A Distributed Strategy for Near-Optimal Network Topology Design

Dong Xue[1], Azwirman Gusrialdi[2] and Sandra Hirche[1]

Abstract— **In this paper, we propose a systematic strategy for distributedly maximizing the connectivity of a network. A graph process is introduced to formulate the problem in which edges are added to an existing graph in a step-wise manner to maximize the algebraic connectivity of underlying network topology. This problem turns into an intractable combinatorial optimization and hence we explore a heuristic rule to approximately solve it. In dependence upon global network structure information, we first provide a local criterion of near-optimization by involving the elements of the Fiedler eigenvector of graph Laplacian. To conduct the graph process in a distributed fashion, a decentralized estimation algorithm is devised for locally attaining the knowledge of the Fiedler eigenvector. The proofs of convergence and convergence rate are provided. Synthetically, a novel heuristic strategy is proposed to deal with the network optimization problem in a distributed and generalized fashion. The effectiveness of proposed methods are demonstrated via some examples.**

I. Introduction

Networks with complex structure describe a broad range of systems in society and civil engineering such as the Internet, metabolic networks, smart power grids, supply chains, water distribution and traffic systems, and large arrays of micro-electro-mechanical systems (MEMS).

A fundamental challenge in the study of complex networks is to explore topological properties of information diffusion throughout the network. As shown in a number of works e.g. [1]–[3], the network connectivity is a key structural metric which shows how well a graph is connected. The algebraic connectivity is one of the most prominent parameters to evaluate the connectivity of a network in contrast to the conventional metrics, such as *vertex* connectivity and *edge* connectivity. Due to the closed relevance to the convergence speed of the consensus protocol, structural robustness and synchronization of communication networks, the optimal topology design involving maximization of the algebraic connectivity has received much more attention, see e.g. [4], [5] and references therein. There are two main approaches to optimize the algebraic connectivity. One is to tune weights of the edges in a weighted graph, which leads to the absolute algebraic connectivity of the graph defined by Fiedler [6], see for instance [7], [8]. Another approach is to add or delete edges in a network, where an optimal topological structure problem is solved, see for instance [4], [9]. In this paper, we focus on the latter case of designing an optimal strategy for edge allocation to maximize the network connectivity. In the aforementioned papers, the structure optimization problems are typically formulated as combinatorial problems and - applying relaxation approaches - solved by using Semi-Definite Programming (SDP). In [3], a Mixed-Integer Quadratic Constraint Programming (MIQCP) is introduced to further facilitate the solution of this problem. While the SDP methodology does provide an upper bound on the optimal value for moderate problem size, the computational complexity level of this approach increases rapidly as the size of the networks gets larger (in some cases, not even tractable). Furthermore, SDP formulation may yield solutions that does not directly provide insight into and physical and graphical characteristics for topology design. Alternatively, based on matrix perturbation analysis of the graph Laplacian, some greedy heuristics are proposed to approximately solve this problem. In [4], a greedy perturbation heuristic is analyzed to provide a local optimum in a fast iterative manner. Analogous result applied to airport transportation network is extended to solve the weighted graph setting in [9]. However, this approach requires knowledge of the entire network and is inherently centralized. Specifically, the objective function (algebraic connectivity) and constraint conditions (e.g. candidate edge set) of the reformulated optimization are both computed based on the stricture knowledge of entire networks.

In practice, the global network structure information may be difficult to accumulate in a centralized fashion because of either technological matters or privacy concerns. Indeed, a node usually only has a limited perspective, and hence it cannot straightforwardly acquire what the rest of the network looks like. Thus, a distributed design scheme is more appealing from both aspects of numerical tractability and practical application. The distributed power iteration is employed to achieve this goal. Several approaches are available for estimating the eigenvectors and eigenvalues of an underlying network topology. For instance, a decentralized orthogonal iteration approach is proposed in [10] to estimate some leading eigenvectors but with a restriction of centralized initialization. Furthermore, a distributed power iteration approach is introduced in [2] to estimate the components of the Fiedler eigenvector and then the Fiedler value in a continuous-time fashion. However, the continuous time nature of the algorithm makes its implementation in a real-world scenario particularly challenging. In the work of [11], the authors propose a distributed approach for the estimation of the eigenvalues of the weight matrix and analogous work can be found in [12], without estimating the associated eigenvector. To the best of our knowledge, little literature

[1]D. Xue and S. Hirche are with the Institute for Information-Oriented Control, Technische Universität München, D-80290 München, Germany; {dong.xue, hirche}@tum.de

[2]A. Gusrialdi is with Department of Electrical Engineering and Computer Science, University of Central Florida, USA; Azwirman.Gusrialdi@ucf.edu

is available regarding the decentralized estimation of the eigenvector associated with the algebraic connectivity in the optimal topology design problem.

The main contribution of this paper is to provide a distributed strategy for approximately solving the optimal topology design. A graph process is introduced for constructively generalizing the edge operation in topology design problem. By developing eigenvalue sensitivity in graph process, we reformulate the optimization problem such that involving the components of the Fiedler eigenvector. In order to facilitate the distributed implementation, a decentralized pattern of power iteration algorithm is proposed, which is aimed at locally estimating the Fiedler eigenvector for all nodes in the network. In this distributed estimation scheme, each node only accounts for computing one component of the Fiedler eigenvector, which dramatically decreases communication and computational load for nodes. By providing the proofs of convergence and convergence rate, we achieve full distributed scheme that nodes are individually aware of explicit estimates of the Fiedler eigenvector by only communicating and sharing data with their neighbors in the network. Finally, we synthesize the obtained results and propose a strategy of edge operation to solve the optimal topology problem in a distributed manner.

The organization of this paper is as follows: preliminary results of graph theory and the problem formulation are presented in Section II. The main results are proposed in Section III including a heuristic rule of topology design, a distributed estimation scheme of Fiedler vector, and a synthetic strategy of optimal edges allocation. Finally, the proposed strategies are evaluated via numerical examples in Section IV. In Section V, conclusion of the underlying problem is reached.

Notation: Let $\mathbb{R}$ be the set of real numbers; $\mathbf{1}$ $(\mathbf{0})$ denotes the column vector of all ones (zeros). $\text{diag}(a,b)$ represents the diagonal matrix $\left[\begin{smallmatrix} a & 0 \\ 0 & b \end{smallmatrix}\right]$, where $a,b\in\mathbb{R}$. A identity matrix is given by $\mathcal{I}$, i.e., $\mathcal{I}=\text{diag}(1,\dots,1)$. For a set $\mathcal{S}$, $|\mathcal{S}|$ is the number of the elements in this set. For any $n\times n$ symmetric matrix $\mathcal{B}$, a partial order is operated in its spectrum, more specifically, $\lambda_1(\mathcal{B})\le\lambda_2(\mathcal{B})\le\ldots\le\lambda_n(\mathcal{B})$.

