Convergence speed in distributed consensus and averaging

CONVERGENCE SPEED IN DISTRIBUTED CONSENSUS AND AVERAGING*

ALEX OLSHEVSKY† AND 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.

1. Introduction. Given a set of autonomous agents — which may be sensors, nodes of a communication network, cars, or unmanned aerial vehicles — the distributed *consensus* problem asks for a distributed algorithm that the agents can use to agree on an opinion (represented by a scalar or a vector), starting from different initial opinions among the agents, and in the presence of possibly severely restricted communications.

Algorithms that solve the distributed consensus problem provide the means by which networks of agents can be coordinated. Although each agent may have access to different local information, the agents can agree on a decision (e.g., on a common direction of motion, on the time to execute a move, etc.). Such synchronized behavior often been observed in biological systems [15].

The distributed consensus problem has historically appeared in many diverse areas, such as parallel computation [30, 31, 3], control theory [18, 28], and communication networks [24, 22]. Recently, the problem has attracted significant attention [18, 22, 2, 11, 24, 7, 14, 25, 26, 13, 8, 5, 1], motivated by new contexts and open problems in communications, sensor networks, and networked control theory. We briefly describe some more of the more recent applications.

Reputation management in ad hoc networks: It is often the case that the nodes of a wireless multi-hop 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: 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 [22], where a variation of one of the methods we examine here — the "agreement algorithm" — was used as a basis for an empirical investigation.

Sensor networks: A sensor network designed for detection or estimation needs to combine various measurements into a decision or into a single estimate. Distributed computation of this decision/estimate has the advantage of being fault-tolerant (network operation is not dependent on a small set of nodes) and self-organizing (network

*This research was supported by the National Science Foundation under a Graduate Research Fellowship and grants ECS-0312921 and ECS-0426453. A preliminary version of this paper was presented at the 45th IEEE Conference on Decision and Control, San Diego, USA, 2006.

†Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; alex_o@mit.edu

‡Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; jnt@mit.edu

functionality does not require constant supervision) [31, 2, 11].

Control of autonomous agents: It is often necessary to coordinate collections of autonomous agents (e.g., cars or UAVs). For example, one may wish for the agents to agree on a direction or speed. Even though the data related to the decision may be distributed through the network, it is usually desirable that the final decision depend on all the known data, even though most of them are unavailable at each node. A model motivated by such a context was empirically investigated in [32].

In this paper, we focus on a special case of the distributed consensus problem, the distributed *averaging* problem. Averaging algorithms guarantee that the final global value will be the exact average of the initial individual values. 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 the main contributions. In Section 2, we provide some background material, by reviewing the agreement algorithm of [30, 31] for the distributed consensus problem. In Sections 3-8, we consider the case of fixed graphs. In Section 3, we discuss three different ways that 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 provide a variational characterization of the convergence rate.

In Section 5, we use results from [23] 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 non-symmetric 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 our knowledge, none of the existing consensus or averaging algorithms in the literature, has 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 [10] for the time-invariant case, and in [30, 31] for the case of "asynchronous" and time-

[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 $f(n) = \Theta(g(n))$.

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.

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, in p. 517 of [3]). 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 [10], 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 to 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. [32]. Note that here the constant α of Assumption 2.1 can be take 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 [31] and proved in [30], in a more general setting that allows for communication delays, under a slightly stronger version of Assumption 2.3; see also Ch. 7 of [3], and [31, 4] for extensions to the cases of communication delays and probabilistic dropping of packets. The above version of Assumption 2.3 was introduced in [18]. Under the additional assumption of a bidirectional model, the bounded interconnectivity time assumption is unnecessary, as established in [20, 6] for the bidirectional equal-neighbor model, and in [17, 25] 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 that it can be used to solve the averaging problem. We show that an iteration $x := Ax$ that solves the consensus 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 $\pi_i = 1/n$ or every i. Since π is a left eigenvector of A, with eigenvalue equal to 1, this requirement translates to 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 [27] (see also [8]); it 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 to an algorithm for the averaging problem as well. There 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 [16].
(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 an eigenvalue of A with multiplicity 1, and the corresponding left eigenvector, denoted by π, has nonzero entries. Because of the above assumed convergence property, 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 maginitude. 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 $\lim_{t\to\infty} x(t)$. As is well known, we have $\rho = \max\{|\lambda_2|, |\lambda_n|\}$.

