Randomized Gossip Algorithms

Stephen Boyd, *Fellow, IEEE*, Arpita Ghosh, *Student Member, IEEE*, Balaji Prabhakar, *Member, IEEE*, and Devavrat Shah

***Abstract*—Motivated by applications to sensor, peer-to-peer, and *ad hoc* networks, we study distributed algorithms, also known as *gossip* algorithms, for exchanging information and for computing in an arbitrarily connected network of nodes. The topology of such networks changes continuously as new nodes join and old nodes leave the network. Algorithms for such networks need to be robust against changes in topology. Additionally, nodes in sensor networks operate under limited computational, communication, and energy resources. These constraints have motivated the design of "gossip" algorithms: schemes which distribute the computational burden and in which a node communicates with a randomly chosen neighbor.**

We analyze the averaging problem under the gossip constraint for an arbitrary network graph, and find that the averaging time of a gossip algorithm depends on the second largest eigenvalue of a doubly stochastic matrix characterizing the algorithm. Designing the fastest gossip algorithm corresponds to minimizing this eigenvalue, which is a semidefinite program (SDP). In general, SDPs cannot be solved in a distributed fashion; however, exploiting problem structure, we propose a distributed subgradient method that solves the optimization problem over the network.

The relation of averaging time to the second largest eigenvalue naturally relates it to the mixing time of a random walk with transition probabilities derived from the gossip algorithm. We use this connection to study the performance and scaling of gossip algorithms on two popular networks: Wireless Sensor Networks, which are modeled as Geometric Random Graphs, and the Internet graph under the so-called Preferential Connectivity (PC) model.

***Index Terms*—Distributed averaging, gossip, random walk, scaling laws, sensor networks, semidefinite programming.**

I. Introduction

THE advent of sensor, wireless *ad hoc* and peer-to-peer networks has necessitated the design of distributed and fault-tolerant computation and information exchange algorithms. This is mainly because such networks are constrained by the following operational characteristics: i) they may not have a centralized entity for facilitating computation, communication, and time-synchronization, ii) the network topology may not be completely known to the nodes of the network, iii) nodes may join or leave the network (even expire), so that the network topology itself may change, and iv) in the case of sensor networks, the computational power and energy resources may be very limited. These constraints motivate the design of simple decentralized algorithms for computation where each node exchanges information with only a few of its immediate neighbors in a time instance (or, a round). The goal in this setting is to design algorithms so that the desired computation and communication is done as quickly and efficiently as possible.

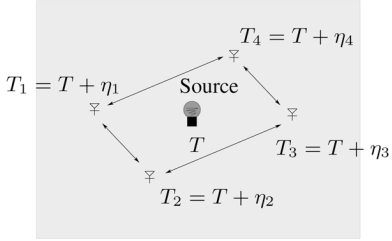

Fig. 1. Sensor nodes deployed to measure ambient temperature.

We study the problem of averaging as an instance of the distributed computation problem.[1] A toy example to motivate the averaging problem is sensing the temperature of some small region of space using a network of sensors. For example, in Fig. 1, sensors are deployed to measure the temperature T of a source. Sensor i, $i = 1,\ldots,4$, measures $T_i = T + \eta_i$, where the η_i are independent and identically distributed (i.i.d.), zero-mean Gaussian sensor noise variables. The unbiased, minimum mean-squared error (MMSE) estimate is the average $\hat{T} = \frac{\sum_i T_i}{4}$. Thus, to combat minor fluctuations in the ambient temperature and the noise in sensor readings, the nodes need to average their readings.

The problem of distributed averaging on a network comes up in many applications such as coordination of autonomous agents, estimation, and distributed data fusion on *ad hoc* networks, and decentralized optimization.[2] For one of the earliest references on distributed averaging on a network, see [45]. Fast distributed averaging algorithms are also important in other contexts; see Kempe *et al.* [22], for example. For an extensive body of related work, see [11], [16], [17], [20], [23], [25], [26], [29], [34], [42], [44], [46].

This paper undertakes an in-depth study of the design and analysis of gossip algorithms for averaging in an *arbitrarily connected* network of nodes. (By a gossip algorithm, we mean specifically an algorithm in which each node communicates

Manuscript received March 13, 2005; revised November 11, 2005. This work is supported in part by a Stanford Graduate Fellowship, and by C2S2, the MARCO Focus Center for Circuit and System Solution, under MARCO Contract 2003-CT-888.

S. Boyd, A. Ghosh, and B. Prabhakar are with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: fboyd@stanford.edu; arpitag@stanford.edu; balajig@stanford.edu).

D. Shah is with the LIDS, Departments of Elecrtrical Engineering and Computer Science, and ESD, the Massachusetts Institute of Technology, Cambridge, MA 02138 USA (e-mail: devavrat@mit.edu).

Communicated by M. Méderad, Guest Editor.

Digital Object Identifier 10.1109/TIT.2006.874516

[1]Preliminary versions of this paper appeared in [2]–[4].

[2]The theoretical framework developed in this paper is not restricted merely to averaging algorithms. It easily extends to the computation of other functions which can be computed via pairwise operations; e.g., the maximum, minimum, or product functions. It can also be extended for analyzing information exchange algorithms, although this extension is not as direct. For concreteness and for stating our results as precisely as possible, we shall consider averaging algorithms in the rest of the paper.

with no more than one neighbor in each time slot.) Given a graph G, we determine the averaging time T_{ave}, which is the time taken for the value at each node to be close to the average value (a more precise definition is given later). We find that the averaging time depends on the second largest eigenvalue of a doubly stochastic matrix characterizing the averaging algorithm: the smaller this eigenvalue, the faster the averaging algorithm. The *fastest* averaging algorithm is obtained by minimizing this eigenvalue over the set of allowed gossip algorithms on the graph. This minimization is shown to be a semidefinite program (SDP), which is a convex problem, and therefore can be solved efficiently to obtain the global optimum.

The averaging time T_{ave} is closely related to the mixing time T_{mix} of the random walk defined by the matrix that characterizes the algorithm. This means we can also study averaging algorithms by studying the mixing time of the corresponding random walk on the graph. The recent work of Boyd *et al.* [1] shows that the ratio of the mixing times of the natural random walk to the fastest mixing random walk can grow without bound as the number of nodes increases; correspondingly, therefore, the optimal averaging algorithm can perform arbitrarily better than the one based on the natural random walk. Thus, computing the optimal averaging algorithm is important: however, this involves solving a SDP, which requires a knowledge of the complete network topology. Surprisingly, we find that we can exploit problem structure to devise a distributed subgradient method to solve the SDP and obtain a near-optimal averaging algorithm, with only local communication.

Finally, we study the performance of gossip algorithms on two network graphs which are very important in practice: Geometric Random Graphs, which are used to model wireless sensor networks, and the Internet graph under the Preferential Connectivity model. We find that for geometric random graphs, the averaging time of the natural and the optimal averaging algorithms are of the same order. As remarked earlier, this need not be the case in a general graph.

We shall state our main results after setting out some notation and definitions in Section I.

A. Problem Formulation and Definitions

Consider a connected graph $G = (V, E)$, where the vertex set V contains n nodes and E is the edge set. The ith component of the vector $x(0) = [x_1(0), \ldots, x_n(0)]^T$ represents the initial value at node i. Let $x_{\text{ave}} = \sum_i x_i(0)/n$ be the average of the entries of $x(0)$. Our goal is to compute x_{ave} in a distributed manner.