II. Problem Formulation

Let $\mathcal{G}=(\mathcal{V},\mathcal{E},\mathcal{A})$ be a weighted undirected graph with a set of vertices $\mathcal{V}=\{1,2,\dots,n\}$, a set of edges $\mathcal{E}\subseteq\mathcal{V}\times\mathcal{V}$, and a weighted adjacency matrix $\mathcal{A}=[a_{ij}]_{n\times n}$. Complementally,a weight matrix $\mathbb{A}=[A_{ij}]_{n\times n}$ is given such that $a_{ij}=A_{ij}$ if $(i,j)\in\mathcal{E}$ and $a_{ij}=0$ otherwise, where $\underline{A}\le A_{ij}\le\overline{A}$, $A_{ii}=0$ and $A_{ij}=A_{ji}$. This weight may be the available bandwidth between two machines, the number of links between two web pages, or an estimate of the strength of a social tie between two individuals. In this paper, $a_{ij}\sim(i,j)$ is used to illustrate the combination of the edge with weight a_{ij} and the correspondingly connecting node pair (i,j) if $(i,j)\in\mathcal{E}$. A neighborhood set $\mathcal{N}_i\subseteq\mathcal{V}$ of vertex i is given by $\{j\in\mathcal{V}|a_{ij}\neq 0\}$. $|\mathcal{E}|$ is the cardinality of the edge set, i.e. the number of edges in graph $\mathcal{G}$. In order to guarantee that the problem is well-defined, the complement of a weighted graph is introduced first. The complement of a graph $\mathcal{G}=(\mathcal{V},\mathcal{E},\mathcal{A})$ is denoted by $\mathcal{G}^c=(\mathcal{V},\mathcal{E}^c,\mathcal{A}^c)$ with the following definition.

Definition 2.1: The complement $\mathcal{G}^c$ of a graph $\mathcal{G}$ is a graph with the same vertex set and with the property that two vertices are adjacent in $\mathcal{G}^c$ if and only if they are not adjacent in $\mathcal{G}$. The weighted adjacent matrix of $\mathcal{G}^c$ is given by $\mathcal{A}^c=[\tilde{a}_{ij}]_{n\times n}=\mathbb{A}-\mathcal{A}$, i.e. $\tilde{a}_{ij}=A_{ij}-a_{ij}$.

The Laplacian matrix $\mathcal{L}\in\mathbb{R}^{n\times n}$ of $\mathcal{G}$ is positive semi-definite matrix $\mathcal{L}(\mathcal{G})=\text{diag}(\mathcal{A}\mathbf{1})-\mathcal{A}$. It follows that the Laplacian matrix has a simple zero eigenvalue and all the other eigenvalues are positive values if and only if the graph is connected, i.e., for connected graph, $0=\lambda_1(\mathcal{L})<\lambda_2(\mathcal{L})\le\ldots\le\lambda_n(\mathcal{L})$., with which the corresponding eigenvectors $\nu_1,\nu_2,\dots,\nu_n$, where $\nu_1=\frac{\mathbf{1}}{\sqrt{n}}$. The second smallest eigenvalue $\lambda_2(\mathcal{L})$ of $\mathcal{L}$ is the *weighted algebraic connectivity* and the corresponding normalized right eigenvector ν_2 is called the *Fiedler vector*. Since $\lambda_2(\mathcal{G})$ is computed based on the weighted Laplacian matrix of $\mathcal{G}$, we equivalently use $\lambda_2(\mathcal{G})$ and $\lambda_2(\mathcal{L})$ in the following analysis.

Given the graph $\mathcal{G}_0=(\mathcal{V},\mathcal{E}_0,\mathcal{A}_0)$ of an existing weighted network, we assume it is connected throughout the paper. The objective of this paper is to maximize Fiedler value by selecting a edge subset $\triangle\mathcal{E}\subseteq\mathcal{E}_0^c$ with fixed edge number l from $\mathcal{E}_0^c$. Thus, the following optimization problem is solved in this paper:

$$\begin{aligned} \max\quad & \lambda_2(\mathcal{G}(\mathcal{V},\mathcal{E}_0+\triangle\mathcal{E},\mathcal{A}_0+\triangle\mathcal{A})) \\ \text{s.t.}\quad & |\triangle\mathcal{E}|=l, \\ & \triangle\mathcal{E}\subseteq\mathcal{E}_0^c, \end{aligned} \tag{P1}$$

where $\triangle\mathcal{A}$ is the weight matrix associated with $\triangle\mathcal{E}$ and the weights are derived from prespecified weighted matrix $\mathbb{A}$.

For a graph $\mathcal{G}$ with m edges and n vertices, the Laplacian matrix $\mathcal{L}$ can be factorized as $\mathcal{L}=\nabla^\top\nabla=\sum_{e=1}^m a_e\nabla_e\nabla_e^\top$, where a_e is the weight of edge e and $\nabla=[\nabla_1,\dots,\nabla_m]$ is the *incidence matrix* whose column vector ∇_e (also called *edge vector*) concerning edge e (connecting vertex i and j) is given by

$$\nabla_{e,v}=\begin{cases}1 & \text{if } (i,j)\in\mathcal{E} \text{ and } v=i,\\ -1 & \text{if } (i,j)\in\mathcal{E} \text{ and } v=j,\\ 0 & \text{otherwise,}\end{cases}$$

where $\nabla_{e,v}$ is the v-th element of ∇_e and $v=\{1,\dots,n\}$. The notations of the edge weights a_e and a_{ij} are abused in the rest of paper if their correlation is inferred by context. In practice, this correlation is determined by the graph labeling that is the assignment of labels to edges and vertices.

