SIAM Review
Vol. 53, No. 4, pp. 747–772

Convergence Speed in Distributed Consensus and Averaging*

Alex Olshevsky†
John N. Tsitsiklis‡

Abstract. We study the convergence speed of distributed iterative algorithms for the consensus and averaging problems, with emphasis on the latter. We first consider the case of a fixed communication topology. We show that a simple adaptation of a consensus algorithm leads to an averaging algorithm. We prove lower bounds on the worst-case convergence time for various classes of linear, time-invariant, distributed consensus methods, and provide an algorithm that essentially matches those lower bounds. We then consider the case of a time-varying topology, and provide a polynomial-time averaging algorithm.

Key words. consensus algorithms, distributed averaging, cooperative control

AMS subject classification. 93A14

DOI. 10.1137/110837462

1. Introduction.

Given a set of autonomous agents—which may be sensors, nodes of a communication network, cars, or unmanned aerial vehicles—with each agent i having a real-valued initial opinion $x_i(0)$, the distributed *averaging problem* asks for a distributed algorithm that the agents can use to compute the average of their opinions in the presence of severely restricted and time-varying communication capabilities. A solution to the averaging problem can be obtained, under suitable assumptions, by iterative algorithms of the form

$$x_i(t+1) = \sum_j a_{ij}(t)x_j(t), \tag{1.1}$$

if, at each time t, the matrix $A(t)$ formed by the coefficients $a_{ij}(t)$ is doubly stochastic. In a less demanding variant of the distributed averaging problem, known as the *consensus problem,* it is only required that the agents converge to a common opinion which is a convex combination of the initial opinions. In this case, the matrices $A(t)$ only need to be stochastic (as opposed to doubly stochastic). The subject of this paper is the design and analysis of averaging and consensus algorithms of the form (1.1), or of closely related forms, with a focus on the resulting convergence rates.

*Published electronically November 7, 2011. This paper originally appeared in *SIAM Journal on Control and Optimization*, Volume 48, Number 1, 2009, pages 33–55. This research was supported by the National Science Foundation under a Graduate Research Fellowship and grants ECS-0312921 and ECS-0426453.

http://www.siam.org/journals/sirev/53-4/83746.html

†Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ (aolshevs@princeton.edu).

‡Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA (jnt@mit.edu).

Distributed averaging and consensus algorithms are often useful as subroutines in more complicated network coordination problems. Although each agent may have access to different local information, the agents can fuse this information to agree on a collective decision (e.g., on a common direction of motion, on the time to execute a move, etc.). For this reason, such algorithms have historically appeared in many and diverse areas, such as parallel computation [57, 58, 5], control theory [35, 49], and communication networks [44, 41]. Recently, these problems have attracted considerable attention [35, 41, 7, 23, 44, 12, 28, 46, 45, 26, 29, 18, 3, 14, 10, 11, 2, 21, 15, 42, 13, 16, 17], motivated by new contexts and open problems in communications, sensor networks, and networked control theory. In the remainder of this section, we discuss some representative motivating applications, without attempting to exhaust the vast relevant literature.

1.1. Distributed Optimization. We illustrate here the use of distributed consensus and averaging algorithms in distributed optimization.

Consider a stochastic gradient algorithm for minimizing a smooth convex function f of a single variable:

$$y(t+1) = y(t) - \gamma(t)\big(f'(y(t)) + w(t)\big),$$

where $\gamma(t)$ is a positive stepsize, f' stands for the derivative of f, and $w(t)$ is a zero mean noise term, independently drawn at each time t. In a distributed version, each processor i obtains an independent noisy measurement of $f'(y(t))$ and updates according to

$$x_i(t) = y(t) - \gamma_i(t)\big(f'(y(t)) + w_i(t)\big).$$

Before proceeding to the next iteration, the processors can run a consensus algorithm of the form (1.1) to reconcile their different values $x_i(t)$ and agree on a common value $y(t+1)$, which is a convex combination $\sum_i \pi_i(t)x_i(t)$ of the values $x_i(t)$. The net effect is an update of the form

$$y(t+1) = y(t) - \sum_i \gamma_i(t)\pi_i(t)\big(f'(y(t)) + w_i(t)\big).$$

This net update retains the desirable property of moving in a direction of cost decrease (in expectation), while reducing the effect of the noise (because the noise terms $w_i(t)$ are averaged). In a more attractive version, introduced in [57, 58], which avoids excessive delay between consecutive updates of y, the execution of the consensus algorithm is interleaved with the stochastic gradient updates, resulting in an algorithm of the form

$$x_i(t+1) = \sum_j a_{ij}(t)x_j(t) - \gamma_i(t)\big(f'(x_i(t)) + w_i(t)\big).$$

As long as the consensus algorithm converges in a time scale faster than the time scale of the stochastic gradient updates (the latter being determined by the stepsizes $\gamma_i(t)$), this interleaved algorithm retains the desirable convergence properties of single-processor stochastic gradient algorithms [57, 58].

An alternative setting involves a separable cost function of the form

$$F(y) = \sum_i f_i(y),$$

where the functions f_i are convex, but not necessarily differentiable. Suppose that the form of the function f_i is known only by processor i. For example, f_i could represent a statistical loss function associated with data collected locally at sensor i. Alternatively, f_i could correspond to the loss function associated with a subblock of data in a large-scale machine learning problem.

In order to carry out a subgradient update, we can have each processor compute a local subgradient f_i', and then use an averaging algorithm to average these local subgradients. Similar to our earlier development, one can consider interleaving an averaging algorithm with the subgradient updates to obtain an algorithm of the form [48]

$$x_i(t+1) = \sum_j a_{ij}(t)x_j(t) - \gamma_i(t)f_i'(x_i(t)).$$

(See [24] for related methods.)

Because the gradient of F gives equal weight to each of the local cost functions f_i, the coefficients $a_{ij}(t)$ must conform to an averaging algorithm (consensus will not suffice) and the stepsizes $\gamma_i(t)$ must be the same for all i. Under these conditions and some additional assumptions, the interleaved algorithm retains the desirable convergence properties of single-processor subgradient algorithms [48]. We note that the available convergence time estimates for distributed subgradient methods [48, 24, 54] typically involve two contributions: one reflecting the convergence time of a centralized subgradient algorithm, and one reflecting the convergence time of the underlying averaging algorithm. This suggests that even if the convergence time of an averaging algorithm is studied in isolation, it has an immediate bearing on the distributed optimization setting.

1.2. Additional Motivating Problems.

Formation Control. Consider a collection of UAVs or other types of vehicles that wish to maintain a formation in the face of random disturbances. Every once in a while, a pair of vehicles succeeds in getting an accurate measurement of their relative positions. What is desired is a protocol for taking actions (e.g., changing positions) based on these measurements, so that over time, as more and more measurements arrive, the vehicles arrange themselves according to the desired formation.

Distributed averaging updates can be used to design such protocols. This is because averaging updates cope well with time-varying connectivity and intermittent measurements. For more details, we refer the reader to Chapter 2 of [51] for an exposition of some ideas originating in [49].

Coverage Control. Coverage control is the problem of optimally positioning a set of robots to monitor an area. A typical case involves a polygon-shaped area along with robots that can measure distances to the boundary as well as to each other. Based on these distances, it is desirable to construct controllers which cover the entire area, yet assign as little area as possible to each robot. A common addition to this setup involves associating a number $f(x)$ to each point x in the polygon, representing the importance of monitoring this point. The robots then optimize a corresponding objective function that weighs regions according to the importance of the points in them.

It turns out that averaging algorithms are very useful in designing distributed controllers for such systems. We refer the reader to [29] for the connection between distributed controllers and averaging over a certain class of graphs defined by Voronoi

diagrams. A similar approach was adopted in [56]. Note also the extension to nonuniform coverage in [37].

Clock Synchronization. A collection of clocks that are constantly drifting apart would like to maintain a common time to the extent that this is possible. Various pairs of clocks can measure noisy versions of the time offsets between them. This is a common scenario, because clocks drift randomly depending on various factors within their environment (e.g., temperature), and also because clocks often have a (nonzero) drift relative to the "true" time. In this context, maintaining a common time is important for a number of estimation problems (for example, in direction of arrival problems).

This problem has a natural similarity to averaging, except that one does not care about getting the average right; just reaching consensus is enough. On the other hand, the constantly evolving clock times present a challenge. A natural approach, explored in some of the recent literature, is to adopt averaging techniques to work in this setting. We refer the reader to [38, 55, 4, 27].

Reputation Management in Ad Hoc Networks. It is often the case that the nodes of a wireless multihop network are not controlled by a single authority or do not have a common objective. Selfish behavior among nodes (e.g., refusing to forward traffic meant for others) is possible, and some mechanism is needed to enforce cooperation. One way to detect selfish behavior is reputation management; i.e., each node forms an opinion by observing the behavior of its neighbors. One is then faced with the problem of combining these different opinions into a single globally available reputation measure for each node. The use of distributed consensus algorithms for doing this was explored in [41], where a variation of the algorithm (1.1) was used as a basis for an empirical investigation.

Social Network Models. All the contexts discussed so far are in the spirit of engineering system design: an algorithm is constructed to accomplish a certain objective. In a philosophically different but mathematically similar line of research, consensus algorithms of the form (1.1) are used as stylized models of opinion formation and updating in social networks. This type of research was initiated by DeGroot [20] who proposed a time-invariant version of (1.1) as a model of the merging of expert opinions. A number of recent works has taken this line of analysis further by analyzing how the graph-theoretic structure of social networks and the detailed features of the update rule affect the outcome. In particular, the seminal paper [31] introduced a popular and mathematically appealing model of opinion fragmentation. The paper [22] explored applications of these types of models to a variety of situations in politics, and the related work [1] quantified the extent to which "stubborn" agents (who maintain fixed opinions) could influence the opinions of other agents. Finally, [30] studied the effectiveness of social networks at aggregating distributed information in terms of certain graph-theoretic properties.

1.3. Summary and Contributions. Our general objective is to characterize the worst-case convergence time of various averaging algorithms, as a function of the number n of agents, and to understand their fundamental limitations by providing lower bounds on the convergence time.

We now outline the remainder of this paper and preview its main contributions. In section 2, we provide some background material by reviewing the agreement algorithm of [57, 58] for the distributed consensus problem. In sections 3–8, we consider the case of fixed graphs. In section 3, we discuss three different ways in which the agreement

algorithm can provide a solution to the averaging problem. In particular, we show how an averaging algorithm can be constructed based on two parallel executions of the agreement algorithm. In section 4, we define the notions of convergence rate and convergence time, and we provide a variational characterization of the convergence rate.

