Available online at www.sciencedirect.com
SCIENCE DIRECT®
Systems & Control Letters 53 (2004) 65–78
SYSTEMS & CONTROL LETTERS
www.elsevier.com/locate/sysconle

Fast linear iterations for distributed averaging

Lin Xiao*, Stephen Boyd

Information Systems Laboratory, Stanford University, Stanford, CA 94305-9510, USA

Received 21 March 2003; received in revised form 24 February 2004; accepted 25 February 2004

Abstract

We consider the problem of finding a linear iteration that yields distributed averaging consensus over a network, i.e., that asymptotically computes the average of some initial values given at the nodes. When the iteration is assumed symmetric, the problem of finding the fastest converging linear iteration can be cast as a semidefinite program, and therefore efficiently and globally solved. These optimal linear iterations are often substantially faster than several common heuristics that are based on the Laplacian of the associated graph.

We show how problem structure can be exploited to speed up interior-point methods for solving the fastest distributed linear iteration problem, for networks with up to a thousand or so edges. We also describe a simple subgradient method that handles far larger problems, with up to 100 000 edges. We give several extensions and variations on the basic problem.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Distributed consensus; Linear system; Spectral radius; Graph Laplacian; Semidefinite programming

1. Introduction

We consider a network (connected graph) $\mathscr{G}=(\mathscr{N},\mathscr{E})$ consisting of a set of nodes $\mathscr{N}=\{1,\ldots,n\}$ and a set of edges $\mathscr{E}$, where each edge $\{i,j\}\in\mathscr{E}$ is an unordered pair of distinct nodes. The set of neighbors of node i is denoted $\mathscr{N}_i=\{j|\{i,j\}\in\mathscr{E}\}$.

Each node i holds an initial scalar value $x_i(0)\in\mathbf{R}$, and $x(0)=(x_1(0),\ldots,x_n(0))$ denotes the vector of the initial node values on the network. (We can think of $x_i(0)$ as the amount of a certain resource allocated to node i.) The network gives the allowed communication between nodes: two nodes can communicate with each other, if and only if they are neighbors. We are interested in computing the average of the initial values, $(1/n)\sum_{i=1}^{n}x_i(0)$, via a distributed algorithm, in which the nodes only communicate with their neighbors. (If we think of the node values as the amount of a resource, then the average is a fair or uniform allocation of the resource across the network.)

Distributed averaging can be done in many ways. One straightforward method is flooding. Each node maintains a table of the initial node values of all the nodes, initialized with its own node value only. At each step, the nodes exchange information from their own tables and the tables of their neighbors. After a number of steps equal to the diameter of the network, every node knows all the initial values of all the nodes, so the average (or any other function of the initial node values) can be computed.

* Corresponding author.
E-mail addresses: lxiao@stanford.edu (L. Xiao), boyd@stanford.edu (S. Boyd).

0167-6911/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysconle.2004.02.022

In this paper, we only consider *distributed linear iterations*, which have the form

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

where $t = 0, 1, 2, \ldots$ is the discrete time index, and W_{ij} is the *weight* on x_j at node i. Setting $W_{ij} = 0$ for $j \notin \mathcal{N}_i$, this iteration can be written in vector form as

$$x(t+1) = Wx(t). \tag{1}$$

The constraint on the sparsity pattern of the matrix W can be expressed as $W \in \mathcal{S}$, where

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

The distributed averaging problem arises in the context of coordination of networks of autonomous agents, in particular, the *consensus* or *agreement* problem among the agents. Distributed consensus problems have been studied extensively in the computer science literature (see, e.g., [23]). Recently, it has found a wide range of applications, in areas such as formation flight of unmanned air vehicles and clustered satellites, and coordination of mobile robots. The recent paper [33] studies linear and nonlinear consensus protocols in these new applications with fixed network topology. Related coordination problems with time-varying topologies have been studied in [21] using a switched linear systems model, and in [27] using set-valued Lyapunov theory.

In previous work, the edge weights used in the linear consensus protocols are either constant or only dependent on the degrees of their incident nodes. With these simple methods of choosing edge weights, many concepts and tools from algebraic graph theory (e.g., [5,17]), in particular the *Laplacian* matrix of the associated graph, appear to be very useful in the convergence analysis of consensus protocols (see, e.g., [33] and Section 4 of this paper). The graph Laplacian has also been used in control of distributed dynamic systems (e.g., [13,14,25]).

This paper is concerned with general conditions on the weight matrix W for the linear iteration (1) to converge to the average at each node, and how we choose W to make the convergence as fast as possible.

1.1. Fastest distributed linear averaging problem

The linear iteration (1) implies that $x(t) = W^t x(0)$ for $t = 0, 1, 2, \ldots$. We want to choose the weight matrix W so that for any initial value $x(0)$, $x(t)$ converges to the average vector

$$\bar{x} = (\mathbf{1}^{\mathrm{T}}x(0)/n)\mathbf{1} = (\mathbf{1}\mathbf{1}^{\mathrm{T}}/n)x(0),$$

i.e.,

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} W^t x(0) = \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}x(0). \tag{2}$$

(Here $\mathbf{1}$ denotes the vector with all coefficients one.) This is equivalent to the matrix equation

$$\lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}.$$

Assuming this holds, we define the *asymptotic convergence factor* as

$$r_{\mathrm{asym}}(W) = \sup_{x(0)\neq\bar{x}} \lim_{t\to\infty} \left(\frac{\|x(t)-\bar{x}\|_2}{\|x(0)-\bar{x}\|_2}\right)^{1/t},$$

and the associated *convergence time*

$$\tau_{\mathrm{asym}} = \frac{1}{\log(1/r_{\mathrm{asym}})}, \tag{3}$$

which gives the (asymptotic) number of steps for the error to decrease by the factor $1/e$.

Another measure of the speed of convergence is the *per-step convergence factor* which is defined as

$$r_{\mathrm{step}}(W) = \sup_{x(t)\neq\bar{x}} \frac{\|x(t+1)-\bar{x}\|_2}{\|x(t)-\bar{x}\|_2}.$$

We can also define the associated convergence time τ_{step}, as in (3).

In this paper we consider the following problem: find the weight matrix $W \in \mathcal{S}$, consistent with the given network, that makes the convergence as fast as possible. In terms of the asymptotic convergence factor, this can be posed as the following optimization problem:

$$\begin{array}{ll} \text{minimize} & r_{\mathrm{asym}}(W) \\ \text{subject to} & W \in \mathcal{S}, \quad \lim_{t\to\infty} W^t = \mathbf{1}\mathbf{1}^{\mathrm{T}}/n. \end{array} \tag{4}$$

Here W is the optimization variable, and the network is the problem data. A similar optimization problem can be formulated by replacing the objective function $r_{\mathrm{asym}}(W)$ with the per-step convergence factor $r_{\mathrm{step}}(W)$. We call problem (4) (and in general its

variations, e.g., with the objective function $r_{\text{step}}(W)$, or additional constraints on the weight matrix) the *fastest distributed linear averaging* (FDLA) problem.

