated by Algorithm 1 is bounded.

(ii) The error $\boldsymbol{\varepsilon}(t) = \mathbf{y}(t) - \mathbf{y}^\dagger$ satisfies the condition in Theorem 2 for any $\mathbf{y}^\dagger \in Y^*$.

(iii) Finally, $\mathbf{y}(t)$ converges geometrically to some $\mathbf{y}^* \in Y^*$.

Step (i): Consider any $\{\mathbf{y}(t)\}$ generated by Algorithm 1. Let

$$g_i(\mathbf{y}) := \tfrac{1}{2}\log\big(\textstyle\sum_{k\in\mathcal{V}} a_{ki}e^{y_k} / \sum_{j\in\mathcal{V}} a_{ij}e^{-y_j}\big), \quad \forall i \in \mathcal{V}$$
$$\mathbf{y}^{(i)}(t) := [y_1^{(i)}(t), \ldots, y_N^{(i)}(t)]^\mathsf{T}, \quad \forall t \geq 0, i \in \mathcal{V},$$

where $y_j^{(i)}(t)$ is given by (7) and $y_j^{(i)}(t) = 0$ whenever $j \notin \mathcal{N}_i^{\pm}$. Clearly, g_i is continuously differentiable and by (8),

$$g_i(\mathbf{y}^{(i)}(t)) = \tilde{y}_i(t), \quad \forall t \in \mathcal{T}_i.$$

Fix a $\mathbf{y}^\dagger \in Y^*$ (existence follows from Lemma 1) and define

$$\boldsymbol{\varepsilon}(t) := \mathbf{y}(t) - \mathbf{y}^\dagger.$$

It then follows from (6)–(9) that if $t \in \mathcal{T}_i$, then $\varepsilon_i(t+1) = (1-\alpha_i)\varepsilon_i(t) + \alpha_i\big(g_i(\mathbf{y}^{(i)}(t)) - y_i^\dagger\big)$; otherwise $\varepsilon_i(t+1) = \varepsilon_i(t)$. Using the fact that $g_i(\mathbf{y}^\dagger) = y_i^\dagger$ (i.e., the optimality condition in (4)) and the Mean Value Theorem, we have

$$g_i(\mathbf{y}^{(i)}(t)) - y_i^\dagger = \nabla g_i\big(\mathbf{z}^{(i)}(t)\big)^\mathsf{T}(\mathbf{y}^{(i)}(t) - \mathbf{y}^\dagger)$$

with $\mathbf{z}^{(i)}(t) = [1 - c_t^{(i)}]\mathbf{y}^{(i)}(t) + c_t^{(i)}\mathbf{y}^\dagger$ for some $c_t^{(i)} \in [0,1]$. Thus,

$$\varepsilon_i(t+1) = (1-\alpha_i)\varepsilon_i(t) + \alpha_i \sum_{l\in\mathcal{V}\setminus\{i\}} \partial_l g_i\big(\mathbf{z}^{(i)}(t)\big)\varepsilon_l(t - \tau_l^{(i)}(t))$$

for any $t \in \mathcal{T}_i$. Here, straightforward calculations show that

$$\partial_l g_i(\mathbf{z}) = \frac{1}{2}\Bigg[\frac{a_{li}e^{z_l}}{\sum_{k\in\mathcal{V}} a_{ki}e^{z_k}} + \frac{a_{il}e^{-z_l}}{\sum_{j\in\mathcal{V}} a_{ij}e^{-z_j}}\Bigg] \tag{14}$$

for all $\mathbf{z} \in \mathbb{R}^N$. Clearly, $\partial_l g_i(\mathbf{z}) = 0, \forall l \notin \mathcal{N}_i^{\pm}$ and

$$\partial_l g_i(\mathbf{z}) > 0 \text{ and } \partial_i g_l(\mathbf{z}) > 0, \quad \forall l \in \mathcal{N}_i^{\pm}. \tag{15}$$

Moreover, $\sum_{l\in\mathcal{V}} \partial_l g_i(\mathbf{z}) = 1, \forall \mathbf{z} \in \mathbb{R}^N, \forall i \in \mathcal{V}$.

