Structural Analysis and Optimal Design of Distributed System Throttlers

Milad Siami, *Member, IEEE*, and Joëlle Skaf

***Abstract*—In this paper, we investigate the performance analysis and synthesis of distributed system throttlers (DST). A throttler is a mechanism that limits the flow rate of incoming metrics, e.g., byte per second, network bandwidth usage, capacity, traffic, etc. This can be used to protect a service's backend/clients from getting overloaded, or to reduce the effects of uncertainties in demand for shared services. We study performance deterioration of DSTs subject to demand uncertainty. We then consider network synthesis problems that aim to improve the performance of noisy DSTs via communication link modifications as well as server update cycle modifications.**

***Index Terms*—Cooperative control, distributed system throttler, network analysis and control, optimization, traffic control.**

I. Introduction

System throttling (also known as rate-limiting) aims to limit the total number of requests from all clients to a shared service and provide a harmonized and fair quota allocation among them (where the definition of fairness is application-specific). Examples of systems in need of throttling protection include cloud-based services and traffic management services. A number of works on rate-limiting systems and congestion control have been published in the recent literature [1]–[8].

System throttlers can be classified into centralized and distributed types. In a centralized system throttler (CST), there is a single decision maker that sets the per-client limits according to aggregated metrics it receives from multiple servers, which in turn aggregate them from metrics reported by the clients. CSTs are designed based on a globally aggregated view of usage metrics. On the other hand, a distributed system throttler (DST) does not have a centralized mechanism for setting per-client limits: It consists of multiple servers, each of which makes autonomous decisions and updates its own limit based on measurements it takes as well as local information.

While the centralized approach has benefits, including consistency and ease of implementation and analysis, it also has drawbacks relative to a decentralized version: 1) Less local adaptability: in a centralized version, each server needs to send information to the decision-making server and wait for its command, which means a delayed response time. 2) Limited communication: there is no inter-server communication except to the decision-making server. Moreover, we want to facilitate information propagation in order to improve the performance and to make the network more flexible and fast when handling uncertainty in demand.

There are some related works in the literature that study performance and robustness issues in noisy linear distributed systems; for example, see [9]–[17] and the references therein. In [9], Bamieh *et al.* investigate the deviation from the mean of states of a continuous-time consensus network on tori with additive noise inputs. A rather comprehensive performance analysis of noisy linear consensus networks with arbitrary graph topologies has been recently reported in [11]. In [11], several fundamental tradeoffs between a $\mathcal{H}_2$-based performance measure and sparsity measures of a continuous-time linear consensus network are studied. Moreover, [18] studies a $\mathcal{H}_2$-based performance measure of continuous-time linear consensus system in the presence of a time-delay and additive noise inputs. Most of these papers treat continuous-time systems only; In discrete-time networks, however, the time-step size along with the topology of the network plays an important role on the network performance.

We should mention that papers [7] and [8] investigate the notion of distributed rate-limiting as a mechanism that controls the aggregate service used by a client of a cloud-based service. The main idea is to improve a set of cloud servers with the ability to exchange information with them toward the common purpose: control of the aggregate usage that a cloud-based service experiences. However, comprehensive performance analysis and synthesis have yet to be done for these networks with an arbitrary underlying graph.

In this paper, our goal is to develop a unified framework for analysis and design of discrete-time distributed rate-limiting systems with a local aggregated view of usage metrics. We investigate performance deterioration (e.g., over-throttling, mismatch, convergence rate) of DSTs with respect to external uncertainties and the update cycle of servers. We develop a graph-theoretic framework to relate the underlying structure of the system to its overall performance measure. We then compare the performance/robustness of DSTs with different topologies. In this work, in addition to the overall performance measure for a network, each node has its own performance measure, which is one of the main differentiators between this work and some other related work [9]–[11], [18].

The rest of this paper is organized as follows. In Section II, we present some basic mathematical concepts and notations employed in this paper. In Section III, we define and study a DST. In Section IV, we evaluate the overall performance of a DST with a given nodal performance measure. In Subsection V-A, we study the impact of the server update cycle on performance. In Subsection V-B, two synthesis problems are studied. In Section VI, some numerical results are demonstrated. In Section VII, we focus on throttling algorithms that are used by servers. In Section VIII, we conclude our work and suggest directions for future research.

II. Mathematical Notation

Throughout the paper, the discrete time index is denoted by k. The sets of real (integer), positive real (integer), and strictly positive real (integer) numbers are represented by $\mathbb{R}$ ($\mathbb{Z}$), $\mathbb{R}_+$ ($\mathbb{Z}_+$) and $\mathbb{R}_{++}$ ($\mathbb{Z}_{++}$),

Manuscript received October 1, 2016; revised October 3, 2016 and May 15, 2017; accepted July 5, 2017. Date of publication July 17, 2017; date of current version January 26, 2018. Recommended by Associate Editor Ming Cao. *(Corresponding author: Milad Siami.)*.

M. Siami is with the Institute for Data, Systems, and Society, Massachusetts Institute of Technology, Cambridge, MA 02319, USA. He also performed the work as a summer intern at Google, NYC. (e-mail: siami@mit.edu).

J. Skaf is with Google Inc., New York, NY 10011 USA (e-mail: jskaf@google.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2017.2728002

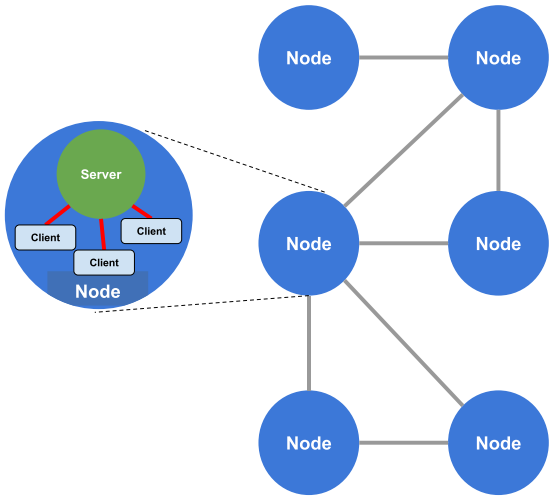

Fig. 1. Example of a DST with six servers. Nodes show servers and links present communication links between servers.