The FDLA problem (4) is closely related to the problem of finding the fastest mixing Markov chain on a graph [9]; the only difference in the two problem formulations is that in the FDLA problem, the weights can be (and the optimal ones often are) negative, hence faster convergence could be achieved compared with the fastest mixing Markov chain on the same graph. Despite the similarity in the problem formulations, this paper gives new results on convergence conditions for the weights without non-negative constraint, considers the per-step convergence factor, discusses in detail how to exploit structure in an interior-point method for solving the associated semidefinite programms, and introduces several interesting extensions.

1.2. Outline

In Section 2, we give necessary and sufficient conditions for a distributed linear iteration to converge to the average vector, and characterize the asymptotic and per-step convergence factors. In Section 3, we formulate the FDLA problem (with asymptotic convergence factor) as a spectral radius minimization problem, and the FDLA problem, with per-step convergence factor, as a spectral norm minimization problem. We then show that a variation on the FDLA problem, in which the weights are assumed to be symmetric, can be formulated as a semidefinite program (SDP). We also show how to formulate the FDLA problem, with per-step convergence factor, as an SDP. In Section 4, we describe some simple heuristics for the FDLA problem, based on the graph Laplacian, which at least guarantee convergence. In Section 5, we give some numerical examples, and show that the optimal weights often result in substantially faster convergence than those obtained from the simple heuristics. In Section 6, we show how to exploit structure in solving the symmetric FDLA problem (or the per-step FDLA problem) by an interior-point method, and also give a simple subgradient method that handles large-scale problems. In Section 7, we describe several extensions of the FDLA problem.

2. Convergence conditions

As we have seen, the distributed linear iteration (1) converges to the average, i.e., Eq. (2) holds, for any initial vector $x(0) \in \mathbf{R}^n$, if and only if

$$\lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}. \tag{5}$$

We have the following necessary and sufficient conditions for this matrix equation to hold.

Theorem 1. *Eq.* (5) *holds, if and only if*

$$\mathbf{1}^{\mathrm{T}} W = \mathbf{1}^{\mathrm{T}}, \tag{6}$$

$$W\mathbf{1} = \mathbf{1}, \tag{7}$$

$$\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) < 1, \tag{8}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix. Moreover,

$$r_{\text{asym}}(W) = \rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n), \tag{9}$$

$$r_{\text{step}}(W) = \|W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\|_2. \tag{10}$$

(*Here $\|\cdot\|_2$ denotes the spectral norm, or maximum singular value.*)

Before proving the theorem, we first give some interpretations:

- Eq. (6) states that $\mathbf{1}$ is a left eigenvector of W associated with the eigenvalue one. This condition implies that $\mathbf{1}^{\mathrm{T}}x(t+1) = \mathbf{1}^{\mathrm{T}}x(t)$ for all t, i.e., the sum (and therefore the average) of the vector of node values is preserved at each step.
- Eq. (7) states that $\mathbf{1}$ is also a right eigenvector of W associated with the eigenvalue one. This condition means that $\mathbf{1}$ (or any vector with constant entries) is a fixed point of the linear iteration (1).
- Together with the first two conditions, condition (8) means that one is a simple eigenvalue of W, and that all other eigenvalues are strictly less than one in magnitude.
- If the elements of W are nonnegative, then (6) and (7) state that W is doubly stochastic, and (8) states that the associated Markov chain is irreducible and aperiodic.

Proof. First we prove sufficiency. If W satisfies conditions (6) and (7), then

$$\begin{aligned} W^t - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n &= W^t\left(I - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\right) \\ &= W^t\left(I - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\right)^t \\ &= \left(W\left(I - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\right)\right)^t \\ &= \left(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\right)^t, \end{aligned} \tag{11}$$

where in the second equality, we use the fact that $I - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n$ is a projection matrix. Now applying condition (8) leads to the desired convergence (5).

To prove necessity, we use the fact that $\lim_{t\to\infty} W^t$ exists (such matrices are called *semi-convergent*) if and only if, there is a nonsingular matrix T such that

$$W = T\begin{bmatrix} I_\kappa & 0 \\ 0 & Z \end{bmatrix} T^{-1},$$

where I_κ is the κ-dimensional identity matrix $(0 \leqslant \kappa \leqslant n)$ and Z is a convergent matrix, i.e., $\rho(Z) < 1$. (This can be derived using the Jordan canonical form; see [32,26].) Let $u_1, \ldots, u_n$ be the columns of T and $v_1^{\mathrm{T}}, \ldots, v_n^{\mathrm{T}}$ be the rows of T^{-1}. Then we have

$$\begin{aligned} \lim_{t\to\infty} W^t &= \lim_{t\to\infty} T\begin{bmatrix} I_\kappa & 0 \\ 0 & Z^t \end{bmatrix} T^{-1} \\ &= T\begin{bmatrix} I_\kappa & 0 \\ 0 & 0 \end{bmatrix} T^{-1} = \sum_{i=1}^{\kappa} u_i v_i^{\mathrm{T}}. \end{aligned} \tag{12}$$

Since each $u_i v_i^{\mathrm{T}}$ is a rank-one matrix and their sum $\sum_{i=1}^{n} u_i v_i^{\mathrm{T}} = TT^{-1} = I$ has rank n, the matrix $\sum_{i=1}^{\kappa} u_i v_i^{\mathrm{T}}$ must have rank κ. Comparing Eqs. (5) and (12) gives $\kappa = 1$ and $u_1 v_1^{\mathrm{T}} = \mathbf{1}\mathbf{1}^{\mathrm{T}}/n$, which implies that both u_1 and v_1 are multiples of $\mathbf{1}$. In other words, one is a simple eigenvalue of W and $\mathbf{1}$ is its associated left and right eigenvectors, i.e., Eqs. (6) and (7) hold. Moreover,

$$\rho\left(W - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}\right) = \rho\left(T\begin{bmatrix} 0 & 0 \\ 0 & Z \end{bmatrix} T^{-1}\right) = \rho(Z) < 1,$$

which is precisely condition (8).

Finally Eqs. (9) and (10) can be derived directly from the error dynamics

$$\begin{aligned} x(t+1) - \bar{x} &= \left(W - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}\right) x(t) \\ &= \left(W - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}\right)(x(t) - \bar{x}). \end{aligned}$$

In other words, the asymptotic convergence factor r_{asym} is the spectral radius of $W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n$, and the per-step convergence factor r_{step} is its spectral norm. □

3. Fastest distributed linear averaging problems

Using Theorem 1, the FDLA problem (4) can be formulated as the following spectral radius minimization problem:

$$\begin{array}{ll} \text{minimize} & \rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) \\ \text{subject to} & W \in \mathscr{S}, \quad \mathbf{1}^{\mathrm{T}} W = \mathbf{1}^{\mathrm{T}}, \quad W\mathbf{1} = \mathbf{1}, \end{array} \tag{13}$$

with optimization variable W.