Now let $W(t) = [w_{ij}(t)]$ be defined as follows for $t \geq 0$

$$w_{ij}(t) := \begin{cases} 1, & t \notin \mathcal{T}_i, j = i \\ 1 - \alpha_i, & t \in \mathcal{T}_i, j = i \\ \alpha_i \partial_j g_i\big(\mathbf{z}^{(i)}(t)\big), & t \in \mathcal{T}_i, j \in \mathcal{N}_i^{\pm} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Then, it follows that $W(t) \geq \mathbf{0}$ and $W(t)\mathbf{1} = \mathbf{1}, \forall t \geq 0$, i.e., $W(t)$ is always a stochastic matrix, showing that

$$\varepsilon_i(t+1) = \textstyle\sum_{j\in\mathcal{V}} w_{ij}(t)\varepsilon_j(t - \tau_j^{(i)}(t)), \quad \forall i \in \mathcal{V} \tag{17}$$

is an asynchronous linear consensus iteration with delays as in (12). This means that at any iteration, $\varepsilon_i(t+1)$ is always a convex combination of $\{\varepsilon_j(t - \tau_j^{(i)}(t))\}_{j\in\mathcal{V}}$. Define

$$\underline{\varepsilon}(t) = \min_{i\in\mathcal{V}, t-T_1+1\leq s\leq t} \varepsilon_i(s), \quad \bar{\varepsilon}(t) = \max_{i\in\mathcal{V}, t-T_1+1\leq s\leq t} \varepsilon_i(s),$$

for $t \geq T_1$, where T_1 is the maximum delay as per Assumption 3. Clearly, $\{\underline{\varepsilon}(t)\}$ is a nondecreasing sequence and $\{\bar{\varepsilon}(t)\}$ a nonincreasing one. Indeed, the following hold

$$\underline{\varepsilon}(t) \leq \underline{\varepsilon}(t+1) \leq \bar{\varepsilon}(t+1) \leq \bar{\varepsilon}(t), \quad \forall t \geq T_1. \tag{18}$$

Thus, both sequences $\{\underline{\varepsilon}(t)\}_{t\geq 0}$ and $\{\bar{\varepsilon}(t)\}_{t\geq 0}$ are bounded. As a result, $\{\boldsymbol{\varepsilon}(t)\}_{t\geq 0}$ is bounded and then so is $\{\mathbf{y}(t)\}_{t\geq 0}$.

Step (ii): From above, it remains to verify that both conditions (a) and (b) in Theorem 2 hold.

Clearly, condition (b) holds because of (15), (16) and Assumption 3 (a). For condition (a), we need to show that $w_{ij}(t)$ in (16) is either zero or strictly positive and bounded away from zero. Equivalently, we will demonstrate that $\partial_j g_i\big(\mathbf{z}^{(i)}(t)\big)$ is either 0 or bounded below by a positive constant for all i, j, t. To this end, let $\Omega \subset \mathbb{R}^N$ be any closed box containing $\{\mathbf{y}(t)\}_{t\geq 0}$ and $\mathbf{y}^\dagger$. Thus, Ω is convex and compact (because of the boundedness of $\{\mathbf{y}(t)\}_{t\geq 0}$) and $\{\mathbf{y}^{(i)}(t)\}_{t\geq 0} \subset \Omega, \forall i \in \mathcal{V}$. Since $\mathbf{z}^{(i)}(t)$ is a convex combination of $\mathbf{y}^{(i)}(t)$ and $\mathbf{y}^\dagger$, it follows that $\mathbf{z}^{(i)}(t) \in \Omega, \forall t \in \mathcal{T}_i$. Hence, we now consider $\partial_j g_i$ on Ω for any $(i,j) \in \mathcal{E}$. By continuity and positivity of $\partial_j g_i$ (see (14) and (15), respectively) and compactness of Ω, we have $\inf_{\mathbf{z}\in\Omega} \min_{l\in\mathcal{N}_i^\pm} \partial_l g_i(\mathbf{z})$ is achieved and strictly positive. Thus, there exists $\tilde{\gamma} > 0$ such that

$$\tilde{\gamma} := \min_{(i,l)\in\mathcal{E}} \min_{\mathbf{z}\in\Omega}\{1-\alpha_i, \alpha_i \partial_l g_i(\mathbf{z})\}.$$

This and (16) imply that $W(t)$ is stochastic and satisfies $w_{ij}(t) \geq \tilde{\gamma}$ if $w_{ij}(t) > 0$, thereby verifying condition (a).

