Distributed Average Consensus with Least-Mean-Square Deviation

Lin Xiao, Stephen Boyd, and Seung-Jean Kim

***Abstract*— We consider a stochastic model for distributed average consensus, which arises in applications such as load balancing for parallel processors, distributed coordination of mobile autonomous agents, and network synchronization. In this model, each node updates its local variable with a weighted average of its neighbors' values, and each new value is corrupted by an additive noise with zero mean. The quality of consensus can be measured by the total mean-square deviation of the individual variables from their average, which converges to a steady-state value. We consider the problem of finding the (symmetric) edge weights that result in the least mean-square deviation in steady state. We show that this problem can be cast as a convex optimization problem, so the global solution can be found efficiently. We describe some computational methods for solving this problem, and compare the weights and the mean-square deviations obtained by this method and several other weight design methods.**

***Keywords*— average consensus, distributed algorithm, least mean square, convex optimization, eigenvalue optimization.**

I. Introduction

A. Asymptotic distributed average consensus

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be an undirected connected graph with node set $\mathcal{N} = \{1, \ldots, n\}$ and edge set $\mathcal{E}$, where each edge $\{i, j\} \in \mathcal{E}$ is an unordered pair of distinct nodes. Let $x_i(0)$ be a real scalar assigned to node i at time $t = 0$. The *distributed average consensus* problem is to compute (iteratively) the average $(1/n)\sum_{i=1}^n x_i(0)$ at every node, allowing only local communication on the graph. Thus, node i carries out its update, at each step, based on its local state and communication with its neighbors $\mathcal{N}_i = \{j \mid \{i, j\} \in \mathcal{E}\}$.

Distributed average consensus is an important problem in algorithm design for distributed computing. It has been extensively studied in computer science, for example in distributed agreement and synchronization problems (see, *e.g.*, [1]). It is a central topic for load balancing (with divisible tasks) in parallel computers (see, *e.g.*, [2], [3]). More recently, it has also found applications in distributed coordination of mobile autonomous agents (*e.g.*, [4], [5], [6], [7]), and distributed data fusion in sensor networks (*e.g.*, [8], [9], [10]).

Appearing in Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS), Kyoto, Japan, July 24-28, 2006.

Lin Xiao is with the Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA 91125-9300, USA. Email: lxiao@caltech.edu

Stephen Boyd and Seung-Jean Kim are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9510, USA. Emails: boyd@stanford.edu, sjkim@stanford.edu

The following linear iterative algorithm is widely used in the applications cited above:

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}(x_j(t) - x_i(t)), \qquad (1)$$

for $i = 1, \ldots, n$, and $t = 0, 1, \ldots$. Here W_{ij} is a weight associated with the edge $\{i, j\}$. These weights are algorithm parameters. Since we associate weights with undirected edges, we have $W_{ij} = W_{ji}$. (It is also possible to consider nonsymmetric weights, associated with ordered pairs of nodes.) Setting $W_{ij} = 0$ for $j \notin \mathcal{N}_i$ and $W_{ii} = 1 - \sum_{j \in \mathcal{N}_i} W_{ij}$, the algorithm in (1) can be expressesd as

$$x(t+1) = Wx(t), \qquad t = 0, 1, \ldots,$$

with initial condition $x(0) = (x_1(0), \ldots, x_n(0))$. By construction, the matrix W satisfies

$$W = W^T, \qquad W\mathbf{1} = \mathbf{1}, \qquad W \in \mathcal{S}, \qquad (2)$$

where $\mathbf{1}$ denotes the vector of all ones, and $\mathcal{S}$ denotes the set of matrices with sparsity patterns compatible with the graph:

$$\mathcal{S} = \{W \in \mathbf{R}^{n \times n} \mid W_{ij} = 0 \text{ if } i \neq j \text{ and } \{i, j\} \notin \mathcal{E}\}.$$

To achieve asymptotic average consensus no matter what the initial node values are, we must have

$$\lim_{t \to \infty} x(t) = \lim_{t \to \infty} W^t x(0) = (1/n)\mathbf{1}\mathbf{1}^T x(0)$$

for all $x(0)$; equivalently $\lim_{t \to \infty} W^t = (1/n)\mathbf{1}\mathbf{1}^T$. The (rank one) matrix on the right is the averaging matrix: $(1/n)\mathbf{1}\mathbf{1}^T z$ is the vector all of whose components are the average of the entries of z. We will use this matrix often, so we will denote it as

$$J = (1/n)\mathbf{1}\mathbf{1}^T.$$

The condition that we have asymptotic average consensus, in addition to those in (2), is

$$\|W - J\| < 1, \qquad (3)$$

where the norm is the spectral or maximum singular value norm (see, *e.g.*, [11]). The norm $\|W - J\|$ gives a measure of the worst-case, asymptotic rate of convergence to consensus. Indeed, the Euclidean deviation of the node values from their average is guaranteed to be reduced by the factor $\|W - J\|$ at each step:

$$\|x(t+1) - Jx(0)\| \leq \|W - J\|\|x(t) - Jx(0)\|,$$

(The vector norm here is the Euclidean norm, $\|u\| = (u^T u)^{1/2}$.)

Weights that satisfy the basic constraints (2), as well as the convergence condition (3), always exist. For example, we can take the *Metropolis-Hastings* weights:

$$W_{ij} = \begin{cases} 1/(\max\{d_i, d_j\}+1) & \{i,j\} \in \mathcal{E} \\ 1 - \sum_{j\in\mathcal{N}_i} 1/(\max\{d_i, d_j\}+1) & i = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $d_i = |\mathcal{N}_i|$ is the degree of node i in the graph [11].

Many variations of the model (1) have also been studied. These include problems where the weights are not symmetric, problems where final agreement is achieved, but not necessarily to the average (*e.g.*, [4], [6], [7]) Convergence conditions have also been established for distributed consensus on dynamically changing graphs (*e.g.*, [4], [6], [8], [7]) and with asynchronous communication and computation ([12], [13]) Alternative algorithm for average consensus has been studied in, *e.g.*, [14].

In [11], we formulated the *fastest distributed linear averaging* (FDLA) problem: choose the weights to obtain fastest convergence, *i.e.*, to minimize the asymptotic convergence factor $\|W - J\|$. We showed that (for symmetric weights) this FDLA problem is convex, and hence can be solved globally and efficiently. In this paper we study a similar optimal weight design problem, based on a stochastic extension of the simple averaging model (1).