Even though the constraints in problem (13) are linear equalities, the problem in general is very hard. The reason is that the objective function, i.e., the spectral radius of a matrix, is not a convex function; indeed it is not even Lipschitz continuous (see, e.g., [35]). Some related spectral radius minimization problems are NP-hard [6,29].

We can also formulate the FDLA problem, with per-step convergence factor, as the following spectral norm minimization problem:

$$\begin{array}{ll} \text{minimize} & \|W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\|_2 \\ \text{subject to} & W \in \mathscr{S}, \quad \mathbf{1}^{\mathrm{T}} W = \mathbf{1}^{\mathrm{T}}, \quad W\mathbf{1} = \mathbf{1}. \end{array} \tag{14}$$

In contrast to the spectral radius formulation (13), this problem is convex, and can be solved efficiently and globally.

Now suppose we add the additional constraint that weights are symmetric, i.e., $W_{ij} = W_{ji}$ for all $\{i, j\} \in \mathscr{E}$. In this case the spectral radius minimization problem (13) and the spectral norm minimization problem (14) coincide (since the spectral norm of a symmetric matrix is also its spectral radius). In this case,

both problems can be cast as

$$\begin{array}{ll} \text{minimize} & \rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) \\ \text{subject to} & W \in \mathcal{S}, \quad W = W^{\mathrm{T}}, \quad W\mathbf{1} = \mathbf{1}, \end{array} \tag{15}$$

which is a convex problem. We refer to this problem as the symmetric FDLA problem.

The problem of minimizing the per-step convergence factor, i.e., the spectral norm minimization problem (14), can be expressed as an SDP, by introducing a scalar variable s to bound the spectral norm $\|W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\|_2$, and expressing the norm bound constraint as a linear matrix inequality (LMI):

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & \begin{bmatrix} sI & W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n \\ W^{\mathrm{T}} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n & sI \end{bmatrix} \succcurlyeq 0, \\ & W \in \mathcal{S}, \quad \mathbf{1}^{\mathrm{T}}W = \mathbf{1}^{\mathrm{T}}, \quad W\mathbf{1} = \mathbf{1}. \end{array} \tag{16}$$

Here the symbol $\succcurlyeq$ denotes matrix inequality, i.e., $X \succcurlyeq Y$ means that $X - Y$ is positive semidefinite. For background on SDP and LMIs, see, e.g., [1,2,11,12,16,41–43]. Related background on eigenvalue optimization can be found in, e.g., [10,22,36].

Similarly, the symmetric FDLA problem (15) can be expressed as the SDP

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & -sI \preccurlyeq W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n \preccurlyeq sI, \\ & W \in \mathcal{S}, \quad W = W^{\mathrm{T}}, \quad W\mathbf{1} = \mathbf{1}, \end{array} \tag{17}$$

with variables $s \in \mathbf{R}$ and $W \in \mathbf{R}^{n\times n}$.

4. Heuristics based on the Laplacian

There are some simple heuristics for choosing $W \in \mathcal{S}$ that guarantee convergence of the distributed linear averaging iteration, and sometimes give reasonably fast convergence. These heuristics are based on the Laplacian matrix of the associated graph and assign symmetric edge weights. To describe these heuristics, we first need to introduce the incidence matrix of the graph and an alternative representation of the FDLA problem based on it.

Suppose the graph has m edges, which we label $1,\ldots,m$. We arbitrarily assign a reference direction for each edge. (We will see later that this choice has no effect on the weight assignment, analysis, or algorithm.) The *incidence matrix* $A \in \mathbf{R}^{n\times m}$ is defined as

$$A_{il} = \begin{cases} 1 & \text{if edge } l \text{ starts from node } i, \\ -1 & \text{if edge } l \text{ ends at node } i, \\ 0 & \text{otherwise.} \end{cases}$$

The *Laplacian matrix* of the graph is defined as $L = AA^{\mathrm{T}}$ (and does not depend on the choice of reference directions). The Laplacian matrix is a useful tool in algebraic graph theory, and its eigenstructure reveals many important properties of the graph (see, e.g., [17,24]). We note for future use that L is positive semidefinite, and since our graph is assumed connected, L has a simple eigenvalue zero, with corresponding eigenvector $\mathbf{1}$.

We can use the incidence matrix to derive an alternative representation of the symmetric FDLA problem. Since we consider symmetric weights, each edge l of the graph is associated with the single weight $w_l = W_{ij} = W_{ji}$, where edge l is incident to nodes i and j, which we denote $l \sim \{i,j\}$. We let $w \in \mathbf{R}^m$ denote the vector of weights on the edges. Using this notation, the matrix W can be written as

$$W = I - A\,\mathbf{diag}(w)A^{\mathrm{T}}. \tag{18}$$

The advantage of expressing W in the form (18) is that it automatically satisfies the constraints $W \in \mathcal{S}$, $W = W^{\mathrm{T}}$, and $W\mathbf{1} = \mathbf{1}$. Therefore the (symmetric) FDLA problem (15) can be expressed as the unconstrained minimization problem

$$\text{minimize} \quad \|I - A\,\mathbf{diag}(w)A^{\mathrm{T}} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\|_2, \tag{19}$$

with variable $w \in \mathbf{R}^m$. This representation will also be used in the discussion of computational methods in Section 6.

4.1. Constant edge weights

The simplest approach is to set all the edge weights (for neighboring nodes) equal to a constant α; the self-weights on the nodes are then chosen to satisfy the condition $W\mathbf{1} = \mathbf{1}$. This corresponds to $w = \alpha\mathbf{1}$, and the associated weight matrix

$$W = I - \alpha AA^{\mathrm{T}} = I - \alpha L, \tag{20}$$

which can be expressed as

$$W_{ij} = \begin{cases} \alpha, & \{i,j\} \in \mathscr{E}, \\ 1 - d_i\alpha, & i = j, \\ 0, & \text{otherwise}, \end{cases}$$

where d_i is the degree of node i (i.e., the number of neighbors of the node i). Distributed linear averaging with this weight matrix is thus equivalent to the following commonly used iteration (see, e.g., [33]):

$$x_i(t+1) = x_i(t) + \alpha \sum_{j \in \mathscr{N}_i} (x_j(t) - x_i(t)),$$

$$i = 1, \ldots, n.$$

Since L is positive semidefinite, we must have $\alpha > 0$ for the convergence condition $\rho(W - \mathbf{1}\mathbf{1}/n) < 1$ to hold. From Eq. (20), we can express the eigenvalues of W in terms of those of L:

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

where $\lambda_i(\cdot)$ denotes the ith largest eigenvalue of a symmetric matrix. In particular, the eigenvalue zero of L corresponds to the eigenvalue one of W (i.e., $\lambda_n(L)=0$, $\lambda_1(W)=1$). The spectral radius of $W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n$ can then be expressed as

$$\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) = \max\{\lambda_2(W), -\lambda_n(W)\}$$
$$= \max\{1 - \alpha\lambda_{n-1}(L), \alpha\lambda_1(L) - 1\}. \quad (21)$$

From this we can determine the range of α over which convergence is obtained: we have $\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) < 1$, if and only if

$$0 < \alpha < \frac{2}{\lambda_1(L)}. \quad (22)$$

The choice of α that minimizes (21) is given by

$$\alpha^* = \frac{2}{\lambda_1(L) + \lambda_{n-1}(L)}. \quad (23)$$

This gives the best possible constant edge weight.