Step (iii): It follows from Theorem 2 that $\exists \bar{\varepsilon} \in \mathbb{R}$ such that as $t \to \infty$, $\forall i \in \mathcal{V}$, $y_i(t) \to y_i^\dagger + \bar{\varepsilon}$ at a rate bounded above by $\mu = (1-\tilde{\gamma}^T)^{\frac{1}{T}}$. Now, define $\mathbf{y}^* := \mathbf{y}^\dagger + \bar{\varepsilon}\mathbf{1}$. Clearly, $F(\mathbf{y}^*) = F(\mathbf{y}^\dagger)$, hence $\mathbf{y}^* \in Y^*$. Moreover, $\mathbf{y}^* \in \bar{Y}$ since $\lim_{t\to\infty} \mathbf{y}(t) = \mathbf{y}^*$ and $\{\mathbf{y}(t)\}_{t\geq 0} \subset \bar{Y}$ (which is closed by definition). Therefore, $Y^* \cap \bar{Y}$ must be nonempty. Since the above arguments hold for any $\mathbf{y}^\dagger \in Y^*$, we can consider $\mathbf{y}^\dagger \in \bar{Y}$. Thus, the convergence rate μ can be refined by selecting $\Omega = \bar{Y}$ and $\tilde{\gamma} = \gamma$, as in the theorem statement. ■

Clearly, the convergence rate bound μ in (11) depends on the asynchronism degree, characterized by T and bounded above by $N(T_0+T_1)$, the network $\mathcal{G}$ through parameters N and $\{\eta_i\}$, the step sizes $\{\alpha_i\}$, and initializations $\mathbf{y}(0)$. In general, solving for η_i exactly may not be practical since it requires not only A but also $\bar{Y}$, which depends on $\{\mathbf{y}(t)\}$.

Remark 3: (Step Size Selection) First, each $\mathcal{C}_c$ can choose local step sizes $\{\alpha_i\}_{i\in\mathcal{C}_c}$ arbitrarily in $(0,1)$, and the algorithm still converges geometrically. On the one hand, this brings about simplicity in the algorithm implementation; on the other hand, optimal step sizes that maximize the convergence rate are hard to determine, even in a centralized manner. Second, it can be seen from the proof above that Algorithm 1 still converges geometrically under time-varying step sizes $\alpha_i(t) \in [\underline{\alpha}_i, \bar{\alpha}_i] \subset (0,1)$ for some $\underline{\alpha}_i$ and $\bar{\alpha}_i$.

Remark 4: (Relation to Network Flow Problem) The dual of (3) is a strict convex network flow problem with entropy cost functions, which was studied in [18, § 7.2] and Algorithm 1 can be viewed as an extension of the partially asynchronous algorithm therein that uses a uniform step size. It should be emphasized, however, that our convergence analysis is totally different from that in [18]; moreover, while the convergence rate was not established in [18], our approach gives rise to the geometric convergence.

V. Application to Epidemic Control

Applications of matrix balancing appear in diverse fields [2]. To the best of our knowledge, application to an optimal epidemic control problem is presented here for the first time.

A. Epidemic Network

Epidemic spreading processes appear in a range of network phenomena, such as social behaviors, diffusion of infections in people or computers, and cascading failures; see, e.g., [20]–[23]. Here, we adopt a continuous-time model, called the N-intertwined SIS, proposed in [22], [24]; see also [25]. Given a network of N agents with the contact graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each agent $i \in \mathcal{V}$ can be either *susceptible* or *infected*. The transition between these two states is characterized by its curing rate $\delta_i > 0$ and infection rates of its neighbors $\beta_{ij} > 0, (i,j) \in \mathcal{E}$. Thus, the network can be represented by a Markov model with 2^N states. Since an exact analysis of this model is difficult for large networks, the following first order mean-field approximation ([22], [24]) is often used:

$$\dot{p}(\tau) = \big(1 - p_i(\tau)\big)\big(\textstyle\sum_{j\in\mathcal{N}_i^+} \beta_{ij} p_j(\tau)\big) - \delta_i p_i(\tau), \quad \forall i \in \mathcal{V},$$