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

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

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

Under the above assumptions, a result from [33] states

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

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 [23]. 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 maginitude. 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 proves 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 (Chapter 7, Theorem 4.3 of [29]) 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

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

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, Eq. (4.3) follows from (4.2) by considering the vector $x_i = y_i / \sqrt{(\sum_{j=1}^{n} \pi_j y_j^2)}$. □

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

5. Convergence time for the algorithm of Section 3.3. For the equal neighbor, time-invariant, bidirectional model, tight bounds on the convergence rate were derived in [23].

THEOREM 5.1. *[23] 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. 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 [23] for the case of symmetric graphs without self-arcs. It is not hard to check that essentially the same proof goes through 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 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, symmetric 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 a 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 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 that (Theorem 5.1[3]of [21]),

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

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

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

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

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

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

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

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

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

[3]Theorem 5.1 of [21] 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 of Eq. (3.1) in [21] for details.

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 5.1 is satisfied. It follows that A has real eigenvalues. Let $x(0)$ be a (real) eigenvector of A corresponding to the eigenvalue ρ. Then, $x(t) = A^t x(0) = \rho^t x(0)$, 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 $\rho \geq 1 - \gamma n^{-3}$, leading to the inequality $T_n(\epsilon) \geq cn^3 \log(1/\epsilon)$, 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 [33], 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 aim 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 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 [33]. 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 (b) if we delete the self-arcs, ignore the orientation of the arcs and remove duplicate arcs, we are 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 $\overline{a}_{ii} = 0$ for $i \neq r$. This corresponds to a Markov chain in which the state moves deterministically towards 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 $T_n(\epsilon) \leq n$. 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. 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 picking δ suitably small, we can choose a (stochastic) matrix A_δ with arbitrarily favorable convergence rate, and which 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 $T_n(\epsilon';0) \leq n$ for every $\epsilon' > 0$[4].

However, note that as n gets larger, $n\pi_i$ may approach 0 at the non-root nodes. The implementation of the scaling in Eq. (3.2) will involve division by a number which 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: 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, and require

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

where C is a given constant with $C > 1$. 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 has a nonnegligible impact on the final value $\lim_{t\to\infty} x_k(t)$, on which consensus is reached[5].

We will now show that under the additional condition (6.1), there are graphs for which the convergence time is $\Omega(n^2 \log(1/\epsilon))$. One may wonder whether a better convergence time is possible by allowing some of the entries of A to be negative. As the following result shows, negative entries do not help. 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 $|i-j| > 1$, and such that the graph with edge set $\{(i,j) \in \mathcal{E} \mid a_{ij} \neq 0\}$ is connected. 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 Eq. (6.1) such*

[4]Indeed, it is easy to see that by suitably choosing the root, we can make sure that convergence time is at most $\lceil d(G)/2 \rceil$ where $d(G)$ is the diameter of the graph G defined as the largest distance between any two vertices.

[5]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 an vertex as the sum total of of the weights incident upon it, the condition $n\pi_i \geq C$ is equivalent to requiring that each degree is lower bounded by a constant times the average degree.

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

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

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,

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

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 the inequality (4.3) and Eqs. (6.2)-(6.3), we obtain the desired bound on ρ.

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 $x(t+1) = Ax(t)$ not to converge at all. Indeed, nothing in the argument precludes the possibility that the smallest eigenvalue is -1, for example. In such a case, the lower bounds of the theorem - derived based on 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 $E = \sum_{i=1}^n d_i = 2(n-1)+n < 3n$. (The factor of 2 is 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 Eq. (4.2) can be rewritten in the form

$$\lambda_2 = 1 - \frac{1}{2} \min_{\sum_i^n d_i x_i = 0, \sum_i^n d_i x_i^2 = 1} \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2. \tag{6.4}$$

We use the methods of [23] 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 such that $|x_{\max}| = \max_i |x_i|$. Then,

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

and it follows that $|x_{\max}| \ge 1/\sqrt{3n}$. Without loss of generality, assume that $x_{\max} > 0$ (else, 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,

$$\frac{1}{\sqrt{3n}} \le x_{\max} - x_{\text{neg}} = (x_{\max} - x_{k_1}) + (x_{k_1} - x_{k_2}) + \cdots + (x_{k_{r-1}} - x_{\text{neg}}), \tag{6.5}$$

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

$$\frac{1}{3n} \le \frac{n}{2} \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2. \tag{6.6}$$

(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,

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

which proves the bound for the second largest eigenvalue.

For the smallest eigenvalue, recall that $a_{ii} \ge 1/n$ 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

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

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,

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

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

Let us 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 $n\pi_i = nd_i/E \geq 2/3$, and Theorem 6.1 provides a $\Omega(n^2 \log(1/\epsilon))$ lower bound.