There are some simple bounds that give choices for α that do not require exact knowledge of the Laplacian spectrum. For example, we have the following bounds:

$$\lambda_1(L) \leqslant \max_{\{i,j\} \in \mathscr{E}} (d_i + d_j),$$

with equality if and only if the graph is bipartite and semiregular (see, e.g., [17,24]). It follows that distributed linear averaging (with constant edge weight α) converges for α in the range

$$0 < \alpha < \frac{2}{\max_{\{i,j\} \in \mathscr{E}} (d_i + d_j)}. \quad (24)$$

In particular, convergence is guaranteed if $\alpha \in (0, 1/d_{\max})$, where $d_{\max}$ is the maximum degree over the nodes in the network. In fact, convergence is guaranteed using the *maximum-degree weights*,

$$\alpha^{\mathrm{md}} = \frac{1}{d_{\max}}, \quad (25)$$

provided the graph is not bipartite.

Compared with the optimal weights, the maximum-degree weights often lead to much slower convergence when there are bottle-neck links in the graph. In [8], we give an example of two complete graphs connected by a bridge, where the optimal weight matrix W^* can perform arbitrarily better than the maximum-degree weights, in the sense that the ratio $(1 - r^*_{\mathrm{asym}})/(1 - r^{\mathrm{md}}_{\mathrm{asym}})$ can be unbounded as n (the number of nodes in the graph) increases.

4.2. Local-degree weights

Another simple method is to assign the weight on an edge based on the larger degree of its two incident nodes:

$$w_l = \frac{1}{\max\{d_i, d_j\}}, \quad l \sim \{i,j\}.$$

We call these weights the *local-degree weights*, since they depend only on the degrees of the incident nodes. This method comes from the Metropolis–Hastings algorithm, used to simulate a Markov chain with uniform equilibrium distribution; see, e.g., [9]. Similar to the maximum-degree weights, the local-degree weights guarantee convergence provided the graph is not bipartite.

5. Examples

We first consider the small network, with 8 nodes and 17 edges, shown in Fig. 1. For this network, the best constant edge weight, found from (23), is $\alpha^* = 0.227$. By solving the SDP (17), we found the optimal symmetric edge weights, which are labeled in

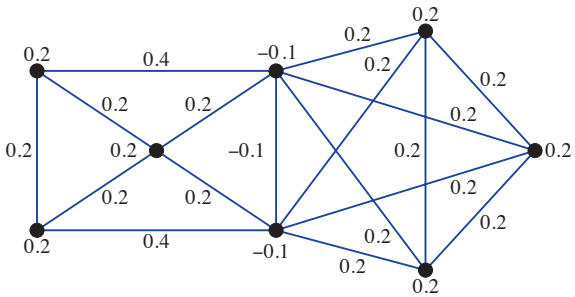

Fig. 1. A small graph with 8 nodes and 17 edges. Each edge and node is labeled with the optimal symmetric weights, which give the minimum asymptotic convergence factor.

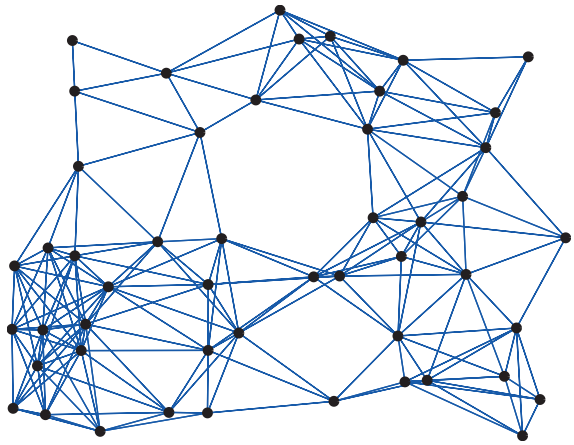

Fig. 2. A randomly generated network with 50 nodes and 200 edges.

Fig. 1. Note that the optimal symmetric weights for the two middle nodes, and the edge connecting them, are negative. To say the least, this is not an obvious choice of weights.

The asymptotic convergence factors and convergence times of the four choices of weights are summarized in Table 1. For this example the maximum-degree and local-degree weights result in the slowest convergence, with the optimal symmetric weights substantially faster than the best constant weight.

5.1. A larger network

Next we consider the graph shown in Fig. 2, which has 50 nodes and 200 edges. This graph was randomly generated as follows. First we randomly generate 50 nodes, uniformly distributed on the unit square. Two nodes are connected by an edge if their distance is less than a specified threshold. Then we increase the threshold until the total number of edges is 200. (The resulting graph is connected.)

The asymptotic convergence factors and convergence times, for the four different sets of weights, are summarized in Table 2. It can be seen that the convergence with the optimal symmetric weights is roughly twice as fast as with the best constant edge weight and the local-degree weights, and is more than three times faster than the maximum-degree weights.

Fig. 3 shows the eigenvalue distribution for the four weight matrices. Each of the distributions has a single eigenvalue at one. The convergence of the

Table 1
Convergence factors/times of different weights for the graph in Fig. 1

	Maximum-degree	Local-degree	Best constant	Optimal symmetric
$\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n)$	0.746	0.743	0.655	0.600
$\tau = 1/\log(1/\rho)$	3.413	3.366	2.363	1.958

Table 2
Convergence factors/times of different weights for the graph in Fig. 2

	Maximum-degree	Local-degree	Best constant	Optimal symmetric
$\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n)$	0.971	0.949	0.947	0.902
$\tau = 1/\log(1/\rho)$	33.980	19.104	18.363	9.696

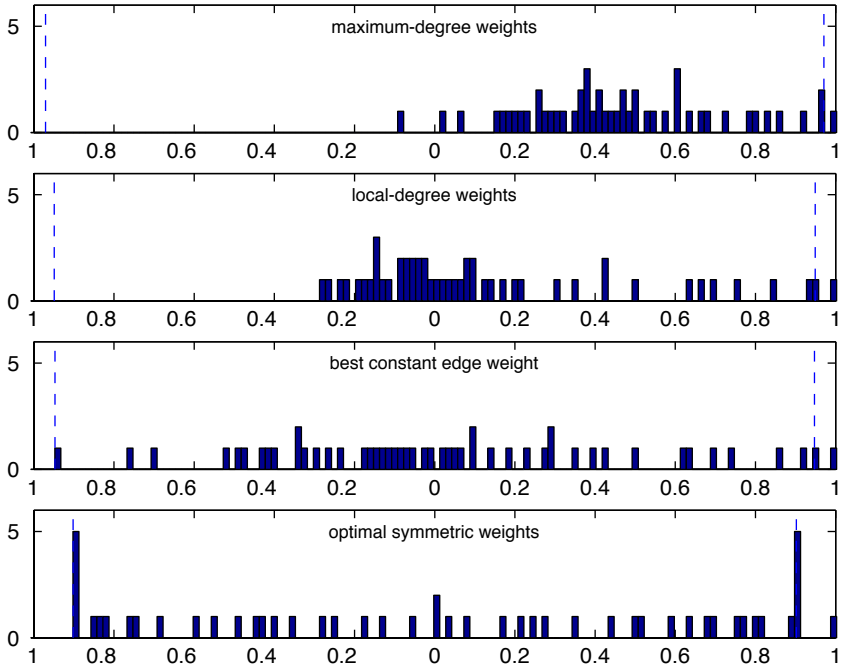

Fig. 3. Distribution of the eigenvalues of W with four different strategies. The dashed lines indicate $\pm\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n)$.