respectively. Capital letters, such as A or B, stand for real-valued matrices. We use $\mathrm{diag}(x_1, x_2, \ldots, x_n)$ to denote a n-by-n diagonal square matrix with x_1 to x_n on its diagonal. For a square matrix X, $Trace(X)$ refers to the summation of on-diagonal elements of X. We represent the n-by-1 vector of ones by $\mathbf{1}$. The n-by-n identity matrix is denoted by I. The Moore–Penrose pseudoinverse of matrix A is denoted by $A^\dagger$, i.e., $A^\dagger = \left(A + \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)^{-1} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$. We assume that all graphs are connected, undirected, simple graphs. We represent graph $\mathcal{G}$ by (V, E, w), where V is the node set, E is the edge set, and $w : E \to \mathbb{R}_+$ is the link weight function. We denote by L the Laplacian matrix of the weighted graph $\mathcal{G}$ with the following eigenvalues

$$\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_n . \tag{1}$$

Since we assume in this work that all graphs are connected, it follows that $\lambda_2 > 0$.

The *effective resistance* between nodes i and j is defined by

$$r_{ij} := l^\dagger_{ii} + l^\dagger_{jj} - l^\dagger_{ji} - l^\dagger_{ij} \tag{2}$$

where $l^\dagger_{ji}$ is the (i, j)th entry in $L^\dagger$. The white Gaussian noise with zero mean and variance σ^2 is represented by $v \sim N(0, \sigma^2)$.

III. Distributed System Throttler

A DST is a graph $\mathcal{G}$ with n nodes. Each node in the graph is a server with assigned clients that can send it requests. Links in the graph represent communication channels between servers. The global goal of a DST is to keep the aggregate number of accepted requests from all clients for a shared service at or below a prescribed level. The DST does not have a centralized mechanism for setting per-client limits. Instead it consists of multiple servers, each of which makes its own decisions and updates its own limit based on its own measurements and local information from its neighbors (on graph $\mathcal{G}$). Fig. 1 depicts an example of a distributed throttler with six nodes (servers).

Let's denote by $r_i(k)$ the total number of client requests received by server i at time k. Each node has a total limit on the number of requests that it is allowed to service at time k represented by $x_i(k)$. It is also associated with a performance measure $p_i(k)$ which represents how well that node is working at time k. Examples of typical performance measures are: over-throttling at time k, the ratio of the total allowed usage to total requested usage, or any function of $r_i(k), x_i(k)$, and time. We will talk about functional properties of the performance measure later on in this paper (see Table I).

In this setup, we assume that each node updates its state $x_i(k)$ based on its neighbors' states and performance measures (i.e., a local aggregated view of usage metrics). The update law is given by the following difference equation:

$$x_i(k+1) \;=\; x_i(k) \;+\; \gamma \sum_{i \sim j} w_{ij}\big(p_i(k) - p_j(k)\big), \;\; k \in \mathbb{Z}_+ \tag{3}$$

where $i \sim j$ denotes that nodes i and j are connected by a link in the underlying graph, $w_{ij} = w(\{i, j\})$ is the weight of link $\{i, j\}$ in graph $\mathcal{G}$, and parameter γ is a positive number that depends on the size of the time step (i.e., $x(k) := x(k\Delta t)$ where $\Delta t = \gamma$). In Subsection V-A, γ is referred to as the server update cycle, and its effect on the performance analysis will be discussed.

The dynamics of the entire network can be written in the following compact form

$$x(k+1) \;=\; x(k) \;+\; \gamma\, L\, p(k), \;\; k \in \mathbb{Z}_+ \tag{4}$$

where $x(k)$ is an n-by-1 vector of node limits at time k, $p(k)$ is an n-by-1 vector of nodal performance measures at time k, and L is the Laplacian matrix of the coupling graph $\mathcal{G}$. Then, the accepted number of requests at server i at time k is given by $a_i(k) := \min\{x_i(k), r_i(k)\}$. The total number of requests, the total limit, and the total number of accepted requests for the entire network are defined by

$$r_{\text{total}}(k) \;:=\; \sum_{i=1}^{n} r_i(k) \tag{5}$$

$$l_{\text{total}} \;:=\; \sum_{i=1}^{n} x_i(0) \tag{6}$$

and

$$a_{\text{total}}(k) \;:=\; \sum_{i=1}^{n} \min\{x_i(k), r_i(k)\} \tag{7}$$

respectively. The ideal curve for total number of accepted requests is given by

$$a_{\text{ideal}}(k) \;:=\; \min\{l_{\text{total}}, r_{\text{total}}(k)\}\,. \tag{8}$$

We should note that in compact form (4), weights do not disappear, and are encoded in matrix L. Here L is the Laplacian matrix of weighted graph $\mathcal{G}$. Hence, off-diagonal elements of the matrix represent $-w_{ij}$'s.

The following lemma shows that the total nodal limit is fixed over time.

Lemma 1: The total summation of nodal limits is fixed over time, which means

$$\sum_{i=1}^{n} x_i(k) \;=\; \sum_{i=1}^{n} x_i(0), \quad \text{for all } k \in \mathbb{Z}_{++}\,. \tag{9}$$

Proof: We multiply both sides of (4) by $\mathbf{1}^T$ on the left and get

$$\sum_{i=1}^{n} x_i(k+1) \;=\; \sum_{i=1}^{n} x_i(k) \;+\; \gamma\, \mathbf{1}^T\, L\, p(t). \tag{10}$$

Assume that $p_i(k)$'s are bounded. Since L is the Laplacian matrix of an undirected graph, its row and column sums are zero which completes the proof. ■

Based on this result, the total sum of nodal limits is constant and it depends only on initial values, i.e., l_{total}. A similar result is reported in [8], which guarantees the capacity constraint for a generalized distributed rate-limiting system. The condition (10) holds for any linear consensus network even for those over directed graphs.