In section 5, we use results from [36] to show that the worst-case convergence time of an averaging algorithm introduced in section 3 is essentially $\Theta(n^3)$.[1] In section 6, we show that for one of our methods the convergence rate can be made arbitrarily fast. On the other hand, under an additional restriction that reflects numerical stability considerations, we show that the convergence time of a certain class of algorithms (and by extension of a certain class of averaging algorithms) is $\Omega(n^2)$, in the worst case. We also provide a simple method (based on executing the agreement algorithm on a spanning tree) whose convergence time essentially matches the $\Omega(n^2)$ lower bound. In section 7, we discuss briefly particular methods that employ doubly stochastic matrices and their potential drawbacks.

Then, in section 8, we turn our attention to the case of dynamic topologies. For the agreement algorithm, we show that its convergence time for the case of nonsymmetric topologies can be exponentially large in the worst case. On the other hand, for the case of symmetric topologies, we provide a new averaging algorithm (and, therefore, an agreement algorithm as well) whose convergence time is $O(n^3)$. To the best of our knowledge, none of the previously available consensus or averaging algorithms had a similar guarantee of polynomial-time convergence in the presence of dynamically changing topologies. In section 9, we report on some numerical experiments illustrating the advantages of two of our algorithms. Section 10 contains some brief concluding remarks.

2. The Agreement Algorithm. The "agreement algorithm" is an iterative procedure for the solution of the distributed consensus problem. It was introduced in [20] for the time-invariant case and in [57, 58] for the case of "asynchronous" and time-varying environments. We briefly review this algorithm and summarize the available convergence results.

Consider a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of nodes. Each node i starts with a scalar value $x_i(0)$; the vector with the values of all nodes at time t is denoted by $x(t) = (x_1(t), \ldots, x_n(t))$. The agreement algorithm updates $x(t)$ according to the equation $x(t+1) = A(t)x(t)$, or

$$x_i(t+1) = \sum_{j=1}^{n} a_{ij}(t)x_j(t),$$

where $A(t)$ is a nonnegative matrix with entries $a_{ij}(t)$. The row-sums of $A(t)$ are equal to 1, so that $A(t)$ is a stochastic matrix. In particular, $x_i(t+1)$ is a weighted average of the values $x_j(t)$ held by the nodes at time t.

We next state some conditions under which the agreement algorithm is guaranteed to converge.

ASSUMPTION 2.1. *There exists a positive constant α such that*
(a) $a_{ii}(t) \geq \alpha$ *for all i, t;*
(b) $a_{ij}(t) \in \{0\} \cup [\alpha, 1]$ *for all i, j, t;*
(c) $\sum_{j=1}^{n} a_{ij}(t) = 1$ *for all i, t.*

[1] Let f and g be two positive functions on the positive integers. We write $f(n) = O(g(n))$ (respectively, $f(n) = \Omega(g(n))$) if there exists a positive constant c and some n_0 such that $f(n) \leq cg(n)$ (respectively, $f(n) \geq cg(n)$) for all $n \geq n_0$. If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ both hold, we write $f(n) = \Theta(g(n))$.

Intuitively, whenever $a_{ij}(t) > 0$, node j communicates its current value $x_j(t)$ to node i. Each node i updates its own value by forming a weighted average of its own value and the values it has just received from other nodes. We represent the sequence of communications between nodes by a sequence $G(t) = (\mathcal{N}, \mathcal{E}(t))$ of directed graphs, where $(j,i) \in \mathcal{E}(t)$ if and only if $a_{ij}(t) > 0$. Note that $(i,i) \in \mathcal{E}(t)$ for all t, and this condition will remain in effect throughout the paper.

Our next assumption requires that, following an arbitrary time t and for any i, j, there is a sequence of communications through which node i will influence (directly or indirectly) the value held by node j.

ASSUMPTION 2.2 (connectivity). *For every $t \geq 0$, the graph $(\mathcal{N}, \cup_{s \geq t} \mathcal{E}(s))$ is strongly connected.*

Assumption 2.2 by itself is not sufficient to guarantee consensus (see Exercise 3.1 on page 517 of [5]). This motivates the following stronger version.

ASSUMPTION 2.3 (bounded interconnectivity times). *There is some B such that, for all k, the graph $\big(\mathcal{N}, \mathcal{E}(kB) \cup \mathcal{E}(kB+1) \cup \cdots \cup \mathcal{E}((k+1)B-1)\big)$ is strongly connected.*

We note various special cases of possible interest.

Time-invariant model. In this model, introduced by DeGroot [20], the set of arcs $\mathcal{E}(t)$ is the same for all t; furthermore, the matrix $A(t)$ is the same for all t. In this case, we are dealing with the iteration $x := Ax$, where A is a stochastic matrix; in particular, $x(t) = A^t x(0)$. Under Assumptions 2.1 and 2.2, A is the transition probability matrix of an irreducible and aperiodic Markov chain. Thus, A^t converges to a matrix, all of whose rows are equal to the (positive) vector $\pi = (\pi_1, \ldots, \pi_n)$ of steady-state probabilities of the Markov chain. Accordingly, we have $\lim_{t \to \infty} x_i(t) = \sum_{i=1}^n \pi_i x_i(0)$.

Bidirectional model. In this case, we have $(i,j) \in \mathcal{E}(t)$ if and only if $(j,i) \in \mathcal{E}(t)$, and we say that the graph G is *symmetric*. Intuitively, whenever i communicates with j, there is a simultaneous communication from j to i.

Equal-neighbor model. Here,

$$a_{ij}(t) = \begin{cases} 1/d_i(t) & \text{if } j \in \mathcal{N}_i(t), \\ 0 & \text{if } j \notin \mathcal{N}_i(t), \end{cases}$$

where $\mathcal{N}_i(t) = \{j \mid (j,i) \in \mathcal{E}(t)\}$ is the set of nodes j (including i) whose value is taken into account by i at time t, and $d_i(t)$ is its cardinality. This model is a linear version of a model considered by Vicsek et al. [59]. Note that here the constant α of Assumption 2.1 can be taken to be $1/n$.

THEOREM 2.4. *Under Assumptions 2.1 and 2.3, the agreement algorithm guarantees asymptotic consensus; that is, there exists some c (depending on $x(0)$ and on the sequence of graphs $G(\cdot)$) such that $\lim_{t \to \infty} x_i(t) = c$ for all i.*

Theorem 2.4 is presented in [58] and proved in [57], in a more general setting that allows for communication delays, under a slightly stronger version of Assumption 2.3; see also Chapter 7 of [5] and [58, 6] for extensions to the cases of communication delays and probabilistic dropping of packets. The above version of Assumption 2.3 was introduced in [35]. Under the additional assumption of a bidirectional model, the bounded interconnectivity time assumption is unnecessary, as established in [39, 9] for the bidirectional equal-neighbor model and in [32, 46] for the general case.

3. Averaging with the Agreement Algorithm in Fixed Networks.

In this section, as well as in sections 4–8, we assume that the network topology is fixed, i.e., $G(t) = G$ for all t, and known. We consider the time-invariant version, $x := Ax$, of the agreement algorithm and discuss various ways in which it can be used to solve the averaging problem. We show that an iteration $x := Ax$ that solves the consen-

sus problem can be used in a simple manner to provide a solution to the averaging problem as well.

3.1. Using a Doubly Stochastic Matrix. As remarked in section 2, with the time-invariant agreement algorithm $x := Ax$, we have

$$\lim_{t\to\infty} x_i(t) = \sum_{i=1}^{n} \pi_i x_i(0) \qquad \forall\, i, \tag{3.1}$$

where π_i is the steady-state probability of node i in the Markov chain associated with the stochastic matrix A. It follows that we obtain a solution to the averaging problem if and only if $\pi_i = 1/n$ for every i. Since π is a left eigenvector of A, with eigenvalue equal to 1, this requirement translates into the property $\mathbf{1}^T A = \mathbf{1}^T$, where $\mathbf{1}$ is the vector with all components equal to 1. Equivalently, the matrix A needs to be doubly stochastic. A particular choice of a doubly stochastic matrix has been proposed in [50] (see also [29]); this is discussed further in sections 7 and 9.

3.2. The Scaled Agreement Algorithm. Suppose that the graph G is fixed a priori and that there is a system designer or other central authority who chooses a stochastic matrix A offline, computes the associated steady-state probability vector (assumed unique and positive), and disseminates the value of $n\pi_i$ to each node i.

Suppose next that the nodes execute the agreement algorithm $\overline{x} := A\overline{x}$ using the matrix A, but with the initial value $x_i(0)$ of each node i replaced by

$$\overline{x}_i(0) = \frac{x_i(0)}{n\pi_i}. \tag{3.2}$$

Then the value $\overline{x}_i(t)$ of each node i converges to

$$\lim_{t\to\infty} \overline{x}_i(t) = \sum_{i=1}^{n} \pi_i \overline{x}_i(0) = \frac{1}{n}\sum_{i=1}^{n} x_i(0),$$

and we therefore have a valid averaging algorithm. This establishes that any (time-invariant) agreement algorithm for the consensus problem translates into an algorithm for the averaging problem as well. The following are two possible drawbacks of the scheme we have just described:

(a) If some of the $n\pi_i$ are very small, then some of the initial $\overline{x}_i(0)$ will be very large, which can lead to numerical difficulties [33].
(b) The algorithm requires some central coordination in order to choose A and compute π.

The algorithm provided in the next subsection provides a remedy for both of the above drawbacks.

3.3. Using Two Parallel Passes of the Agreement Algorithm. Given a fixed graph G, let A be the matrix that corresponds to the time-invariant, equal-neighbor, bidirectional model (see section 2 for definitions); in particular, if $(i,j) \in \mathcal{E}$, then $(j,i) \in \mathcal{E}$, and $a_{ij} = 1/d_i$, where d_i is the cardinality of $\mathcal{N}_i$. Under Assumptions 2.1 and 2.2, the stochastic matrix A is irreducible and aperiodic (because $a_{ii} > 0$ for every i). Let $E = \sum_{i=1}^{n} d_i$. It is easily verified that the vector π with components $\pi_i = d_i/E$ satisfies $\pi^T = \pi^T A$ and is therefore equal to the vector of steady-state probabilities of the associated Markov chain.

The following averaging algorithm employs two parallel runs of the agreement algorithm, with different, but locally determined, initial values.

ALGORITHM 3.1.
(a) *Each node i sets $y_i(0) = 1/d_i$ and $z_i(0) = x_i(0)/d_i$.*
(b) *The nodes run the agreement algorithms $y(t+1) = Ay(t)$ and $z(t+1) = Az(t)$.*
(c) *Each node sets $x_i(t) = z_i(t)/y_i(t)$.*
We have

$$\lim_{t\to\infty} y_i(t) = \sum_{i=1}^{n} \pi_i y_i(0) = \sum_{i=1}^{n} \frac{d_i}{E} \cdot \frac{1}{d_i} = \frac{n}{E}$$

and

$$\lim_{t\to\infty} z_i(t) = \sum_{i=1}^{n} \pi_i z_i(0) = \sum_{i=1}^{n} \frac{d_i}{E} \cdot \frac{x_i(0)}{d_i} = \frac{1}{E} \sum_{i=1}^{n} x_i(0).$$

This implies that

$$\lim_{t\to\infty} x_i(t) = \frac{1}{n} \sum_{i=1}^{n} x_i(0);$$

i.e., we have a valid averaging algorithm. Note that the iteration $y := Ay$ need not be repeated if the network remains unchanged and the averaging algorithm is to be executed again with different initial opinions. Finally, if n and E are known by all nodes, the iteration $y := Ay$ is unnecessary, and we could just set $y_i(t) = n/E$.