maximum-degree method is determined by its second eigenvalue, although it has a negative eigenvalue. For the local degree weights, the second eigenvalue is smaller, and the smallest eigenvalue is more negative, but still not enough to affect the convergence factor. The best constant edge weights always make $\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) = \lambda_2(W) = -\lambda_n(W)$, as shown in the figure. For the optimal symmetric weights, the eigenvalues (other than 1) have an approximately symmetric distribution, with many at or near the two critical values $\pm\rho(W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n)$. Fig. 4 shows the distribution of the optimal symmetric edge and node weights. It shows that many of the weights are negative.

We also solved problem (16) to find the optimal (possibly asymmetric) weights that minimize the per-step convergence factor; We found that $r^*_{\mathrm{step}} = 0.902$, and the solution W^* also turned out to be symmetric (the solution is non-unique). Our computational experience shows that allowing asymmetric weights usually does not lead to meaningful improvement of the per-step convergence factor.

6. Computational methods

6.1. Interior-point method

Standard interior-point algorithms for solving SDPs work well for problems with up to a thousand or so edges (see, e.g., [1,12,31,41–44]). The particular structure of the SDPs encountered in FDLA problems can be exploited for some gain in efficiency, but problems with more than a few thousand edges are probably beyond the capabilities of current interior-point SDP solvers.

We consider a simple primal barrier method, with the standard log-determinant barrier function (see, e.g., [12, Chapter 11]); the same techniques can be used to compute the search directions in other interior-point methods (e.g., primal–dual). In the primal barrier method, at each step we need to compute a Newton step for the problem of minimizing the function

$$\phi(s, w) = \mu s - \log\det(sI + W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) - \log\det(sI - W + \mathbf{1}\mathbf{1}^{\mathrm{T}}/n), \tag{26}$$

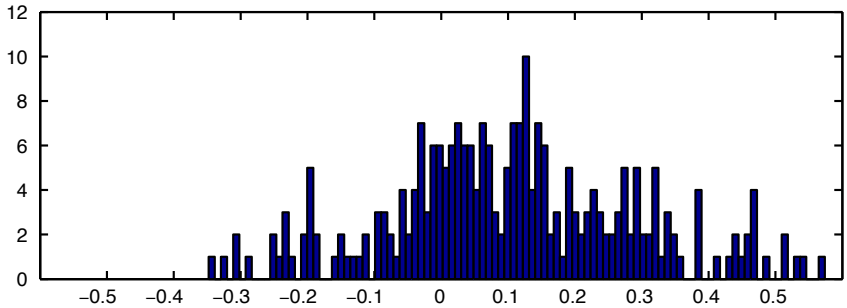

Fig. 4. Distribution of the optimal symmetric edge and node weights, found by solving the SDP (17). Note that many weights are negative.

where $\mu > 0$ is a parameter and W denotes $I - A\,\mathbf{diag}(w)A^{\mathrm{T}}$. The main effort is in forming the Hessian matrix $H = \nabla^2\phi(s,w)$, the gradient vector $g = \nabla\phi(s,w)$, and then computing the Newton step $-H^{-1}g$. We will show that the Hessian and gradient can be formed efficiently, by exploiting the structure of W, namely,

$$W = I - A\,\mathbf{diag}(w)A^{\mathrm{T}} = I - \sum_{l=1}^{m} w_l a_l a_l^{\mathrm{T}}$$

$$= I - \sum_{l=1}^{m} w_l (e_i - e_j)(e_i - e_j)^{\mathrm{T}}, \tag{27}$$

where e_i denotes the ith standard unit vector, and $a_l = e_i - e_j$, with $l \sim \{i,j\}$, is the lth column of the incidence matrix A.

We index the entries of the gradient vector from 0 to m, with g_0 denoting the derivative with respect to s. Similarly, we index the rows and columns of the Hessian H from 0 to m, with index 0 referring to the variable s. Each Newton step can be computed as follows:

1. Compute the matrices

$$U = (sI + W - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n)^{-1},$$

$$V = (sI - W + \mathbf{1}\mathbf{1}^{\mathrm{T}}/n)^{-1}.$$

Since the matrices to be inverted are positive definite, they can be computed by dense Cholesky factorization and back substitution. This costs $(5/3)n^3$ flops per inversion, so the total cost of this step is $(10/3)n^3$ flops. (Alternatively, we can exploit the structure of the matrices, which are sparse plus a rank one matrix, to form U and V more efficiently.)

2. Form the gradient and Hessian of ϕ as follows:

$$g_0 = \mu - \mathrm{tr}U - \mathrm{tr}V,$$

$$g_l = \mathrm{tr}(Ua_la_l^{\mathrm{T}}) - \mathrm{tr}(Va_la_l^{\mathrm{T}})$$
$$= (U_{ii} + U_{jj} - 2U_{ij}) - (V_{ii} + V_{jj} - 2V_{ij}),$$
$$l \sim \{i,j\},\ l = 1,\ldots,m,$$

$$H_{00} = \mathrm{tr}U^2 + \mathrm{tr}V^2,$$

$$H_{0l} = -\mathrm{tr}(Ua_la_l^{\mathrm{T}}U) + \mathrm{tr}(Va_la_l^{\mathrm{T}}V)$$
$$= -\left((U^2)_{ii} + (U^2)_{jj} - 2(U^2)_{ij}\right)$$
$$+ \left((V^2)_{ii} + (V^2)_{jj} - 2(V^2)_{ij}\right),$$
$$l \sim \{i,j\}, l = 1,\ldots,m,$$

$$H_{ll'} = \mathrm{tr}(Ua_la_l^{\mathrm{T}}Ua_{l'}a_{l'}^{\mathrm{T}}) + \mathrm{tr}(Va_la_l^{\mathrm{T}}Va_{l'}a_{l'}^{\mathrm{T}})$$
$$= (U_{ii'} + U_{jj'} - U_{ij'} - U_{i'j})^2$$
$$+ (V_{ii'} + V_{jj'} - V_{ij'} - V_{i'j})^2,$$
$$l \sim \{i,j\}, l' \sim \{i',j'\},$$
$$l = 1,\ldots,m, l' = 1,\ldots,m.$$

These formulas are derived using Eq. (27). The structure exploited here is similar to the methods used in the dual-scaling algorithm for large-scale combinatorial optimization problems, studied in [3]. The total costs of this step (number of flops) is on the order of m^2 (negligible compared with steps 1 and 3).