- **Asynchronous time model**: Each node has a clock which ticks at the times of a rate 1 Poisson process. Thus, the inter-tick times at each node are rate 1 exponentials, independent across nodes and over time. Equivalently, this corresponds to a single clock ticking according to a rate n Poisson process at times Z_k, $k \geq 1$, where $\{Z_{k+1} - Z_k\}$ are i.i.d. exponentials of rate n. Let $I_k \in \{1, \ldots, n\}$ denote the node whose clock ticked at time Z_k. Clearly, the I_k are i.i.d. variables distributed uniformly over $\{1, \ldots, n\}$. We discretize time according to clock ticks since these are the only times at which the value of $x(\cdot)$ changes. Therefore, the interval $[Z_k, Z_{k+1})$ denotes the kth time-slot and, on average, there are n clock ticks per unit of absolute time. Lemma 1 states a precise translation of clock ticks into absolute time.

- **Synchronous time model**: In the synchronous time model, time is assumed to be slotted commonly across nodes. In each time slot, each node contacts one of its neighbors independently and (not necessarily uniformly) at random. Note that in this model all nodes communicate simultaneously, in contrast to the asynchronous model where only one node communicates at a given time. On the other hand, in both models each node contacts only one other node at a time.

  Previous work, notably that of [22], [29], considers the synchronous time model. The qualitative and quantitative conclusions are unaffected by the type of model; we start with the asynchronous time model for convenience, and then analyze the synchronous model and show that the same kind of results hold in this case as well.

- **Algorithm** $\mathcal{A}(P)$: We consider a particular class of time-invariant gossip algorithms, denoted by $\mathcal{A}$. An algorithm in this class is characterized by an $n \times n$ matrix $P = [P_{ij}]$ of nonnegative entries with the condition that $P_{ij} > 0$ only if $(i, j) \in E$. For technical reasons, we assume that P is a stochastic matrix with its largest eigenvalue equal to 1, and all remaining $n-1$ eigenvalues strictly less than 1 in magnitude. (Such a matrix can always be found if the underlying graph is connected and nonbipartite; we will assume that the network graph G satisfies these conditions for the remainder of the paper.) Depending on the time model, two types of algorithms arise: 1) asynchronous, and 2) synchronous. Next, we describe the asynchronous algorithm associated with P to explain the role of the matrix P in the algorithm. As we shall see, asynchronous algorithms are rather intuitive and easy to explain. We defer the description of the synchronous algorithm to Section III-C.

  The asynchronous algorithm associated with P, denoted by $\mathcal{A}(P)$, is described as follows: In the kth time slot, let node i's clock tick and let it contact some neighboring node j with probability P_{ij}. At this time, both nodes set their values equal to the average of their current values. Formally, let $x(k)$ denote the vector of values at the end of the time slot k. Then

$$x(k) = W(k)x(k-1) \tag{1}$$

  where with probability $\frac{1}{n}P_{ij}$ (the probability that the ith node's clock ticks is $1/n$, and the probability that it contacts node j is P_{ij}) the random matrix $W(k)$ is

$$W_{ij} = I - \frac{(e_i - e_j)(e_i - e_j)^T}{2} \tag{2}$$

  where $e_i = [0 \ \ldots \ 0 \ 1 \ 0 \ \ldots 0]^T$ is an $n \times 1$ unit vector with the ith component equal to 1.

- **Quantity of Interest**: Our interest is in determining the (absolute) time it takes for $x(t)$ to converge to $x_{\text{ave}}\mathbf{1}$, where $\mathbf{1}$ is the vector of all ones.

  Definition 1: For any $0 < \epsilon < 1$, the ϵ-averaging time of an algorithm $\mathcal{A}(P)$ is denoted by $T_{\text{ave}}(\epsilon, P)$, and is defined as

$$\sup_{x(0)} \inf \left\{ t : \Pr\left(\frac{\|x(t) - x_{\text{ave}}\mathbf{1}\|}{\|x(0)\|} \geq \epsilon \right) \leq \epsilon \right\} \tag{3}$$

  where $\|v\|$ denotes the l_2 norm of the vector v.

  Thus, the ϵ-averaging time is the smallest time it takes for $x(\cdot)$ to get within ϵ of $x_{\text{ave}}\mathbf{1}$ with high probability, regardless of the initial value $x(0)$.

The following lemma relates the number of clock ticks to absolute time. This relation allows us to use clock ticks instead of absolute time when we deal with asynchronous algorithms.

Lemma 1: For any $k \geq 1$, $E[Z_k] = k/n$. Further, for any $\delta > 0$

$$\Pr\left(\left| Z_k - \frac{k}{n} \right| \geq \frac{\delta k}{n} \right) \leq 2 \exp\left(-\frac{\delta^2 k}{2} \right). \tag{4}$$

Proof: By definition

$$E[Z_k] = \sum_{j=1}^{k} E[Z_j - Z_{j-1}] = \sum_{j=1}^{k} 1/n = k/n.$$

Equation (4) follows directly from Cramer's theorem (see [10, pp. 30 and 35]). □

As a consequence of Lemma 1, for $k \geq n$

$$Z_k = \frac{k}{n}\left(1 \pm \sqrt{\frac{2 \log n}{n}} \right)$$

with high probability (i.e., probability at least $1 - 1/n^2$). In this paper, all ϵ-averaging times are at least n. Hence, dividing the quantities measured in terms of the number of clock ticks by n gives the corresponding quantities when measured in absolute time (for an example, see Corollary 2).

B. Previous Results

A general lower bound for any graph G and any averaging algorithm was obtained in [29] in the synchronous setting. Their result is as follows.

Theorem 1: For any gossip algorithm on any graph G and for $0 < \epsilon < 0.5$, the ϵ-averaging time (in synchronous steps) is lower-bounded by $\Omega(\log n)$.

The recent work [22] studies the gossip-constrained averaging problem for the special case of the complete graph. A randomized gossiping algorithm is proposed which is shown to converge to the vector of averages on the complete graph. For a synchronous averaging algorithm, [22] obtain the following result.

Theorem 2: For a complete graph, there exists a gossip algorithm such that the $1/n$-averaging time of the algorithm is $O(\log n)$.

In Section III-C, we obtain a synchronous averaging algorithm which is simpler than the one described in [22], with ϵ-averaging time $\Theta(\log \epsilon^{-1})$ for the complete graph (from Corollary 3).

The problem of fast distributed averaging without the gossip constraint on an arbitrary graph is studied in [48]; here, the matrices $W(t)$ are constant, i.e., $W(t) = W$ for all t. It is shown that the problem of finding the (constant) W that converges fastest to $\mathbf{1}\mathbf{1}^T/n$ (where $\mathbf{1}\mathbf{1}^T$ is the matrix of all ones) can be written as a SDP (under a symmetry constraint), and can therefore be solved numerically.

Distributed averaging has also been studied in the context of distributed load balancing ([43]), where nodes (processors) exchange tokens in order to uniformly distribute tokens over all the processors in the network (the number of tokens is constrained to be integral, so exact averaging is not possible). An analysis based on Markov chains is used to obtain bounds on the time required to achieve averaging up to a certain accuracy. However, each iteration is governed either by a constant stochastic matrix, or a fixed sequence of matchings is considered. This differs from our work (in addition to the integral constraint) in that we consider an arbitrary sequence $W(t)$ drawn i.i.d. from some distribution, and try to characterize the properties the distribution must possess for convergence. Some other results on distributed averaging can be found in [6], [21], [30], [36], [37].