B. Distributed average consensus with additive noise

We now consider an extension of the averaging iteration (1), with a noise added at each node, at each step:

$$x_i(t+1) = x_i(t) + \sum_{j\in\mathcal{N}_i} W_{ij}(x_j(t) - x_i(t)) + v_i(t), \tag{5}$$

for $i = 1, \ldots, n$, and $t = 0, 1, \ldots$. Here $v_i(t)$, $i = 1, \ldots, n$, $t = 0, 1, \ldots$ are independent random variables, identically distributed, with zero mean and unit variance. We can write this in vector form as

$$x(t+1) = Wx(t) + v(t),$$

where $v(t) = (v_1(t), \ldots, v_n(t))$. In the sequel, we will assume that W satisfies the conditions required for asymptotic average consensus without the noises, *i.e.*, that the basic constraints (2) and the convergence condition (3) hold.

With the additive noise terms, the sequence of node values $x(t)$ becomes a stochastic process. The expected value of $x(t)$ satisfies $\mathbf{E}\, x(t+1) = W\, \mathbf{E}\, x(t)$, so it propagates exactly like the node values without the noise term. In particular, each component of the expected value converges to the average of the initial node values. But the node values do not converge to the average of the initial node values in any useful sense. To see this, let $a(t) = (1/n)\mathbf{1}^T x(t)$ denote the average of the node values. Thus, $a(0)$ is the average of $x_i(0)$, and we have

$$a(t+1) = a(t) + (1/n)\mathbf{1}^T v(t),$$

using $\mathbf{1}^T W = \mathbf{1}^T$. The second term $(1/n)\mathbf{1}^T v(t)$ is a sequence of independent, zero mean, unit variance random variables. Therefore the average $a(t)$ undergoes a random walk, starting from the initial average value $a(0) = (1/n)\mathbf{1}^T x(0)$. In particular, we have

$$\mathbf{E}\, a(t) = a(0), \qquad \mathbf{E}(a(t) - \mathbf{E}\, a(t))^2 = t.$$

This shows that the additive noises induce a (zero mean) error in the average, which has variance that increases linearly with time, independent of the particular weight matrix used. In particular, we do not have average consensus (except in the mean), for any choice of W.

There is, however, a more useful measure of consensus for the sequence $x(t)$. We define $z(t)$ to be the vector of deviations of the components of $x(t)$ from their average. This can be expressed in component form as $z_i(t) = x_i(t) - a(t)$, or as

$$z(t) = x(t) - Jx(t) = (I - J)x(t).$$

We define the (total) mean-square deviation as

$$\delta(t) = \mathbf{E} \sum_{i=1}^{n} (x_i(t) - a(t))^2 = \mathbf{E}\, \|(I - J)x(t)\|^2.$$

This is a measure of relative deviation of the node values from their average, and can also be expressed as

$$\delta(t) = \frac{1}{n}\, \mathbf{E} \sum_{i<j} (x_i(t) - x_j(t))^2,$$

i.e., it is proportional to the average pairwise expected deviation among the node values (the exact average need a factor $2/(n(n-1))$ instead of $1/n$). Therefore $\delta(t)$ can be interpreted as a measure of how far the components of $x(t)$ are from consensus.

We will show that, assuming W satisfies (2) and (3), the mean-square deviation $\delta(t)$ converges to a finite (steady-state) value as $t \to \infty$, which we denote δ_{ss}:

$$\delta_{\text{ss}} = \lim_{t\to\infty} \delta(t).$$

This steady-state mean-square deviation is a function of the weights W, so we will denote it as $\delta_{\text{ss}}(W)$. It is a measure of how well the weight matrix W is able to enforce consensus, despite the additive noises introduced at each node at each step.

C. Least-mean-square consensus problem

In this paper we study the following problem: given the graph, find edge weights that yield the smallest steady-state mean-square deviation. This can be posed as the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \delta_{\text{ss}}(W) \\ \text{subject to} & W = W^T, \quad W\mathbf{1} = \mathbf{1} \\ & \|W - J\| < 1, \quad W \in \mathcal{S}, \end{array} \tag{6}$$

with variable $W \in \mathbf{R}^{n\times n}$. We call the problem (6) the *least-mean-square consensus* (LMSC) problem.

For future use, we describe an alternative formulation of the LMSC problem that is parametrized by the edge weights, instead of the weight matrix W. We enumerate the edges $\{i,j\} \in \mathcal{E}$ by integers $k = 1, \ldots, m$, where $m = |\mathcal{E}|$. We write $k \sim \{i,j\}$ if the edge $\{i,j\}$ is labeled k. We assign an arbitrary direction or orientation for each edge. Now suppose $k \sim \{i,j\}$, with the edge direction being from i to j. We associate with this edge the vector a_{ij} in $\mathbf{R}^n$ with ith element $+1$, jth element -1, and all other elements zero. We can then write the weight matrix as

$$W = I - \sum_{\{i,j\}\in\mathcal{E}} W_{ij} a_{ij} a_{ij}^T = I - \sum_{k=1}^{m} w_k a_k a_k^T, \qquad (7)$$

where w_k denotes the weight on the kth edge. It can be verified that the parametrization (7) of W automatically satisfies the basic constraints (2), and that conversely, any W that satisfies the basic constraints (2) can be expressed in the form (7). Thus, we can express the LMSC problem (6) as

$$\begin{array}{ll} \text{minimize} & \delta_{\rm ss}\left(I - \sum_{k=1}^m w_k a_k a_k^T\right) \\ \text{subject to} & \left\|I - J - \sum_{k=1}^m w_k a_k a_k^T\right\| < 1, \end{array} \qquad (8)$$

with variable $w \in \mathbf{R}^m$. In this formulation the only constraint is $\|W - J\| < 1$.

D. Applications

The model of average consensus with additive noises (5) and the LMSC problem (6) arise naturally in many practical applications. Here we briefly discuss its role in load balancing, coordination of autonomous agents, and network synchronization.

In the literature of load balancing, most work has focused on the *static* model (1), which is called a *diffusion scheme* because it can be viewed as a discretized diffusion equation (Poisson equation) on the graph [3]. Nevertheless, the stochastic version (5) is often more relevant in practice, in particular, for *dynamic* load balancing problems where a random amount of (divisible) tasks are generated during the load balancing process. In fact, one of the first models for a diffusion scheme proposed in [2] is of this kind:

$$q_i(t+1) = q_i(t) + \sum_{j\in\mathcal{N}_i} W_{ij}(q_j(t) - q_i(t)) - c + u_i(t).$$

Here $q_i(t)$ is the amount of (divisible) tasks waiting to be processed at node i at time t (the queue length), c is the constant number of tasks that every processor can complete in unit time, and $u_i(t)$ is a nonnegative random variable that accounts for new tasks generated at time t for processor i. The quantity $W_{ij}(q_j(t) - q_i(t))$ is the amount of tasks transfered from processor j to i (a negative number means transfering in the opposite direction). As discussed in [2], the most interesting case is when $\mathbf{E}\, u_i(t) = c$, and this is precisely the model (5) with the substitutions $v_i(t) = u_i(t) - c$ and $x_i(t) = q_i(t) - q_i(0)$. (Instead of adding the constraint $q_i(t) \geq 0$, we asumme the initial queue lengths $q_i(0)$ are large so that $q_i(t)$ remain nonnegative with very high probability.)