3. Compute the Newton step $-H^{-1}g$ by Cholesky factorization and back substitution. The cost of this step is $(1/3)m^3$ flops.

The primal barrier method usually finds the optimal solution within 20–80 N steps, at a cost of $(10/3)n^3 + (1/3)m^3$ flops per step.

6.2. Subgradient method

We now give a simple subgradient method that can solve the symmetric FDLA problem on a large-scale graph, with 100 000 or more edges. The disadvantage, compared to interior-point methods, is that this algorithm is relatively slow, and has no simple stopping criterion that guarantees a certain level of suboptimality.

Again consider problem (19), repeated here as

$$\text{minimize} \quad r(w) = \|I - A\,\mathbf{diag}(w)A^{\mathrm{T}} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n\|_2.$$

Here r represents r_{asym} or r_{step}, which are the same for symmetric weight matrices. The objective function $r(w)$ is a nonsmooth convex function.

A *subgradient* of r at w is a vector $g \in \mathbf{R}^m$ that satisfies the inequality

$$r(\tilde{w}) \geqslant r(w) + g^{\mathrm{T}}(\tilde{w} - w)$$

for any vector $\tilde{w} \in \mathbf{R}^m$. If r is differentiable at w, then $g = \nabla r(w)$ is the only subgradient; but when r is not differentiable at w, it can have multiple subgradients. Subgradients play a key role in convex analysis, and are used in several algorithms for convex optimization (see, e.g., [4,7,20,38,40]).

We can compute a subgradient of r at w as follows. If $r(w) = \lambda_2(W)$ and u is the associated unit eigenvector, then a subgradient g is given by

$$g_l = -(u_i - u_j)^2, \quad l \sim \{i,j\},\ l = 1,\ldots,m.$$

Similarly, if $r(w) = \lambda_n(W)$ and v is a unit eigenvector associated with $\lambda_n(W)$, then

$$g_l = (v_i - v_j)^2, \quad l \sim \{i,j\},\ l = 1,\ldots,m.$$

A more detailed derivation of these formulas can be found in [9]. For large sparse symmetric matrices W, we can compute a few extreme eigenvalues and their corresponding eigenvectors very efficiently using Lanczos methods (see, e.g., [37,39]). Thus, we can compute a subgradient of r very efficiently.

The subgradient method is very simple:

given a feasible $w^{(1)}$ (e.g., from the maximum-degree or local-degree heuristics)
$k := 1$
repeat
1. Compute a subgradient $g^{(k)}$ of r at $w^{(k)}$, and set $w^{(k+1)} = w^{(k)} - \beta_k g^{(k)}/\|g^{(k)}\|$.
2. $k := k + 1$.

Here the stepsize β_k is nonnegative and satisfies the *diminishing stepsize rule*:

$$\lim_{k\to\infty} \beta_k = 0 \quad \text{and} \quad \sum_{k=1}^{\infty} \beta_k = \infty.$$

The convergence of this algorithm is proved in [40, Section 2.2]. Some closely related methods for solving large-scale SDPs and eigenvalue problems are the spectral bundle method [19] and a prox-method [30]; see also [34].

To demonstrate the subgradient method, we apply it to a large-scale network with 10 000 nodes and 10 0000 edges. The graph is generated as follows. First we generate a 10 000 by 10 000 symmetric matrix R, whose entries R_{ij}, for $i \leqslant j$, are independent and uniformly distributed on the interval $[0,1]$. Then we choose a threshold value $\delta \in [0,1]$, and construct the graph by placing an edge between nodes i and j if $R_{ij} \leqslant \delta$. We choose δ such that there are precisely 100 000 edges (this graph is connected).

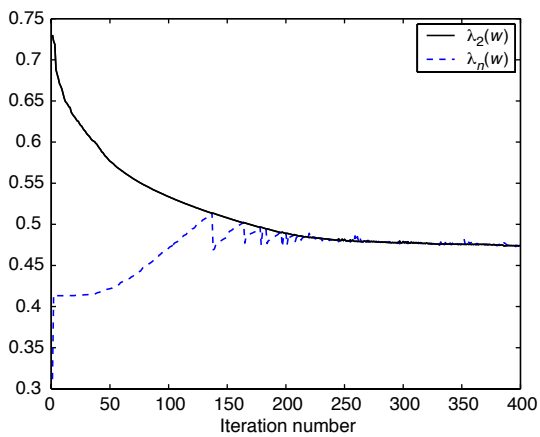

Fig. 5. Progress of the subgradient method for FDLA problem on a large network with 10 000 nodes and 100 000 edges.

We applied the subgradient method with stepsize $\beta_k = 1/(4\sqrt{k})$, starting with the local-degree weights, which has convergence factor $r = 0.730$. Fig. 5 shows the progress of the algorithm, plotting the magnitude of the two extreme eigenvalues λ_2 and λ_n of the matrix W. After 400 iterations, the algorithm gives a convergence factor $r = 0.473$, which is a significant reduction compared with the local-degree weights.

7. Extensions

In this section we describe several extensions of the basic FDLA problems described above.

7.1. Central weights

It is possible to consider criteria other than the asymptotic convergence factor or per-step factor in the selection of the weight matrix. Here we describe one simple example: we define the *central weights* as those that result in asymptotic convergence and minimize the logarithmic barrier function

$$\log\det(I - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n + W)^{-1} + \log\det(I + \mathbf{1}\mathbf{1}^{\mathrm{T}}/n - W)^{-1}. \tag{28}$$

(The terminology follows control theory; see, e.g., [28].) In terms of the eigenvalues λ_i of W, the central weights minimize the objective

$$\sum_{i=2}^{n} \log \frac{1}{1-\lambda_i^2}$$

(subject to $|\lambda_i| < 1$, $i = 2, \ldots, n$); in contrast, the fastest converging symmetric weights minimize the objective function

$$\max_{i \in \{2,\ldots,n\}} |\lambda_i| = \max\{\lambda_2, -\lambda_n\}.$$

The weight design problem with objective (28) is an unconstrained smooth convex minimization problem, readily solved using Newton's method.

7.2. Sparse graph design

An interesting variation on the FDLA problem is to find a sparse subgraph of the given graph, while guaranteeing a certain convergence factor. In other words, we seek an edge weight vector with as many zero entries as possible, subject to a prescribed maximum for the convergence factor. This is a difficult combinatorial problem, but one very effective heuristic to achieve this goal is to minimize the ℓ_1 norm of the vector of edge weights; see, e.g., [18] and [12, Chapter 6]. For example, given the maximum allowed asymptotic convergence factor $r^{\max}$, the ℓ_1 heuristic for the sparse graph design problem (with symmetric edge weights) can be posed as the convex problem

$$\text{minimize} \quad \sum_{l=1}^{m} |w_l| \tag{29}$$

subject to

$$-r^{\max} I \preccurlyeq I - A\,\mathbf{diag}(w)A^{\mathrm{T}} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n \preccurlyeq r^{\max} I.$$

It is also possible to assign weights to the edges, to achieve (hopefully) some desired sparsity pattern. More sophisticated heuristics for sparse design and minimum rank problems can be found in, e.g., [15].