An interesting result regarding products of random matrices is found in [12]. The authors prove the following result on a sequence of iterations $x(t+1) = W(t)x(t)$, where the $W(t)$ belong to a finite set of paracontracting matrices (i.e., $W(t)x \neq x \Leftrightarrow \|W(t)x\| < \|x\|$). If $\mathcal{I}$ is the set of matrices W_i that appear infinitely often in the sequence $W(t)$, and for $i \in \mathcal{I}$, $\mathcal{H}(W_i)$ denotes the eigenspace of W_i associated with eigenvalue 1, then the sequence of vectors $x(t)$ has a limit $x^\star$ in $\cap_{i \in \mathcal{I}} \mathcal{H}(W_i)$. This result can be used to find conditions for convergence of distributed averaging algorithms.

Not much is known about good randomized gossip algorithms for averaging on arbitrary graphs. The algorithm of [22] is quite dependent on the fact that the underlying graph is a complete graph, and the general result of [29] is a nonconstructive lower bound.

C. Our Results

In this paper, we design and characterize the performance of averaging algorithms for arbitrary graphs for both the asynchronous and synchronous time models. The following result characterizes the averaging time of asynchronous algorithms.

Theorem 3: The averaging time $T_{\text{ave}}(\epsilon, P)$ of the asynchronous algorithm $\mathcal{A}(P)$ (in terms of number of clock ticks) is bounded as follows:

$$T_{\text{ave}}(\epsilon, P) \leq \frac{3 \log \epsilon^{-1}}{\log \lambda_2(W)^{-1}} \quad \text{and} \tag{5}$$

$$T_{\text{ave}}(\epsilon, P) \geq \frac{0.5 \log \epsilon^{-1}}{\log \lambda_2(W)^{-1}} \tag{6}$$

where

$$W \triangleq I - \frac{1}{2n}D + \frac{P+P^T}{2n} \tag{7}$$

and D is the diagonal matrix with entries

$$D_i = \sum_{j=1}^{n}[P_{ij} + P_{ji}].$$

Theorem 3 is proved in Section III, using results on convergence of moments that we derive in Section II.

For synchronous algorithms, the averaging time is characterized by Theorems 4 and 5, which are stated and proved in Section III-C. As the reader may notice, the statements of Theorem 3 and Theorems 4–5 are qualitatively the same.

The above tight characterization of the averaging time leads us to the formulation of the question of the fastest averaging algorithm. In Section IV, we show that the problem of finding the fastest averaging algorithm can be formulated as an SDP. In general, it is not possible to solve an SDP in a distributed fashion. However, we exploit the structure of the problem to propose a completely distributed algorithm, based on a subgradient method, that solves the optimization problem on the network. The algorithm and proof of convergence are found in Section IV-A.

Section V relates the averaging time of an algorithm on a graph G with the mixing time of an associated random walk on G. This is used in Section VI to study applications of our results in the context of two networks of practical interest: wireless networks and the Internet. The result for wireless networks involves bounding the mixing times of the natural and optimal random walks on the geometric random graph; these results are derived in Section VI-A. Finally, we conclude in Section VII.

II. Convergence of Moments

In this section, we will study the convergence of randomized gossip algorithms. We will not restrict ourselves here to any particular algorithm; but rather consider convergence of the iteration governed by a product of random matrices, each of which satisfies certain (gossip-based) constraints described below.

The vector of estimates is updated as

$$x(t+1) = W(t)x(t)$$

where each $W(t)$ must satisfy the following constraints imposed by the gossip criterion and the graph topology.

If nodes i and j are not connected by an edge, then $W_{ij}(t)$ must be zero. Further, since every node can communicate with only one of its neighbors per time slot, each column of $W(t)$ can have only one nonzero entry other than the diagonal entry.

The iteration intends to compute the average, and therefore must preserve sums: this means that $\mathbf{1}^T W(t) = \mathbf{1}^T$, where $\mathbf{1}$ denotes the vector of all ones. Also, the vector of averages must be a fixed point of the iteration, i.e., $W(t)\mathbf{1} = \mathbf{1}$.

We will consider matrices $W(t)$ drawn i.i.d. from some distribution on the set of nonnegative matrices satisfying the above constraints, and investigate the behavior of the estimate $x(t)$

$$\begin{aligned} x(t) &= W(t-1)W(t-2)\cdots W(0)x(0) \\ &= \phi(t-1)x(0). \end{aligned}$$

If $x(t)$ must converge to the vector of averages $\frac{\mathbf{1}\mathbf{1}^T}{n}x(0)$ for every initial condition $x(0)$, we must have

$$\lim_{t\to\infty}\phi(t) = \frac{\mathbf{1}\mathbf{1}^T}{n}. \tag{8}$$

A. Convergence in Expectation

Let the mean of the (i.i.d.) matrices $W(t)$ be denoted by $\overline{W}$. We have

$$E(\phi(t)) = \prod_{i=0}^{t} E(W(i)) = \overline{W}^t \tag{9}$$

so $\phi(t)$ converges in expectation to $\frac{\mathbf{1}\mathbf{1}^T}{n}$ if $\overline{W}^t \to \frac{\mathbf{1}\mathbf{1}^T}{n}$. The conditions on $\overline{W}$ for this to happen are stated in [48]; they are

$$\mathbf{1}^T\overline{W} = \mathbf{1}^T \tag{10}$$

$$\overline{W}\mathbf{1} = \mathbf{1} \tag{11}$$

$$\rho\left(\overline{W} - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) < 1 \tag{12}$$

where $\rho(\cdot)$ is the spectral radius of a matrix. The first two conditions will be automatically satisfied by $\overline{W}$, since it is the expected value of matrices each of which satisfies this property. Therefore, if we pick any distribution on the $W(t)$ whose mean satisfies (12), the sequence of estimates will converge in expected value to the vector of averages.

In fact, if $\overline{W}$ is invertible, by considering the martingale $\overline{W}^{-t}\phi(t)x(0)$, we can obtain almost sure convergence of $x(t)$ to $x_\infty = x_{\text{ave}}$. However, neither result tells us the *rate* at which $x(t)$ converges to x_∞.

B. Convergence of Second Moment

To obtain the rate of convergence of $x(t)$ to x_∞, we will investigate the rate at which the error $y(t) = x(t) - x_\infty$ converges to 0. Consider the evolution of y

$$\begin{aligned} y(k+1) &= x(k+1) - x_{\text{ave}}\mathbf{1} \\ &\overset{\text{(a)}}{=} W(k+1)x(k) - x_{\text{ave}}W(k)\mathbf{1} \\ &= W(k+1)\left(x(k) - x_{\text{ave}}\mathbf{1}\right) \\ &= W(k+1)y(k). \end{aligned} \tag{13}$$

Here (a) follows from the fact that $\mathbf{1}$ is an eigenvector for all $W(k+1)$. Thus, y evolves according to the same linear system as x. Therefore, we can write

$$E[y(t+1)^T y(t+1)|y(t)] = y(t)^T E[W(t)^T W(t)]y(t). \tag{14}$$

Since $W(t)$ is doubly stochastic, so is $W(t)^T W(t)$, and therefore, $E[W(t)^T W(t)]$ is doubly stochastic. Since the matrices W are identically distributed we will shorten $E[W(t)^T W(t)]$ to $E[W^T W]$.