As a result, we can rewrite the Problem (P1) as:

$$\begin{aligned} \max_{y}\quad & \lambda_2(\mathcal{L}_0+\triangle\mathcal{L}) \\ \text{s.t.}\quad & \triangle\mathcal{L}=\sum_{e=1}^{m_c} y_e\tilde{a}_e\nabla_e\nabla_e^\top, \\ & \mathbf{1}^\top y=l \\ & y\in\{0,1\}^{m_c}, \end{aligned} \tag{P1$'$}$$

where $m_c = |\mathcal{E}_0^c|$ and $y = [y_1, \ldots, y_e, \ldots, y_{m_c}]^\top$ is a Boolean vector, in which 1 means that edge e in $\mathcal{E}_0^c$ is in the set of $\triangle\mathcal{E}$ and 0 means that not. The Problem (P1$'$) is combinatorial, and hence, can be solved exactly by brute-force search. Furthermore, by replacing the Boolean constraint $y \in \{0,1\}^{m_c}$ by $y \in [0,1]^{m_c}$ and convex relaxation, it can be reformulated as a semi-definite program (SDP) and solved by employing a standard SDP solver for moderate size of adding edges. For more details and proofs, the readers are referred to [4], [13]. However, when the size of network gets larger, these algorithms exhibit slow convergence and have high computational requirements. Moreover, note that in order to compute a solution, the optimization problem (P1$'$) requires the entire network structural information, whereas in reality it is difficult to get the global network structure. The goal of this paper is to provide an explicit rule for adding edges to maximize algebraic connectivity. In particular, we aim at solving Problem (P1$'$) in the absence of global network structure information.

III. Main Results

In this section, a heuristic algorithm to approximately solve the optimal topology design problem is proposed. In order to execute the heuristic in a distributed fashion, a decentralized power iteration scheme is established to locally estimate the Fiedler vector.

A. Eigenvalue Sensitivity Analysis

Motivated by the work on eigenvalue sensitivity [14], we investigate explicit solutions for topology design problems. The eigenvalue sensitivity offers an insight on the effect of the eigenvalues of a perturbed matrix, in this case, when certain edges are added or removed from network.

For any undirected connected graph $\mathcal{G}$, the following *Courant Fischer* principle holds:

$$\lambda_2(\mathcal{L}) = \min_{\substack{\mathbf{1}^\top z=0 \\ z\neq \mathbf{0}}} \frac{z^\top \mathcal{L} z}{\|z\|^2}. \tag{1}$$

Furthermore, by substituting vector z with the normalized Fiedler vector $\nu_2(\mathcal{L})$, it follows from (1) that

$$\lambda_2(\mathcal{L}) = \nu_2^\top \mathcal{L} \nu_2. \tag{2}$$

Note that given any positively weighted undirected graph $\mathcal{G}$, $\lambda_2(\mathcal{G})$ is a nondecreasing function of each edge addition.

Let

$$\mathcal{L}(y) = \mathcal{L}_0 + \triangle\mathcal{L}. \tag{3}$$

The partial derivative of $\lambda_2(\mathcal{L}(y))$ with respect to y_e gives the first-order approximation of the increase of $\lambda_2(\mathcal{L}(y))$, if the edge e is added to the graph $\mathcal{G}$. According to (2) and using the definition in (3), it follows

$$\frac{\partial}{\partial y_e}\lambda_2(\mathcal{L}(y)) = \nu_2^\top \frac{\partial}{\partial y_e}\mathcal{L}(y)\nu_2 = \tilde{a}_{ij}(\nu_{2,i} - \nu_{2,j})^2. \tag{4}$$

This implies that adding a candidate edge $\tilde{a}_{ij} \sim (i,j)$ which maximizes $\tilde{a}_{ij}(\nu_{2,i} - \nu_{2,j})^2$ to a graph, results in the largest displacement of $\lambda_2(\mathcal{L}(y))$, where $\nu_{2,i}$ and $\nu_{2,j}$ are the i-th and j-th elements of the Fiedler vector ν_2 of the current Laplacian $\mathcal{L}(y)$. For simplicity of presentation in the following analysis, we denote $n(n-1)/2$ variables as

$$\pi_{ij} = \tilde{a}_{ij}(\nu_{2,i} - \nu_{2,j})^2, \quad j \neq i. \tag{5}$$

To consider the topology design problem in a generic case, a graph process is introduced as follows:

Definition 3.1: (**Graph Process**): within a time interval $[t_0, t_{m_l}]$, $0 \le l \le m_c$ edges are added into an initial graph $\mathcal{G}_0$. At each discrete-time instant t_k ($k = 1, \ldots, m_l$), l_k edges are picked up from the remaining candidate edges and added into the graph $\mathcal{G}(t_{k-1})$, where m_l is the amount of rounds and $\sum_{k=1}^{m_l} l_k = l$.

Assumption 3.1: The time sequence has the property that $(t_{k+1} - t_k) \ge T$ for some known constant $T > 0$.

Note that the graph is time-invariant within each interval (t_{k-1}, t_k), and therefore its corresponding Laplacian matrix is piece-wise constant. In contrast to the one-by-one edge addition in [4], the definition of graph process allows multi-links addition in each step and encompasses more application scenarios.

Algorithm 3.1: Consider a graph process as shown in Definition 3.1. Within the time interval $(t_{k-1}, t_k]$ ($k = 1, \ldots, m_l$), the optimization problem (P1$'$) can be approximated as to successively select l_k edges such that the following optimization problem is solved

$$\begin{aligned} &\underset{(i,j),\ldots,(h,r)}{\arg\max} && \underbrace{\pi_{ij} + \cdots + \pi_{hr}}_{l_k \text{ pairs}} \\ &\text{s.t.} && \{(i,j),\ldots,(h,r)\} \subseteq \mathcal{E}_0^c(t_{k-1}), \end{aligned} \tag{P2}$$

where $\pi_{2,i}$ is given in (5) involving the weighted difference between the components of Fiedler eigenvector at instant t_{k-1}. The edges addition is implemented at time t_k.

Algorithm 3.1 provides a heuristic of allocating edges by evaluating the deviation of elements in the Fiedler vector. The criterion in Problem (P2) requires to compute the Fiedler eigenvector in each round based on the global structure of the network, specifically, the Laplacian matrix. However, it is difficult to extend the results in the distributed control systems. In most cases, nodes only can communicate and share local information with their connected neighbors in the network. Based on the above consideration, we propose a distributed strategy to locally estimate the Fiedler vector and consequently achieve the self-organized topology design for each subsystem.

B. Distributed Estimation of Fiedler Eigenvector

In this subsection, we propose a distributed algorithm for estimating the right eigenvector (Fiedler eigenvector) of the Laplacian.