To demonstrate this idea, we applied the ℓ_1 heuristic (29) to the example described in Section 5.1. We set the guaranteed convergence factor $r^{\max} = 0.910$, which is only slightly larger than the minimum factor 0.902. The resulting edge weight vector is relatively sparse; the number of edges with non-zero weights is reduced from 200 to 96.[1] This is illustrated in Fig. 6. Fig. 7 shows the distribution of the edge and node weights for the sparse network, and should be compared to the distribution shown in Fig. 4.

7.3. Distributed redistribution

In this section we consider a distributed redistribution problem, which is a generalization of the distributed averaging problem. Here we are interested in rearranging the scalar values at the nodes of a network to some desired distribution via distributed linear iterations, while maintaining a fixed weighted sum of the values at the nodes.

[1] In the numerical solution, a weight is considered zero if its magnitude is smaller than 10^{-3}. Actually there are only three weights with magnitude between 10^{-3} and 10^{-6}, and all remaining 101 weights have magnitudes less than 10^{-6}. We substituted the sparse weights (ignoring weights with magnitude less than 10^{-3}) back to compute the corresponding convergence factor, and we got exactly 0.910 as required.

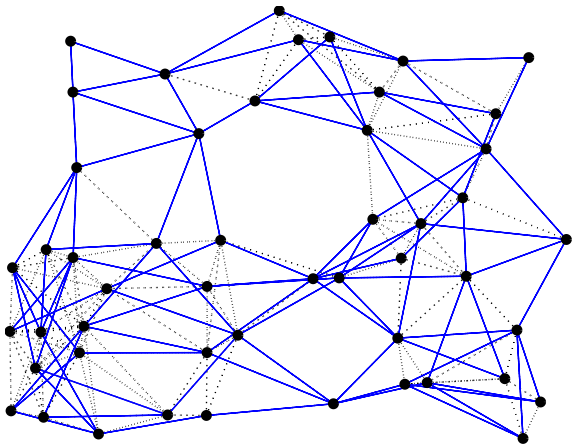

Fig. 6. Sparse network design with guaranteed convergence factor 0.910. The dotted lines show edges that are not used. The number of edges used (i.e., with non-zero edge weights) is reduced from 200 to 96.

More specifically, let $d \in \mathbf{R}^n$ be the desired distribution vector and $c \in \mathbf{R}^n$ be a specified weight vector. We will assume that $c^{\mathrm{T}} d \neq 0$. Then we want to find the distributed linear iteration $x(t+1) = Wx(t)$ such that starting with any $x(0) \in \mathbf{R}^n$, the vector $x(t)$ converges to γd, where the constant γ should satisfy $c^{\mathrm{T}} x(0) = c^{\mathrm{T}}(\gamma d)$, i.e., $\gamma = c^{\mathrm{T}} x(0)/(c^{\mathrm{T}} d)$. In other words, we want to find $W \in \mathscr{S}$ such that

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} W^t x(0) = \frac{dc^{\mathrm{T}}}{c^{\mathrm{T}} d} x(0)$$

for all $x(0) \in \mathbf{R}^n$. This is equivalent to the matrix equation

$$\lim_{t\to\infty} W^t = \frac{dc^{\mathrm{T}}}{c^{\mathrm{T}} d}. \tag{30}$$

Similar to Theorem 1, we have the following necessary and sufficient conditions for the convergence of the distributed redistribution:

Theorem 2. *Eq.* (30) *holds, if and only if*

$$c^{\mathrm{T}} W = c^{\mathrm{T}}, \quad Wd = d, \quad \rho(W - dc^{\mathrm{T}}/(c^{\mathrm{T}} d)) < 1.$$

Moreover, we have

$$r_{\mathrm{asym}}(W) = \rho(W - dc^{\mathrm{T}}/(c^{\mathrm{T}} d)),$$

$$r_{\mathrm{step}}(W) = \|W - dc^{\mathrm{T}}/(c^{\mathrm{T}} d)\|_2.$$

Similar to the distributed averaging problem, we can formulate the fastest distributed redistribution problem (minimizing the asymptotic convergence factor) as

$$\begin{array}{ll} \text{minimize} & \rho(W - dc^{\mathrm{T}}/(c^{\mathrm{T}} d)) \\ \text{subject to} & W \in \mathscr{S}, \quad c^{\mathrm{T}} W = c^{\mathrm{T}}, \quad Wd = d, \end{array} \tag{31}$$

where W is the optimization variable. As before, this problem is, in the general case, very hard. If, however, we minimize the spectral norm of $W - dc^{\mathrm{T}}/(c^{\mathrm{T}} d)$, which gives the per-step convergence factor, the problem is convex and can be expressed as an SDP similar to (16).

Finally we consider a special case of problem (31) that can be converted into a convex optimization problem. This special case is motivated by the fastest mixing reversible Markov chain problem studied in [9]. Specifically, we assume that

$$c_i d_i > 0, \quad i = 1, \ldots, n,$$

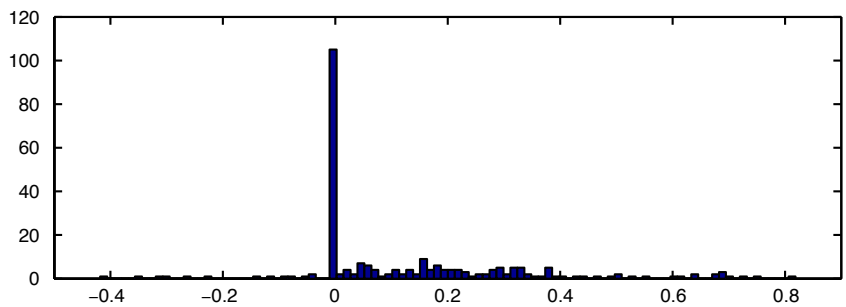

Fig. 7. Distribution of edge and node weights found by the ℓ_1 heuristic for sparse network design, with guaranteed convergence factor 0.910.

and define a matrix T and a vector q as follows:

$$T = \mathbf{diag}\left(\sqrt{d_1/c_1}, \ldots, \sqrt{d_n/c_n}\right),$$

$$q = Tc = T^{-1}d = \left(\sqrt{c_1 d_1}, \ldots, \sqrt{c_n d_n}\right).$$

We also restrict the weight matrix W to have the form $W = T\tilde{W}T^{-1}$, where $\tilde{W} \in \mathscr{S}$ and $\tilde{W} = \tilde{W}^{\mathrm{T}}$. Evidently W and $\widetilde{W}$ have the same eigenvalues, and q is the eigenvector of $\tilde{W}$ associated with the eigenvalue one.