Since $y(t) \perp \mathbf{1}$, and $\mathbf{1}$ is the eigenvector corresponding to the largest eigenvalue 1 of $E[W^T W]$

$$y(t-1)^T E[W^T W] y(t-1) \leq \lambda_2(E[W^T W]) \|y(t-1)\|^2. \tag{15}$$

Repeatedly conditioning and using (15), we finally obtain the bound

$$E[y(t)^T y(t)] \leq \lambda_2^{2t}(E[W^T W]) \|y(0)\|^2. \tag{16}$$

From this, we see that the second moment of the error $y(t)$ converges to 0 at a rate governed by $\lambda_2^2(E[W^T W])$. This means that any scheme of choosing the $W(t)$ which corresponds to a $E[W^T W]$ with second largest eigenvalue *strictly* less than 1 (and, of course, with $\rho(E[W] - \mathbf{1}\mathbf{1}^T/n)$ less than 1) provably converges in the second moment.

This condition is only a sufficient condition for the convergence of the second moment. We can, in fact, obtain a necessary and sufficient condition by considering the evolution of $E[yy^T]$ rather than $E[y^T y]$.

Since $y(t+1) = W(t)y(t)$, $y(t+1)y(t+1)^T = W(t)y(t)y(t)^T W(t)$. Let $Y(t) = y(t)y(t)^T$. Then

$$Y(t+1) = W(t)Y(t)W(t)^T$$

i.e., $Y(t)$ evolves according to a (random) linear system. Now

$$\begin{aligned} Y(t+1)_{ij} &= (W(t)y(t)y(t)^T W(t)^T)_{ij} \qquad (17)\\ &= \left((W(t)y(t))(W(t)y(t))^T\right)_{ij}\\ &= (W(t)y(t))_i \, (W(t)y(t))_j\\ &= \sum_{p,q=1}^{n} W(t)_{ip} W(t)_{jq} y_p(t) y_q(t). \qquad (18) \end{aligned}$$

Collect the entries of the matrix Y into a vector $\tilde{Y} \in \boldsymbol{R}^{n^2}$, with entries drawn columnwise from Y. Then, using (18), we see that

$$\begin{aligned} \tilde{Y}(t+1) &= (W(t) \otimes W(t))\tilde{Y}(t)\\ \Rightarrow E[\tilde{Y}(t+1)|\tilde{Y}(t)] &= E[W(t) \otimes W(t)]\tilde{Y}(t) \end{aligned}$$

where $\otimes$ stands for Kronecker product. Conditioning repeatedly, we see that

$$E[\tilde{Y}(t)] = E[W \otimes W]^t \tilde{Y}(0). \tag{19}$$

Since each $W(t)$ has $\lambda_1 = 1$ with corresponding eigenvector $v_1 = \mathbf{1}$, each $W(t) \otimes W(t)$ also has $\lambda_1 = 1$, with eigenvector $\tilde{v}_1 = \mathbf{1} \in \boldsymbol{R}^{n^2}$. Also, each $\tilde{Y}(t)$ is orthogonal to $\mathbf{1}$, since

$$\sum_{i=1}^{n^2} \tilde{Y}(t)_i = \sum_i \left(y_i(t) \sum_j y_j(t) \right) = \sum y_i(t) \cdot 0 = 0$$

since $y(t) \perp \mathbf{1}$.

Therefore, the convergence of $\tilde{Y}(t)$ is governed by $\rho\left(E(W \otimes W) - \frac{1}{n^2}\mathbf{1}\mathbf{1}^T\right)$, where $\mathbf{1} \in \boldsymbol{R}^{n^2}$. If

$$\rho\left(E(W \otimes W) - \frac{1}{n^2}\mathbf{1}\mathbf{1}^T\right) < 1$$

then $\tilde{Y}(t)$, and therefore $E[yy^T]$ converges to zero.

Note that $E[yy^T]$ converges to the zero matrix if and only if $E[y^T y]$ converges to 0. If $E[yy^T] \to 0$, then each $E[y_i^2] \to 0$, which means that

$$E[y^T y] = \sum_{i=1}^{n} E\left[y_i^2\right] \to 0$$

as well. Conversely, suppose $E[y^T y] \to 0$. Then each $y_i^2 \to 0$ as well. From the Cauchy–Schwartz inequality

$$E^2[|y_i y_j|] \leq E\left[y_i^2\right] E\left[y_j^2\right] \to 0$$

so that each entry in the matrix $E[yy^T]$ converges to 0.

Thus, a necessary and sufficient condition for second moment convergence is that $\rho\left(E(W \otimes W) - \frac{1}{n^2}\mathbf{1}\mathbf{1}^T\right) < 1$. However, despite having an exact criterion for convergence of the second moment, we will use $\lambda_2(E[W^T W])$ in our analysis. This is because $E[W^T W]$ is much easier to evaluate for a given algorithm than the expected value of the Kronecker product $E[W \otimes W]$.

III. High Probability Bounds On Averaging Time

We prove an upper bound (5) and a lower bound (6) in Lemmas 2 and 3 on the discrete time (or equivalently, number of clock ticks) required to get within ϵ of $x_{\text{ave}}\mathbf{1}$ (analogous to (5) and (6)) for the asynchronous averaging algorithm.

A. Upper Bound

Lemma 2: For algorithm $\mathcal{A}(P)$, for any initial vector $x(0)$, for $k \geq K^*(\epsilon)$

$$\Pr\left(\frac{\|x(k) - x_{\text{ave}}\mathbf{1}\|}{\|x(0)\|} \geq \epsilon\right) \leq \epsilon$$

where

$$K^*(\epsilon) \triangleq \frac{3\log\epsilon^{-1}}{\log\lambda_2(W)^{-1}}. \tag{20}$$

Proof: Recall that under algorithm $\mathcal{A}(P)$

$$x(k+1) = W(k+1)x(k) \tag{21}$$

where, with probability $\frac{1}{n}P_{ij}$, the random matrix $W(k)$ is

$$W_{ij} = I - \frac{(e_i - e_j)(e_i - e_j)^T}{2}. \tag{22}$$

Note that $W(k)$ are doubly stochastic matrices for all (i,j). That is, for all $k \geq 1$

$$x(k)^T\mathbf{1} = x(0)^T\mathbf{1} = \sum_{i=1}^{n} x_i(0). \tag{23}$$

Given our assumptions on the matrix of transition probabilities P, we can conclude from the previous section that $x(k) \to x_{\text{ave}}\mathbf{1}$. We want to find out how fast x_k converges; in particular, we want to obtain probabilistic bounds on $y(k) = x(k) - x_{\text{ave}}\mathbf{1}$. For this, we will use the second moment of $y(k)$ to apply Markov's inequality as below.

Computing W:

Let W denote the expected value of $W(0)$ (which is the same as $E[W(k)]$)

$$W = \frac{1}{n}\sum_{i,j} P_{ij}W_{ij}. \tag{24}$$

Then, the entries of W are as follows:

1) for $i \neq j$, $W_{ij} = \frac{P_{ij}+P_{ji}}{2n}$ and
2) $W_{ii} = 1 - \frac{\left[\sum_{j=1}^{n}(P_{ij}+P_{ji})\right]-2P_{ii}}{2n}$.

This yields the W defined in (7), that is,

$$W = I - \frac{1}{2n}D + \frac{P+P^T}{2n} \tag{25}$$

where $D = \text{diag}([D_1 \cdots D_n])$ is the diagonal matrix with entries

$$D_i = \left(\sum_{j=1}^{n}[P_{ij} + P_{ji}]\right).$$

Note that if $P = P^T$, then P is doubly stochastic. This implies that $D_i = 2$, which in turn means that $W = (1-1/n)I + (1/n)P$.