7. Convergence time when using a doubly stochastic matrix. We provide here a brief comparison of our methods with two methods that have been proposed in the literature, and which 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 [33] 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 [33] may yield better convergence times.

(b) The following method was proposed in [27]. 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 required in order to have an averaging algorithm.

This algorithm has the disadvantage of uniformly small step sizes. 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 step sizes of order of $1/n$. On the other hand, if only a small number of nodes have large degree, then the algorithm in [27] 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 [27] with Algorithm 3.1 is carried out, through simulation experiments, in Section 8.

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 have been considered previously in [24, 26].

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, even though it is an open question whether this is also the case when we restrict to symmetric graphs. 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 bound on its convergence time (Section 8.3).

8.1. Non-polynomial 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 when each node is within an ϵ-neighborhood of the final consensus, i.e., $\|x(t) - \lim_{t\to\infty} x(t)\|_\infty \leq \epsilon$. 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 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 not a new result, but we include it for completeness, and for comparison with the lower bound, together with a proof sketch. Similar upper bounds have also been provided recently in [7], 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 $1 > \epsilon > 0$,*

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

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 $1/n$ (c.f. [30, 4]).

We now prove the lower bound by exhibiting a sequence of graphs $G(t)$ and an initial vector $x(0)$, with $\|x(0)\|_\infty \leq 1$ for which $T_{G(\cdot)}(x(0), \epsilon) \geq c_1 nB(n/2)^{B-1} \log(1/\epsilon)$. We assume that n is even and $n \geq 4$. 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 in 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 in 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: $G(t + kB) = G(t)$ 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.

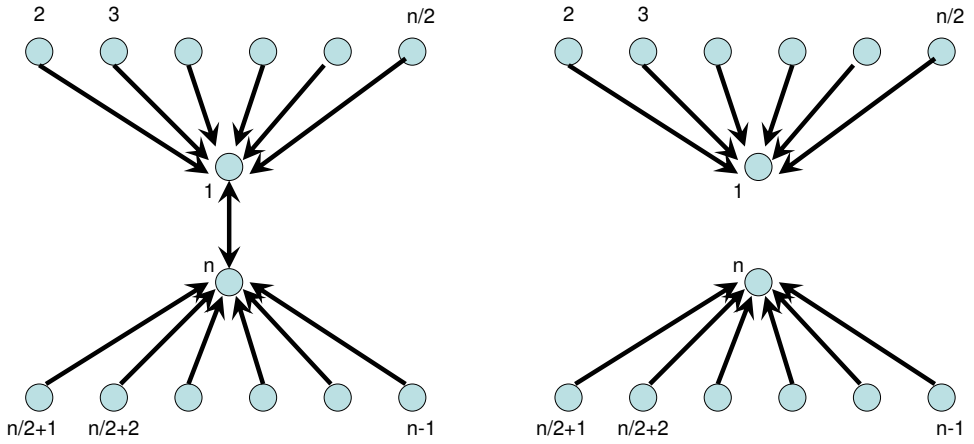

FIG. 8.1. *The diagram on the left is the graph $G(0)$. The diagram on the right is the graph $G(t)$ at times $t = 1, \ldots, B-2$. Self-arcs are not drawn but should be assumed present at every node.*

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 Eq. (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

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

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

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

This concludes the proof. □

Both the upper and lower bounds in Theorem 8.1 display an 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 [9] and Chapter 7.3 of [3]). 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 load 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 only make 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 is 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 a little different from the definition of $\mathcal{N}_i(t)$ in earlier sections, in that i is no longer considered a neighbor of itself.

ALGORITHM 8.2. If $\mathcal{N}_A(t)$ is empty, node A does nothing at time t. Else, 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 above described steps. 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 "Lyapunov" function that quantifies the distance of the state $x(t)$ of the agents form 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 $\epsilon V(0)$:

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

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 all initial conditions $x(0)$.

THEOREM 8.4. *There exists a constant $c > 0$ such that for every n and $1 > \epsilon > 0$, 2.3, we have*

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

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

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

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 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 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 we will for simplicity 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 $1/(2n^3)V(0)$. 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 Eq. (8.4), which concludes the proof. □

9. Simulations. We have proposed several 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 [27], which uses a step size proportional to the inverse of the largest degree. (Our implementation used a step size of $\epsilon = 1/2d_{\max}$.) Figure 9.1 presents simulation results for the two algorithms.