TABLE I
EXAMPLES OF NODAL PERFORMANCE MEASURES

Case I	Amount of throttled traffic	$p_i(k) := r_i(k) - x_i(k)$
Case II	Throttled-to-requested traffic ratio	$p_i(k) := \big(r_i(k) - x_i(k)\big)/r_i(k)$
Case III	Logarithm of requested-to-allowed traffic ratio	$p_i(k) := \log\big(r_i(k)/x_i(k)\big)$
Case IV	Amount of allowed traffic	$p_i(k) := x_i(k)$

In the next section, we study the overall performance of DST networks based on their nodal performance measure and the behavior of incoming network traffic.

IV. PROPERTIES OF TYPICAL NODAL PERFORMANCE MEASURES

Each node i is associated with a performance measure $p_i(k)$, which shows the performance of server i at time k. Some examples of performance measures are presented in Table I.

In this section, we choose $p_i(k)$ to be the number of throttled requests at node i at time k

$$p_i(k) \;:=\; r_i(k) - x_i(k). \tag{11}$$

Then, (3) can be rewritten as

$$p(k+1) \;=\; (I - \gamma\, L)\, p(k) \;+\; \big(r(k+1) - r(k)\big),\; k \in \mathbb{Z}_+. \tag{12}$$

Based on the behavior of incoming network traffic/requests, two cases are considered.

A. Steady Loads

Let us assume that the number of client requests incoming at node i is constant across time

$$r_i(k+1) - r_i(k) \;=\; 0,\; k \in \mathbb{Z}_+. \tag{13}$$

Equation (12) can then be simplified as below

$$p(k+1) \;=\; (I - \gamma\, L)\, p(k),\; k \in \mathbb{Z}_+. \tag{14}$$

Lemma 2 ([19]): For any $i, j \in \{1, 2, \ldots, n\}$, we have

$$\lim_{k\to\infty} |p_i(k) - p_j(k)| \;=\; 0$$

if and only if $\max\{1 - \gamma\lambda_2, \gamma\lambda_n - 1\} < 1$

Proof: It is straightforward. ■

Based on this result, as long as graph $\mathcal{G}$ is connected we can find a positive γ, which guarantees reaching a consensus state (for a small enough positive number γ).

We can now study the convergence rate based on properties of the underlying graph and the design parameter γ.

Let us define the following performance measure that shows the convergence rate of the DST

$$\Phi_{\text{cr}} \;=\; \max_{i\geq 2} |1 - \gamma\lambda_i| \tag{15}$$

a smaller Φ_{cr} meaning faster asymptotic convergence.

Remark 1 (Role of Topologies for Small γ): Networks with n servers can be ranked based on their convergence rates; consequently, the path graph topology has the worst convergence rate and the complete graph has the best convergence rate (for small enough γ). Also, it can be shown that among tree graphs, star graphs have the best rate and path graphs have the worst. ◇

B. Nonsteady Loads

Assumption (13) is strong and can be relaxed. Let us assume that

$$v_i(k+1) \;:=\; r_i(k+1) - r_i(k) \tag{16}$$

where $v(k) \in \mathbb{R}^n$ is a zero mean random vector such as

$$\begin{aligned} \mathbb{E}\,[v(k)] \;&=\; \mathbf{0}, \\ \mathbb{E}\,\big[v(k)v^T(k)\big] \;&=\; \text{Cov}(v) \\ \mathbb{E}\,\big[v(k)v^T(s)\big] \;&=\; \mathbf{0},\; k \neq s. \end{aligned} \tag{17}$$

Then, (12) can then be simplified as below

$$p(k+1) \;=\; (I - \gamma L)\, p(k) \;+\; v(k+1),\; k \in \mathbb{Z}_+. \tag{18}$$

We can now define the following overall performance measure for the network

$$\Phi_{\text{ss}} \;=\; \lim_{k\to\infty} \mathbb{E}\left[\frac{1}{2n}\sum_{i,j} \big(p_i(k) - p_j(k)\big)^2\right]. \tag{19}$$

The quantity (19) shows the steady-state dispersion of p_i's from their average [9]–[11].

The following theorem presents a closed-form formula for the overall performance of DST (18), based on the Laplacian matrix of the underlying graph and the covariance matrix of the input vector v.

Theorem 1: For a given DST (18), the overall performance measure (19) can be quantified as

$$\Phi_{\text{ss}} = \frac{1}{2\gamma}\text{Trace}\left[\Big(L - \frac{\gamma}{2}L^2\Big)^{\dagger}\,\text{Cov}(v)\right] \tag{20}$$

where $\text{Cov}(v)$ is the covariance matrix of random vector $v(k)$.

Proof: The overall performance measure is the same as the squared $\mathcal{H}_2$-norm of a discrete linear time invariant system (18). Therefore, the measure can be quantified as follows:

$$\Phi_{\text{ss}} \;=\; \text{Trace}\,[Q\,\text{Cov}(v)] \tag{21}$$

where $Q \succeq \mathbf{0}$ is the solution of the following discrete Lyapunov equation

$$(I - \gamma L)Q(I - \gamma L)^T \;-\; Q \;+\; \big(I - 1/n\mathbf{1}\mathbf{1}^T\big) \;=\; \mathbf{0}.$$

By doing some calculation it follows that

$$Q \;=\; (2\gamma L - \gamma^2 L^2)^{\dagger}. \tag{22}$$

Using (21) and (22) we get the desired result. ■

Remark 2 (Independent v_i's): In the case where v_i's are independent then $\text{Cov}(v)$ is a diagonal matrix $\gamma\,\text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ where σ_i depends on the property of signal r_i. We get

$$\Phi_{\text{ss}} \;=\; \frac{1}{2}\sum_{i=1}^{n} c_{ii}^{\dagger}\sigma_i^2 \tag{23}$$

where $(L - \frac{\gamma}{2}L^2)^{\dagger} = [c_{ij}^{\dagger}]$. Based on (23), we can obtain a centrality measure for servers. Indeed, $c_{ii}^{\dagger}$ shows the impact of server i on the overall performance. See [10] for more details on centrality measures

with respect to $\mathcal{H}_2$-norm of the system (the focus of [10] is on the class of continuous-time linear consensus networks however.) ◇