Computing the second moment $E[y(k)^T y(k)]$:

With probability $\frac{P_{ij}}{n}$, the edge (i,j) is chosen to average, that is, $W(k) = W_{ij}$. Then

$$W(k)^T W(k) = \left(I - \frac{(e_i - e_j)(e_i - e_j)^T}{2}\right)^2 \tag{26}$$

$$= \left(I - \frac{(e_i - e_j)(e_i - e_j)^T}{2}\right) \tag{27}$$

$$= W(k). \tag{28}$$

It is not an accident that $W(k)^T W(k) = W(k)$: each W_{ij} is a *projection* matrix, which projects a vector x onto the subspace $x_i = x_j$. The entries of x except the i and jth stay unchanged, and x_i and x_j average their values. Since every projection matrix Q satisfies $Q^2 = Q$, and W_{ij} are symmetric, we have $W_{ij}^T W_{ij} = W_{ij}$.

Since (28) holds for each instance of the random matrix W, we have

$$E[W(0)^T W(0)] = E[W(0)] = W. \tag{29}$$

Note that this means that W is symmetric[3] positive-semidefinite (since $W = W^T W$) and hence it has nonnegative real eigenvalues.

[3]The symmetry of W does not depend on P being symmetric.

From (16) and (29)

$$E[y(k)^T y(k)] \leq \lambda_2(W)^k y(0)^T y(0). \tag{30}$$

Now,

$$y(0)^T y(0) = x(0)^T x(0) - nx_{\text{ave}}^2 \leq x(0)^T x(0). \tag{31}$$

Application of Markov's inequality:

From (30), (31), and an application of Markov's inequality, we have

$$\Pr\left(\frac{\|x(k) - x_{\text{ave}}\mathbf{1}\|}{\|x(0)\|} \geq \epsilon\right) = \Pr\left(\frac{y(k)^T y(k)}{x(0)^T x(0)} \geq \epsilon^2\right) \leq \epsilon^{-2}\frac{E[y(k)^T y(k)]}{x(0)^T x(0)} = \epsilon^{-2}\lambda_2(W)^k. \tag{32}$$

From (32), it follows that for $k \geq K(\epsilon) \triangleq \frac{3\log\epsilon^{-1}}{\log\lambda_2(W)^{-1}}$

$$\Pr\left(\frac{\|x(k) - x_{\text{ave}}\mathbf{1}\|}{\|x(0)\|} \geq \epsilon\right) \leq \epsilon. \tag{33}$$

This proves the lemma, and gives us an upper bound on the ϵ-averaging time. □

B. A Lower Bound on the Averaging Time

Here, we will prove a lower bound for the ϵ-averaging time, which is only a factor of 6 away from the upper bound in the previous section. We have the following result.

Lemma 3: For algorithm $\mathcal{A}(P)$, there exists an initial vector $x(0)$, such that for $k < K_*(\epsilon)$

$$\Pr\left(\frac{\|x(k) - x_{\text{ave}}\mathbf{1}\|}{\|x(0)\|} \geq \epsilon\right) > \epsilon$$

where

$$K_*(\epsilon) \triangleq \frac{0.5\log\epsilon^{-1}}{\log\lambda_2(W)^{-1}}. \tag{34}$$

Proof: Since $y(k+1) = W(k)y(k)$, we obtain from (29)

$$E[y(k)] = W^k y(0). \tag{35}$$

By definition, W is a symmetric positive-semidefinite doubly stochastic matrix with nonnegative real eigenvalues

$$1 = \lambda_1(W) \geq \lambda_2(W) \geq \cdots \geq \lambda_n(W) \geq 0$$

and corresponding orthonormal eigenvectors $\frac{1}{\sqrt{n}}\mathbf{1}, v_2, v_3, \ldots, v_n$. Select

$$x(0) = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{n}}\mathbf{1} + v_2\right) \Rightarrow y(0) = \frac{1}{\sqrt{2}}v_2.$$

For this choice of $x(0)$, $\|x(0)\| = 1$. Now from (35)

$$E[y(k)] = \frac{1}{\sqrt{2}}\lambda_2^k(W)v_2. \tag{36}$$

For this particular choice of $x(0)$, we will lower-bound the ϵ-averaging time by lower-bounding $E[\|y(k)\|^2]$, and using Lemma 4 as stated below.

By Jensen's inequality and (36)

$$\begin{aligned} E\left[\sum_{i=1}^{n} y_i(k)^2\right] &\geq \sum_{i=1}^{n} E[y_i(k)]^2 \\ &= E[y(k)]^T E[y(k)] \\ &= \frac{1}{2}\lambda_2^{2k}(W) v_2^T v_2 \\ &= \frac{1}{2}\lambda_2^{2k}(W). \end{aligned} \tag{37}$$

Lemma 4: Let X be a random variable such that $0 \leq X \leq B$. Then, for any $0 < \epsilon < B$

$$\Pr(X \geq \epsilon) \geq \frac{E[X] - \epsilon}{B - \epsilon}.$$

Proof:

$$\begin{aligned} E[X] &\leq \epsilon \Pr(X < \epsilon) + B \Pr(X \geq \epsilon) \\ &= \Pr(X \geq \epsilon)(B - \epsilon) + \epsilon. \end{aligned}$$

Rearranging terms gives us the lemma. □

From (36), $\|y(k)\|^2 \leq \|y(0)\|^2 \leq 1/2$. Hence, Lemma 4 and (37) imply that for $k < K_*(\epsilon)$

$$\Pr\left(\|y(k)\| \geq \epsilon\right) > \epsilon. \tag{38}$$

This completes the proof of Lemma 3. □

Combining the results in the previous two lemmas, we have the result of Theorem 3.

The following corollaries are immediate.

Corollary 1: For large n and symmetric P, $T_{\text{ave}}(\epsilon, P)$ is bounded as follows:

$$T_{\text{ave}}(\epsilon, P) \leq \frac{3n \log \epsilon^{-1}}{(1 - \lambda_2(P))} \triangleq T^*(\epsilon, P) \quad \text{and} \tag{39}$$

$$T_{\text{ave}}(\epsilon, P) \geq \frac{0.5n \log \epsilon^{-1}}{(1 - \lambda_2(P))} \triangleq T_*(\epsilon, P). \tag{40}$$

Proof: By definition, $\lambda_2(W) = \left(1 - \frac{1}{n}(1 - \lambda_2(P))\right)$. For large n, $\frac{1}{n}(1 - \lambda_2(P))$ is small, and hence,

$$\log\left(1 - \frac{1}{n}\left(1 - \lambda_2(P)\right)\right) \approx -\frac{1}{n}\left(1 - \lambda_2(P)\right).$$

This along with Theorem 3 completes the proof. □

Corollary 2: For a symmetric P, the absolute time $Z_{T^\cdot(\epsilon,P)}$ it takes for $T^\cdot(\epsilon, P)$ clock ticks to happen is given by

$$Z_{T^\cdot(\epsilon,P)} = \frac{T^\cdot(\epsilon, P)}{n}\left(1 \pm \frac{2}{\sqrt{n}}\right) \tag{41}$$

with probability at least $1 - 2\epsilon$.

Proof: For $\delta = \frac{\sqrt{2(1-\lambda_2(P))}}{\sqrt{3n}}$ and $k = T^\cdot(\epsilon, P)$ and using (39), the right-hand side of (4) evaluates to

$$2\exp\left(-\frac{2(1 - \lambda_2(P))}{3n} \star \frac{3n \log \epsilon^{-1}}{2(1 - \lambda_2(P))}\right) = 2\epsilon.$$

Since $-1 < \lambda_2(P) < 1$ for the nonnegative doubly stochastic symmetric matrix P, $\delta = \frac{2}{\sqrt{n}}$ is larger than the above choice of δ. This completes the proof. □

Note that the proof of Lemma 3 uses only two features of the algorithm $\mathcal{A}(P)$:

- $E[W(0)] = W$ is symmetric, which allows us to choose an orthonormal set of eigenvectors;
- W is positive semidefinite, which means that the convergence of $y(k)$ to 0 is governed by $\lambda_2(W)$.