In dynamic load balancing problems, it is desirable to keep the mean-square deviation as small as possible, *i.e.*, to distribute the loads most evenly in a stochastic sense. This is precisely the LMSC problem (6), which (to our knowledge) has not been addressed before.

For distributed coordination of autonomous vehicles, the variable $x_i(t)$ can represent the position or velocity of each individual vehicle (*e.g.*, in the context of [4], [5]). The additive noises $v_i(t)$ in (5) can model random variations, *e.g.*, caused by disturbances on the dynamics of each local vehicle. Here the LMSC problem (6) is to obtain the best coordination in steady-state by optimizing the edge weights.

Another possible application of the LMSC problem is *drift-free* clock synchronization in distributed systems (*e.g.*, [15]). Here $x_i(t)$ represents the reading of a local relative clock (with the constant rate deducted), corrupted by random noise $v_i(t)$. Each node of the network adjusts its local clock via the diffusion scheme (5). The LMSC problem (6) amounts to finding the optimal edge weights that give the smallest (mean-square) synchronization error.

E. Outline

In §II, we derive several explicit expressions for the steady-state mean-square deviation $\delta_{\rm ss}(W)$, and show that the LMSC problem is a convex optimization problem. In §III we discuss computational methods for solving the LMSC problem, and explain how to exploit problem structure such as sparsity in computing the gradient and Hessian of $\delta_{\rm ss}$. In §IV, we consider a special case of the LMSC problem where all edge weights are taken to be equal, and illustrate its application to edge-transitive graphs. In §V, we present some numerical examples of the LMSC problem, and compare the resulting mean-square deviation with those given by other weight design methods, including the FDLA weights in [11].

II. Steady-state mean-square deviation

In this section we give a detailed analysis of the steady-state mean-square deviation $\delta_{\rm ss}$, including several useful and interesting formulas for it. We start with

$$x(t+1) = Wx(t) + v(t)$$

where W satisfies the basic constraints (2) and the convergence condition (3), and $v_i(t)$ are independent, identically distributed random variables with zero mean and unit variance. The deviation vector $z(t)$, defined as $z(t) = (I - J)\,x(t)$, satisfies $\mathbf{1}^T z(t) = 0$, and the recursion

$$z(t+1) = (W - J)z(t) + (I - J)v(t). \qquad (9)$$

Therefore we have

$$\mathbf{E}\, z(t) = (W - J)^t\, \mathbf{E}\, z(0) = (W - J)^t (I - J)x(0),$$

which converges to zero as $t \to \infty$, since $\|W - J\| < 1$.

Let $\Sigma(t) = \mathbf{E}\, z(t)z(t)^T$ be the second moment matrix of the deviation vector. The total mean-square deviation can be expressed in terms of $\Sigma(t)$ as

$$\delta(t) = \mathbf{E}\, \|z(t)\|^2 = \mathbf{Tr}\, \Sigma(t).$$

By forming the outer products of both sides of the equation (9), we have

$$\begin{aligned} z(t+1)z(t+1)^T &= (W-J)z(t)z(t)^T(W-J) \\ &\quad +(I-J)v(t)v(t)^T(I-J) \\ &\quad +2(W-J)z(t)v(t)^T(I-J). \end{aligned}$$

Taking the expectation on both sides, and noticing that $v(t)$ has zero mean and is independent of $z(t)$, we obtain a difference equation for the deviation second moment matrix,

$$\begin{aligned} \Sigma(t+1) &= (W-J)\Sigma(t)(W-J) + (I-J)I(I-J) \\ &= (W-J)\Sigma(t)(W-J) + I - J. \end{aligned} \tag{10}$$

(The second equality holds since $(I-J)$ is a projection matrix.) The initial condition is

$$\Sigma(0) = \mathbf{E}\, z(0)z(0)^T = (I-J)x(0)x(0)^T(I-J).$$

Since $\|W-J\| < 1$, the difference equation (10) is a stable linear recursion. It follows that the recursion converges to a steady-state value $\Sigma_{\text{ss}} = \lim_{t\to\infty}\Sigma(t)$, that is independent of $\Sigma(0)$ (and therefore $x(0)$), which satisfies discrete-time Lyapunov equation

$$\Sigma_{\text{ss}} = (W-J)\Sigma_{\text{ss}}(W-J) + (I-J). \tag{11}$$

We can express Σ_{ss} as

$$\begin{aligned} \Sigma_{\text{ss}} &= \sum_{t=0}^{\infty}(W-J)^t(I-J)(W-J)^t \\ &= (I-J) + \sum_{t=1}^{\infty}\left(W^2-J\right)^t \\ &= \sum_{t=0}^{\infty}\left(W^2-J\right)^t - J \\ &= \left(I+J-W^2\right)^{-1} - J. \end{aligned}$$

In several steps here we use the conditions (2) and (3), which ensure the existence of the inverse in the last line. We also note, for future use, that $I+J-W^2$ is positive definite, because it can be expressed as $I+J-W^2 = (\Sigma_{\text{ss}}+J)^{-1}$.

A. Expressions for steady-state mean-square deviation

Now we can write the steady-state mean-square deviation as an explicit function of W:

$$\delta_{\text{ss}}(W) = \mathbf{Tr}\,\Sigma_{\text{ss}} = \mathbf{Tr}\left(I+J-W^2\right)^{-1} - 1, \tag{12}$$

which we remind the reader holds assuming W satisfies the consensus averaging conditions $W=W^T$, $W\mathbf{1}=\mathbf{1}$, and $\|W-J\|<1$. This expression shows that δ_{ss} is an analytic function of W, since the inverse of a matrix is a rational function of the matrix (by Cramer's formula). In particular, it has continuous derivatives of all orders.