Before proceeding, a distributed algorithm for the computation of the powers of a matrix is revisited. As it is well known from [15], *power iteration* (sometimes called *Von Mises iteration*) is an effective and widely applied approach in calculating the dominant (in terms of magnitude) eigenvalue of any matrix. Power iteration first estimates the eigenvector associated with the dominant eigenvalue, then the corresponding eigenvalue can be calculated as a result. For instance, suppose $\lambda_n(\mathcal{C})$ and κ_n are the dominant eigenvalue (with strictly greater magnitude than other eigenvalues) and its associated eigenvector of matrix $\mathcal{C}$. It follows that κ_n can be estimated by power iteration at the $(s+1)$th step

$$\hat{\kappa}_n(s+1) = \frac{\mathcal{C}\hat{\kappa}_n(s)}{\|\mathcal{C}\hat{\kappa}_n(s)\|}, \quad s = 0, 1, \ldots \tag{6}$$

where $\hat{\kappa}_n(s)$ is estimate of ν_n at the s-th iteration, and $\hat{\kappa}_n(0)$ is the initial vector with the property $\hat{\kappa}_n(0)^\top \kappa_n \neq 0$, namely, $\hat{\kappa}_n$ has a nonzero component in the direction of the dominant eigenvalue. If λ_n and κ_n are known, one can estimate $\lambda_{n-1}(\mathcal{C})$ by running the power iteration on the deflated matrix $\mathcal{C}_{n-1} = \mathcal{C} - \lambda_n(\mathcal{C})\kappa_n\kappa_n^\top$.

Remark 3.1: Note that by deflation (or sometimes affine transformation [11]), the estimation of any eigenvalue $\lambda_i(\mathcal{C})$ can be accomplished provided that eigenvalues from $\lambda_n(\mathcal{C})$ to $\lambda_{i+1}(\mathcal{C})$ and the corresponding left and right eigenvectors are known explicitly. In other words, estimation of left (right) eigenvector(s) is always completed before estimating corresponding eigenvalue.

Obviously, traditional power iteration (6) is a recursive procedure with a strict requirement on initial conditions, and normalization is inevitable at each step in order to guarantee the estimate converges to a finite vector. What prevents its distributed implementation is the normalization and therefore a distributed power iteration scheme associated with a consensus observer is introduced in this paper. The estimator is essentially a distributed and discrete power iteration scheme, which enables each node to compute a component of the estimate of the Fiedler eigenvector, while the observer is designed to propagate current estimates over the entire network. Since the consensus algorithm is a naturally distributed operation, which only needs local communication, the proposed scheme is fully distributed in the sense that each node only uses the previous values of its one-hop neighbors and itself to update the estimate variable. As a result, each agent has the estimate of the eigenvector associated with the second smallest eigenvalue of the Laplacian matrix.

To use the power iteration for estimating the Fiedler vector, we first introduce the *Perron matrix* $C = [c_{ij}]_{n\times n}$ [1]

$$C = \mathcal{I} - \beta\mathcal{L}, \tag{7}$$

where $0 < \beta < \frac{1}{d_{\max}}$ with $d_{\max} := \max_i l_{ii}$.

The eigenvalues of matrix C and the eigenvalues of the Laplican matrix $\mathcal{L}$ are related as $\lambda_i(C) = 1 - \beta\lambda_{n+1-i}(\mathcal{L})$. It follows that matrices C and $\mathcal{L}$ also share the same eigenspace which is spanned by the eigenvectors $\{\nu_2, \ldots, \nu_n\}$ of $\mathcal{L}$. Note that by the introduction of Perron matrix, the eigenvalues of Laplacian matrix are conversely reordered in the matrix C. In other words, the useful first and second smallest eigenvalues of Laplacian matrix $\mathcal{L}$, as well as their associated eigenvectors, become the first and second maximum eigenvalues in the matrix C, on which the power iteration

TABLE I: Eigenstructures of matrices

	$\mathcal{L}$		C		C_1	
i	λ_i	ν_i	λ_i	ν_i	λ_i	ν_i
1	0	$\frac{\mathbf{1}}{\sqrt{n}}$	$1-\beta\lambda_n^{\mathcal{L}}$	ν_n	0	$\frac{\mathbf{1}}{\sqrt{n}}$
2	$\lambda_2^{\mathcal{L}}$	ν_2	$1-\beta\lambda_{n-1}^{\mathcal{L}}$	ν_{n-1}	$1-\beta\lambda_n^{\mathcal{L}}$	ν_n
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$\lambda_n^{\mathcal{L}}$	$\nu_n^{\mathcal{L}}$	1	$\frac{\mathbf{1}}{\sqrt{n}}$	$1-\beta\lambda_2^{\mathcal{L}}$	ν_2

$*$ $\lambda_i^{\mathcal{L}}$ represents $\lambda_i(\mathcal{L})$ for simplicity of exposition.

algorithm of matrix can be used. According to *Perron-Frobenius* theorem, the following convergent property holds for matrix C

$$\lim_{s\to\infty}(C)^s = \nu_1\nu_1^\top = \frac{1}{n}\mathbf{1}\mathbf{1}^\top. \tag{8}$$

Since the goal is to estimate the eigenvector associated to the second maximum eigenvalue of C, the Perron matrix is deflated as follows:

$$C_1 = C - \lambda_n(C)\nu_1\nu_1^\top, \tag{9}$$

where the eigenvalue $\lambda_{n-2}(C)$ involved with algebraic connectivity becomes the leading eigenvalue in matrix C_1.

Lemma 3.1: The spectrum relationship between the deflated Perron matrix C_1 and the Laplacian $\mathcal{L}$ is

$$\lambda_1(C_1) = \lambda_1(\mathcal{L}), \quad \lambda_i(C_1) = 1 - \beta\lambda_{n+2-i}(\mathcal{L}) = \lambda_{i-1}(C),$$

where $i \in \{2, \ldots, n\}$. In addition, $\frac{\mathbf{1}}{\sqrt{n}}$ is the eigenvector associated to the eigenvalue $\lambda_1(C_1)$.

Proof: The proof can be obtained by straightforwardly implementing spectrum analysis on these two matrices. For the sake of completeness, the eigenstructures of matrices $\mathcal{L}$, C and its deflated matrix C_1 are shown in Table I. ■

Now, we need to solve a decentralized estimation problem of the eigenvector associated with the dominant eigenvalue of deflated Perron matrix C_1, i.e. $\lambda_n(C_1)$. To achieve it, let a vector variable $\hat{\nu}_2^{(i)}(t) = \left[\hat{\nu}_{2,1}^{(i)}(t), \ldots, \hat{\nu}_{2,n}^{(i)}(t)\right]^\top$ be the estimate of eigenvector ν_2 at vertex i. In addition, we provide an auxiliary variable $\omega_2(t) = [\omega_{2,1}(t), \ldots, \omega_{2,n}(t)]^\top \in \mathbb{R}^n$ which is the explicit estimated vector in the power iteration. The initialization $\omega_2(0)$ is given by selecting a random vector with nonzero component in the direction of the dominant eigenvalue of C_1 and it can be decomposed along the eigenspace

$$\omega_2(0) = c_1\nu_1 + \cdots + c_n\nu_n, \ c_1, \ldots, c_n \in \mathbb{R}, \ c_2 \neq 0. \tag{10}$$

Thus, each vertex $i \in \mathcal{V}$ maintains the estimation $\hat{\nu}_2^{(i)}(t)$ together with $\omega_{2,i}(t)$.