where $p_i(\tau) \in [0,1]$ denote the infection probability of node i at time $\tau \in \mathbb{R}_+$. This model exhibits two equilibria, namely, a *disease-free state*, where the network is cured eventually, i.e., $\mathbf{p}_\infty := \lim_{\tau\to\infty} \mathbf{p}(\tau) = \mathbf{0}$, and an *endemic state*, where each agent is infected with a strictly positive probability, i.e., $\mathbf{p}_\infty > \mathbf{0}$; see [24]. The phase transition is characterized by

$$\lambda_c := \max_{1\leq k\leq N} \mathsf{Re}\{\lambda_k(B - D)\}, \tag{19}$$

where $D = \mathrm{diag}(\delta_i)$, $B = [\beta_{ij}]$, and $\mathsf{Re}\{\lambda_k(B–D)\}$ is the real part of $\lambda_k(B–D)$. Specifically, if $\lambda_c > 0$, the network enters an endemic state, while the disease-free state is reached asymptotically if $\lambda_c \leq 0$, in which case, λ_c is also an upper bound on the decay rate of $\mathbf{p}(\tau)$. (Note that the *susceptible-infected-removed* model in [26] also has the same threshold. Thus, our results below can as well be applied to this model.)

B. Optimal Allocation of Curing Rates

Suppose the cost per curing unit associated with node $i \in \mathcal{V}$ is given by $w_i > 0$. Then, for a given curing profile $\boldsymbol{\delta} = [\delta_1, ..., \delta_N]^\mathsf{T}$, the total network-wide cost is simply $\mathbf{w}^\mathsf{T}\boldsymbol{\delta}$ where $\mathbf{w} = [w_1, ..., w_N]^\mathsf{T}$. This linear cost is similar to those in [27], [28]. We consider the problem of finding least amount of resources to prevent the epidemic spread:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}\geq\mathbf{0}} \quad \big\{\mathbf{w}^\mathsf{T}\boldsymbol{\delta} \mid \lambda_c \leq 0\big\}. \tag{20}$$

It has been shown [29] that (20) is equivalent to the following

$$\min_{\mathbf{u}>0} \quad \tilde{G}(\mathbf{u}) := \textstyle\sum_{(i,j)\in\mathcal{E}} w_i \beta_{ij} u_j / u_i \tag{21}$$

in the sense that $\delta_i^* = \sum_{j\in\mathcal{V}} \beta_{ij} u_j^* / u_i^*$, and $\tilde{G}^* = \sum_{i\in\mathcal{V}} w_i \delta_i^*$. Now, let $\mathbf{x}$ and $A = [a_{ij}]$ be such that $x_i = 1/u_i$ and $a_{ij} = w_i\beta_{ij}$, $\forall i, j$. Then, (21) becomes exactly (1) and thus can be solved in a distributed fashion by Algorithm 1. In this connection, Algorithm 1 can be seen as an extension of the synchronous algorithm in [29] to the asynchronous setting.

Finally, we remark that (20) is closely related to the following problems: (i) Determine a cost-optimal curing profile for a desired recovery rate $\lambda_0 \leq 0$, i.e., $\min_{\boldsymbol{\delta}\geq\mathbf{0}} \big\{\mathbf{w}^\mathsf{T}\boldsymbol{\delta} \mid \lambda_c \leq \lambda_0\big\}$; (ii) Given a sufficient budget $U > \tilde{G}^*$, optimize the recovery rate, i.e., $\min_{\boldsymbol{\delta}\geq\mathbf{0}} \big\{\lambda_c \mid \mathbf{w}^\mathsf{T}\boldsymbol{\delta} \leq U\big\}$; and (iii) When budget U is insufficient, suppress the endemic size, namely,