We give another useful variation of the formula (12). We start with the identity

$$I+J-W^2 = (I-J+W)(I+J-W),$$

which can be verified by multiplying out, and noting that $J^2=J$ and $JW=WJ=J$. Then we use the identity

$$((I-B)(I+B))^{-1} = \frac{1}{2}(I+B)^{-1} + \frac{1}{2}(I-B)^{-1},$$

with $B=W-J$ to obtain

$$\left(I+J-W^2\right)^{-1} = \frac{1}{2}(I+W-J)^{-1} + \frac{1}{2}(I-W+J)^{-1}.$$

Therefore we can express $\delta_{\text{ss}}(W)$ as

$$\delta_{\text{ss}}(W) = \frac{1}{2}\,\mathbf{Tr}(I+J-W)^{-1} + \frac{1}{2}\,\mathbf{Tr}(I-J+W)^{-1} - 1. \tag{13}$$

The condition $\|W-J\|<1$ is equivalent to $-I \prec W-J \prec I$, where $\prec$ denotes (strict) matrix inequality. These inequalities can be expressed as

$$I+J-W \succ 0, \qquad I-J+W \succ 0.$$

This shows that the two matrices inverted in the expression (13) are positive definite. We can therefore conclude that δ_{ss} is a convex function of W, since the trace of the inverse of a positive definite symmetric matrix is a convex function of the matrix [16, exercise 3.57]. This, in turn, shows that the LMSC problem (6), and its formulation in terms of the edge weights (8), are convex optimization problems.

Finally, we give an expression for δ_{ss} in terms of the eigenvalues of W. From (13), and using the fact that the trace of a matrix is the sum of its eigenvalues, we have

$$\begin{aligned} \delta_{\text{ss}}(W) &= \frac{1}{2}\sum_{i=1}^{n}\frac{1}{\lambda_i(I+J-W)} \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}\frac{1}{\lambda_i(I-J+W)} - 1, \end{aligned}$$

where $\lambda_i(\cdot)$ denotes the ith largest eigenvalue of a symmetric matrix. Since $W\mathbf{1}=\mathbf{1}$ (which corresponds to the eigenvalue $\lambda_1(W)=1$), the eigenvalues of $I+W-J$ are one, together with $\lambda_2(W),\ldots,\lambda_n(W)$. A similar analysis shows that the eigenvalues of $I-W-J$ are one, together with $-\lambda_2(W),\ldots,-\lambda_n(W)$. Therefore we can write

$$\begin{aligned} \delta_{\text{ss}}(W) &= (1/2)\sum_{i=2}^{n}\frac{1}{1-\lambda_i(W)} + (1/2)\sum_{i=2}^{n}\frac{1}{1+\lambda_i(W)} \\ &= \sum_{i=2}^{n}\frac{1}{1-\lambda_i(W)^2}. \end{aligned} \tag{14}$$

This simple formula has a nice interpretation. To achieve asymptotic average consensus, the weight matrix W is required to have $\lambda_1(W)=1$, with the other eigenvalues strictly between -1 and 1 (since $\|W-J\|<1$). It is the eigenvalues $\lambda_2(W),\ldots,\lambda_n(W)$ that determine the dynamics of the average consensus process. The asymptotic convergence factor is given by

$$\|W-J\| = \max\{\lambda_2(W), -\lambda_n(W)\},$$

and so is determined entirely by the largest (in magnitude) eigenvalues (excluding $\lambda_1(W)$=1). The formula (14) shows that the steady-state mean-square deviation is also a function of the eigenvalues (excluding $\lambda_1(W)$=1), but one that depends on all of them, not just the largest and smallest. The function $1/(1-\lambda^2)$ can be considered a barrier function for the interval $(-1,1)$ (*i.e.*, a smooth convex function that grows without bound as the boundary is approached). The steady-state mean-square deviation $\delta_{\rm ss}$ is thus a barrier function for the constraint that $\lambda_2(W),\ldots,\lambda_n(W)$ must lie in the interval $(-1,1)$. In other words, $\delta_{\rm ss}$ grows without bound as W approaches the boundary of the convergence constraint $\|W-J\|<1$.

B. Some bounds on steady-state mean-square deviation

Our expression for $\delta_{\rm ss}$ can be related to a bound obtained in [2]. If the covariance matrix of the additive noise $v(t)$ is given by $\sigma^2 I$, then it is easy to show that

$$\delta_{\rm ss}(W)=\sum_{i=2}^n \frac{\sigma^2}{1-\lambda_i(W)^2}.$$

The upper bound on $\delta_{\rm ss}$ in [2] is

$$\delta_{\rm ss}(W)\le \frac{(n-1)\sigma^2}{1-\|W-J\|^2},$$

which is a direct consequence of the fact

$$|\lambda_i(W)|\le \|W-J\|,\quad i=2,\ldots,n.$$

We can give a similar bound, based on both the spectral norm $\|W-J\|$ (which is $\max\{\lambda_2(W),-\lambda_n(W)\}$), and the Frobenius norm $\|W-J\|_F$,

$$\|W-J\|_F^2=\sum_{i,j=1}^n (W-J)_{ij}^2=\sum_{i=2}^n \lambda_i(W)^2.$$

For $|u|\le a<1$, we have

$$1+u^2\le \frac{1}{1-u^2}\le 1+\frac{1}{1-a^2}u^2.$$

Using these inequalities with $a=\|W-J\|$ and $u=\lambda_i$, for $i=2,\ldots,n$, we obtain

$$n-1+\sum_{i=2}^n \lambda_i(W)^2\le \delta_{\rm ss}(W)=\sum_{i=2}^n\frac{1}{1-\lambda_i(W)^2}$$

and

$$\delta_{\rm ss}(W)\le n-1+\frac{1}{1-\|W-J\|^2}\sum_{i=2}^n\lambda_i(W)^2.$$

Thus we have

$$n-1+\|W-J\|_F^2\le\delta_{\rm ss}(W)\le n-1+\frac{\|W-J\|_F^2}{1-\|W-J\|^2}.$$

C. Computing the steady-state mean-square deviation

Here we briefly discuss methods that can be used to compute $\delta_{\rm ss}$ for a fixed W (the weight matrix) or w (the vector of edge weights). One straightforward method is to compute all the eigenvalues of W, which allows us to check the convergence condition $\|W-J\|<1$, as well as evaluate $\delta_{\rm ss}(W)$ using (14). If we exploit no structure in W (other than, of course, symmetry), the computational cost of this approach is $O(n^3)$.

A more efficient method is based on formula (13), where we can exploit the sparse plus rank-one structure of the matrices $I+J-W$ and $I-J+W$. Let N be the number of nonzero elements in the Cholesky factor, after re-ordering to reduce the fill-in, of $I+W$ (which is sparse and positive definite). We can compute $\delta_{\rm ss}(W)$ with a total flop count $O(nN)$ and storage requirement $O(N)$. When N is on the order of n, this gives an $O(n^2)$ algorithm, one order faster than the methods that do not exploit sparsity. For details, see [17].