Algorithm 3.2: (**Distributed Power Iteration**) Consider the power iteration (6) with the deflated Perron matrix C_1 and the estimate vector $\omega_2(t)$ initialized by (10). The dominant eigenvector of C_1 can be iteratively estimated in a distributed fashion

$$\omega_{2,i}(s+1) = \frac{\sum_{j=1}^n c_{ij}(t_k)\omega_{2,j}(s) - \mathbf{1}^\top\phi_i(s)}{n\|\phi_i(s)\|} \tag{11}$$

and $\omega_{2,i}(s) \in \mathbb{R}$ is executed as follows

$$\omega_{2,i}(s) = \omega_{2,i}(sT_e), \quad s = 0, 1, \ldots, M_s$$

where c_{ij} are (i, j)-entries of Perron matrix C, T_e and M_s are respectively the iterative interval and iterative step of estimator (11), and $\phi_i \in \mathbb{R}^n$ is the state of the observer executed at node i, depending on the following update rule

$$\begin{aligned} \phi_i(q+1) &= \sum_{j=1}^{n} c_{ij}\phi_j(q), \quad \phi_i(q) = \phi_i(sT_e + qT_o), \\ \phi_i(0) &= \omega_{2,i}(s)\delta_i, \qquad\qquad q = 0, 1, \ldots, M_q, \end{aligned} \tag{12}$$

where T_o is the sample interval of observer (12) with $T_o \ll T_e$, $M_q = \lfloor T_e/T_o \rfloor$ and δ_i is i-th column vector of $\mathcal{I}$.

Proposition 3.1: Consider a connected graph $\mathcal{G}(t_k)$ within time interval $t \in [t_k, t_{k+1})$. Given the appropriate setting of parameters parameters T_e, T_o, M_q and $M_s = \lfloor (t_{k+1} - t_k - T_oM_q)/T_e \rfloor$, all the estimates $\hat{\nu}_2^{(i)} \in \mathbb{R}^n$ asymptotically converge to the Fiedler vector $\nu_2(t_k)$ by executing Algorithm 3.2 at each node $i \in \mathcal{V}$.

Proof: Let $\phi(q) = [\phi_1^\top(q), \ldots, \phi_n^\top(q)]^\top$, then the observer scheme given in (12) can be written by the stack vector form as follows

$$\phi(q+1) = (C(t_k) \otimes \mathcal{I})\, \phi(q).$$

Taking (8) into consideration, it holds that

$$\lim_{q\to\infty} \phi(q) = \lim_{q\to\infty} (C^q(t_k) \otimes \mathcal{I})\, \phi(0) = \left(\frac{1}{n}\mathbf{1}\mathbf{1}^\top \otimes \mathcal{I}\right) \phi(0).$$

According to the initial condition for $\phi_i(0) = \omega_{2,i}(s)\delta_i$, we can obtain

$$\lim_{q\to\infty} \phi_i(q) = \frac{1}{n}\sum_i \phi_i(0) = \frac{1}{n}\omega_2(s),$$

which implies that $\omega_2(s) = n\phi_i(s)$.

As a result, the closed form of estimator (11) can be given by

$$\begin{aligned} \omega_2(s+1) &= \frac{1}{\|\omega_2(s)\|} \underbrace{\left(C(t_k) - \nu_1\nu_1^\top\right)}_{=C_1} \omega_2(s) \\ &= \frac{C_1^s \omega_2(0)}{\|C_1^{s-1}\omega_2(0)\|}, \end{aligned} \tag{13}$$

where $\omega_2(0)$ is the initial condition defined in (10). The deflated Perron matrix C_1 can be decomposed as

$$C_1 = \underbrace{\left(\begin{bmatrix} \frac{\mathbf{1}^\top}{\sqrt{n}} \\ \vdots \\ \nu_2^\top \end{bmatrix}^\top\right)^{-1}}_{:=V} \underbrace{\begin{bmatrix} 0 & & \\ & \ddots & \\ & & \lambda_n(C_1) \end{bmatrix}}_{:=J} \underbrace{\begin{bmatrix} \frac{\mathbf{1}^\top}{\sqrt{n}} \\ \vdots \\ \nu_2^\top \end{bmatrix}}_{(V^{-1})^\top},$$

where V is invertible matrix and J is the Jordan normal form of C_1 as Regarding the fact that $C_1^s = VJ^sV^{-1}$, it yields

$$\omega_2(s+1) = \frac{(VJ^sV^{-1})(\frac{c_1}{\sqrt{n}}\mathbf{1} + c_2\nu_2 + \cdots + c_n\nu_n)}{\|(VJ^{s-1}V^{-1})(\frac{c_1}{\sqrt{n}}\mathbf{1} + c_2\nu_2 + \cdots + c_n\nu_n)\|}.$$

Due to $\lambda_n(C_1) > 0$ is the dominant eigenvalue of C_1, we have

$$\lim_{s\to\infty} \frac{J^s}{\lambda_n^s(C_1)} = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$

which yields

$$\begin{aligned} \lim_{s\to\infty} \omega_2(s+1) &= \frac{c_2\lambda_n^s(C_1)\nu_2}{\|c_2\lambda_n^{s-1}(C_1)\nu_2\|} \\ &= \operatorname{sgn}(c_2)\lambda_n(C_1)\frac{\nu_2}{\|\nu_2\|}. \end{aligned} \tag{14}$$

As a sequel, (14) leads to

$$\lim_{s\to\infty} \|\omega_2(s)\| = \lambda_n(C_1),$$

and then substituting it to (14), we have

$$\lim_{s\to\infty} \frac{\omega_2(s)}{\|\omega_2(s)\|} = \operatorname{sgn}(c_2)\frac{\nu_2}{\|\nu_2\|}. \tag{15}$$

The convergence is geometric with ratio $\frac{\lambda_{n-1}(C_1)}{\lambda_n(C_1)}$.