4. Definition of the Convergence Rate and the Convergence Time. The convergence rate of any of the algorithms discussed in section 3 is determined by the convergence rate of the matrix powers A^t. In this section, we give a definition of the convergence rate (and convergence time) and provide a tool for bounding the convergence rate. As should be apparent from the discussion in section 3, there is no reason to restrict to doubly stochastic matrices, or even to nonnegative matrices. We therefore start by specifying the class of matrices that we will be interested in.

Consider a matrix A with the following property: For every $x(0)$, the sequence generated by letting $x(t+1) = Ax(t)$ converges to $c\mathbf{1}$ for some scalar c. Such a matrix corresponds to a legitimate agreement algorithm and can be employed in the scheme of section 3.2 to obtain an averaging algorithm, as long as 1 is a simple eigenvalue of A, so that there exists a unique corresponding left eigenvector, denoted by π, which we assume to have nonzero entries. Because of the convergence property assumed above, all other eigenvalues must have magnitude less than 1. Note, however, that we allow A to have some negative entries.

Suppose that A has the above properties. Let $1 = \lambda_1, \lambda_2, \ldots, \lambda_n$, be the eigenvalues of A, sorted in order of decreasing magnitude. We also let X be the set of vectors of the form $c\mathbf{1}$, i.e., with equal components. Given such a matrix A, we define its *convergence rate*, ρ, by

$$\rho = \sup_{x(0)\notin X} \lim_{t\to\infty} \left(\frac{\|x(t) - x^*\|_2}{\|x(0) - x^*\|_2} \right)^{1/t}, \tag{4.1}$$

where x^* stands for $\lim_{t\to\infty} x(t)$.

We also define the *convergence time*, $T_n(\epsilon)$, by

$$T_n(\epsilon) = \min \left\{ \tau : \frac{\|x(t) - x^*\|_\infty}{\|x(0) - x^*\|_\infty} \leq \epsilon \ \ \forall\, t \geq \tau, \ \forall\, x(0) \notin X \right\}.$$

Although we use the infinity norm to define the convergence time, bounds for other norms can be easily obtained from our subsequent results, by using the equivalence of norms.

Under the above assumptions, a result from [60] states that

$$\rho = \max\{|\lambda_2|, |\lambda_n|\}.$$

To study ρ, therefore, we must develop techniques to bound the eigenvalues of the matrix A. To this end, we will be using the following result from [36]. We present here a slightly more general version and include a proof for completeness.

THEOREM 4.1. *Consider an $n \times n$ matrix A, and let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be its eigenvalues sorted in order of decreasing magnitude. Suppose that the following conditions hold:*

(a) *We have $\lambda_1 = 1$ and $A\mathbf{1} = \mathbf{1}$.*
(b) *There exists a positive vector π such that $\pi^T A = \pi^T$.*
(c) *For every i and j, we have $\pi_i a_{ij} = \pi_j a_{ji}$.*

Let

$$S = \left\{ x \;\middle|\; \sum_{i=1}^n \pi_i x_i = 0,\ \sum_{i=1}^n \pi_i x_i^2 = 1 \right\}.$$

Then all eigenvalues of A are real and

$$\lambda_2 = 1 - \frac{1}{2} \min_{x \in S} \sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} (x_i - x_j)^2. \tag{4.2}$$

In particular, for any vector y that satisfies $\sum_{i=1}^n \pi_i y_i = 0$, we have

$$\lambda_2 \geq 1 - \frac{\sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} (y_i - y_j)^2}{2 \sum_{i=1}^n \pi_i y_i^2}. \tag{4.3}$$

Proof. Let D be a diagonal matrix whose ith diagonal entry is π_i. Condition (c) yields $DA = A^T D$. We define the inner product $\langle \cdot, \cdot \rangle_\pi$ by $\langle x, y \rangle_\pi = x^T D y$. We then have

$$\langle x, Ay \rangle_\pi = x^T DAy = x^T A^T Dy = \langle Ax, y \rangle_\pi.$$

Therefore, A is self-adjoint with respect to this inner product, which implies that A has real eigenvalues.

Since the largest eigenvalue is 1, with an eigenvector of $\mathbf{1}$, we use the variational characterization of the eigenvalues of a self-adjoint matrix [34] to obtain

$$\begin{aligned}
\lambda_2 &= \max_{x \in S} \langle x, Ax \rangle_\pi \\
&= \max_{x \in S} \sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} x_i x_j \\
&= \frac{1}{2} \max_{x \in S} \sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} (x_i^2 + x_j^2 - (x_i - x_j)^2).
\end{aligned}$$

For $x \in S$, we have, using the assumption $\pi_i a_{ij} = \pi_j a_{ji}$,

$$\sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} (x_i^2 + x_j^2) = 2 \sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} x_i^2 = 2 \sum_{i=1}^n \pi_i x_i^2 = 2 \langle x, x \rangle_\pi = 2,$$

which yields

$$\lambda_2 = 1 - \frac{1}{2} \min_{x \in S} \sum_{i=1}^{n} \sum_{j=1}^{n} \pi_i a_{ij} (x_i - x_j)^2.$$

Finally, (4.3) follows from (4.2) by considering the vector

$$x_i = y_i \Big/ \sqrt{\left(\sum_{j=1}^{n} \pi_j y_j^2 \right)}. \qquad \square$$

Note that the bound in (4.3) does not change if we replace the vector y with αy for any $\alpha \neq 0$.

5. Convergence Time for the Equal-Neighbor, Time-Invariant, Bidirectional Model. For the equal-neighbor, time-invariant, bidirectional model, tight bounds on the convergence rate were derived in [36].

THEOREM 5.1 (see [36]). *Consider the equal-neighbor, time-invariant, bidirectional model on a connected graph with n nodes. The convergence rate satisfies*

$$\rho \leq 1 - \gamma_1 n^{-3},$$

where γ_1 is a constant independent of n or the particular graph. Moreover, there exists some $\gamma_2 > 0$ such that, for every positive integer n, there exists an n-node connected symmetric graph for which

$$\rho \geq 1 - \gamma_2 n^{-3}.$$

Theorem 5.1 is proved in [36] for the case of symmetric graphs without self-arcs. It is not hard to check that essentially the same proof holds when self-arcs are present, the only difference being in the values of the constants γ_1 and γ_2. This is intuitive because the effect of the self-arcs is essentially a "slowing down" of the associated Markov chain by a factor of at most 2, and therefore the convergence rate should stay the same.

Using some additional results on random walks, Theorem 5.1 leads to a tight bound (within a logarithmic factor) on the convergence time.

COROLLARY 5.2. *The convergence time for the equal-neighbor, time-invariant, bidirectional model on a connected graph on n nodes satisfies*[2]

$$T_n(\epsilon) = O(n^3 \log(n/\epsilon)).$$

Furthermore, for every positive integer n, there exists an n-node connected graph for which

$$T_n(\epsilon) = \Omega(n^3 \log(1/\epsilon)).$$

Proof. The matrix A is the transition probability matrix for a random walk on the given graph, where, given the current state i, the next state is equally likely to be

[2]Throughout, log will stand for the base-2 logarithm.

any of its neighbors (including i itself). Let $p_{ij}(t)$ be the (i,j)th entry of the matrix A^t. It is known (see Theorem 5.1 of [43]) that[3]

$$|p_{ij}(t) - \pi_j| \le \sqrt{\frac{d_j}{d_i}}\rho^t. \tag{5.1}$$

Since $1 \le d_i$ and $d_j \le n$, we have

$$|p_{ij}(t) - \pi_j| \le \sqrt{n}\rho^t$$

for all i, j, and t. Using Theorem 5.1, we obtain

$$|p_{ij}(t) - \pi_j| \le \sqrt{n}(1 - n^{-3})^t. \tag{5.2}$$

This implies that by taking $t = cn^3 \log(n/\epsilon)$, where c is a sufficiently large absolute constant, we will have $|p_{ij}(\tau) - \pi_j| \le \epsilon/n$ for all i, j, and $\tau \ge t$.

Let $A^* = \lim_{t\to\infty} A^t$, and let $x^* = \lim_{t\to\infty} A^t x(0)$. Note that $A^* x(0) = x^* = A^t x^* = A^* x^*$ for all t. Then, with t chosen as above,

$$\begin{aligned}\|x(t) - x^*\|_\infty &= \|A^t(x(0) - x^*)\|_\infty \\ &= \|(A^t - A^*)(x(0) - x^*)\|_\infty \\ &\le \|A^t - A^*\|_1 \cdot \|x(0) - x^*\|_\infty \\ &\le \epsilon\|x(0) - x^*\|_\infty.\end{aligned}$$

This establishes the upper bound on $T_n(\epsilon)$.

For the lower bound, note that for every $(i,j) \in \mathcal{E}$, we have $\pi_i a_{ij} = (d_i/E)(1/d_i) = 1/E$, so that condition (c) in Theorem 4.1 is satisfied. It follows that A has real eigenvalues. Let $x(0)$ be a (real) eigenvector of A corresponding to the eigenvalue with magnitude ρ. Then $\|x(t)\|_\infty = \|A^t x(0)\|_\infty = \rho^t \|x(0)\|_\infty$, which converges to zero, i.e., $x^* = 0$. We then have

$$\frac{\|x(t) - x^*\|_\infty}{\|x(0) - x^*\|_\infty} = \rho^t.$$

By the second part of Theorem 5.1, there exists a graph for which $\rho \ge 1 - \gamma n^{-3}$, leading to the inequality $T_n(\epsilon) \ge cn^3 \log(1/\epsilon)$ for some absolute constant c. □

The $\Omega(n^3)$ convergence time of this algorithm is not particularly attractive. In the next section, we explore possible improvements in the convergence time by using different choices for the matrix A.

6. Convergence Time for the Scaled Agreement Algorithm. In this section, we consider the scaled agreement algorithm introduced in section 3.2. As in [60], we assume the presence of a system designer who chooses the matrix A so as to obtain a favorable convergence rate, subject to the condition $a_{ij} = 0$ whenever $(i,j) \notin \mathcal{E}$. The latter condition is meant to represent the network topology through which the nodes are allowed to communicate. Our goal is to characterize the best possible convergence rate guarantee. We will see that the convergence rate can be brought arbitrarily close to zero. However, if we impose a certain "numerical stability" requirement, the

[3] Theorem 5.1 of [43] is proved for symmetric graphs without self-arcs. However, the proof does not use the absence of self-arcs, and when they are present the same proof yields the same result. We refer the reader to the derivation in [43, section 3.1] for details.

convergence time becomes $\Omega(n^2 \log(1/\epsilon))$ for a worst-case choice of the underlying graph. Furthermore, this worst-case lower bound applies even if we allow for matrices A in a much larger class than that considered in [60]. Finally, we will show that a convergence time of $O(n^2 \log(n/\epsilon))$ can be guaranteed in a simple manner, using a spanning tree.