Consider *any* randomized gossip algorithm with symmetric expectation matrix $E[W]$ (and, of course, satisfying the gossip constraints stated in Section II). For such an algorithm, the rate of convergence of $y(k)$ to 0 is governed by $\lambda_{\max}(E[W])$, the second largest eigenvalue in absolute value, rather than $\lambda_2(E[W])$. Exactly the same proof can be used to derive a lower bound for this gossip algorithm, with the only difference being that $\lambda_2(W)$ is replaced by $\lambda_{\max}(E[W])$. Thus, we can state the following lower bound for the performance of an arbitrary randomized gossip algorithm with symmetric $E[W]$.

Lemma 5: For any randomized gossip algorithm with symmetric expectation $E[W]$, there exists an initial vector $x(0)$, such that for $k < K_*(\epsilon)$

$$\Pr\left(\frac{\|x(k) - x_{\text{ave}}\mathbf{1}\|}{\|x(0)\|} \geq \epsilon\right) > \epsilon$$

where

$$K_*(\epsilon) \triangleq \frac{0.5 \log \epsilon^{-1}}{\log \lambda_{\max}(E[W])^{-1}}. \tag{42}$$

The proof of the upper bound relies on more specific properties of the algorithm $\mathcal{A}$, and thus cannot be duplicated for an arbitrary algorithm. Note also that while the expressions for the lower bounds for our algorithm $\mathcal{A}$, and an arbitrary algorithm with symmetric expectation are very similar, this does not mean that $\mathcal{A}$ has the same lower bound as any other randomized gossip algorithm with symmetric expectation: the lower bound depends on the value of $E[W]$, and the set of matrices that can be $E[W]$ for some instance of the algorithm $\mathcal{A}$ is a subset of the set of all doubly stochastic symmetric matrices.

C. Synchronous Averaging Algorithms

In this subsection, we consider the case of synchronous averaging algorithms. Unlike the asynchronous case, in the synchronous setting, multiple node pairs communicate at exactly the same time. Gossip constraints require that these simultaneously active node pairs are disjoint. That is, the edges of the network graph corresponding to the pair-wise operations form a (not necessarily complete) matching. This makes the

synchronous case harder to deal with, as it requires the algorithm to form a matching in a distributed manner.

We first present a *centralized* synchronous gossip algorithm that achieves the same performance as the asynchronous algorithm. This algorithm requires a centralized entity to choose matchings of the nodes each time. Then, we present a completely distributed synchronous gossip algorithm that finds matchings in a distributed manner without any additional computational burden. We show that this algorithm performs as well as the centralized gossip algorithm for any graph with bounded degree. We extend this result for unbounded degree regular graphs, for example, the complete graph.

1) Centralized Synchronous Algorithm: Let P be any $n \times n$ doubly-stochastic symmetric matrix corresponding to the probability matrix of the algorithm, as before. By Birkhoff–Von Neumann's theorem [18], a nonnegative doubly-stochastic matrix P can be decomposed into permutation matrices (equivalently matchings) as

$$P = \sum_{m=1}^{n^2} \alpha_m \Pi_m, \quad \alpha_m \geq 0, \quad \sum_{m=1}^{n^2} \alpha_m = 1.$$

Define a (matrix) random variable Π with distribution $\Pr(\Pi = \Pi_m) = \alpha_m, 1 \leq m \leq n^2$.

The *centralized* synchronous algorithm corresponding to P is as follows: in each time step, choose one of the n^2 permutations (matchings) in an i.i.d. fashion with distribution identical to Π. Note that the permutation Π need not be symmetric. The update corresponding to a permutation Π is as follows: if $\Pi_{ij} = 1$, then node i averages its current value with the value it receives from node j. Now, we state the theorem that characterizes the averaging time of this algorithm.

Theorem 4: The averaging time of the centralized synchronous algorithm described above is given by

$$\frac{0.5 \log \epsilon^{-1}}{\log \lambda^{-1}} \leq T_{\text{ave}}(\epsilon) \leq \frac{3 \log \epsilon^{-1}}{\log \lambda^{-1}}$$

where $\lambda = \frac{1}{2}(1 + \lambda_2(P))$.

Proof: The proof of Theorem 4 is based on the proofs of Lemmas 2 and 3 presented in Section III. Let $\Pi(k)$ denote the random permutation matrix chosen by the algorithm at time $k \in \mathbb{Z}_+$. The linear iteration corresponding to this update is $x(k+1) = W(k)x(k)$, where $W(k)$ is given by

$$W(k) = \frac{1}{2}\left(I + \Pi(k)\right). \tag{43}$$

Now

$$\begin{aligned} W &= E[W(0)] \\ &= \frac{1}{2}\left(I + E[\Pi(0)]\right) \\ &= \frac{1}{2}\left(I + P\right). \end{aligned} \tag{44}$$

Now since $W(0) = \frac{1}{2}(I + \Pi)$

$$W(0)^T W(0) = \frac{1}{4}\left(2I + \Pi + \Pi^T\right)$$

since $\Pi^T \Pi = I$ (Π is a permutation matrix). Therefore,

$$\begin{aligned} E[W(0)^T W(0)] &= \frac{1}{2} E\left[I + \frac{\Pi + \Pi^T}{2}\right] \\ &= \frac{1}{2}(I + P) \qquad (45) \\ &= W. \qquad (46) \end{aligned}$$

Using the arguments of Lemmas 2 and 3, exactly as in the asynchronous case, it can be easily shown that for any averaging algorithm

$$\begin{aligned} \frac{0.5 \log \epsilon^{-1}}{\log \lambda_{\max}^{-1}(E[W(0)])} &\leq T_{\text{ave}}(\epsilon) \\ &\leq \frac{3 \log \epsilon^{-1}}{\log \lambda_{\max}^{-1}(E[W(0)^T W(0)])}. \end{aligned} \tag{47}$$

From (44) and (46)

$$\lambda_{\max}(E[W(0)]) = \lambda_{\max}(E[W(0)^T W(0)]) = \lambda_{\max}(W).$$

Further, all eigenvalues of $W = \frac{1}{2}(I + P)$ are nonnegative. Hence,

$$\lambda_{\max}(W) = \frac{1}{2}\left(1 + \lambda_2(P)\right) \triangleq \lambda. \tag{48}$$

From (47) and (48), the statement of Theorem 4 follows. □

2) Distributed Synchronous Algorithm: The centralized synchronous algorithm needs a centralized entity to select a permutation matrix (or matching) at each time step, corresponding to the matrix P. Here we describe a way to obtain such a permutation matrix in a distributed manner for bounded degree network graphs. Later we extend this result for unbounded degree regular graphs for a particular class of P (corresponding to the natural random walk).

Given a network graph G, let d^* be the maximum node degree. We assume that all nodes know d^* (a justification for this assumption is given at the end of the proof of Theorem 5). Now we describe the algorithm based on P as follows.