Moreover, note that the consensus-based observer (12) is established to broadcast the last step local estimate $\omega_{2,i}$ to the overall network. Whenever the network is connected at each interval, each node gets explicit knowledge of last entire estimate ω_2. Based on the above consideration, the observer algorithm can be used to collect the entire estimates and update the $\hat{\nu}_2^{(i)}$ at each node. An extra iterative time M_qT_o is required to execute the observer in (12) after finishing the estimation. Then, the estimate $\hat{\nu}_2^{(i)}$ maintained at node i becomes

$$\hat{\nu}_2^{(i)}(\eta) = \frac{\phi_i(\eta)}{\left\|\phi_i\left(\eta\right)\right\|}, \tag{16}$$

where $\eta(t_k) = t_k + M_sT_e + M_qT_o$. In other words, all estimates $\hat{\nu}_2^{(i)}$ $(i \in \mathcal{V})$ converge to the true Fiedler eigenvector. ■

The scheme proposed in Proposition 3.1 consists of two components: estimator (11) and observer (12). Both components are executed using the same information topology while evolving at a different pace. Note that, the convergence of the observer needs to achieve within a time period of T_s such that estimate values $\omega_{2,i}$ can be broadcasted accurately to all the systems, provided that the graph is connected. In fact, the auxiliary observer (12) is established to propagate the local estimates over the entire network and is not necessary to reflect any practical reality.

Remark 3.2: Note that the total amount of nodes n is involved in the estimator (11). In case this number is not available to nodes, a labeling technique as shown in [12] can be employed. In particular, this labeling method takes place in a fully distributed manner, and hence it is consistent with our framework.

C. Distributed Algorithm for Edge Addition

In order to complete the local addition of edges by individual nodes, we slightly modify the decentralized estimation in Proposition 3.1 to facilitate the graph process. Before proceeding further, the following assumption is made in this paper.

Assumption 3.2: Each node i knows the i-th row of the weighted matrix $\mathbb{A}$ and the connected neighbors in $\mathcal{N}_i(t)$. Additionally, the nodes have the knowledge of the total number l of added links and the set $\{l_k\}_{k=1}^{m_l}$ during the graph process.
The assumption implies that the individual system knows the changes of the corresponding communication links connected with itself while such a change is generally not known to any other systems.

First, let $\mu(k) = \max\{n-1, l_k\}$. We introduce a descending sequence $\{\hat{\pi}_{i,p}(t)\}_{p=1}^{\mu}$ which is made up by elements

$$(A_{ij} - a_{ij})\left(\hat{\nu}_{2,i}^{(i)} - \hat{\nu}_{2,j}^{(i)}\right)^2, \ j \neq i \text{ and } j = 1, \ldots, n.$$

and it updates when new estimate $\hat{\nu}_2^{(i)}(t)$ is available. Additionally, we denote an edge set $\mathcal{E}^i(t) = \{(i,j) | j \notin \mathcal{N}_i(t)\}$ for each node i and induce a partial order in this set according to to the partial order of $\{\hat{\pi}_{i,p}(t)\}$, specifically, when $l_k > n-1$, $\{\hat{\pi}_{i,p}(t)\}_{p=n}^{l_k} = 0$ and accordingly setting null in edge set. Before entering into the details of the entire algorithm, we present an example in Figure 1 that illustrates how to construct this sequence and its associated edge set at nodes.

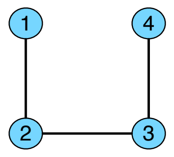


Fig. 1: Example

Example: For this graph, it has

$$\mathbb{A} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad \mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

and the number of added edge is $l_k = 2$. Then, refer to node 1, for instance, we have

$$\{\hat{\pi}_{1,p}\}_{p=1}^2 = \{\underbrace{1.7072}_{\hat{\pi}_{1,1}}, \underbrace{0.8536}_{\hat{\pi}_{1,2}}\}$$

which leads to edge set $\mathcal{E}^i = \{(1,4), (1,3)\}$.

In analogue with a distributed estimation of certain maximum value in [16], [17], nodes execute the following algorithm to individually explore the admissible edges to be added in the network.

Algorithm 3.3: **(Optimal Edges Search)**

1) *Initialization*: at initial step $\tau = 0$, update $\{\hat{\pi}_{i,p}(t_\tau)\}$ and $\mathcal{E}^i(t_\tau)$ based on the resultant estimates $\hat{\nu}_2^{(i)}$, where $t_\tau = t + \tau$.
2) *Transmission*: sends $\{\hat{\pi}_{i,p}(t_\tau)\}$ and $\mathcal{E}^i(t_\tau)$ to its neighbors.
3) *Update*: at step $\tau \geq 1$, node i first looks for new edge in received information from its neighbors which provides a set as follows:

$$\Phi_i(t_{\tau-1}) = \Psi_i(t_{\tau-1}) / \left\{\mathcal{E}^i(t_{\tau-1}) \bigcap \Psi_i(t_{\tau-1})\right\}$$

where $\Psi_i(t_{\tau-1}) = \bigcup_{j \in \mathcal{N}_i(t)} \mathcal{E}^j(t_{\tau-1})$

```
for any edge in $\Phi_i(t_{\tau-1})$ do
    if its corresponding $\hat{\pi}_{j,p'} < \forall \hat{\pi}_{i,p}$ then
        return
    else
        $\hat{\pi}_{i,\mu(k)} \leftarrow \hat{\pi}_{j,p'}$ and update $\mathcal{E}^i$
    end if
end for
```

Then,

$$\{\hat{\pi}_{i,p}(t_\tau)\} \leftarrow \{\hat{\pi}_{i,p}(t_{\tau-1})\}, \quad \mathcal{E}^i(t_\tau) \leftarrow \mathcal{E}^i(t_{\tau-1}).$$

Finally, permute the $\{\hat{\pi}_{i,p}(t_\tau)\}$ and correspondingly rearrange $\mathcal{E}^i(t_\tau)$.

4) *Loop*: $\tau \leftarrow \tau + 1$

```
if $\tau < M_q$ then
    return to Step 2
else
    Return $\mathcal{E}^i(t + M_q)$
end if
```

5) *End algorithm.*

Theorem 3.1: Consider a graph process in $[t_0, t_{m_l}]$ with l-edges addition and the nodes executing the Proposition 3.1 and Algorithm 3.3. In each time interval $(t_{k-1}, t_k]$ $(k \geq 1)$, nodes maximize the algebraic connectivity by adding l_k edges at discrete time t_k in a distributed fashion.