Remark 3 (Independent and identically distributed v_i's): Based on Theorem 1, the overall performance measure of the network can be calculated based on spectral eigenvalues of the coupling graph and the variance of changing demands (i.e., $r_i(k+1) - r_i(k) \sim N(0, \gamma\,\sigma^2)$) as follows

$$\Phi_{\rm ss} \;=\; \begin{cases} \sum_{i=2}^{n} \frac{\sigma^2}{\lambda_i\,(2-\gamma\,\lambda_i)}, \; 0 < \lambda_i < 2/\gamma \text{ for } i = 2, \ldots, n \\ \infty, \qquad\qquad \text{otherwise.} \end{cases} \tag{24}$$

We note that condition $0 < \lambda_i < 2/\gamma$ for $i = 2, \ldots, n$ is the same as the one needed for the system without noise to converge (cf., Lemma 2). ◇

The quantity (24) has a close connection with the "total effective resistance" of an electric network as follows

$$\lim_{\gamma \to 0} \Phi_{\rm ss} \;=\; \frac{\sigma^2}{2n} \sum_{i>j} r_{ij} \tag{25}$$

where r_{ij} is the effective resistance between node i and j, i.e.

$$r_{ij} \;:=\; l^\dagger_{ii} \;+\; l^\dagger_{jj} \;-\; l^\dagger_{ij} \;-\; l^\dagger_{ji}, \;\; L^\dagger = [l^\dagger_{ij}].$$

For more details see [20].

Remark 4 (Another interpretation of the overall measure): Let us assume that $r_i(0)$'s are given with the normal distribution, and r_i's remain constant. Then the expected total mismatch loss can be obtained based on

$$\mathbb{E}\left[\frac{1}{n}\sum_{k=0}^{\infty}\sum_{i>j}\left(p_i(k) - p_j(k)\right)^2 \Delta t\right] \;= \\ \frac{1}{2\gamma}\text{Trace}\left[\left(L - \frac{\gamma}{2}L^2\right)^\dagger \text{Cov}(v)\right] \;=\; \Phi_{\rm ss}. \tag{26}$$

◇

Due to space limitations, other nodal performance measures defined in Table I are briefly analyzed in the Appendix.

V. DST Optimization Problems

A. Impact of the Server Update Cycle

In this subsection, we study the effect of the server update cycle γ on our analysis. As shown in Section IV, the overall performance measure of a DST depends on its Laplacian eigenvalues and the server update cycle. To enhance the overall performance of the network, one can obtain the optimal update cycle for all servers.

The following theorem presents the optimal update cycle for a DST in the case of steady loads (i.e., when the number of client requests is constant across time).

Theorem 2: For a given DST (14) with a graph $\mathcal{G}$, the optimal update cycle is given by

$$\gamma_{\rm optimal} \;=\; \frac{2}{\lambda_2 + \lambda_n}. \tag{27}$$

Proof: We need to solve the following convex optimization

$$\underset{\gamma>0}{\text{minimize}} \;\; \max_{i\geq 2} \left|1 - \gamma\lambda_i\right|. \tag{28}$$

It is not difficult to see that $2(\lambda_2 + \lambda_n)^{-1}$ minimizes the cost function. We have

$$0 < \lambda_2 \leq \cdots \leq \lambda_n$$

and, accordingly, we can rewrite the cost function as follows

$$\max_{i\geq 2} \left|1 - \gamma\lambda_i\right| = \max\left\{1 - \gamma\lambda_2, \gamma\lambda_n - 1\right\}. \tag{29}$$

To minimize (29), we need

$$1 - \gamma\lambda_2 \;=\; \gamma\lambda_n - 1 \tag{30}$$

since if $1 - \gamma\lambda_2 \neq \gamma\lambda_n - 1$, one can decrease the cost function by increasing or decreasing γ. Therefore, the optimal γ is the solution of (30). This completes the proof. ■

In the case of nonsteady loads, having a closed-form formula for the optimal update time based on the Laplacian eigenvalues seems difficult. However, one can obtain the solution by solving the following convex optimization problem:

$$\underset{\gamma>0}{\text{minimize}} \;\; \frac{1}{2\gamma}\text{Trace}\left[\left(L - \frac{\gamma}{2}L^2\right)^\dagger \text{Cov}(v)\right]. \tag{31}$$

In the case where v_i's are independent and identically distributed (i.e., $\text{Cov}(v) = \gamma\,\text{diag}(\sigma^2, \ldots, \sigma^2)$), the optimal update time can be bounded from above and below by $1/\lambda_2$ and $1/\lambda_n$, respectively.

B. DST Synthesis Problems

In this subsection, we present our main results on the design of optimal distributed rate-limiting systems. We formulate our problems as convex optimization problems. The questions we are trying to answer in this section are

1) What are the optimal link weights for the *fastest* DST network?
2) What are the optimal link weights for the *most robust* DST network?

Depending on which nodal and overall performance measures are chosen, one can come up with different optimal topologies.

1) Fastest DST Process: Here we briefly describe the problem of finding the fastest DST on a given underlying topology, where "fastest" means the one with the smallest $\Phi_{\rm cr}$. The optimal weights can be found by solving the following optimization problem

$$\begin{aligned} &\underset{w(e)}{\text{minimize}} \;\; \max_{i\geq 2} \left|1 - \gamma\lambda_i\right| \\ &\text{subject to} \;\; w(e) \;\geq\; 0, \;\; \text{for all } e \in E. \end{aligned} \tag{32}$$

This problem was studied before in [21]. Problem (32) can be cast as a semidefinite programming (SDP) problem as follows

$$\begin{aligned} &\underset{w(e),\theta}{\text{minimize}} \;\; \theta \\ &\text{subject to} \;\; -\theta I \preceq I - \gamma\sum_{e\in E} w(e)L_e - \frac{1}{n}\mathbf{1}\mathbf{1}^T \preceq \theta I \\ &\qquad w(e) \geq 0, \;\; e \in E \end{aligned} \tag{33}$$

where L_e is the unweighted Laplacian matrix of link e.