6.1. Favorable but Impractical Convergence Rates. In this section, we show that given a connected symmetric directed graph $G = (\mathcal{N}, \mathcal{E})$, there is an elementary way of choosing a stochastic matrix A for which ρ is arbitrarily close to zero.

We say that a directed graph is a *bidirectional spanning tree* if (a) it is symmetric, (b) it contains all self-arcs (i, i), and (c) if we were to delete the self-arcs, ignore the orientation of the arcs, and remove duplicate arcs, we would be left with a spanning tree.

Without loss of generality, we assume that G is a bidirectional spanning tree; since G is symmetric and connected, this amounts to deleting some of its arcs or, equivalently, setting $a_{ij} = 0$ for all deleted arcs (i, j).

Pick an arbitrary node, denoted by r, and designate it as the root. Consider an arc (i, j) and suppose that j lies on the path from i to the root. Let $\overline{a}_{ij} = 1$ and $\overline{a}_{ji} = 0$. Finally, let $\overline{a}_{rr} = 1$, and let $\overline{a}_{ii} = 0$ for $i \neq r$. This corresponds to a Markov chain in which the state moves deterministically toward the root. We have $\overline{A}^t = e_r \mathbf{1}^T$ for all $t \geq n$, where e_i is the ith basis vector. It follows that $\rho = 0$ and $T_n(\epsilon) \leq n - 1$. However, this matrix $\overline{A}$ is not useful because the corresponding vector of steady-state probabilities has mostly zero entries, which prohibits the scaling discussed in section 3.2. Nevertheless, this is easily remedied by perturbing the matrix $\overline{A}$ as follows. For every $(i, j) \in \mathcal{E}$ with $i \neq j$ and $\overline{a}_{ij} = 0$, let $a_{ij} = \delta$, where δ is a small positive constant. Similarly, let us set all the self-coefficients (except the one at the root) to δ, i.e., $a_{ii} = \delta$ for all $i \neq r$. Finally, for every i, there exists a unique j for which $\overline{a}_{ij} = 1$. For any such pair (i, j), we set $a_{ij} = 1 - \sum_{k=1}^{n} a_{ik}$ (which is nonnegative as long as δ is chosen small enough). We have thus constructed a new matrix A_δ which corresponds to a Markov chain whose transition diagram is a bidirectional spanning tree. Since the convergence rate ρ is an eigenvalue of the iteration matrix, and eigenvalues are continuous functions of matrix elements, we see that, for the matrix A_δ, the convergence rate ρ can be made as small as desired by choosing δ sufficiently small. Finally, since A_δ is a positive matrix, the corresponding vector of steady-state probabilities is positive.

To summarize, by choosing δ suitably small, we can choose a (stochastic) matrix A_δ with an arbitrarily favorable convergence rate that allows the application of the scaled agreement algorithm of section 3.2. It can be shown that the convergence time is linear in the following sense: For every ϵ, there exists some δ such that, for the matrix A_δ, the corresponding convergence time, denoted by $T_n(\epsilon; \delta)$, satisfies $T_n(\epsilon; \delta) \leq n$. Indeed, this is an easy consequence of the facts $\lim_{\delta \to 0}(A_\delta^n - \overline{A}^n) = 0$ and $T_n(\epsilon'; 0) \leq n - 1$ for every $\epsilon' > 0$.

However, we note that as n gets larger, $n\pi_i$ may approach 0 at the nonroot nodes, exponentially fast. The implementation of the scaling in (3.2) will involve division by a number which quickly approaches 0, possibly leading to numerical difficulties. Thus, the resulting averaging algorithm may be undesirable. Setting averaging aside, the agreement algorithm based on A_δ, with δ small, is also undesirable; i.e., despite its favorable convergence rate, the final value on which consensus is reached is approximately equal to the initial value $x_r(0)$ of the root node. Such a "dictatorial" solution runs contrary to the motivation behind consensus algorithms.

6.2. A Lower Bound. In order to avoid the numerical issues raised above, we will now impose a condition on the dominant (and positive) left eigenvector π of the matrix A. We require

$$n\pi_i \geq \frac{1}{C} \quad \forall\, i, \tag{6.1}$$

where C is a given constant with $C \geq 1$. This condition ensures that $n\pi_i$ does not approach 0 as n gets large, so that the initial conditions in the scaled agreement algorithm of section 3.2 are well behaved. Furthermore, in the context of consensus algorithms, condition (6.1) has an appealing interpretation: it requires that the initial value $x_i(0)$ of every node i have a nonnegligible impact on the final value $\lim_{t\to\infty} x_k(t)$ on which consensus is reached.[4] We will now show that, under the additional condition (6.1), there are graphs for which the convergence time is $\Omega(n^2 \log(1/\epsilon))$. The graph that we employ is a *line graph*, with arc set $\mathcal{E} = \{(i,j) \mid |i-j| \leq 1\}$.

THEOREM 6.1. *Consider an $n \times n$ matrix A such that $a_{ij} = 0$ whenever $|i-j| > 1$. Let $\lambda_1, \lambda_2, \ldots$ be its eigenvalues in order of decreasing modulus. Suppose that $\lambda_1 = 1$ and $A\mathbf{1} = 1$. Furthermore, suppose that there exists a vector π satisfying* (6.1) *such that $\pi^T A = \pi^T$. Then there exist absolute constants c_1 and c_2 such that*

$$\rho \geq 1 - c_1 \frac{C}{n^2}$$

and

$$T_n(\epsilon) \geq c_2 \frac{n^2}{C} \log\left(\frac{1}{\epsilon}\right).$$

Proof. If the entries of A were all nonnegative, we would be dealing with a birth-death Markov chain. Such a chain is reversible, i.e., satisfies the detailed balance equations $\pi_i a_{ij} = \pi_j a_{ji}$ (condition (c) in Theorem 4.1). In fact, the derivation of the detailed balance equations does not make use of nonnegativity; thus, detailed balance holds in our case as well.

Without loss of generality, we can assume that $\sum_{i=1}^n \pi_i = 1$. For $i = 1, \ldots, n$, let $y_i = i - \beta$, where β is chosen so that $\sum_{i=1}^n \pi_i y_i = 0$. We will make use of the inequality (4.3). Since $a_{ij} = 0$ whenever $|i-j| > 1$, we have

$$\sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} (y_i - y_j)^2 \leq \sum_{i=1}^n \sum_{j=1}^n \pi_i a_{ij} = 1. \tag{6.2}$$

Furthermore,

$$\sum_{i=1}^n \pi_i y_i^2 \geq \frac{1}{nC} \sum_{i=1}^n y_i^2 = \frac{1}{nC} \sum_{i=1}^n (i-\beta)^2 \geq \frac{1}{nC} \sum_{i=1}^n \left(i - \frac{n+1}{2}\right)^2 \geq \frac{n^2-1}{12C}. \tag{6.3}$$

The next-to-last inequality above is an instance of the general inequality $\mathbf{E}[(X-\beta)^2] \geq \mathrm{var}(X)$ applied to a discrete uniform random variable X. The last inequality follows from the well-known fact $\mathrm{var}(X) = (n^2-1)/12$. Using inequalities (4.3) and (6.2)–(6.3), we obtain the desired bound on λ_2 and, therefore, on ρ.

[4] In the case where A is the transition matrix of a reversible Markov chain, there is an additional interpretation. A reversible Markov chain may be viewed as a random walk on an undirected graph with edge-weights. Defining the degree of a vertex as the sum total of the weights incident upon it, the condition $n\pi_i \geq C$ is equivalent to requiring that each degree be lower bounded by a constant times the average degree.

For the bound on $T_n(\epsilon)$, we let $x(0)$ be a (real) eigenvector of A associated with the eigenvalue λ_2, and proceed as in the end of the proof of Corollary 5.2. ∎

Remark. Note that if the matrix A is as in the previous theorem, it is possible for the iteration $x(t+1) = Ax(t)$ to not converge at all. Indeed, nothing in the argument precludes the possibility that the smallest eigenvalue is -1, for example. Furthermore, the graph could also be disconnected, and convergence to consensus could fail. In both cases, the lower bounds of the theorem—derived by bounding the second largest eigenvalue—still hold, as the convergence rate and time are infinite.

6.3. Convergence Time for Spanning Trees. We finally show that an $O(n^2)$ convergence time guarantee is easily obtained by restricting to a spanning tree.

THEOREM 6.2. *Consider the equal-neighbor, time-invariant, bidirectional model on a bidirectional spanning tree. We have*

$$\rho \leq 1 - \frac{1}{3n^2}$$

and

$$T_n(\epsilon) = O\big(n^2 \log(n/\epsilon)\big).$$

Proof. In this context, we have $\pi_i = d_i/E$, where

$$E = \sum_{i=1}^{n} d_i = 2(n-1) + n < 3n. \tag{6.4}$$

(The factor of 2 arises because we have arcs in both directions; the additional term n corresponds to the self-arcs.) As in the proof of Theorem 6.1, the detailed balance conditions $\pi a_{ij} = \pi_j a_{ji}$ hold, and we can apply Theorem 4.1. Note that (4.2) can be rewritten in the form

$$\lambda_2 = 1 - \frac{1}{2} \min_{\sum_i^n d_i x_i = 0, \sum_i^n d_i x_i^2 = 1} \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2. \tag{6.5}$$

We use the methods of [36] to show that for trees, λ_2 can be upper bounded by $1 - 1/3n^2$. Indeed, suppose that x satisfies $\sum_i^n d_i x_i = 0$ and $\sum_i^n d_i x_i^2 = 1$, and let $x_{\max}$ be a component of x such that $|x_{\max}| = \max_i |x_i|$. Then, using (6.4),

$$1 = \sum_i d_i x_i^2 \leq 3n x_{\max}^2,$$

and it follows that $|x_{\max}| \geq 1/\sqrt{3n}$. Without loss of generality, assume that $x_{\max} > 0$ (otherwise, replace each x_i by $-x_i$). Since $\sum_i d_i x_i = 0$, there exists some i for which $x_i < 0$; let us denote such a negative x_i by x_{neg}. Then

$$\frac{1}{\sqrt{3n}} \leq x_{\max} - x_{\text{neg}} = (x_{\max} - x_{k_1}) + (x_{k_1} - x_{k_2}) + \cdots + (x_{k_{r-1}} - x_{\text{neg}}), \tag{6.6}$$

where $k_1, k_2, \ldots, k_{r-1}$ are the nodes on the path from $x_{\max}$ to x_{neg}. By the Cauchy–Schwarz inequality,

$$\frac{1}{3n} \leq \frac{n}{2} \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2. \tag{6.7}$$

(The factor of 1/2 in the right-hand side arises because the sum includes both terms $(x_{k_i} - x_{k_{i+1}})^2$ and $(x_{k_{i+1}} - x_{k_i})^2$.) Thus,

$$\sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2 \geq \frac{2}{3n^2},$$

which proves the bound for the second largest eigenvalue.