Proof: In fact, the obtained set $\{\hat{\pi}_{i,p}(t_k)\}$ derived from Algorithm 3.3 can be used as the local estimates of π_{ij} for node i. Selecting the first l_k elements from $\mathcal{E}^i(t_k)$ associated with this local estimate set and using Algorithm 3.1, all nodes has access to the optimal combination of candidate edges and locally implement the edge addition. ■

Moreover, in order ensure that the proposed distributed framework of edges addition is effectively implemented, we need to discuss the compatibility of the time scales among graph process, estimator (11) and self-diagnosis scheme of potentially adding edges in Algorithm 3.3. Evidently, the implementation of edge addition is made after achieving the output of Algorithm 3.3 that turns to call for an available estimate of the Fiedler vector from Algorithm 3.2. In other words, estimation iteration has already converged before implementing edge-addition. As a result, the schedule of the components involved in the distributed edge-addition should be appropriately arranged such that

$$M_s T_e + 2T_{\text{mix}}(t) < T, \quad \forall t \in [t_0, t_{m_l}], \tag{17}$$

where $T_{\text{mix}}(t)$ is the mixing time of a random work on a network. Regarding the fact that both observing scheme (12) and Algorithm 3.3 are derived from distributed consensus

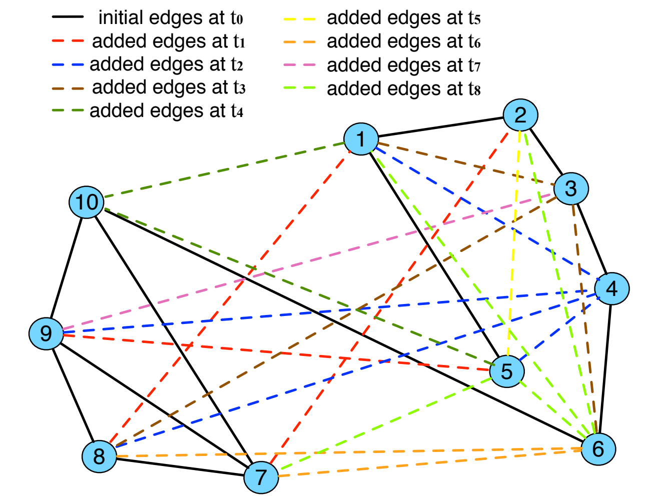


Fig. 2: The resulting weighted graph with 10 nodes under a graph process

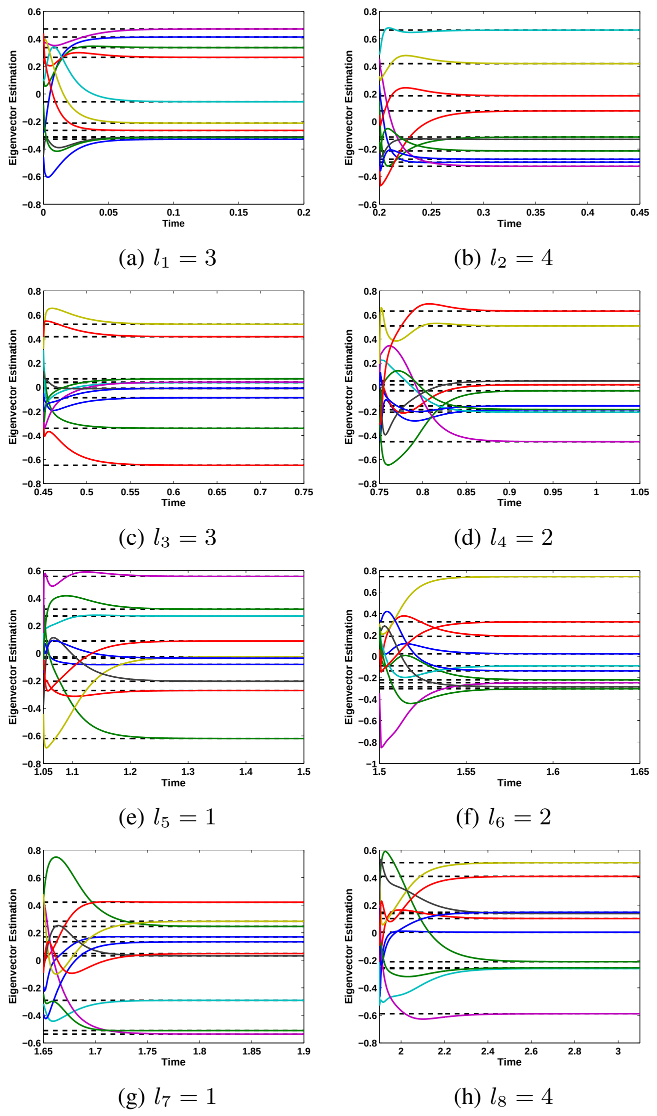
algorithms which has the close relationship with Markov chains. As a consequence, the running times for both components are closely related to the mixing time $T_{\text{mix}} = M_q T_o$ of the random walk on the corresponding network graph.

Remark 3.3: If the lower bound constraint in Assumption 3.1 does not hold most of the time, any online estimation schemes are available to work. However, the proposed distributed strategy of adding edges depend on the deviation between elements of eigenvector estimates and by specifying an error band, this scheme can effectively perform based on the output of the estimators at the settling time.

IV. Numerical Examples

In this section, some simulation examples are presented to show the performance of the algorithms. An initial graph $\mathcal{G}_0(\mathcal{V}, \mathcal{E}_0, \mathcal{A}_0)$ with $n = 10$ nodes and $|\mathcal{E}_0| = 11$ edges is randomly generated such that forming a connected topology, as shown as solid lines in Fig. 2. The weights of edges are randomly prespecified as $A_{ij} \in [1, 10]$. Once $\mathcal{G}_0$ and weighted matrix $\mathbb{A}$ are fixed, the set of numbers of added edges $\{l_k\}$ is the input of the algorithm which is set as$\{3, 4, 3, 2, 1, 2, 1, 4\}$ in this case. The decentralized estimation is randomly initiated and the resulting graph process is shown in Fig. 2. In Fig. 3, the corresponding eigenvector estimation processes are illustrated for each edge-adding round. The nodes achieve the same estimated the components of Fiedler eigenvector in a short time as shown in Fig. 4.

The topology design problem (P1) can be exactly solved by exhaustive search which leads to a global optimum. In the context of the graph process, we introduce a pseudo exhaustive search which computes λ_2 for $\binom{m_c - \sum_{j=1}^{k-1} l_k}{l_k}$ Laplacian matrices based the previous resulting graph $\mathcal{G}(t_{k-1})$. Fig. 5 shows the result from Theorem 3.1 offer better algebraic connectivity than pseudo exhaustive search does, and it is very close to the actual optimal values from exhaustive search.