2) Most Robust DST Process: Here we briefly describe the problem of finding the most robust DST on a given underlying topology, where "most robust" means the one with the smallest $\Phi_{\rm ss}$. The optimal weights can be found by solving the following problem:

$$\begin{aligned} &\underset{w(e)}{\text{minimize}} \;\; \frac{1}{2\gamma}\text{Trace}\left[\left(L - \frac{\gamma}{2}L^2\right)^\dagger \text{Cov}(v)\right] \\ &\text{subject to} \;\; w(e) \;\geq\; 0, \;\; \text{for all } e \in E \\ &\qquad L \;=\; \sum_{e\in E} w(e)L_e \\ &\qquad \max_{i\geq 2} \left|1 - \gamma\lambda_i\right| \;\leq\; 1. \end{aligned} \tag{34}$$

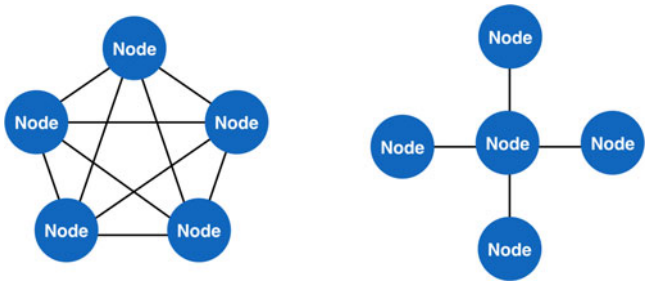

Fig. 2. Two DST networks with five servers over a complete graph and a star graph.

We note that $\Phi_{\rm ss} = \frac{1}{2\gamma}\text{Trace}\left(\left(L - \frac{\gamma}{2}L^2\right)^{\dagger}\text{Cov}(v)\right)$ is a convex function of the link weights. To find the solution of (34) one can use a variety of standard methods for convex optimization (e.g., interior-point methods and subgradient-based methods).

Theorem 3: Problem (34) can be formulated as a SDP problem as follows

$$\begin{aligned}
&\underset{w(e),Y}{\text{minimize}} \ \frac{1}{2\gamma}\text{Trace}\left[Y\,\text{Cov}(v)\right] - \frac{\mathbf{1}^T\text{Cov}(v)\mathbf{1}}{2n\gamma^2}\\
&\text{subject to} \ \ w(e) \geq 0, \ \text{for all } e \in E\\
&\qquad L = \sum_{e\in E} w(e)L_e\\
&\qquad \mathbf{0} \preceq I - \frac{1}{2}\left(\gamma L + (1/n)\mathbf{1}\mathbf{1}^T\right) \preceq I\\
&\qquad \begin{bmatrix} L + \frac{1}{\gamma n}\mathbf{1}\mathbf{1}^T & L & I \\ L & \frac{2}{\gamma}I & \mathbf{0} \\ I & \mathbf{0} & Y \end{bmatrix} \succeq 0. \qquad (35)
\end{aligned}$$

Proof: We need the following condition to hold in order to guarantee that the network is marginally stable:

$$\mathbf{0} \preceq I - \frac{1}{2}\left(\gamma L + (1/n)\mathbf{1}\mathbf{1}^T\right) \preceq I. \qquad (36)$$

Then, according to (36) and the Schur complement condition for positive semidefiniteness it follows that

$$\begin{bmatrix} L + \frac{1}{\gamma n}\mathbf{1}\mathbf{1}^T & L \\ L & \frac{2}{\gamma}I \end{bmatrix} \succeq 0. \qquad (37)$$

Again, using the Schur complement condition for positive semidefiniteness, (35) and (37), we get the following equivalent condition

$$Y - \frac{1}{\gamma n}\mathbf{1}\mathbf{1}^T \succeq \left(L - \frac{\gamma}{2}L^2\right)^{\dagger} \qquad (38)$$

which completes the proof. ■

VI. Illustrative Numerical Simulations

In this section, we support our theoretical developments with illustrative examples that provide better insight into the role of the underlying graph topology in the DST network.

Example 1: Consider two DST networks with five servers over complete graph $\mathcal{K}_5$ and star graph $\mathcal{S}_5$ as depicted in Fig. 2. Let us assume that the update cycle is given and fixed (without loss of generality $\gamma = 1$). Based on the results presented in Theorem 2, one can obtain the optimal weight links for both networks to get the fastest DST (see Table II).

TABLE II
Optimal Link Weights

	Complete Graph $\mathcal{K}_5$	Star Graph $\mathcal{S}_5$
Optimal Weight	$w(e) = 1/5$	$w(e) = 1/3$

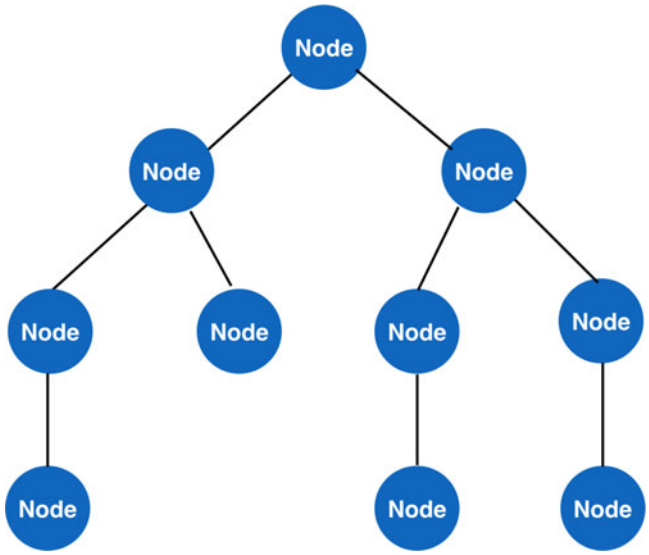

Fig. 3. DST network with ten servers over a tree graph (graph #1).

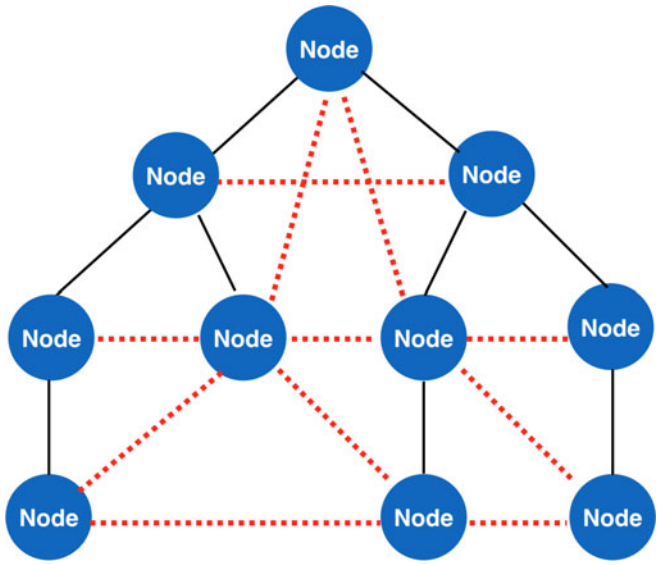

Fig. 4. DST network with ten servers over a tree graph with some additional red dotted links (graph #2).