In each time step, every node becomes *active* with probability $1/2$ independently. Consider an active node i. Let $d(i)$ be its degree (i.e., the number of its neighbors). Active node i contacts at most one of its neighbors to average, as follows. With probability $1 - \frac{d(i)}{d^*}$, node i does nothing, i.e., it does not contact any neighbor. With equal probabilities $\frac{1}{d^*}$, it chooses one of its $d(i)$ neighbors to contact.

All active nodes ignore the nodes that contact them. An inactive node, say j, ignores the requests of active nodes if contacted by more than one active node. If active node i contacts inactive node j but no other active node contacts j, then i and j average their values with probability $\Phi_j P_{ij}$, where

$$\Phi_j = \left(1 - \frac{1}{2d^*}\right)^{d^* - d(j)}.$$

We state the following result for this algorithm.

Theorem 5: The averaging time of the distributed synchronous algorithm described above is given by

$$\frac{0.5 \log \epsilon^{-1}}{\log \lambda^{-1}} \leq T_{\text{ave}}(\epsilon) \leq \frac{3 \log \epsilon^{-1}}{\log \lambda^{-1}}$$

where $\lambda = \lambda_2(W)$, with

$$W = I - \frac{\bar{d}}{8}D + \frac{\bar{d}}{8}\left(P + P^T\right)$$
$$\bar{d} = \frac{1}{d^*}\left(1 - \frac{1}{2d^*}\right)^{d^*-1}$$

and D a diagonal matrix with $D_{ii} = \sum_j P_{ij} + P_{ji}$.

Before we prove Theorem 5, note that for bounded d^*, $\bar{d}$ is a constant away from 0. Hence,

$$\begin{aligned}\lambda_2(W) &= 1 - \frac{\bar{d}}{8}\lambda_2(D - P - P^T) \\ &= \Theta\left(1 - \lambda_2(D - P - P^T)\right).\end{aligned}$$

Thus, the averaging time of the distributed synchronous algorithm is of the same order as that of the centralized synchronous algorithm for any bounded degree graph.

Proof of Theorem 5: The proof follows using Theorem 4. We first note that the algorithm, as described above, only allows pair-wise averaging for distinct node pairs. Let $W(k)$ be the random matrix corresponding to the algorithm at time k, that is,

$$x(k) = W(k)x(k-1).$$

Since $W(k)$ averages values of distinct node pairs, it is a symmetric projection matrix, projecting onto the intersection of the subspaces $x_i = x_j$ where (i, j) is an averaging pair. Therefore, $W(k)^T W(k) = W(k)$ for all k, and therefore $E[W^T W] = E[W]$. Using this property, as argued in Theorem 4, the averaging time is bounded as

$$\begin{aligned}\frac{0.5\log\epsilon^{-1}}{\log\lambda_{\max}^{-1}(E[W(0)])} &\le T_{\text{ave}}(\epsilon) \\ &\le \frac{3\log\epsilon^{-1}}{\log\lambda_{\max}^{-1}(E[W(0)])}. \qquad (49)\end{aligned}$$

Next, we evaluate $W = E[W(0)]$. First we compute the probability that node pair (i, j) average. Denote this probability Q_{ij}. We claim that

$$Q_{ij} = \frac{\bar{d}}{4}(P_{ij} + P_{ji})$$

where $\bar{d} = \frac{1}{d^*}\left(1 - \frac{1}{2d^*}\right)^{d^*-1}$. The reason is as follows: (i, j) average when a) i is active, j is inactive, i contacts j but no other node contacts j, and they decide to average; b) j is active, i is inactive, j contacts i but no other node contacts i, and they decide to average.

We compute the probability of a): i is active and j is inactive with probability $1/4$; i contacts j with probability $1/d^*$; no other node contacts j with probability $\left(1 - \frac{1}{2d^*}\right)^{d(j)-1}$; after which (i, j) average with probability $\Phi_j P_{ij}$. Since all these events are independent, the probability of a) turns out to be $\frac{\bar{d}P_{ij}}{4}$. Similarly, the probability of event b) is $\frac{\bar{d}P_{ij}}{4}$. Since events a) and b) are disjoint, the net probability of (i, j) averaging is as claimed.

Now, it is easy to see that

$$W = I - \frac{\bar{d}}{8}D + \frac{\bar{d}}{8}\left(P + P^T\right) \qquad (50)$$

where D is the diagonal matrix defined in the statement of the theorem. From the argument preceding (49), we have that $E[W] = E[W^T W]$, so that all eigenvalues of W are nonnegative. Hence from (50), the statement of Theorem 5 follows. □

Note. The assumption of nodes knowing d^* is not restrictive for the following reason: all nodes can compute the maximum node degree via a gossip algorithm in which each node contacts its neighbors in a round-robin fashion, and informs them of its current estimate of the maximum degree (its initial estimate is its own degree). Since the order of pair-wise comparisons to compute the maximum of many numbers is not important, each node can compute the maximum of the received information from other nodes in any order to update its own estimate. It is not hard to see that such an algorithm requires $O(d^* D)$ time for all nodes to know maximum degree, where D is the diameter of the graph. Now, consider a node pair (i, j) such that the shortest path between them is D. Now consider $x(0)$ such that $x_i(0) = 1$ and $x_l = 0$ for all $l \neq i$. Then, under any averaging algorithm, for $m < D/2$, $\|x(m) - x_{\text{ave}}\mathbf{1}\| \ge \sqrt{D}/n$. Hence, for $\epsilon = o(1/n)$, the ϵ-averaging time is at least $\Omega(D)$. Since we are considering bounded degree graphs, $O(d^* D) = O(D)$. Hence, we can ignore the pre-processing time for $\epsilon = o(1/n)$ in order notation. For clean presentation of our results, we ignore this pre-processing time in general.

Consider a d-regular graph, where each node degree is exactly d. Now, modify the above algorithm as follows: when an active node i contacts an inactive node j and j is not contacted by any other node, then (i, j) always average. The following result follows using arguments of Theorem 5.

Corollary 3: The averaging time of the algorithm described above for a d-regular graph is bounded as

$$\frac{0.5\log\epsilon^{-1}}{\log\lambda_2^{-1}(W)} \le T_{\text{ave}}(\epsilon) \le \frac{3\log\epsilon^{-1}}{\log\lambda_2^{-1}(W)} \qquad (51)$$

where

$$W = \left(1 - \frac{\hat{d}}{4}\right)I + \frac{\hat{d}}{4}P$$
$$\hat{d} = \left(1 - \frac{1}{2d}\right)^{d-1}$$

and P is defined as

$$P_{ij} = \begin{cases} 1/d, & \text{if } i \text{ and } j \text{ are neighbors} \\ 0, & \text{otherwise.}\end{cases}$$

Note that

$$\lambda_2(W) = \left(1 - \frac{\hat{d}}{4}(1 - \lambda_2(P))\right).$$

As a consequence, for the complete graph, $\lambda_2(W) \approx 1 - 0.25e^{-0.5} < 1$. Thus, the averaging time $T_{\text{ave}}(\epsilon) = \Theta(\log\epsilon^{-1})$. For $\epsilon = 1/n$, this implies the main results of [29] and [22].