For the smallest eigenvalue, recall that $a_{ii} \geq 1/n$ for every i. It follows that the matrix A is of the form $(I/n) + Q$, where I is the identity matrix and Q is a nonnegative matrix whose row sums are equal to $1 - 1/n$. Thus, all of the eigenvalues of Q have magnitude bounded above by $1 - 1/n$, which implies that the smallest eigenvalue of Q is bounded below by $-1 + 1/n$. We conclude that λ_n, the smallest eigenvalue of $(I/n) + Q$, satisfies

$$\lambda_n \geq -1 + \frac{2}{n} \geq -1 + \frac{2}{n^3}.$$

For the bound on the convergence time, we proceed as in the proof of Corollary 5.2. Let $p_{ij}(t)$ be the (i,j)th entry of A^t. Then

$$|p_{ij}(t) - \pi_j| \leq \sqrt{n}\left(1 - \frac{1}{3}n^{-2}\right)^t.$$

For a suitable absolute constant c and for $t \geq cn^2 \log(n/\epsilon)$, we obtain $|p_{ij}(t) - \pi(j)| \leq \epsilon/n$. The rest of the proof of Corollary 5.2 holds unchanged. □

In light of the preceding theorem, we propose the following simple heuristic, with worst-case convergence time $O(n^2 \log(n/\epsilon))$, as an alternative to a more elaborate design of the matrix A.

ALGORITHM 6.3. We are given a symmetric graph G. We delete enough arcs to turn G into a bidirectional spanning tree and then carry out the equal-neighbor, time-invariant, bidirectional consensus algorithm, with initial value $x_i(0)/n\pi_i$ at node i.

We note that this heuristic satisfies the desired condition (6.1) with $C = 1/3$. Indeed, $E < 3n$ and $d_i \geq 1$, so that $n\pi_i = nd_i/E \geq 1/3$. Let us also remark that the $O(n^2 \log(n/\epsilon))$ bound (Theorem 6.2) on the convergence time of this heuristic is essentially tight (within a factor of $\log n$). Indeed, if the given graph is a line graph, then with our heuristic we have $n\pi_i = nd_i/E \geq 1/2$, and Theorem 6.1 provides an $\Omega(n^2 \log(1/\epsilon))$ lower bound.

7. Convergence Time When Using a Doubly Stochastic Matrix. We provide here a brief comparison of our methods with the following two methods proposed in the literature that rely on doubly stochastic matrices. Recall that doubly stochastic matrices give rise directly to an averaging algorithm, without the need for scaling the initial values.

(a) Reference [60] considers the case where the graph G is given and studies the problem of choosing a doubly stochastic matrix A for which the convergence rate ρ is smallest. In order to obtain a tractable (semidefinite programming) formulation, this reference imposes the further restriction that A be symmetric. For a doubly stochastic matrix, we have $\pi_i = 1/n$ for all i, so that condition (6.1) holds with $C = 1$. According to Theorem 6.1, there exists a sequence of graphs for which we have $T_n(\epsilon) = \Omega(n^2 \log(1/\epsilon))$. We conclude that, despite the sophistication of this method, its worst-case guarantee is no

better (ignoring the $\log n$ factor) than the simple heuristic we have proposed (Algorithm 6.3). On the other hand, for particular graphs, the design method of [60] may yield better convergence times.

(b) The following method was proposed in [50]. The nodes first agree on some value $\epsilon \in (0, 1/\max_i d_i)$. (This is easily accomplished in a distributed manner.) Then the nodes iterate according to the equation

$$x_i(t+1) = (1 - \epsilon d_i)x_i(t) + \epsilon \sum_{j \in \mathcal{N}(i) \setminus \{i\}}^{n} x_j(t). \tag{7.1}$$

Assuming a connected graph, the iteration converges to consensus (this is a special case of Theorem 2.4). Furthermore, this iteration preserves the sum $\sum_{i=1}^{n} x_i(t)$. Equivalently, the corresponding matrix A is doubly stochastic, as is required in order to have an averaging algorithm.

This algorithm has the disadvantage of uniformly small stepsizes. If many of the nodes have degrees of the order of n, there is no significant theoretical difference between this approach and our Algorithm 3.1, as both have effective stepsizes of order $1/n$. On the other hand, if only a few nodes have large degrees, then the algorithm in [50] will force *all* the nodes to take small steps. This drawback is avoided by our Algorithms 3.1 (section 3.3) and 6.3 (section 6.3). A comparison of the method of [50] with Algorithm 3.1 is carried out, through simulation experiments, in section 9.

8. Averaging with Dynamic Topologies. In this section, we turn our attention to the more challenging case where communications are bidirectional but the network topology changes dynamically. Averaging algorithms for such a context were considered previously in [44, 45].

As should be clear from the previous sections, consensus and averaging algorithms are intimately linked, with the agreement algorithm often providing a foundation for the development of an averaging algorithm. For this reason, we start by investigating the worst-case performance of the agreement algorithm in a dynamic environment. Unfortunately, as shown in section 8.1, its convergence time is not polynomially bounded, in general. Motivated by this negative result, we approach the averaging problem differently: we introduce an averaging algorithm based on "load balancing" ideas (section 8.2) and prove a polynomial upper bound on its convergence time (section 8.3).

8.1. Nonpolynomial Convergence Time for the Agreement Algorithm. We begin by formally defining the notion of "convergence time" for the agreement algorithm on dynamic graph sequences. Given a sequence of graphs $G(t)$ on n nodes such that Assumption 2.3 of section 2 is satisfied for some $B > 0$, and an initial condition $x(0)$, we define the convergence time $T_{G(\cdot)}(x(0), \epsilon)$ (for this particular graph sequence and initial condition) as the first time t after which each node is within an ϵ-neighborhood of the final consensus, i.e., $\|x(t\tau) - \lim_{s\to\infty} x(s)\|_\infty \leq \epsilon$ for all $\tau \geq t$. We then define the (worst-case) convergence time, $T_n(B, \epsilon)$, as the maximum value of $T_{G(\cdot)}(x(0), \epsilon)$ over all graph sequences $G(\cdot)$ on n nodes that satisfy Assumption 2.3 for that particular B, and over all initial conditions that satisfy $\|x(0)\|_\infty \leq 1$.

We focus our attention on the equal-neighbor version of the agreement algorithm. The result that follows shows that its convergence time is not bounded by a polynomial in n and B. In particular, if B is proportional to n, the convergence time increases faster than an exponential in n. We note that the upper bound in Theorem 8.1 is

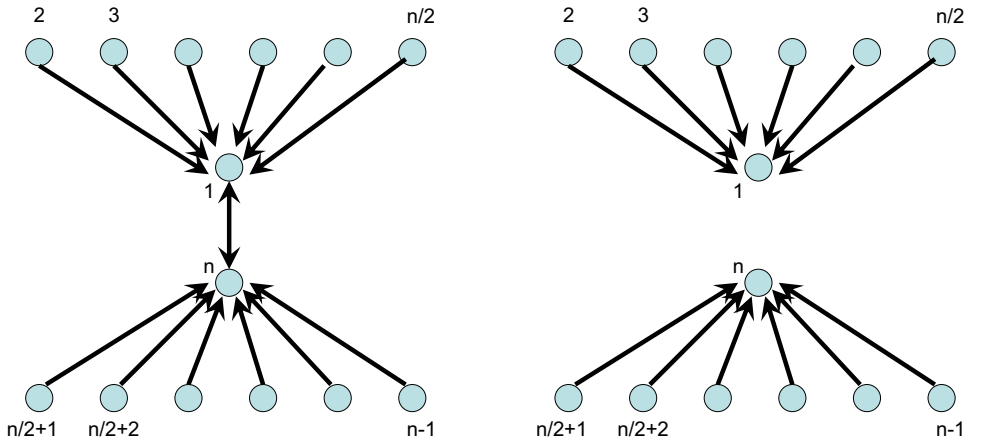

Fig. 8.1 *The diagram on the left is the graph $G(0)$. The diagram on the right is the graph $G(t)$ at times $t = 1, \ldots, B-2$. Self-arcs are not drawn but should be assumed present at every node.*

not a new result, but we include it for completeness and for comparison with the lower bound, together with a comment on its proof. Similar upper bounds were also provided recently in [12], under slightly different assumptions on the graph sequence $G(\cdot)$.

THEOREM 8.1. *For the equal-neighbor agreement algorithm, there exist positive constants c_1 and c_2 such that, for every n, B, and $1 > \epsilon > 0$,*

$$c_1 nB\Big(\frac{n-1}{2}\Big)^{B-1} \log\frac{1}{\epsilon} \le T_n(B,\epsilon) \le c_2 Bn^{nB} \log\frac{1}{\epsilon}. \tag{8.1}$$

Remark. It is possible to exhibit a sequence of *symmetric* graphs such that the equal-neighbor agreement algorithm takes an exponentially long time in n to converge. Such an example was communicated to the authors by Cao [8]. An exposition of that example as well as a simple proof of the exponentially large convergence time can be found in [53].

Proof. The upper bound follows by inspecting the proof of convergence of the agreement algorithm with the constant α of Assumption 2.1 set to $1/n$ (cf. [57, 6]).

We now prove the lower bound by exhibiting a sequence of graphs $G(t)$ and an initial vector $x(0)$, with $\|x(0)\|_\infty \le 1$, for which $T_{G(\cdot)}(x(0),\epsilon) \ge c_1 nB(n/2)^{B-1}\log(1/\epsilon)$. We assume that n is even and $n \ge 4$. The initial condition $x(0)$ is defined as $x_i(0) = 1$ for $i = 1, \ldots, n/2$ and $x_i(0) = -1$ for $i = n/2+1, \ldots, n$.

(i) The graph $G(0)$, used for the first iteration, is shown on the left-hand side of Figure 8.1.

(ii) For $t = 1, \ldots, B-2$, we perform an equal-neighbor iteration, each time using the graph $G(t)$ shown on the right-hand side of Figure 8.1.

(iii) Finally, at time $B-1$, the graph $G(B-1)$ consists of the complete graph over the nodes $\{1, \ldots, n/2\}$ and the complete graph over the nodes $\{n/2+1, \ldots, n\}$.

(iv) This sequence of B graphs is then repeated, i.e., $G(t+kB) = G(t)$ for every positive integer k.

It is easily seen that this sequence of graphs satisfies Assumption 2.3 and that convergence to consensus is guaranteed.