D. Derivation via spectral functions

In this section we show how convexity of $\delta_{\rm ss}(W)$, with the expression (14), can be derived using the theory of *convex spectral functions* [18, §5.2]. For $y\in\mathbf{R}^n$, we write $[y]$ as the vector with its components rearranged into nonincreasing order; *i.e.*, $[y]_i$ is the ith largest component of y. A function $g:\mathbf{R}^n\to\mathbf{R}$ is called *symmetric* if $g(y)=g([y])$ for all vectors $y\in\mathbf{R}^n$. In other words, a symmetric function is invariant under permutation of its arguments. Let g be a symmetric function and $\lambda(\cdot)$ denote the vector of eigenvalues of a symmetric matrix, arranged in nonincreasing order. The composite function $g\circ\lambda$ is called a *spectral function*. It is easily shown that a spectral function is *orthogonally invariant*; *i.e.*,

$$(g\circ\lambda)(QWQ^T)=(g\circ\lambda)(W)$$

for any orthogonal Q and any symmetric matrix W in $\mathbf{R}^{n\times n}$.

A spectral function $g\circ\lambda$ is closed and convex if and only if the corresponding symmetric function g is closed and convex (see, *e.g.*, [19] and [18, §5.2]). Examples of convex spectral functions include the trace, largest eigenvalue, and the sum of the k largest eigenvalues, for any symmetric matrix; and the trace of the inverse, and log determinant of the inverse, for any positive definite matrix. More examples and details can be found in, *e.g.*, [19], [20].

From the expression (14), we see that the function $\delta_{\rm ss}(W)$ is a spectral function, associated with the symmetric function (with $y_1=1$ always)

$$g(y)=\begin{cases}\displaystyle\sum_{i=2}^n\frac{1}{1-y_i^2}, & \text{if } -1<y_i<1,\ i=2,\ldots,n\\ +\infty, & \text{otherwise.}\end{cases}$$

Since g is closed and convex, we conclude that the spectral function $\delta_{\rm ss}$ is also closed and convex. Furthermore, $\delta_{\rm ss}$ is twice continuously differentiable because the above symmetric function g is twice continuously differentiable.

III. Solving the LMSC problem

In this section we describe computational methods for solving the LMSC problem (6). We will focus on the formulation (8), with edge weights as variables. We have already noted that the steady-state mean-square deviation $\delta_{\rm ss}$ is a barrier function for the convergence condition $\|W-J\|<1$, which can therefore be neglected in the optimization problem (8), provided we interpret $\delta_{\rm ss}$ as ∞ when the convergence condition does not hold. In other words, we must solve the unconstrained problem

$$\text{minimize} \quad f(w)=\delta_{\rm ss}\left(I-\textstyle\sum_{k=1}^m w_k a_k a_k^T\right), \qquad (15)$$

with variable $w\in\mathbf{R}^m$, where we interpret $f(w)$ as ∞ whenever $\|I-\sum_{k=1}^m w_k a_k a_k^T - J\|\geq 1$.

This is a smooth unconstrained convex optimization problem, and so can be solved by many standard methods, such a gradient descent method, quasi-Newton method, conjugate gradient method, or Newton's method. These methods have well known advantages and disadvantages in speed of convergence, computational cost per iteration, and storage requirements; see, *e.g.*, [21], [22], [16], [23]. These algorithms must be initialized with a point, such as the Metropolis-Hastings weight (4), that satisfies $f(w)<\infty$. At each step of these algorithms, we need to compute the gradient $\nabla f(w)$, and for Newton's method, the Hessian $\nabla^2 f(w)$ as well. In the next few sections we derive expressions for the gradient and Hessian, and describe methods that can be used to compute them.

A. Gradient

We start with the formula (13). With the substitution $W=I-\sum_{l=1}^m w_l a_l a_l^T$, we have

$$f(w)=\frac{1}{2}\,\mathbf{Tr}\,F(w)^{-1}+\frac{1}{2}\,\mathbf{Tr}\,G(w)^{-1}-1,$$

where

$$F(w)=2I-\sum_{k=1}^m w_k a_k a_k^T - J,\quad G(w)=\sum_{k=1}^m w_k a_k a_k^T + J.$$

Suppose that weight w_k corresponds to edge $\{i,j\}$, *i.e.*, $k\sim\{i,j\}$. Then we have

$$\begin{aligned}\frac{\partial f}{\partial w_k} &= -\frac{1}{2}\,\mathbf{Tr}\left(F^{-1}\frac{\partial F}{\partial w_k}F^{-1}\right)-\frac{1}{2}\,\mathbf{Tr}\left(G^{-1}\frac{\partial G}{\partial w_k}G^{-1}\right)\\ &= \frac{1}{2}\,\mathbf{Tr}\left(F^{-1}a_k a_k^T F^{-1}\right)-\frac{1}{2}\,\mathbf{Tr}\left(G^{-1}a_k a_k^T G^{-1}\right)\\ &= \frac{1}{2}\|F^{-1}a_k\|^2-\frac{1}{2}\|G^{-1}a_k\|^2\\ &= \frac{1}{2}\left\|(F^{-1})_{:,i}-(F^{-1})_{:,j}\right\|^2\\ &\quad -\frac{1}{2}\left\|(G^{-1})_{:,i}-(G^{-1})_{:,j}\right\|^2, \qquad (16)\end{aligned}$$

where $(F^{-1})_{:,i}$ denotes the ith column of F^{-1} (and similarly for G). In the first line, we use the fact that if a symmetric matrix X depends on a parameter t, then

$$\frac{\partial X^{-1}}{\partial t}=-\left(X^{-1}\frac{\partial X}{\partial t}X^{-1}\right).$$

The formula (16) gives us the optimality conditions for the problem (15): a weight vector $w^\star$ is optimal if and only if $F(w^\star)\succ 0$, $G(w^\star)\succ 0$, and, for all $\{i,j\}\in\mathcal{E}$, we have

$$\begin{aligned}&\left\|\left(F(w^\star)^{-1}\right)_{:,i}-\left(F(w^\star)^{-1}\right)_{:,j}\right\| = \\ &\qquad \left\|\left(G(w^\star)^{-1}\right)_{:,i}-\left(G(w^\star)^{-1}\right)_{:,j}\right\|.\end{aligned}$$

The formula (16) also gives us a simple method for computing the gradient $\nabla f(w)$. We first compute F^{-1} and G^{-1}. Then for each $k=1,\ldots,m$, we compute $\partial f/\partial w_k$, using the last line of (16). For each k, this involves subtracting two columns of F^{-1}, and finding the norm squared of the difference, and the same for G^{-1}, which has a cost $O(n)$, so this step has a total cost $O(mn)$. Assuming no structure is exploited in forming the inverses, the total cost is $O(n^3+mn)$, which is the same as $O(n^3)$, since $m\leq n(n-1)/2$. It is also possible to compute F^{-1} and G^{-1} more efficiently by exploiting their sparse plus rank-one structure (see §II-C).