For each network the weights are uniform since their underlying graphs are edge-transitive.

Example 2: Consider two DST networks with 10 servers over graphs depicted in Figs. 3 and 4. Let us assume that in both graphs all links have a weight of one. Based on the results presented in The-

TABLE III
OPTIMAL UPDATE CYCLES

	Graph #1	Graph #2
Optimal update cycle	$\Delta t = 0.4226$	$\Delta t = 0.2222$

TABLE IV
OVERALL NETWORK PERFORMANCE MEASURES

	Graph #1	Graph #2
$\Phi_{\mathbf{cr}}$	0.9969	0.9727
$\Phi_{\mathbf{ss}}$	334.7965	69.3075
Over-throttling %	6.2%	2.8%

orem 2, one can obtain the optimal update cycle for both networks to get the fastest DST (see Table III).

Moreover, let us consider 1000 clients that are randomly assigned to 10 servers such that each server has 100 clients. Fig. 5 shows the simulation results that are obtained for each of these two DST networks given a randomly generated usage curve over 1000 cycles. As expected, the overall performance of the DST over graph #2 is better (i.e., over-throttling is less severe) than the performance of the DST over graph #1 for small time step $\gamma = 0.02$ (see Table IV).

In Fig. 5, the blue curve shows the total number of requests versus time, i.e., $r_{\text{total}}(k)$, the black dashed line presents the total limit for the entire network, i.e., l_{total}, and the red and green curves show the total accepted requests for graph #1 and graph #2, respectively, i.e., $a_{\text{total}}(k)$. We should note that the ideal curve for total accepted requests is given by (8). Therefore, the percentage of over-throttling can be defined as follows

$$\text{Over-throttling } \% := \frac{\sum_{k=0}^{N} \left(a_{\text{ideal}}(k) - a_{\text{total}}(k)\right)}{\sum_{k=0}^{N} a_{\text{ideal}}(k)} \times 100$$

where N is the number of cycles (in this example 1000).

VII. THROTTLING ALGORITHMS AT THE NODE LEVEL

In this part, we focus on the structure of each node. Each node consists of a server with its clients (see, for example, Fig. 6). Besides update law (3), it has its own throttling algorithm to handle its clients. Let us assume that server i has c_i clients, and $r_i^{(j)}(k)$ is the number of requests received by server i from its jth client at time k. Therefore, the total number of client requests received by server i at time k is

$$r_i(k) = \sum_{j=1}^{c_i} r_i^{(j)}(k).$$

Let's define $x_i^{(j)}(k)$ as a limit on the number of requests of jth client of server i that is allowed to service at time k. The summation of the limits should be less than or equal to the node limit (i.e., $\sum_{j=1}^{c_i} x_i^{(j)}(k) \leq x_i(k)$). In each update cycle, first each server (let's say server i) collects all metrics from its clients (i.e., number of requests $r_i^{(j)}(k)$ for $j \in \{1, 2, \ldots, c_i\}$) as well as collecting its neighbors' states and performance measures (i.e., a local aggregated view of usage metrics), then aggregates all metrics and updates its state, and finally pushes new limits to its clients (i.e., $x_i^{(j)}(k)$ for $j \in \{1, 2, \ldots, c_i\}$). It also communicates its local aggregated view of usage metrics to its neighboring nodes' servers.

At the node level, viable throttling algorithms can be considered to throttle same amount, ratio, or the logarithm of ratio from all tasks until the total limit is reached (please see nodal performance measures in Table I). In what follows, we present two simple throttling algorithms with their high-level examples that can be used by each node.

Algorithm 1: A simple balancing algorithm for keeping throttled ratios uniform at server i at time k.

Input : $r_i^{(j)}(k)$ for $j \in \{1, 2, \cdots, c_i\}$ and $x_i(k)$.

Output: $x_i^{(j)}(k)$ for $j \in \{1, 2, \cdots, c_i\}$.

$r_i(k) := \sum_{j=1}^{c_i} r_i^{(j)}(k)$
if $r_i(k) \leq x_i(k)$ **then**
  for $j = 1$ *to* c_i **do**
    $x_i^{(j)}(k) := r_i^{(j)}(k)$
  end
else
  for $j = 1$ *to* c_i **do**
    $x_i^{(j)}(k) := \frac{x_i(k)}{r_i(k)} r_i^{(j)}(k)$
  end
end

Algorithm 2: A simple load balancing algorithm with throttling large number of requests at server i at time k.

Input : $r_i^{(j)}(k)$ for $j \in \{1, 2, \cdots, c_i\}$ and $x_i(k)$.

Output: $x_i^{(j)}(k)$ for $j \in \{1, 2, \cdots, c_i\}$

$r_i(k) := \sum_{j=1}^{c_i} r_i^{(j)}(k)$
if $r_i(k) \leq x_i(k)$ **then**
  for $j = 1$ *to* c_i **do**
    $x_i^{(j)}(k) := r_i^{(j)}(k)$
  end
else
  $r_i^{(j)\uparrow}(k) \leftarrow$ **sort** $r_i^{(j)}(k)$ for $j \in \{1, 2, \cdots, c_i\}$
  $s := 0$
  $l := \frac{x_i(k)}{c_i}$
  for $j = 1$ *to* c_i **do**
    if $l > r_i^{(j)\uparrow}(k)$ **then**
      $s = s - r_i^{(j)\uparrow}(k) + l$
      $l = \frac{s}{c_i - j} + l$
    end
  end
  for $j = 1$ *to* c_i **do**
    $x_i^{(j)}(k) := l$
  end
end

The first algorithm keeps the throttled ratios uniform over all tasks and is defined in Algorithm 1.