At the end of the first iteration, we have $x_i(1) = x_i(0)$ for $i \neq 1, n$, and

$$x_1(1) = \frac{(n/2)-1}{(n/2)+1} = 1 - \frac{4}{n+2}, \qquad x_n(1) = -x_1(1). \tag{8.2}$$

Consider now the evolution of $x_1(t)$ for $t = 1, \ldots, B-2$, and let $\alpha(t) = 1 - x_1(t)$. We have

$$x_1(t+1) = \frac{1 \cdot (1-\alpha(t)) + (n/2-1) \cdot 1}{n/2} = 1 - (2/n)\alpha(t),$$

so that $\alpha(t+1) = 2\alpha(t)/n$. From (8.2), $\alpha(1) = 4/(n+2)$, which implies that $\alpha(B-1) = (2/n)^{B-2}$, or

$$x_1(B-1) = 1 - \frac{4}{n+2}\Big(\frac{2}{n}\Big)^{B-2}.$$

By symmetry,

$$x_n(B-1) = -1 + \frac{4}{n+2}\Big(\frac{2}{n}\Big)^{B-2}.$$

Finally, at time $B-1$, we iterate on the complete graph over nodes $\{1, \ldots, n/2\}$ and the complete graph over nodes $\{n/2+1, \ldots, n\}$. For $i = 2, \ldots, n/2$, we have $x_i(B-1) = 1$, and we obtain

$$x_i(B-1) = \frac{1 \cdot \Big(\frac{n}{2} - 1\Big) + 1 - \frac{4}{n+2}\Big(\frac{2}{n}\Big)^{B-2}}{n/2} = 1 - \frac{4}{n+2}\Big(\frac{2}{n}\Big)^{B-1}.$$

Similarly, for $i = (n/2)+1, \ldots, n$, we obtain

$$x_i(B-1) = -1 + \frac{4}{n+2}\Big(\frac{2}{n}\Big)^{B-2}.$$

Thus,

$$\frac{|\max_i x_i(B) - \min_i x_i(B)|}{|\max_i x_i(0) - \min_i x_i(0)|} = 1 - \frac{4}{n+2} \cdot \Big(\frac{2}{n}\Big)^{B-1}.$$

Moreover, because $x(B)$ is simply a scaled version of $x(0)$, it is clear that by repeating this sequence of graphs, we will have

$$\frac{|\max_i x_i(kB) - \min_i x_i(kB)|}{|\max_i x_i(0) - \min_i x_i(0)|} = \Big(1 - \frac{4}{n+2} \cdot \Big(\frac{2}{n}\Big)^{B-1}\Big)^k.$$

This readily implies that

$$T_{G(\cdot)}(x(0), \epsilon) = \Omega\Big(nB\Big(\frac{n}{2}\Big)^{B-1} \log \frac{1}{\epsilon}\Big).$$

If n is odd, then $n' = n-1$ is even. We apply the same initial condition and graph sequence as above to nodes $\{1, \ldots, n'\}$. As for the additional node x_n, we let $x_n(0) = 0$ and make extra connections by connecting node n to nodes 1 and n' at

time 0 with a bidirectional link. By repeating the analysis above, it can be verified that

$$T_{G(\cdot)}(x(0),\epsilon) = \Omega\Big(nB\Big(\frac{n-1}{2}\Big)^{B-1}\log\frac{1}{\epsilon}\Big).$$

This concludes the proof. □

Both the upper and lower bounds in Theorem 8.1 display exponential growth of the convergence time as a function of B. It is unclear, however, which of the two terms, n^B or n^{nB}, better captures the behavior of $T_n(B,\epsilon)$.

8.2. Polynomial-Time Averaging in Dynamic Topologies. The algorithm we present here is a variation of an old *load-balancing* algorithm (see [19] and Chapter 7.3 of [5]). Intuitively, a collection of processors with different initial loads try to equalize their respective loads. As some of the highly loaded processors send some of their loads to their less loaded neighbors, the loads at different nodes tend to become equal. Similarly, at each step of our algorithm, each node offers some of its value to its neighbors and accepts or rejects such offers from its neighbors. Once an offer from i to j to send δ has been accepted, the updates $x_i := x_i - \delta$ and $x_j := x_j + \delta$ are executed.

We assume a time-varying sequence of graphs $G(t)$. We make only the following two assumptions on $G(\cdot)$: symmetry and bounded interconnectivity times (see section 2 for definitions). The symmetry assumption is natural if we consider, for example, communication between two nodes to be feasible whenever the nodes are within a certain distance of each other. The assumption of bounded interconnectivity times is necessary for an upper bound on the convergence time to exist (otherwise, we could insert infinitely many empty graphs $G(t)$, in which case convergence would be arbitrarily slow for any algorithm).

We next describe formally the steps that each node carries out at each time t. For definiteness, we refer to the node executing the steps below as node A. Moreover, the instructions below sometimes refer to the "neighbors" of node A; this always means nodes other than A that are neighbors at time t, when the step is being executed (since $G(t)$ can change with t, the set of neighbors of A can also change). Let $\mathcal{N}_i(t) = \{j \neq i : (i,j) \in \mathcal{E}(t)\}$. Note that this is a little different from the definition of $\mathcal{N}_i(t)$ in earlier sections, in that i is no longer considered to be a neighbor of itself.

ALGORITHM 8.2. If $\mathcal{N}_A(t)$ is empty, node A does nothing at time t. Otherwise, node A carries out the following steps:

1. Node A broadcasts its current value x_A to all of its neighbors (every j with $j \in \mathcal{N}_A(t)$).
2. Node A finds a neighboring node B with the smallest value: $x_B = \min\{x_j : j \in \mathcal{N}_A(t)\}$. If $x_A \leq x_B$, then node A does nothing further at this step. If $x_B < x_A$, then node A makes an offer of $(x_A - x_B)/2$ to node B.
3. If node A does not receive any offers, it does nothing further at this step. Otherwise, it sends an acceptance to the sender of the largest offer and a rejection to all the other senders. It updates the value of x_A by adding the value of the accepted offer.
4. If an acceptance arrives for the offer made by node A, node A updates x_A by subtracting the value of the offer.

For concreteness, we use $x_i(t)$ to denote the value possessed by node i at the *beginning* of the steps described above. Accordingly, the value possessed by node i at the end of the above steps will be $x_i(t+1)$. The algorithm we have specified clearly

keeps the value of $\sum_{i=1}^n x_i(t)$ constant. Furthermore, it is a valid averaging algorithm, as stated in Theorem 8.3 below. We do not provide a separate proof, because this result follows from the convergence time bounds in the next subsection.

THEOREM 8.3. *Suppose that each $G(t)$ is symmetric and that Assumption 2.3 (bounded interconnectivity times) holds. Then $\lim_{t\to\infty} x_i(t) = \frac{1}{n}\sum_{k=1}^n x_k(0)$ for all i.*

8.3. Convergence Time. We introduce the following potential function that quantifies the distance of the state $x(t)$ of the agents from the desired limit:

$$V(t) = \left\| x(t) - \frac{1}{n}\sum_{i=1}^n x_i(0)\mathbf{1} \right\|_2^2.$$

Intuitively, $V(t)$ measures the variance of the values at the different nodes. Given a sequence of graphs $G(t)$ on n nodes and an initial vector $x(0)$, we define the convergence time $T_{G(\cdot)}(x(0),\epsilon)$ as the first time t after which $V(\cdot)$ remains smaller than $\epsilon V(0)$:

$$T_{G(\cdot)}(x(0),\epsilon) = \min\left\{t \mid V(\tau) \le \epsilon V(0)\ \forall\ \tau \ge t\right\}.$$

We then define the (worst-case) convergence time, $T_n(B,\epsilon)$, as the maximum value of $T_{G(\cdot)}(x(0),\epsilon)$ over all graph sequences $G(\cdot)$ on n nodes that satisfy Assumption 2.3 for that particular B and over all initial conditions $x(0)$.

THEOREM 8.4. *There exists a constant $c > 0$ such that, for every n and every $\epsilon \in (0,1)$, we have*

$$T_n(B,\epsilon) \le cBn^3 \log\frac{1}{\epsilon}. \tag{8.3}$$

Remark. In later work [47] after this paper originally appeared, we proved the stronger result

$$T_n(B,\epsilon) \le cBn^2 \log\frac{1}{\epsilon}.$$

(The algorithm in [47] has some minor differences, but the same analysis applies to the algorithm as presented here.) Moreover, our subsequent paper [52] proved that for a natural class of update rules (namely, those for which $x(t+1)$ is a smooth function of $x(t)$), it is possible to establish the lower bound

$$T_n(1,\epsilon) \ge \frac{n^2}{30}\log\frac{1}{\epsilon}.$$

Proof. The proof is structured as follows. Without loss of generality, we assume that $\sum_{i=1}^n x_i(0) = 0$; this is possible because adding a constant to each x_i does not change the sizes of the offers or the acceptance decisions. We will show that $V(t)$ is nonincreasing in t and that

$$V((k+1)B) \le \left(1 - \frac{1}{2n^3}\right)V(kB) \tag{8.4}$$

for every nonnegative integer k. These two claims readily imply the desired result. To see this, note that if $V(t)$ decreases by a factor of $1-(1/2n^3)$ every B steps, then it decreases by a $\Theta(1)$ factor in Bn^3 steps. It follows that the time until $V(t)$ becomes less than $\epsilon V(0)$ is $O(Bn^3\log(1/\epsilon))$. Finally, since $V(t)$ is nonincreasing, $V(t)$ stays below $\epsilon V(0)$ thereafter.

We first show that $V(t)$ is nonincreasing. We argue that while rejected offers clearly do not change $V(t)$, each accepted offer at time t results in a decrease of $V(t+1)$. While this would be straightforward to establish if there were a single accepted offer, a more complicated argument is needed to account for the possibility of multiple offers being simultaneously accepted. We will show that we can view the changes at time t as a result of a series of sequentially accepted offers, each of which results in a smaller value of V.

Let us focus on a particular time t. We order the nodes from smallest to largest, so that $x_1(t) \leq x_2(t) \leq \cdots \leq x_n(t)$, breaking ties arbitrarily. Let $A_i(t)$ be the size of the offer accepted by node i at time t (if any). If the node accepted no offers at time t, set $A_i(t) = 0$. Furthermore, if $A_i(t) > 0$, let $\mathcal{A}_i(t)$ be the index of the node whose offer node i accepted.

Let us now break time t into n *periods*. The ith period involves the updates caused by node i accepting an offer from node $\mathcal{A}_i(t)$. In particular, node i performs the update $x_i(t) := x_i(t) + A_i(t)$ and node $\mathcal{A}_i(t)$ performs the update $x_{\mathcal{A}_i(t)}(t) := x_{\mathcal{A}_i(t)}(t) - A_i(t)$.

We note that every offer accepted at time t appears in some period in the above sequence. We next argue that each offer decreases V. This will complete the proof that $V(t)$ is nonincreasing in t.