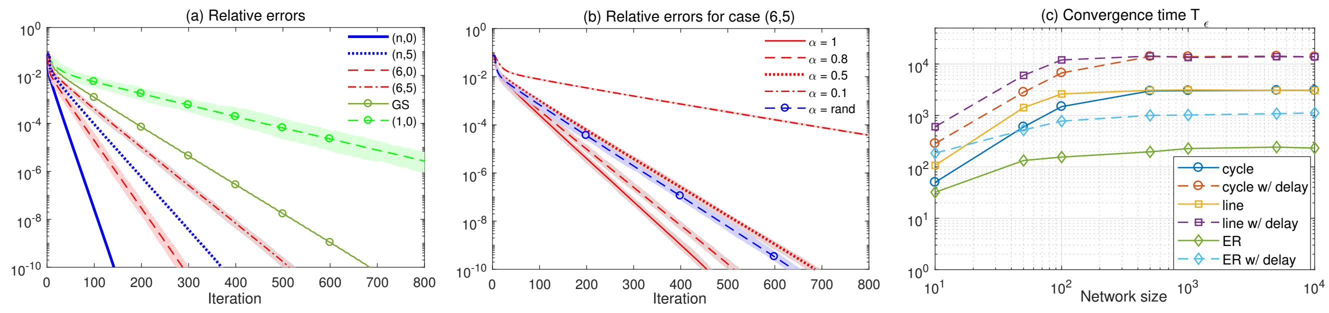

Fig. 2. (a) Relative errors $\frac{G(\mathbf{x}(t))-G^*}{G^*}$ for different parameters (R, D) in Example 1, where GS is the Gauss-Seidel algorithm; (b) Relative errors in Example 1 for different α, where α=rand means that each α_i is selected u.a.r. in $(0, 1)$; (c) ϵ-convergence times in Example 2 with $\epsilon = 10^{-3}$.

$\min_{\boldsymbol{\delta}\geq\mathbf{0}} \left\{\mathbf{1}^\mathsf{T}\mathbf{p}_\infty \,|\, \mathbf{w}^\mathsf{T}\boldsymbol{\delta} \leq U\right\}$. In fact, it is shown in [29] that both (i) and (ii) are equivalent to (20), a solution of which can as well be used to obtain suboptimal solutions to (iii).

VI. Numerical Examples

Example 1: Consider the network in Fig. 1, where the edge weights are selected uniformly at random (u.a.r.) in $(0, 1)$. Fig. 2 (a) shows the performance of Algorithm 1 with $\alpha_i = 0.8, \forall i \in \mathcal{V}$ and under various choices of (R, D), where R is the number of active nodes (selected u.a.r. in $[1, N]$) at each iteration and all delays are selected u.a.r. in $[1, D]$ with $D = T_0 - 1$. In each case (except for $(R, D) = (n, 0)$ and GS), we show the average and dispersion of 100 runs. To assess the effect of α_i, we consider the case $(R, D) = (6, 5)$ with several choices of $\alpha_i \equiv \alpha$. The results are shown in Fig. 2 (b), where it appears that larger α yields faster convergence.

Example 2: To examine the scalability of Algorithm 1, we apply it to a set of topologies {`line`, `cycle`, `ER`} and with various sizes $N \in \{10, 50, 100, 500, 1000, 5000, 10^4\}$, where `ER` networks are giant components (of size at least $0.8N$) of Erdos-Renyi graphs generated with parameter $p = \frac{2}{N-1}$. For each network, we run 100 simulations (each with edge weights selected u.a.r. from $(0, 1)$, $R = N$ and $\tau_j^{(i)} \in \{0, 5\}$) and find the average running time $T_\epsilon = \min\{t \,|\, f(t) \leq \epsilon f(0)\}$ where $\epsilon = 10^{-3}$ and we use $f(t) = \|\nabla F(\mathbf{y}(t))\|_2$ to measure the total imbalance at iteration t. The results shown in Fig. 2 (c) suggest that T_ϵ is not exponential in N.

References

[1] M. H. Schneider and S. A. Zenios, "A comparative study of algorithms for matrix balancing," *Oper. Res.*, vol. 38, no. 3, pp. 439–455, 1990.

[2] M. Idel, "A review of matrix scaling and Sinkhorn's normal form for matrices and positive maps," *arXiv preprint arXiv:1609.06349*, 2016.

[3] S. Robinson, A. Cattaneo, and M. El-Said, "Updating and estimating a social accounting matrix using cross entropy methods," *Economic Systems Research*, vol. 13, no. 1, pp. 47–64, 2001.

[4] E. Osborne, "On pre-conditioning of matrices," *J. ACM*, vol. 7, no. 4, pp. 338–345, 1960.

[5] B. Kalantari, L. Khachiyan, and A. Shokoufandeh, "On the complexity of matrix balancing," *SIAM J. Matrix Anal. Appl.*, vol. 18, no. 2, pp. 450–463, 1997.