The second algorithm demonstrates a simple load balancing algorithm that distributes incoming requests across all tasks as uniformly as possible by throttling large number of requests. We present the steps of this algorithm in Algorithm 2.

We now present two high-level examples according to these algorithms. Fig. 7(a) and (b) depict numbers of requests and throttled requests for server i based on Algorithms 1 and 2, respectively. Each bar shows the number of requests per client. Blue bars show clients' requests. The red area shows the throttled request traffic. The clients are sorted by number of requests in ascending order. The blue dashed line in Fig. 7(b) shows the allowed limit on each task. We should note that the total number of request at this server (server i) is the area of all bars, i.e., $r_i(k)$, and the total allowed request is the area of all blue bars, i.e., $a_i(k)$.

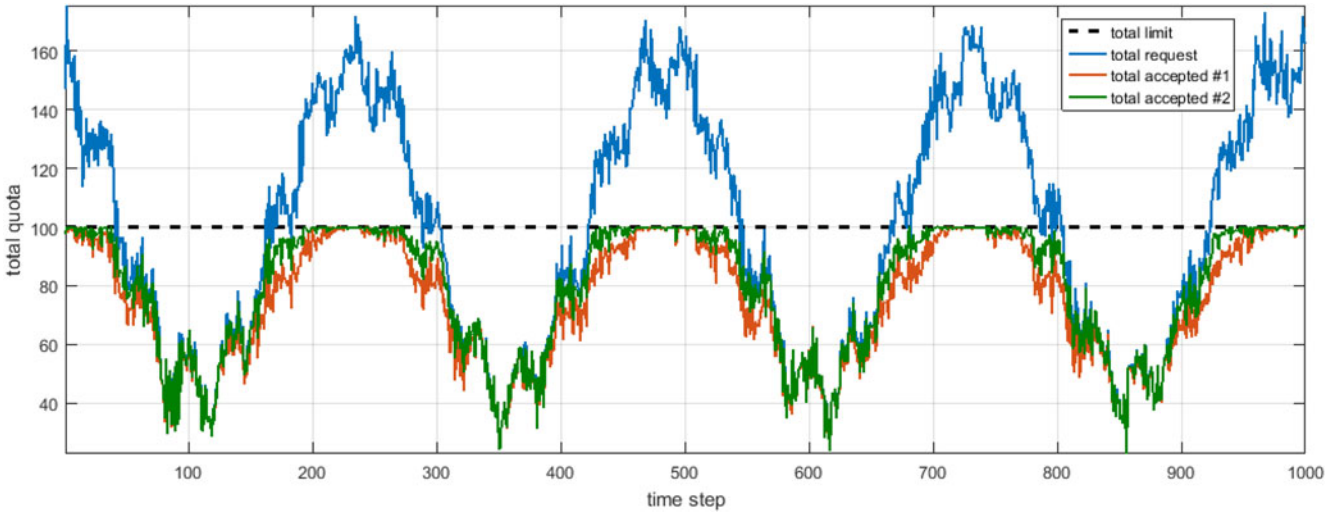

Fig. 5. Simulation results for two DST networks with $\gamma = 0.02$ over graphs $\#1$ and $\#2$ with ten nodes.

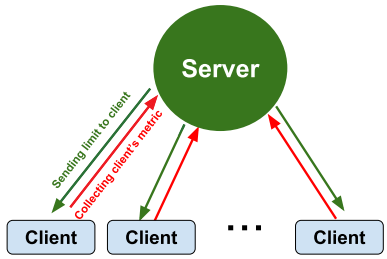

Fig. 6. Server with its clients.

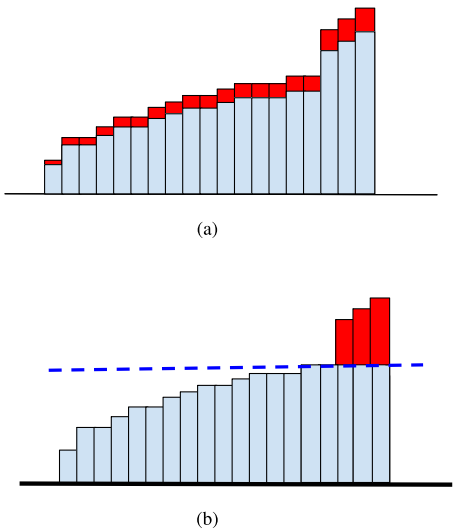

Fig. 7. Requested quota (i.e., number of requests) and throttled requests for server i: (a) Algorithm 1 that keeps the throttled ratios uniform over all tasks, and (b) Algorithm 2 that throttles large number of requests. The blue dashed line shows the resulting allowed level l in Algorithm 2.

VIII. Conclusion

In this paper, we investigated performance deterioration (e.g., over-throttling) of DST with respect to external uncertainties and server time cycles. We developed a graph-theoretic framework to relate the underlying structure of the system to its overall performance measure. We then compared the performance/robustness of the proposed DST with different underlying graphs. A promising research direction is to investigate the overall performance measure of DST networks with respect to the other nodal performance measures.

Appendix
Other Nodal Performance Measures

In this part, we present the dynamics of the DST for other nodal performance measure defined in Table I (*Case I* is studied in Section IV).

Case II: Assume that the performance measure at server i is given by

$$p_i(k) \;=\; \frac{r_i(k) - x_i(k)}{r_i(k)} \tag{39}$$

and $r_i(k) > 0$. Then, we can rewrite (3) in the following form

$$x(k+1) \;=\; \gamma\, L\ \mathrm{diag}\left[r_1(k)^{-1}, \ldots, r_n(k)^{-1}\right](r(k) - x(k)) + x(k),\ \ k \in \mathbb{Z}_+. \tag{40}$$

Let assume that $r_i(k) = \mathfrak{r}$ for all $k \in \mathbb{Z}_+$ and $i \in \{1, 2, \ldots, n\}$. So, we have

$$x_i(k) \;=\; -\mathfrak{r}\,(p_i(k) - 1)\,. \tag{41}$$

Then, it follows that

$$p(k+1) \;=\; \left(I - \frac{\gamma}{\mathfrak{r}} L\right) p(k). \tag{42}$$

In this case, in addition to the coupling graph and the update cycle, the values of $\mathfrak{r}$ plays a role in the convergence rate of the network (same for other overall performance measures).