We formulate a spectral radius minimization problem with the symmetric matrix $\widetilde{W}$ as the variable:

$$\begin{array}{ll} \text{minimize} & \rho(\tilde{W} - qq^{\mathrm{T}}) \\ \text{subject to} & \tilde{W} \in \mathscr{S}, \quad \tilde{W} = \tilde{W}^{\mathrm{T}}, \quad \tilde{W}q = q. \end{array}$$

This problem is convex and can be expressed as an SDP similar to (17). For any optimal solution $\tilde{W}^*$ to this problem, the matrix $W = T\tilde{W}^*T^{-1}$ satisfies (30) and has the same convergence factor, i.e., $\rho(W - dc^{\mathrm{T}}/(c^{\mathrm{T}}d)) = \rho(\widetilde{W}^* - qq^{\mathrm{T}})$.

Acknowledgements

The authors are grateful to Persi Diaconis who initiated our research on the fastest mixing Markov chain problem, which motivated the research in this paper. We also thank Pablo Parrilo for helpful discussions on exploiting graph symmetry in solving these problems.

References

[1] F. Alizadeh, Interior point methods in semidefinite programming with applications to combinatorial optimization, SIAM J. Optim. 5 (1995) 13–51.

[2] A. Ben-Tal, A. Nemirovski, Lectures on Modern Convex Optimization, Analysis, Algorithms, and Engineering Applications, MPS/SIAM Series on Optimization, SIAM, Philadelphia, PA, 2001.

[3] S. Benson, Y. Ye, X. Zhang, Solving large-scale sparse semidefinite programs for combinatorial optimization, SIAM J. Optim. 10 (2000) 443–461.

[4] D.P. Bertsekas, Nonlinear Programming, 2nd Edition, Athena Scientific, Belmont, MA, 1999.

[5] N. Biggs, Algebraic Graph Theory, 2nd Edition, Cambridge University Press, Cambridge, 1993.

[6] V. Blondel, J.N. Tsitsiklis, NP-hardness of some linear control design problems, SIAM J. Control Optim. 35 (1997) 2118–2127.

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

[8] S. Boyd, P. Diaconis, P. Parrilo, L. Xiao, Symmetry analysis of reversible Markov chains, Submitted to Internet Mathematics, December 2003, Also available online at: http://www.stanford.edu/~boyd/symmetry.html.

[9] S. Boyd, P. Diaconis, L. Xiao, Fastest mixing Markov chain on a graph, SIAM Review, in press, Available at: http://www.stanford.edu/~boyd/fmmc.html.

[10] S. Boyd, L. El Ghaoui, Method of centers for minimizing generalized eigenvalues, Linear Algebra Appl. 188 (1993) 63–111.

[11] S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory, SIAM, Philadelphia, 1994.

[12] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, 2003, Available at http://www.stanford.edu/~boyd/cvxbook.html.

[13] A. Fax, R.M. Murray, Graph Laplacians and vehicle formation stabilization, in: Proceedings of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain, July 2002.

[14] A. Fax, R.M. Murray, Information flow and cooperative control of vehicle formations, in: Proceedings of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain, July 2002.

[15] M. Fazel, H. Hindi, S. Boyd, A rank minimization heuristic with application to minimum order system approximation, in: Proceedings American Control Conference, Vol. 6, Arlington, VA, June 2001, pp. 4734–4739.

[16] L. El Ghaoui, S.-I. Niculescu (Eds.), Advances on Linear Matrix Inequality Methods in Control, SIAM, Philadelphia, 1999.

[17] C. Godsil, G. Royle, Algebraic Graph Theory, Graduate Texts in Mathematics, Vol. 207, Springer, Berlin, 2001.

[18] A. Hassibi, J. How, S. Boyd, Low-authority controller design via convex optimization, AIAA J. Guidance Control Dynam. 22 (6) (1999) 862–872.

[19] C. Helmberg, F. Rendl, A spectral bundle method for semidefinite programming, SIAM J. Optim. 10 (3) (2000) 673–696.

[20] J.-B. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms, Springer, Berlin, 1993.

[21] A. Jadbabaie, J. Lin, A.S. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, IEEE Trans. Autom. Control 48(6) (2003) 988–1001.

[22] A.S. Lewis, M.L. Overton, Eigenvalue optimization, Acta Num. 5 (1996) 149–190.

[23] N.A. Lynch, Distributed Algorithms, Morgan Kaufmann, San Francisco, CA, 1996.

[24] R. Merris, Laplacian matrices of graphs: a survey, Linear Algebra Appl. 197 (1994) 143–176.

[25] M. Mesbahi, On a dynamic extension of the theory of graphs, in: Proceedings of the American Control Conference, Anchorange, AL, May 2002.

[26] C.D. Meyer, R.J. Plemmons, Convergence powers of a matrix with applications to iterative methods for singular linear systems, SIAM J. Num. Anal. 14 (4) (1977) 699–705.

[27] L. Moreau, Stability of multi-agent systems with time-dependent communication links, IEEE Trans. Automat. Control, in press.
[28] D. Mustafa, K. Glover, Minimum Entropy H_∞ Control, Lecture Notes in Control and Information Sciences, Vol. 146, Springer, Berlin, 1990.
[29] A. Nemirovskii, Several NP-hard problems arising in robust stability analysis, Math. Control Signals Systems 6 (1993) 99–105.
[30] A. Nemirovskii, Prox-method with rate of convergence $O(1/t)$ for Lipschitz continuous variational inequalities and smooth convex-concave saddle point problems, Personal communication, 2003.
[31] Y. Nesterov, A. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, PA, 1994.
[32] R. Oldenburger, Infinite powers of matrices and characteristic roots, Duke Math. J. 6 (1940) 357–361.
[33] R. Olfati-Saber, R.M. Murray, Consensus protocols for networks of dynamic agents, in: Proceedings of American Control Conference, Denver, CO, June 2003.
[34] M.L. Overton, Large-scale optimization of eigenvalues, SIAM J. Optim. 2 (1992) 88–120.
[35] M.L. Overton, R.S. Womersley, On minimizing the spectral radius of a nonsymmetric matrix function—optimality conditions and duality theory, SIAM J. Matrix Anal. Appl. 9 (1988) 473–498.
[36] M.L. Overton, R.S. Womersley, Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices, Math. Programming 62 (1993) 321–357.
[37] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[38] R.T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, NJ, 1970.
[39] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Manchester University Press, Manchester, UK, 1992.
[40] N.Z. Shor, Minimization Methods for Non-differentiable Functions, Springer Series in Computational Mathematics, Springer, Berlin, 1985.
[41] M. Todd, Semidefinite optimization, Acta Num. 10 (2001) 515–560.
[42] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM Rev. 38 (1) (1996) 49–95.
[43] H. Wolkowicz, R. Saigal, L. Vandengerghe (Eds.), Handbook of Semidefinite Programming, Theory, Algorithms, and Applications, Kluwer Academic Publishers, Dordrecht, 2000.
[44] Y. Ye, Interior Point Algorithms: Theory and Analysis, Wiey-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, 1997.