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 pick $r = \Theta(\sqrt{\log n/n})$, which is a standard choice for modeling wireless networks (c.f. [11]).

We then change the random graph $G(n, r)$ by picking n_d nodes at random ($n_d = 10$ in both figures) and adding edges randomly to make the degree of these nodes linear in n; this is done by making each edge incident to these nodes present 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 [27] grows considerably faster than the one proposed in this paper.

9.2. Averaging in time-varying Erdös-Renyi 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 static geometric graph, we test two time-varying models which simulate movement.

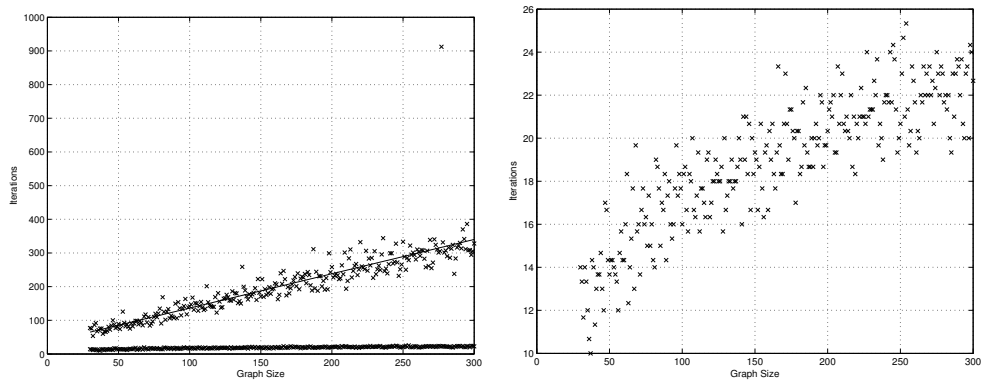

FIG. 9.1. *On the left: comparing averaging algorithms on a geometric random graph. The top line corresponds to the algorithm of [27], and the bottom line (close to the horizontal axis) corresponds to using two parallel passes of the agreement algorithm (Algorithm 3.1). On the right: a blowup of the performance of the agreement algorithm.*

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 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; else, 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.

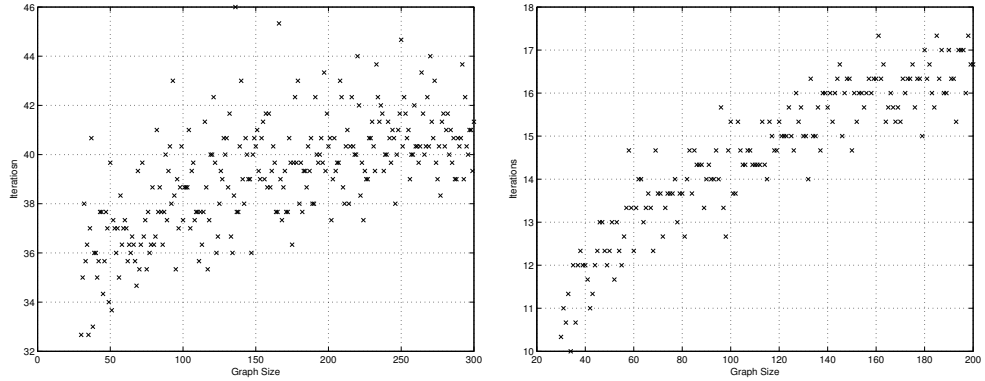

FIG. 9.2. *On the left: averaging in time-varying Erdös-Renyi random graphs with the load balancing algorithm. Here $c = 3/4$ at each time t. On the right: averaging in time-varying geometric random graphs with the load balancing algorithm. Here $r = \sqrt{\log n/n}$.*

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 [3], 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 (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, which 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 each 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] P.-A. Bliman, D. Angeli, "Convergence speed of unsteady distributed consensus: decay estimate along the settling spanning-trees," `http://arxiv.org/abs/math.OC/0610854`

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized Gossip Algorithms," joint issue of the IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking, June 2006, 52(6):2508-2530.

[3] D. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods,* Prentice Hall, 1989.

[4] 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, December 2005.

[5] J. Cortes, "Finite-time convergent gradient flows with applications to network consensus," *Automatica,* 42 (11) (2006), 1993-2000.

[6] M. Cao, A. S. Morse, and B. D. O. Anderson, "Coordination of an Asynchronous, Multi-Agent

System via Averaging," *Proceedings of The 16th International Federation of Automatic Control World Congress(IFAC),* Prague, Czech, July 2005.