Let us suppose that, in the ith period, node i accepts an offer from node $\mathcal{A}_i(t)$, which for simplicity we will denote by j. Because nodes only send offers to lower-valued nodes, the inequality $x_j > x_i$ must hold at the beginning of time t, before time period 1. We claim that this inequality continues to hold when the ith time period is reached. Indeed, x_j is unchanged during periods $1, \ldots, i-1$ (it can only send one offer, which was to x_i; and if it receives any offers, their effects will occur in period j, which is after period i). Moreover, while the value of x_i may have changed in periods $1, \ldots, i-1$, it cannot have increased (since i is not allowed to accept more than one offer at any given time t). Therefore, the inequality $x_j > x_i$ still holds at the beginning of the ith period.

During the ith period, a certain positive amount is transferred from node j to node i. Since the transfer takes place from a higher-valued node to a lower-valued one, it is easily checked that the value of $x_i^2 + x_j^2$ (which is the contribution of these two nodes to V) is reduced. To summarize, we have shown that we can serialize the offers accepted at time t in such a way that each accepted offer causes a reduction in V. It follows that $V(t)$ is nonincreasing.

We will now argue that at some time t in the interval $0, 1, \ldots, B-1$, there will be some update (acceptance of an offer) that reduces $V(t)$ by at least $V(0)/2n^3$. Without loss of generality, we assume $\max_i |x_i(0)| = 1$, so that all the values lie in the interval $[-1, +1]$. It follows that $V(0) \leq n$.

Since $\sum_{i=1}^n x_i(0) = 0$, it follows that $\min_i x_i(0) \leq 0$. Hence, the largest gap between any two consecutive $x_i(0)$ must be at least $1/n$. Thus, there exist some numbers a and b with $b - a \geq 1/n$, and the set of nodes can be partitioned into two disjoint subsets S^- and S_+ such that $x_i(0) \leq a$ for all $i \in S_-$, and $x_i(0) \geq b$ for all $i \in S_+$. By Assumption 2.3, the graph with arcs $\bigcup_{s=0,\ldots,B-1} \mathcal{E}(s)$ is connected. Thus, there exists a first time $\tau \in \{0, 1, \ldots, B-1\}$ at which there is a communication between some node $i \in S_-$ and some node $j \in S_+$, resulting in an offer from j to i. Up until that time, nodes in S_- have not interacted with nodes in S_+. It follows that $x_k(\tau) \leq a$ for all $k \in S_-$ and $x_k(\tau) \geq b$ for all $k \in S_+$. In particular, $x_i(\tau) \leq a$ and $x_j(\tau) \geq b$. There are two possibilities: either i accepts the offer from j, or i accepts some higher offer from some other node in S_+. In either case, we conclude that there is a first time $\tau \leq B-1$ at which a node in S_- accepts an offer from a node in S_+.

Let us use plain x_i and x_j for the values at nodes i and j, respectively, at the beginning of period i of time τ. At the end of that period, the value at both nodes is equal to $(x_i + x_j)/2$. Thus, the Lyapunov function V decreases by

$$x_i^2 + x_j^2 - 2\Big(\frac{x_i + x_j}{2}\Big)^2 = \frac{1}{2}(x_i - x_j)^2 \geq \frac{1}{2}(b-a)^2 \geq \frac{1}{2n^2}.$$

At every other time and period, V is nonincreasing, as shown earlier. Thus, using the inequality $V(0) \leq n$,

$$V(B) \leq V(0) - \frac{1}{2n^2} \leq V(0)\Big(1 - \frac{1}{2n^3}\Big).$$

By repeating this argument over the interval $kB, \ldots, (k+1)B$, instead of the interval $0, \ldots, B$, we establish (8.4), which concludes the proof. □

Remark. The thesis [51] provides a variation of this averaging scheme with the additional property that every node averages with at most one neighbor per round. Averaging algorithms with this property were explored in [40].

9. Simulations. We have proposed three new algorithms for the distributed consensus and averaging problems. For one of them, namely, the spanning tree heuristic of section 6.3 (Algorithm 6.3), the theoretical performance has been characterized completely—see Theorem 6.2 and the discussion at the end of section 6.3. In this section, we provide simulation results for the remaining two algorithms.

9.1. Averaging in Fixed Networks with Two Passes of the Agreement Algorithm. In section 3.3, we proposed a method for averaging in fixed graphs, based on two parallel executions of the agreement algorithm (Algorithm 3.1). We speculated in section 7 that the presence of a small number of high-degree nodes would make the performance of our algorithm attractive relative to the algorithm of [50], which uses a stepsize proportional to the inverse of the largest degree. (Our implementation used a stepsize of $\epsilon = 1/2d_{\max}$.) Figure 9.1 presents simulation results for the two algorithms.

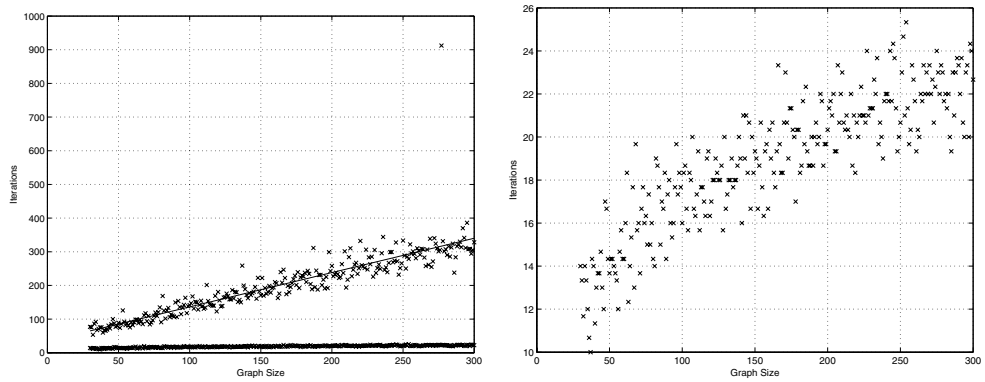

Fig. 9.1 *On the left: Comparison of averaging algorithms on a geometric random graph. The top line corresponds to the algorithm of* [50]*, and the bottom line (close to the horizontal axis) corresponds to using two parallel passes of the agreement algorithm (Algorithm* 3.1*). On the right: A blow-up of the performance of the agreement algorithm.*

In each simulation, we first generate geometric random graph $G(n, r)$ by placing nodes randomly in $[0, 1]^2$ and connecting two nodes if they are at most r apart. We choose $r = \Theta(\sqrt{\log n/n})$, which is a standard choice for modeling wireless networks (cf. [23]).

We then change the random graph $G(n, r)$ by choosing n_d nodes at random ($n_d = 10$ in both parts of Figure 9.1) and adding edges randomly to make the degree of these nodes linear in n; this is done by considering all possible edges incident to at least one node in n_d; each such edge is inserted independently with probability 1/3. We run the algorithm with random starting values, uniformly distributed in $[0, 1]$, until the largest deviation from the mean is at most $\epsilon = 10^{-3}$.

Each outcome recorded in Figure 9.1 (for different values of n) is the average of three runs. We conclude that for such graphs, the convergence time of the algorithm in [50] grows considerably faster than the one proposed in this paper.

9.2. Averaging in Time-Varying Random Graphs. We report here on simulations involving the load-balancing algorithm (Algorithm 8.2) on time-varying random graphs. In contrast to our previous simulations on a static geometric graph, we test two time-varying models.

In both models, we select our initial vector $x(0)$ by choosing each component independently as a uniform random variable over $[0, 1]$. In our first model, at each time t, we independently generate an Erdös–Renyi [25] random graph $G(t) = G(c, n)$ with $c = 3/4$. In the second model, at each time step we independently generate a geometric random graph with $G(n, r)$ with $r = \sqrt{\log n/n}$. In both models, if the largest deviation from the mean is at most $\epsilon = 10^{-3}$, we stop; otherwise, we perform another iteration of the load-balancing algorithm.

The results are summarized in Figure 9.2, where again each point represents the average of three runs. We conclude that in these random models, only a sublinear number of iterations appears to be needed.

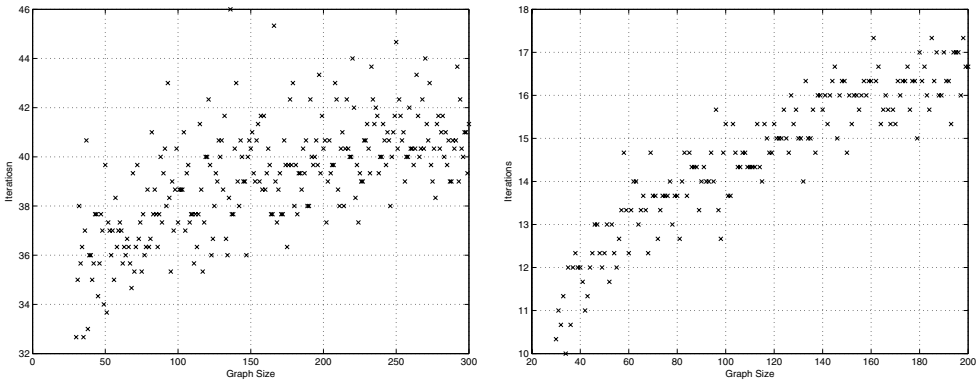

Fig. 9.2 *On the left: Averaging in time-varying Erdös–Renyi random graphs with the load balancing algorithm. Here, $c = 3/4$ at each time t. On the right: Averaging in time-varying geometric random graphs with the load-balancing algorithm. Here, $r = \sqrt{\log n/n}$.*

10. Concluding Remarks. In this paper we have considered a variety of consensus and averaging algorithms and studied their convergence rates. While our discussion was focused on averaging algorithms, several of our results pertain to the closely related consensus problem.

For the case of a fixed topology, we showed that averaging algorithms are easy to construct by using two parallel executions of the agreement algorithm for the consensus problem. We also saw that a reasonable performance guarantee can be obtained by using the equal-neighbor agreement algorithm on a spanning tree, as opposed to a more sophisticated design.

For the case of a fixed topology, the choice of different algorithms is not a purely mathematical issue; one must also take into account the extent to which one is able to design the algorithm offline and provide suitable instructions to each node. After all, if the nodes are able to set up a spanning tree, there are simple distributed algorithms, involving two sweeps along the tree in opposite directions, with which the sum of their initial values can be computed and disseminated [5], thus eliminating the need for an iterative algorithm. On the other hand, in less structured environments, with the possibility of occasional changes in the system topology, iterative algorithms can be more resilient. For example, the equal-neighbor agreement algorithm adjusts itself naturally when the topology changes.