[6] R. Ostrovsky, Y. Rabani, and A. Yousefi, "Matrix balancing in L_p norms: bounding the convergence rate of Osborne's iteration," in *Proc. 28th ACM-SIAM Symp. Discrete Algorithms*, 2017, pp. 154–169.

[7] L. J. Schulman and A. Sinclair, "Analysis of a classical matrix preconditioning algorithm," *J. ACM*, vol. 64, no. 2, p. 9, 2017.

[8] M. Cohen, A. Madry, D. Tsipras, and A. Vladu, "Matrix scaling and balancing via box constrained Newton's method and Interior Point methods," in *IEEE 58th Annu. Symp. Found. Computer Science*, 2017, pp. 902–913.

[9] B. N. Parlett and C. Reinsch, "Balancing a matrix for calculation of eigenvalues and eigenvectors," *Numerische Mathematik*, vol. 13, no. 4, pp. 293–304, 1969.

[10] A. I. Rikos, T. Charalambous, and C. N. Hadjicostis, "Distributed weight balancing over digraphs," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 190–201, 2014.

[11] A. W. Marshall and I. Olkin, "Scaling of matrices to achieve specified row and column sums," *Numerische Mathematik*, vol. 12, no. 1, pp. 83–90, 1968.

[12] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *Ann. Math. Stat.*, vol. 35, no. 2, pp. 876–879, 1964.

[13] N. Linial, A. Samorodnitsky, and A. Wigderson, "A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents," in *Proc. 13th Annu. ACM Symp. Theo. Comput.* ACM, 1998, pp. 644–652.

[14] B. Kalantari and L. Khachiyan, "On the complexity of nonnegative-matrix scaling," *Lin. Algebra Appl.*, vol. 240, pp. 87–103, 1996.

[15] A. D. Dominguez-Garcia and C. N. Hadjicostis, "Distributed matrix scaling and application to average consensus in directed graphs," *IEEE Trans. Autom. Control*, vol. 58, no. 3, pp. 667–681, 2013.

[16] T. Elfving, "On some methods for entropy maximization and matrix scaling," *Linear Algebra Appl.*, vol. 34, pp. 321–339, 1980.

[17] A. Nemirovski and U. Rothblum, "On complexity of matrix scaling," *Lin. Algebra Appl.*, vol. 302, pp. 435–460, 1999.

[18] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall Englewood Cliffs, NJ, 1989.

[19] A. Nedić and A. Ozdaglar, "Convergence rate for consensus with delays," *J. Global Optim.*, vol. 47, no. 3, pp. 437–456, 2010.

[20] M. E. Newman, "Spread of epidemic disease on networks," *Phys. Rev. E*, vol. 66, no. 1, p. 016128, 2002.

[21] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: An eigenvalue viewpoint," in *Proc. 22nd Int. Symp. Reliable Dist. Syst.*, 2003, pp. 25–34.

[22] P. Van Mieghem, J. Omic, and R. Kooij, "Virus spread in networks," *IEEE/ACM Trans. Net.*, vol. 17, no. 1, pp. 1–14, 2009.

[23] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *Rev. Mod. Phys.*, vol. 87, no. 3, p. 925, 2015.

[24] P. Van Mieghem and J. Omic, "In-homogeneous virus spread in networks," *arXiv preprint arXiv:1306.2588*, 2013.

[25] V. Preciado, M. Zargham, C. Enyioha, A. Jadbabaie, and G. Pappas, "Optimal resource allocation for network protection against spreading processes," *IEEE Trans. Contr. Netw. Syst.*, vol. 1, no. 1, pp. 99–108, 2014.

[26] M. Youssef and C. Scoglio, "An individual-based approach to SIR epidemics in contact networks," *J. Theor. Biol.*, vol. 283, no. 1, pp. 136–144, 2011.

[27] C. Borgs, J. Chayes, A. Ganesh, and A. Saberi, "How to distribute antidote to control epidemics," *Random Struct. Algor.*, vol. 37, no. 2, pp. 204–222, 2010.

[28] E. Ramírez-Llanos and S. Martínez, "Distributed discrete-time optimization algorithms with applications to resource allocation in epidemics control," *Optim. Control Appl. Methods*, pp. 1–21, 2017.

[29] V. S. Mai, A. Battou, and K. Mills, "Distributed algorithm for suppressing epidemic spread in networks," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 555–560, 2018.