B. Hessian

From the gradient formula above, we can derive the Hessian of f as

$$\begin{aligned}\frac{\partial^2 f}{\partial w_l \partial w_k} &= \frac{\partial}{\partial w_l}\left(\frac{1}{2}\|F^{-1}a_k\|^2-\frac{1}{2}\|G^{-1}a_k\|^2\right)\\ &= a_k^T F^{-1}\frac{\partial F^{-1}}{\partial w_l}a_k - a_k^T G^{-1}\frac{\partial G^{-1}}{\partial w_l}a_k\\ &= a_k^T F^{-1}\left(F^{-1}a_l a_l^T F^{-1}\right)a_k\\ &\quad + a_k^T G^{-1}\left(G^{-1}a_l a_l^T G^{-1}\right)a_k\\ &= (a_k^T F^{-1}a_l)(a_k^T F^{-2}a_l)\\ &\quad + (a_k^T G^{-1}a_l)(a_k^T G^{-2}a_l).\end{aligned}$$

Suppose that weight w_l corresponds to edge $\{p,q\}$, *i.e.*, $l\sim\{p,q\}$. Then we have

$$\begin{aligned}\frac{\partial^2 f}{\partial w_l \partial w_k} &= \alpha\Big((F^{-1})_{:,i}-(F^{-1})_{:,j}\Big)^T\Big((F^{-1})_{:,p}-(F^{-1})_{:,q}\Big)\\ &\quad +\beta\Big((G^{-1})_{:,i}-(G^{-1})_{:,j}\Big)^T\Big((G^{-1})_{:,p}-(G^{-1})_{:,q}\Big),\end{aligned}$$

where

$$\begin{aligned}\alpha &= a_k F^{-1}a_l = (F^{-1})_{i,p}-(F^{-1})_{i,q}-(F^{-1})_{j,p}+(F^{-1})_{j,q},\\ \beta &= a_k G^{-1}a_l = (G^{-1})_{i,p}-(G^{-1})_{i,q}-(G^{-1})_{j,p}+(G^{-1})_{j,q}.\end{aligned}$$

Once the matrices F^{-1} and G^{-1} are formed, for each $k,l=1,\ldots,m$, we can compute $\partial^2 f/\partial w_l \partial w_k$ using the last formula above, which has a cost $O(n)$. The total cost of forming the Hessian is $O(n^3+m^2 n)$, which is the same as $O(m^2 n)$, irrespective of the sparsity of the graph (and hence W). The computational cost per Newton step is $O(m^2 n+m^3)$, which is the same as $O(m^3)$ (the Hessian is fully dense even when the graph is sparse).

IV. LMSC WITH CONSTANT EDGE WEIGHT

In this section we consider a special case of the LMSC problem, where all edge weights are taken to be equal. This special case is interesting on its own, and in some cases, the optimal solution of the general LMSC problem can be shown to occur when all weights are equal.

When the edge weights are equal we have $w_k = \alpha$, so

$$W = I - \alpha \sum_{k=1}^{m} a_k a_k^T = I - \alpha L,$$

where L is the *Laplacian* matrix of the graph, defined as

$$L_{ij} = \begin{cases} -1 & \{i,j\} \in \mathcal{E}, \\ d_i & i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

The Laplacian matrix is positive semidefinite, and since we assume the graph is connected, it has a single eigenvalue $\lambda_n(L) = 0$, with associated eigenvector $\mathbf{1}$. We have

$$\lambda_i(W) = 1 - \alpha\lambda_{n-i+1}(L), \qquad i = 1, \ldots, n,$$

so the convergence condition $\|W - J\| < 1$ is equivalent to $0 < \alpha < 2/\lambda_1(L)$.

In this case, the steady-state mean-square deviation is

$$\begin{aligned} \delta_{\rm ss}(I - \alpha L) &= \sum_{i=1}^{n-1} \frac{1}{1 - (1 - \alpha\lambda_i(L))^2} \\ &= \sum_{i=1}^{n-1} \frac{1}{\lambda_i(L)\alpha} \, \frac{1}{2 - \lambda_i(L)\alpha}. \end{aligned}$$

Thus the LMSC problem reduces to

$$\text{minimize} \quad \sum_{i=1}^{n-1} \frac{1}{\lambda_i(L)\alpha} \, \frac{1}{2 - \lambda_i(L)\alpha}, \tag{18}$$

with scalar variable α, and the implicit constraint $0 < \alpha < 2/\lambda_1(L)$. The optimality condition is simply $\partial\delta_{\rm ss}/\partial\alpha = 0$, which is equivalent to

$$\sum_{i=1}^{n-1} \frac{1}{\lambda_i(L)} \, \frac{1 - \lambda_i(L)\alpha}{(2 - \lambda_i(L)\alpha)^2} = 0. \tag{19}$$

The lefthand side is monotone decreasing in α, so a simple bisection can be used to find the optimal weight α. A Newton method can be used to obtain very fast final convergence.

From (19) we can conclude that the optimal edge weight $\alpha^\star$ satisfies $\alpha^\star \geq 1/\lambda_1(L)$. To see this, we note that the lefthand side of (19) is nonnegative when $\alpha = 1/\lambda_1(L)$ and is $-\infty$ when $\alpha = 2/\lambda_1(L)$. Thus we have

$$\frac{1}{\lambda_1(L)} \leq \alpha^\star < \frac{2}{\lambda_1(L)}. \tag{20}$$

So we can always estimate $\alpha^\star$ within a factor of two, *e.g.*, with $\alpha = 1/\lambda_1(L)$.

A. LMSC problem on edge-transitive graphs

For graphs with large symmetry groups, we can exploit symmetry in the LMSC problem to develop far more efficient computational methods. In particular, we show that for *edge-transitive* graphs, it suffices to consider constant edge weight in the (general) LMSC problem.

An *automorphism* of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a permutation π of $\mathcal{N}$ such that $\{i,j\} \in \mathcal{E}$ if and only if $\{\pi(i), \pi(j)\} \in \mathcal{E}$. A graph is edge-transitive if given any pair of edges there is an automorphism which transforms one into the other. For example, rings and hypercubes are edge-transitive.