Case III: Next, we assume that the performance measure at server i is given by

$$p_i(k) \;=\; \log r_i(k) - \log x_i(k). \tag{43}$$

Assume that $r_i(k) = \mathfrak{r}$, for all $k \in \mathbb{Z}_+$ and $i \in \{1, 2, \ldots, n\}$. Therefore, we get

$$x_i(k) \;=\; \mathfrak{r}\, e^{-p_i(k)}. \tag{44}$$

Then, it follows that

$$\exp\left(-p(k+1)\right) \;=\; \exp\left(-p(k)\right) + \frac{\gamma}{\mathfrak{r}}\, L\, p(k),\ \ k \in \mathbb{Z}_+ \tag{45}$$

where $\exp p(k) := \left[e^{p_1(k)}, \ldots, e^{p_n(k)}\right]^T$. Let us define

$$\bar{p}(k) := \exp\left(-p(k)\right) \tag{46}$$

using (45) and (46), it follows that

$$\bar{p}(k+1) \;=\; \bar{p}(k) - \frac{\gamma}{\mathfrak{r}}\, L\, \ln \bar{p}(k),\;\; k \in \mathbb{Z}_+ \tag{47}$$

where

$$\ln \bar{p}(k) \;:=\; \left[\ln \bar{p}_1(k), \ldots, \ln \bar{p}_n(k)\right]^T.$$

Case IV: Finally, let us assume that

$$p_i(k) \;=\; x_i(k)$$

for $i = 1, \ldots, n$. Then, dynamics (3) can be rewritten in the following form

$$p(k+1) \;=\; (I + \gamma L)\, p(k). \tag{48}$$

In this case, based on Lemma 2 the system is unstable, which means the state trajectories are unbounded. Therefore, we consider additional constraints to make them bounded as follows: the state of node i at time $k+1$ is not updated (i.e., $x_i(k+1) \;=\; x_i(k)$) and its information at time k is not used for updating the states of neighboring nodes at time $k+1$ when

1) $x_i(k) = r_i(k)$ and $x_i(k+1) - x_i(k) > 0$
2) $x_i(k) = 0$ and $x_i(k+1) - x_i(k) < 0$.

We should note that also in this case the following equality holds

$$\sum_{i=1}^{n} x_i(k) \;=\; \sum_{i=1}^{n} x_i(0).$$

In a steady state, each state x_i reaches its boundaries (i.e., 0 and r_i) or a value between them.

Acknowledgment

The authors would like to thank Dr. S. Vassilvitskii and Dr. H. Azari for their valuable comments and suggestions to improve the paper.

References

[1] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 37–48, 2005.

[2] K. Tan, F. Jiang, Q. Zhang, and X. Shen, "Congestion control in multihop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, pp. 863–873, Mar. 2007.

[3] Y. Zhang, S. R. Kang, and D. Loguinov, "Delay-independent stability and performance of distributed congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 838–851, Aug. 2007.

[4] R. Johari and D. K. H. Tan, "End-to-end congestion control for the internet: Delays and stability," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 818–832, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1109/90.974534

[5] R. Gibbens and F. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, no. 12, pp. 1969–1985, 1999.

[6] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, 1998.

[7] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud control with distributed rate limiting," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2007, pp. 337–348.

[8] R. Stanojevic and R. Shorten, "Generalized distributed rate limiting," in *Proc. 17th Int. Workshop Qual. Serv.*, 2009, pp. 1–9.

[9] B. Bamieh, M. Jovanović, P. Mitra, and S. Patterson, "Coherence in large-scale networks: Dimension-dependent limitations of local feedback," *IEEE Trans. Autom. Control*, vol. 57, no. 9, pp. 2235–2249, Sep. 2012.

[10] M. Siami, S. Bolouki, B. Bamieh, and N. Motee, "Centrality measures in linear consensus networks with structured uncertainties," *IEEE Trans. Control Netw. Syst.*, to be published, doi: 10.1109/TCNS.2017.2655731.

[11] M. Siami and N. Motee, "Fundamental limits and tradeoffs on disturbance propagation in linear dynamical networks," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 4055–4062, Dec. 2016.

[12] D. Zelazo and M. Mesbahi, "Edge agreement: Graph-theoretic performance bounds and passivity analysis," *IEEE Trans. Autom. Control*, vol. 56, no. 3, pp. 544–555, Mar. 2011.

[13] M. Siami and N. Motee, "Systemic measures for performance and robustness of large–scale interconnected dynamical networks," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 5119–5124.

[14] E. Lovisari, F. Garin, and S. Zampieri, "Resistance-based performance analysis of the consensus algorithm over geometric graphs," *SIAM J. Control Optim.*, vol. 51, no. 5, pp. 3918–3945, 2013.

[15] F. Lin, M. Fardad, and M. R. Jovanović, "Design of optimal sparse feedback gains via the alternating direction method of multipliers," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2426–2431, Sep. 2013.

[16] M. Siami and N. Motee, "Fundamental limits on robustness measures in networks of interconnected systems," in *Proc. 52nd IEEE Conf. Decis. Control*, Dec. 2013, pp. 67–72.

[17] A. Jadbabaie and A. Olshevsky, "On performance of consensus protocols subject to noise: Role of hitting times and network structure," in *Proc. IEEE 55th Conf. Decision Control (CDC)*, Las Vegas, NV, USA, 2016, pp. 179–184.

[18] Y. Ghaedsharaf, M. Siami, C. Somarakis, and N. Motee, "Interplay between performance and communication delay in noisy linear consensus networks," in *Proc. 15th Eur. Control Conf.*, Jun. 2016, pp. 1703–1708.

[19] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, pp. 65–78, 2003.

[20] A. Ghosh, S. Boyd, and A. Saberi, "Minimizing effective resistance of a graph," *SIAM Rev.*, vol. 50, no. 1, pp. 37–66, 2008.

[21] S. Boyd, "Convex optimization of graph Laplacian eigenvalues," in *Proc. Int. Congr. Mathematicians*, 2006, pp. 1311–1319.