(a) $l_1 = 3$ (b) $l_2 = 4$ (c) $l_3 = 3$ (d) $l_4 = 2$ (e) $l_5 = 1$ (f) $l_6 = 2$ (g) $l_7 = 1$ (h) $l_8 = 4$

Fig. 3: Fiedler eigenvector estimation for the graph process in Fig. 2

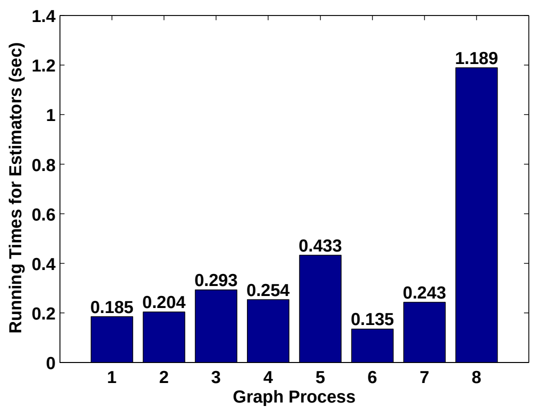


Fig. 4: Running time of distributed estimators for the graph process in Fig. 2

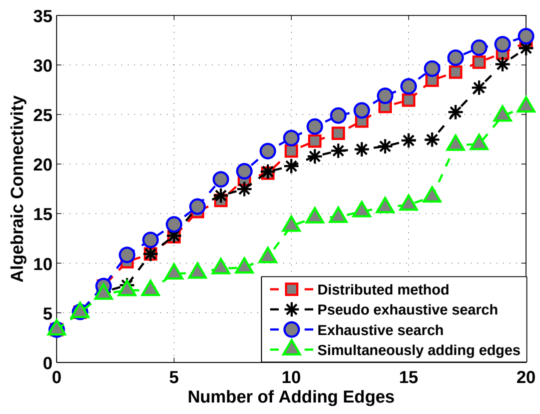


Fig. 5: A comparison of the decentralized methods with other optimal solutions

In addition, the observation in Fig. 5 is consistent with the common intuition that when using greedy perturbation in Algorithm 3.1, the strategy of adding edges one-by-one outperforms than adding them in one time. Indeed, edge-wise strategy of implementing graph process results in the least perturbation and track the evolution of algebraic connectivity in the most accurate way. In order to further affirm this conclusion, an additional example is provide and the consequent results are shown in Fig. 6. Referred to the analysis of matrix perturbation in Algorithm 3.1, we consider five different strategies of adding 20 edges in a $0-1$ unweighed graph. From Fig. 6, the strategy of adding edge one-by-one and simultaneously adding total edges provides the upper and lower bounds for distributed edge addition, respectively. In other words, adding less number of edges for per-step provides better performance but requires more steps.

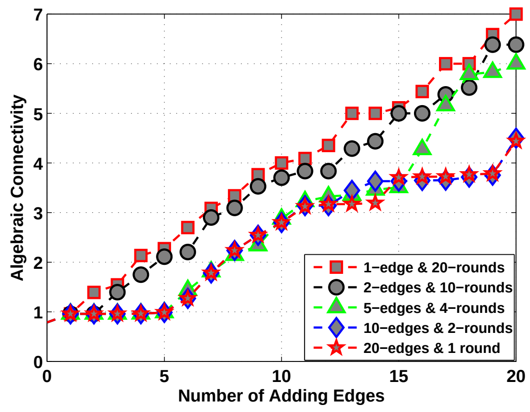


Fig. 6: Algebraic connectivity of adding 20 edges under different edge-addition strategies

V. CONCLUSIONS

This paper proposes a distributed strategy for solving the optimal topology design problem in networks. By introducing a graph process, we algebraically generalize the edge operation for optimal topology design. Based on eigenvalue sensitivity analysis, a heuristic rule of edge allocation is presented to approximately solve the original combinatorial optimization, which provides a simple search algorithm involving the deviation of the elements in Fiedler vector. The decentralized estimation scheme based on power iteration is established to locally estimate the desired eigenvector. As a result, a distributed strategy is synthesized to realize local self-awareness of edge addition for all individuals in networks.

REFERENCES

[1] R. Olfati-Saber and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95(1), pp. 215–233, 2007.

[2] P. Yang, R. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46(2), pp. 390–396, 2010.

[3] R. Dai and M. Mesbahi, "Optimal topology design for dynamic networks." 50th IEEE Conference on Decision and Control and European Control Conference, 2011, pp. 1280–1285.

[4] A. Ghosh and S. Boyd, "Growing well-connected graphs." 45th IEEE Conference on Decision and Control, 2006, pp. 6605–6611.

[5] M. Rafiee, "Optimal network topology design in multi-agent systems for efficient average consensus." 49th IEEE Conference on Decision and Control, 2010, pp. 3877–3883.

[6] M. Fiedler, "Some minimax problems for graphs," *Discrete Mathematics*, vol. 121, pp. 65–74, 1993.

[7] Y. Wan, S. Roy, X. Wang, A. Saberi, T. Yang, M. Xue, and B. Malek, "On the structure of graph edge designs that optimize the algebraic connectivity." 47th IEEE Conference on Decision and Control, 2008, pp. 805–810.

[8] A. Priolo, A. Gasparri, E. Montijano, and C. Sagues, "A decentralized algorithm for balancing a strongly connected weighted digraph." American Control Conference, 2013, pp. 6547–6552.

[9] P. Wei and D. Sun, "Weighted algebraic connectivity: an application to airport transportation network." 18th IFAC World Congress, 2011, pp. 13 846–13 869.

[10] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, vol. 74(1), pp. 70–83, 2008.

[11] C. Li and Z. Qu, "Distributed estimation of algebraic connectivity of directed networks," *Systems and Control Letters*, vol. 62, pp. 517–524, 2013.

[12] R. Aragues, G. Shi, D. Dimarogonas, C. Sagues, and K. Johansson, "Distributed algebraic connectivity estimation for adaptive event-triggered consensus." American Control Conference, 2012, pp. 32–37.

[13] D. Xue, A. Gusrialdi, and S. Hirche, "Robust distributed control design for interconnected systems under topology uncertainty." American Control Conference, 2013, pp. 6556–6561.

[14] A. Gusrialdi, "Performance-oriented communication topology design for distributed control of interconnected systems," *Mathematics of Control, Signals, and Systems, special issue on Control, Communication, and Complexity*, vol. 25(4), pp. 559–585, 2013.

[15] D. Bertseksas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[16] A. Tahbaz-Salehi and A. Jadbabaie, "A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times." 45th IEEE Conference on Decision and Control, 2006, pp. 4664–4669.

[17] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52(6), pp. 2508–2530, 2006.