For edge-transitive graphs, we can assume without loss of generality that the optimal solution to the LMSC problem is a constant weight on all edges. To see this, let $w^\star$ be any optimal weight vector, not necessarily constant on all edges. Let $\pi(w^\star)$ denote the vector whose elements are rearranged by the permutation π. If π is an automorphism of the graph, then $\pi(w^\star)$ is also feasible. Let $\overline{w}$ denote the average of such vectors induced by all automorphisms of the graph. Then $\overline{w}$ is also feasible (because each $\pi(w)$ is feasible and the feasible set is convex), and moreover, using convexity of $\delta_{\rm ss}$, we have $\delta_{\rm ss}(\overline{w}) \leq \delta_{\rm ss}(w^\star)$. It follows that $\overline{w}$ is optimal. By construction, $\overline{w}$ is also invariant under the automorphisims. For edge-transitive graphs, this implies that $\overline{w}$ is a constant vector, *i.e.*, its components are equal. (See [16, exercise 4.4].) More discussion of exploiting symmetry in convex optimization problems can be found in [24], [25], [26].

B. Edge-transitive examples

In this section we consider several examples of graphs that are edge-transitive. The optimal weights are therefore constant, with value α (say) on each edge.

1) Rings: For rings with n nodes, the Laplacian matrix is circulant, and has eigenvalues

$$2\left(1 - \cos\frac{2k\pi}{n}\right), \qquad k = 0, \ldots, n-1.$$

Therefore we have

$$\delta_{\rm ss} = \sum_{k=1}^{n-1} \frac{1}{1 - \left(1 - 2\left(1 - \cos\frac{k\pi}{n}\right)\alpha\right)^2}.$$

For n even, $\lambda_1(L) = 4$ so by (20) we have $1/4 \leq \alpha^\star < 1/2$. For n odd, $\lambda_1(L) = 2(1 + \cos(\pi/n))$, so we have

$$\frac{1}{2(1 + \cos(\pi/n))} \leq \alpha^\star < \frac{1}{1 + \cos(\pi/n)}.$$

2) Meshes: Consider a two-dimensional mesh, with n nodes in each direction, with wraparounds at the edges. This mesh graph is the Cartesian products of two n-node rings (see, *e.g.*, [27]). The Laplacian is the Kroneker product of two circulant matrices, and has eigenvalues

$$4\left(1 - \cos\frac{(k{+}j)\pi}{n}\cos\frac{(k{-}j)\pi}{n}\right), \quad k, j = 0, \ldots, n-1.$$

Therefore $\delta_{\rm ss} =$

$$-1 + \sum_{k,j=0}^{n-1} \frac{1}{1 - \left(1 - 4\left(1 - \cos\frac{(k+j)\pi}{n}\cos\frac{(k-j)\pi}{n}\right)\alpha\right)^2}.$$

Again we can bound the optimal solution $\alpha^\star$ by (20). For example, when n is even, we have $\lambda_1(L) = 8$, therefore $1/8 \leq \alpha^\star < 1/4$.

3) Stars: The star graph with n nodes consists of one center node and $n-1$ peripheral nodes connected to the center. The Laplacian matrix has three distinct eigenvalues: 0, n, and 1. The eigenvalue 1 has multiplicity $n-2$. We have

$$\delta_{\rm ss} = \frac{1}{2n\alpha - n^2\alpha^2} + \frac{n-2}{2\alpha - \alpha^2}.$$

The optimality condition (19) boils down to

$$\frac{1-n\alpha^\star}{n(2-n\alpha^\star)^2} + (n-2)\frac{1-\alpha^\star}{(2-\alpha^\star)^2} = 0.$$

This leads to a cubic equation for $\alpha^\star$, which gives an analytical (but complicated) expression for $\alpha^\star$. In any case, the bounds (20) give $1/n \leq \alpha^\star < 2/n$.

4) Hypercubes: For a d-dimensional hypercube, there are 2^d vertices, each labeled with a binary word with length d. Two vertices are connected by an edge if their words differ in exactly one component. The Laplacian matrix has eigenvalues $2k$, $k = 0, 1, \ldots, d$, each with multiplicity $\binom{d}{k}$ (*e.g.*, [27]). Substituting these eigenvalues into (18), we find that

$$\delta_{\rm ss} = \sum_{k=1}^{d} \binom{d}{k} \frac{1}{4k\alpha - 4k^2\alpha^2},$$

with domain $0 < \alpha < 1/d$. The bounds (20) give $1/(2d) \leq \alpha^\star < 1/d$.

According to our numerical experiments, we conjecture that the optimal solution is $\alpha^\star = 1/(d+1)$, but we haven't been able to prove this yet. The value $\alpha = 1/(d+1)$ is also the solution for the FDLA problem studied in [11] (see also [2], [27], [26]).

V. Examples

In this section, we give some numerical examples of the LMSC problem (6), and compare the solutions obtained with the Metropolis weights (4) and weights that yield fastest asymptotic convergence, *i.e.*, a solution of

$$\begin{array}{ll} \text{minimize} & \|W - J\| \\ \text{subject to} & W \in \mathcal{S}, \quad W = W^T, \quad W\mathbf{1} = \mathbf{1}. \end{array} \qquad (21)$$

This FDLA problem need not have a unique solution, so we simply use *an* optimal solution. (See [11] for details of the FDLA problem.)

For each example, we consider a family of graphs that vary in the number of nodes or edges. For each graph instance, we report both the average mean-square deviation

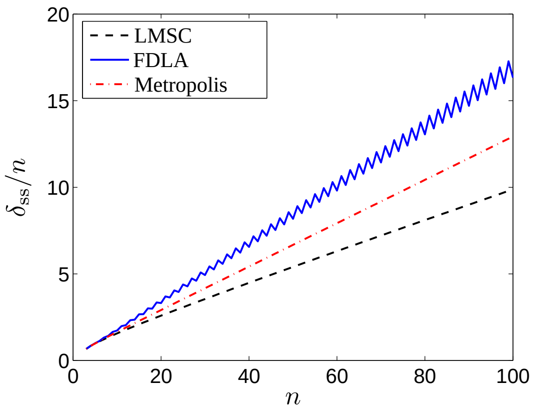

Fig. 1. Average MSD $\delta_{\rm ss}/n$ for rings with n nodes.

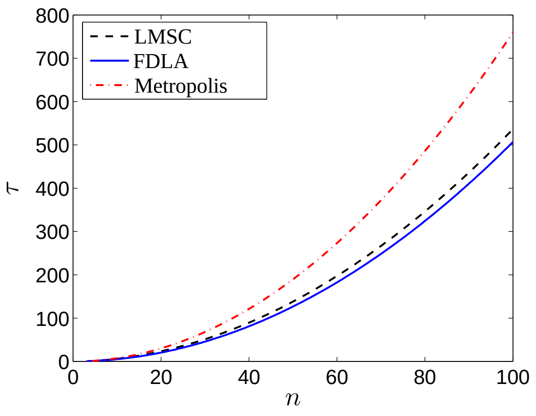

Fig. 2. Convergence time τ for rings with n nodes.