[7] M. Cao, D. A. Spielman, and A. S. Morse, "A Lower Bound on Convergence of a Distributed Network Consensus Algorithm," *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, Seville, Spain, December 2005.

[8] C. Gao, J. Cortes, and F. Bullo, "Notes on Averaging over Acyclic Digraphs and Discrete Coverage Control," preprint.

[9] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *Journal of Parallel and Distributed Computing*, Vol. 7, No. 2, 1989.

[10] M. H. DeGroot, "Reaching a Consensus," *Journal of American Statistical Association*, Vol. 69, No. 345, 1974, pp. 118-121.

[11] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic Gossip: Efficient Aggregation for Sensor Networks," *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN), Nashville, TN,* 2006.

[12] P. Erdös and A. Renyi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci,* 5:17, 1960.

[13] L. Fang, P. Antsaklis, "On Communication Requirements for Multi-agent Consensus Seeking," *Proceedings of the Workshop on Networked Embedded Sensing and Control,* Notre Dame, IN, 2005.

[14] A. Ganguli, S. Susca, S. Martinez, F. Bullo, and J. Cortes, "On collective motion in sensor networks: sample problems and distributed algorithms," *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, pages 4239–4244, December 2005.

[15] D. Grunbaum, S. Viscido, and J. K. Parrish, "Extracting Interactive Control Algorithms from Group Dynamics of Schooling Fish," Cooperative Control, V. Kumar, N. Leonard, A. S. Morse (Eds.), pp.103-117, *Lecture Notes in Control and Information Sciences,* Springer, 2004.

[16] N. Higham, *Accuracy and Stability of Numerical Algorithms,* SIAM Press, 2002.

[17] J. M. Hendrickx and V. D. Blondel, "Convergence of different linear and non-linear Vicsek models," *Proceedings of 17th International Symposium on Mathematical Networks and Systems (MTNS 06*, Kyoto, Japan.

[18] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, Vol. 48, No. 3, pp. 988-1001, 2003.

[19] Y. Kim and M. Mesbahi, "On Maximizing the Second Smallest Eigenvalue of a State Dependent Graph Laplacian," *IEEE Transactions on Automatic Control*, Vol. 51, No. 1, January 2006.

[20] S. Li and H. Wang, "Multi-agent coordination using nearest-neighbor rules: revisiting the Vicsek model"; `http://arxiv.org/abs/cs.MA/0407021`.

[21] L. Lovász, "Random Walks on Graphs: A Survey," *Combinatorics, Paul Erdös is Eighty, Vol. 2*, D. Miklós, V. T. Sós, T. Szõnyi (Eds.), János Bolyai Mathematical Society, Budapest, 1996, pp. 353–398.

[22] Y. Liu and Y. R. Yang, "Reputation Propagation and Agreement in Wireless Ad Hoc Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC),* 2003.

[23] H. J. Landau and A. M. Odlyzko, "Bounds for Eigenvalues of Certain Stochastic Matrices," *Linear Algebra and Its Applications*, Vol. 38, pp. 5-15, 1981.

[24] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Distributed Averaging on Asynchronous Communication Networks," *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, Seville, Spain, December 2005.

[25] L. Moreau, "Consensus seeking in multi-agent systems using dynamically changing interaction topologies," *IEEE Transactions on Automatic Control,* Vol. 50, No. 2, pp.169-182, 2005.

[26] C. Moallemi and B. Van Roy, "Consensus Propagation," *IEEE Transactions on Information Theory,* Vol. 52, No. 11, pp. 4753-4766, 2006.

[27] R. Olfati-Saber and R. M. Murray,"Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," *IEEE Transacations on Automatic Control*, vol. 49(9), pp. 1520-1533, Sep. 2004.

[28] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE,* vol. 95, no. 1, pp. 215-233, Jan. 2007.

[29] S. Treil, *Linear Algebra Done Wrong;* `http://www.math.brown.edu/\~treil/papers/LADW/LADW.html`

[30] J. N. Tsitsiklis,"Problems in Decentralized Decision Making and Computation," Ph.D. Thesis, Department of EECS, MIT, 1984

[31] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms," *IEEE Transactions on Automatic Control,* Vol. 31, No. 9, pp. 803-812, 1986.

[32] T. Vicsek, E. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel Type of Phase Transitions in a System of Self-Driven Particles," *Physical Review Letters,* 75, pp. 1226 - 1229, 1995.

[33] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters,* Vol. 53, pp. 65-78, 2004.