In the face of a changing topology (possibly at each time step), the agreement algorithm continues to work properly under minimal assumptions (see Theorem 2.4). On the other hand, its worst-case convergence time may suffer severely (cf. section 8.1). Furthermore, it is not apparent how to modify the agreement algorithm and obtain an averaging algorithm without sacrificing linearity and/or allowing some additional memory at the nodes. In section 8, we introduced an averaging algorithm that is nonlinear but leads to a rather favorable (and, in particular, polynomial) convergence time bound. In view of the favorable performance observed in our simulation results, it would also be interesting to characterize the average performance of this algorithm under a probabilistic mechanism for generating the graphs $G(t)$, similar to the one in our simulations.

Something to notice about Algorithm 8.2 is that it requires the topology to remain fixed during the exchange of offers and acceptances/rejections that happens at each step. On the other hand, without such an assumption, or without introducing a much larger memory at each node (which would allow for flooding of individual values), an averaging algorithm may well turn out to be impossible.

REFERENCES

[1] D. Acemoglu, A. Ozdaglar, and A. ParandehGheibi, *Spread of (mis)information in social networks*, Games Econom. Behav., 70 (2010), pp. 194–227.

[2] D. Angeli and P.-A. Bliman, *Tight estimates for convergence of some non-stationary consensus algorithms*, Systems Control Lett., 57 (2008), pp. 996–1004.

[3] D. Angeli and P.-A. Bliman, *Convergence speed of unsteady distributed consensus: Decay estimate along the settling spanning-trees*, SIAM J. Control Optim., 48 (2009), pp. 1–32.

[4] P. Baroah, *Estimation and Control with Relative Measurements: Algorithms and Scaling Laws*, Ph.D. Thesis, University of California at Santa Barbara, 2007.

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

[6] 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 (CDC-ECC'05), Seville, Spain, 2005, pp. 2996–3000.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, *Randomized gossip algorithms*, IEEE Trans. Inform. Theory, 52 (2006), pp. 2508–2530.

[8] M. Cao, *Personal communication.*

[9] M. Cao, A. S. Morse, and B. D. O. Anderson, *Coordination of an asynchronous, multi-agent system via averaging*, in Proceedings of the 16th International Federation of Automatic Control World Congress (IFAC), Prague, Czech Republic, 2005.

[10] M. Cao, A. S. Morse, and B. D. O. Anderson, *Reaching a consensus in a dynamically changing environment: A graphical approach*, SIAM J. Control Optim., 47 (2008), pp. 575–600.
[11] M. Cao, A. S. Morse, and B. D. O. Anderson, *Reaching a consensus in a dynamically changing environment: Convergence rates, measurement delays, and asynchronous events*, SIAM J. Control Optim., 47 (2008), pp. 601–623.
[12] M. Cao, D. A. Spielman, and A. S. Morse, *A lower bound on convergence of a distributed network consensus algorithm*, in Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05), Seville, Spain, 2005, pp. 2356–2361.
[13] R. Carli, F. Bullo, and S. Zampieri, *Quantized average consensus via dynamic coding/decoding schemes*, Internat. J. Robust Nonlinear Control, 20 (2010), pp. 156–175.
[14] R. Carli, F. Fagnani, P. Frasca, T. Taylor, and S. Zampieri, *Average consensus on networks with transmission noise or quantization*, in Proceedings of the European Control Conference, Kos, Greece, 2007, pp. 1852–1857.
[15] B. Chazelle, *Natural algorithms*, in Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2009, pp. 422–431.
[16] B. Chazelle, *The geometry of flocking*, in Proceedings of the 27th Annual Symposium on Computational Geometry (SOCG), 2010, pp. 19–28.
[17] B. Chazelle, *Analytical tools for natural algorithms*, in Proceedings of the First Conference on Innovations in Computer Science (ICS), 2010, pp. 32–41.
[18] J. Cortes, *Finite-time convergent gradient flows with applications to network consensus*, Automatica, 42 (2006), pp. 1993–2000.
[19] G. Cybenko, *Dynamic load balancing for distributed memory multiprocessors*, J. Parallel Distrib. Comput., 7 (1989), pp. 279–301.
[20] M. H. DeGroot, *Reaching a consensus*, J. Amer. Statist. Assoc., 69 (1974), pp. 118–121.
[21] J.-C Delvenne, R. Carli, and S. Zampieri, *Optimal strategies in the average consensus problem*, Systems Control Lett., 58 (2009), pp. 759–765.
[22] P. M. DeMarzo, D. Vayanos, and J. Zwiebel, *Persuasion bias, social influence, and unidimensional opinions*, Quart. J. Econom., 118 (2003), pp. 909–968.
[23] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, *Geographic gossip: Efficient aggregation for sensor networks*, in Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN), Nashville, TN, 2006, pp. 69–76.
[24] J. Duchi, A. Agarwal, and M. J. Wainwright, *Dual averaging for distributed optimization: Convergence analysis and network scaling*, IEEE Trans. Automat. Control, to appear.
[25] P. Erdös and A. Renyi, *On the evolution of random graphs*, Magyar Tud. Akad. Mat. Kutató Int. Közl., 5 (1960), pp. 17–61.
[26] L. Fang and P. Antsaklis, *On communication requirements for multi-agent consensus seeking*, in Proceedings of the Workshop on Networked Embedded Sensing and Control, Lecture Notes in Control and Inform. Sci. 331, Springer, Berlin, 2006, pp. 53–67.
[27] N. M. Freris and P. R. Kumar, *Fundamental limits on synchronization of affine clocks in networks,* in Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, 2007, pp. 921–926.
[28] A. Ganguli, S. Susca, S. Martinez, F. Bullo, and J. Cortes, *On collective motion in sensor networks: Sample problems and distributed algorithms*, in Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05), Seville, Spain, 2005, pp. 4239–4244.
[29] C. Gao, J. Cortés, and F. Bullo, *Notes on averaging over acyclic digraphs and discrete coverage control*, Automatica, 44 (2008), pp. 2120–2127.
[30] B. Golub and M.J. Jackson, *Naive learning in social networks and the wisdom of crowds*, Amer. Econom. J. Microeconom., 2 (2010), pp. 112–149.
[31] R. Hegselmann and U. Krause, *Opinion dynamics and bounded confidence: Models, analysis and simulation*, J. Artificial Soc. Social Simul., 5 (2002).
[32] J. M. Hendrickx and V. D. Blondel, *Convergence of linear and non-linear versions of Vicsek's model*, in Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS'06), Kyoto, Japan, pp. 1229–1240. Available online from http://www-ics.acs.i.kyoto-u.ac.jp/mtns06/papers/0156.pdf.
[33] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.
[34] R. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1990.
[35] A. Jadbabaie, J. Lin, and A. S. Morse, *Coordination of groups of mobile autonomous agents using nearest neighbor rules*, IEEE Trans. Automat. Control, 48 (2003), pp. 988–1001.

[36] H. J. Landau and A. M. Odlyzko, *Bounds for eigenvalues of certain stochastic matrices*, Linear Algebra Appl., 38 (1981), pp. 5–15.
[37] F. Lekien and N. E. Leonard, *Nonuniform coverage and cartograms*, SIAM J. Control Optim., 48 (2009), pp. 351–372.
[38] Q. Li and D. Rus, *Global clock synchronization in sensor networks*, in Proceedings of the 23rd Conference of the IEEE Comuter and Communications Societies (INFOCOM), 2004, pp. 564–574.
[39] S. Li and H. Wang, *Multi-Agent Coordination Using Nearest-Neighbor Rules: Revisiting the Vicsek Model*, preprint, 2004; available online from http://arxiv.org/abs/cs.MA/0407021.
[40] J. Liu, S. Mou, A. S. Morse, B. D. O. Anderson, and C. Yu, *Request-Based Gossiping*, preprint.
[41] Y. Liu and Y. R. Yang, *Reputation propagation and agreement in mobile ad hoc networks*, in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2003, pp. 1510–1515.
[42] J. Lorenz and D. A. Lorenz, *On conditions for convergence to consensus*, IEEE Trans. Automat. Control, 55 (2010), pp. 1651–1656.
[43] L. Lovász, *Random walks on graphs: A survey*, in Combinatorics, Paul Erdös Is Eighty, Vol. 2, D. Miklós, V. T. Sós, and T. Szõnyi, eds., János Bolyai Mathematical Society, Budapest, 1996, pp. 353–398.
[44] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, *Distributed averaging on asynchronous communication networks*, in Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05), Seville, Spain, 2005, pp. 7446–7451.
[45] C. Moallemi and B. Van Roy, *Consensus propagation*, IEEE Trans. Inform. Theory, 52 (2006), pp. 4753–4766.
[46] L. Moreau, *Consensus seeking in multi-agent systems using dynamically changing interaction topologies*, IEEE Trans. Automat. Control, 50 (2005), pp. 169–182.
[47] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, *On distributed averaging algorithms and quantization effects*, IEEE Trans. Automat. Control, 54 (2009), pp. 2506–2517.
[48] A. Nedic and A. Ozdaglar, *Distributed subgradient methods for multi-agent optimization,* IEEE Trans. Automat. Control, 54 (2009), pp. 48–61.
[49] R. Olfati-Saber, J. A. Fax, and R. M. Murray, *Consensus and cooperation in networked multi-agent systems*, Proc. IEEE, 95 (2007), pp. 215–233.
[50] R. Olfati-Saber and R. M. Murray, *Consensus problems in networks of agents with switching topology and time-delays*, IEEE Trans. Automat. Control, 49 (2004), pp. 1520–1533.
[51] A. Olshevsky, *Efficient Information Aggregation Strategies in Distributed Control and Signal Processing*, Ph.D. Thesis, Department of EECS, MIT, Cambridge, MA, 2010.
[52] A. Olshevsky and J. N. Tsitsiklis, *A lower bound on distributed averaging*, IEEE Trans. Automat. Control, to appear.
[53] A. Olshevsky and J. N. Tsitsiklis, *Degree Fluctuations and the Convergence Time of Consensus Algorithms*, preprint, 2011.
[54] S. S. Ram, A. Nedic, and V. V. Veeravalli, *Distributed stochastic subgradient projection algorithms for convex optimization*, J. Optim. Theory Appl., 147 (2010), pp. 516–545.
[55] L. Schenato and G. Gamba, *A distributed consensus protocol for clock synchronization in wireless sensor network*, in Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, 2007, pp. 2289–2294.
[56] M. Schwager, J.-J. Slotine, and D. Rus, *Consensus learning for distributed coverage control*, in Proceedings of International Conference on Robotics and Automation (ICRA), Pasadena, CA, 2008, pp. 1042–1048.
[57] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*, Ph.D. Thesis, Department of EECS, MIT, Cambridge, MA, 1984.
[58] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, *Distributed asynchronous deterministic and stochastic gradient optimization algorithms*, IEEE Trans. Automat. Control, 31 (1986), pp. 803–812.
[59] T. Vicsek, E. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet, *Novel type of phase transitions in a system of self-driven particles*, Phys. Rev. Lett., 75 (1995), pp. 1226–1229.
[60] L. Xiao and S. Boyd, *Fast linear iterations for distributed averaging*, Systems Control Lett., 53 (2004), pp. 65–78.