(MSD) $\delta_{\rm ss}/n$, which gives the asymptotic MSD per node. We also report the *convergence time*, defined as

$$\tau = \frac{1}{\log(1/\|W - J\|)}.$$

This gives the asymptotic number of steps for the error $\|x(t) - Jx(0)\|$ to decrease by a factor e, in the absence of noise. The FDLA weights minimize the convergence time τ.

Figures 1 and 2 show $\delta_{\rm ss}/n$ and τ, respectively, for ring graphs with a number of nodes ranging from 3 to 100. We see that the LMSC weights achieve much smaller average MSD than the FDLA weights, and the Metropolis weights (in this case a constant weight $1/3$ on all edges) have an MSD in between. In terms of the convergence time, however, there is not much difference between the LMSC weights and FDLA weights, with the Metropolis weights giving much slower convergence.

As a second example, we generate a random family of graphs, all with 100 nodes, as follows. First we generate a symmetric matrix $R \in \mathbf{R}^{100\times 100}$, whose entries R_{ij}, for $i \leq j$, are independent and uniformly distributed on $[0, 1]$. For each threshold value $c \in [0, 1]$ we construct a graph by placing an edge between vertices i and j for $i \neq j$ if

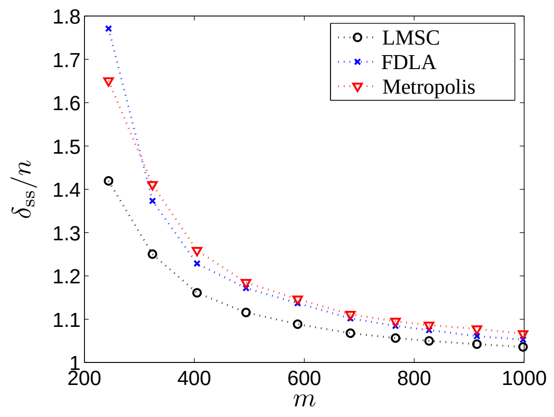

Fig. 3. Average MSD $\delta_{\rm ss}/n$ of a random family of graphs. Here the horizontal axis shows the number of edges (with fixed number of nodes).

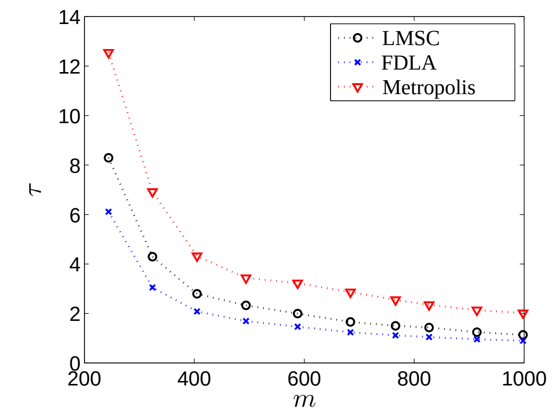

Fig. 4. Convergence time τ of a random family of graphs. The horizontal axis shows the number of edges (the number of nodes n is fixed).

$R_{ij} \leq c$. By increasing c from 0 to 1, we obtain a family of graphs. This family is monotone: the graph associated with a larger value of c contains all the edges of the graph associated with a smaller value of c. We start with a large enough value of c that the resulting graph is connected.

Figures 3 and 4 show $\delta_{\rm ss}/n$ and τ, respectively, for the graphs obtained with ten different values of c (in the range $[0.05,\ 0.2]$). Of course both the average MSD and convergence time decrease as the number of edges m increases. For this random family of graphs, the Metropolis weights often give smaller mean-square deviation than the FDLA weights.

More numerical examples are given in [17].

Acknowledgments

We thank Devavrat Shah and Anders Rantzer for helpful discussions.

References

[1] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, 1996.

[2] G. Cybenko. Load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.

[3] J. Boillat. Load balancing and Poisson equation in a graph. *Concurrency: Practice and Experience*, 2:289–313, 1990.

[4] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions Automatic Control*, 48(6):988–1001, June 2003.

[5] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[6] L. Moreau. Stability of multi-agent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

[7] W. Ren and R. W. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

[8] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks*, pages 63–70, Los Angeles, California, USA, April 2005.

[9] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Distributed sensor fusion using dynamic consensus. Submitted to the 16th IFAC World Congress, 2005.

[10] D. S. Scherber and H. C. Papadopoulos. Locally constructed algorithms for distributed computations in ad-hoc networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 11–19, Berkeley, CA, April 2004.

[11] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53:65–78, 2004.

[12] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, September 1986.

[13] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, 2005.

[14] C. C. Moallemi and B. Van Roy. Consensus propagation. Draft, 2005.

[15] B. Patt-Shamir and S. Rajsbaum. A theory of clock synchronization. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 810–819, Montreal, Canada, 1994.

[16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Available online at `http://www.stanford.edu/~boyd/cvxbook.html`.

[17] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. Submitted to *Journal of Parallel and Distributed Computing*, 2005. Available at `http://www.stanford.edu/~boyd/lms_consensus.html`.

[18] J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization, Theory and Examples*. Canadian Mathematical Society Books in Mathematics. Springer-Verlag, New York, 2000.

[19] A. S. Lewis. Convex analysis on the Hemitian matrices. *SIAM Journal on Optimization*, 6:164–177, 1996.

[20] M. L. Overton and R. S. Womersley. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Mathematical Programming*, 62:321–357, 1993.

[21] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 2nd edition, 1984.

[22] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[23] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 1999.

[24] K. Gatermann and P. A. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Applied Algebra*, 192:95–128, 2004.

[25] S. Boyd, P. Diaconis, P. Parrilo, and L. Xiao. Symmetry analysis of reversible Markov chains. *Internet Mathematics*, 2(1):31–71, 2005.

[26] S. Boyd, P. Diaconis, P. A. Parrilo, and L. Xiao. Fastest mixing Markov chain on graphs with symmetries. Manuscript in preparation, 2005.

[27] B. Mohar. Some applications of Laplace eigenvalues of graphs. In G. Hahn and G. Sabidussi, editors, *Graph Symmetry: Algebraic Methods and Applications*, NATO ASI Ser. C 497, pages 225–275. Kluwer, 1997.