IV. Optimal Averaging Algorithm

We saw in Theorem 3 that the averaging time is a monotonically increasing function of the second largest eigenvalue

of $W(P) = \sum_{i,j=1}^{n} \frac{1}{n} P_{ij} W_{ij}$. Thus, finding the fastest averaging algorithm corresponds to finding P such that $\lambda_2(W)$ is the smallest, while satisfying constraints on P. Thus, we have the optimization problem

$$\begin{aligned} \text{minimize} \quad & \lambda_2(W) \\ \text{subject to} \quad & W = \sum_{i,j=1}^{n} \frac{1}{n} P_{ij} W_{ij} \\ & P_{ij} \geq 0, \quad P_{ij} = 0 \text{ if } \{i,j\} \notin E \\ & \sum\nolimits_j P_{ij} = 1, \quad \forall\, i. \end{aligned} \tag{52}$$

The objective function, which is the second largest eigenvalue of a doubly stochastic matrix, is a convex function on the set of symmetric matrices. Therefore, (52) is a convex optimization problem. This problem can be reformulated as the following SDP:

$$\begin{aligned} \text{minimize} \quad & s \\ \text{subject to} \quad & W - \mathbf{1}\mathbf{1}^T/n \preceq sI \\ & W = \sum_{i,j=1}^{n} \frac{1}{n} P_{ij} W_{ij} \\ & P_{ij} \geq 0, \quad P_{ij} = 0 \text{ if } \{i,j\} \notin E \\ & \sum\nolimits_j P_{ij} = 1, \quad \forall\, i \end{aligned} \tag{53}$$

where $\preceq$ denotes inequality with respect to the cone of symmetric positive semidefinite matrices. For general background on SDPs, eigenvalue optimization, and associated interior-point methods for solving these problems, see, for example, [7], [33], [38], [47], and references therein. Interior point methods can be used to solve problems with a thousand edges or so; subgradient methods can be used to solve the problem for larger graphs that have up to a hundred thousand edges. The disadvantage of a subgradient method compared to a primal-dual interior point method is that the algorithm is relatively slow (in terms of number of iterations), and has no simple stopping criterion that can guarantee a certain level of suboptimality.

In summary, given a graph topology, we can solve the SDP (53) to find the P^* for the fastest averaging algorithm.

A. Distributed Optimization

We have seen that finding the fastest averaging algorithm is a convex optimization problem, and can therefore be solved efficiently to obtain the optimal distribution P^*. Unfortunately, a P^* computed in a centralized fashion is not useful in our setting. It is natural to ask if in this setting, the optimization (like the averaging itself), can also be performed in a decentralized fashion. That is, is it possible for the nodes on the graph, possessing only local information, and with only local communication, to compute the probabilities P_{ij} that lead to the fastest averaging algorithm?

In this subsection, we describe a completely distributed algorithm $\mathcal{S}$ based on an approximate subgradient method which converges to a neighborhood of the optimal; alternately put, each iteration of the algorithm moves P closer to the globally optimal P^*, as stated in this theorem.

Theorem 6: Let m be the number of edges in G. Let the subgradient at iteration k in $\mathcal{S}$ lie within the ϵ_k-subdifferential, and define $\epsilon = \liminf \epsilon_k$. Then, the sequence of iterates in $\mathcal{S}$ converges to a distribution P for which $\lambda_2(W(P))$ is within $8m\epsilon/n^2$ of the globally optimal value $\lambda_2(W(P^*))$.

The required background and notation will be provided as necessary during the proof, which comprises the remainder of this section.

Notation: It will be easier to analyze the subgradient method if we collect the entries of the matrix P_{ij} into a vector, which we will call p. Since there is no symmetry requirement on the matrix P, the vector p will need to have entries corresponding to P_{ij} as well as P_{ji} (this corresponds to replacing each edge in the *undirected* graph G by two directed edges, one in each direction).

The vector p corresponds to the matrix P as follows. Let the total number of (non-self-loop) edges in G be m. Assign numbers to the undirected edges from 1 through m: if edge (i,j), $i < j$, is assigned number l, we denote this as $l \sim (i,j)$. If $l \sim (i,j)$, then define the variable $p_l = P_{ij}$, and $p_{-l} = P_{ji}$.

We will also introduce the notation $\boldsymbol{p_i}$ corresponding to the nonzero entries in the ith row of P (we do this to make concise the constraint that the sum of elements in each row should be 1). That is, we define for $1 \leq i \leq n$

$$\boldsymbol{p_i} = [P_{ij}; (i,j) \in \mathcal{E}]. \tag{54}$$

Define also the matrices E_l, $l \sim (i,j)$, with entries $E_{l_{ij}} = E_{l_{ji}} = +1$, $E_{l_{ii}} = E_{l_{jj}} = -1$, and zeros everywhere else. Then

$$E_l = 2(W_{ij} - I).$$

Finally, denote the degree of node i by m_i.

1) Subgradient Method: We will describe the subgradient method for the optimization problem restated in terms of the variable p. We can state (53) in terms of the variables $p = [p_{-m}, \ldots p_{-1}, p_1, \ldots p_m]$ as follows:

$$\begin{aligned} \text{minimize} \quad & \lambda_2 \left(I + \frac{1}{2n} \sum\nolimits_{l=1}^{m} p_l E_l + p_{-l} E_{-l} \right) \\ \text{subject to} \quad & \mathbf{1}^T \boldsymbol{p_i} \leq 1, \quad \forall\, i \\ & p_l \geq 0, \quad 1 \leq |l| \leq m \end{aligned} \tag{55}$$

where $\boldsymbol{p_i}$ is as defined in (54).

We will use the subgradient method to obtain a distributed solution to this problem. The use of the subgradient method to solve eigenvalue problems is well known; see, for example, [1], [31], [32], [39] for material on nonsmooth analysis of spectral functions, and [8], [5], [19] for more general background on nonsmooth optimization.

Recall that a *subgradient* of λ_2 at W is a symmetric matrix G that satisfies the inequality

$$\begin{aligned} \lambda_2(\tilde{W}) &\geq \lambda_2(W) + \langle G, \tilde{W} - W \rangle \\ &= \lambda_2(W) + \mathbf{Tr}\, G(\tilde{W} - W) \end{aligned}$$

for any feasible, i.e., symmetric stochastic matrix $\tilde{W}$ (here $\langle \cdot, \cdot \rangle$ denotes the matrix inner product, and $\mathbf{Tr}$ denotes the trace of a matrix). Let u be a unit eigenvector associated with $\lambda_2(W)$, then the matrix $G = uu^T$ is a subgradient of $\lambda_2(W)$ (see, for example, [1]). For completeness, we include the proof here. First

note that $u^T\mathbf{1} = 0$. By the variational characterization of the second eigenvalue of W and $\tilde{W}$, we have

$$\lambda_2(W) = u^T W u$$
$$\lambda_2(\tilde{W}) \geq u^T \tilde{W} u.$$

Subtracting the two sides of the above equality from that of the inequality, we have

$$\lambda_2(\tilde{W}) \geq \lambda_2(W) + u^T(\tilde{W} - W)u$$
$$= \lambda_2(W) + \mathbf{Tr}\, uu^T(\tilde{W} - W).$$

So uu^T is a subgradient.

Using

$$W(p) = I + \frac{1}{2n}\left(\sum_{l=1}^{m} p_l E_l + p_{-l}E_{-l}\right)$$
$$= I + \frac{1}{2n}\left(\sum_{|l|=1}^{m} p_l E_l\right)$$

in terms of the probability vector p, we obtain

$$\lambda_2(W(\tilde{p})) \geq \lambda_2(W(p)) + \sum_{|l|=1}^{m}\left(v^T\left(\frac{1}{2n}E_l\right)v\right)(\tilde{p}_l - p_l) \tag{56}$$

so that the subgradient $g(p)$ is given by

$$g(p) = \frac{1}{2n}\left(u^T E_{-m} u, \ldots, u^T E_m u\right) \tag{57}$$

with components

$$g_l(p) = \frac{1}{2n}u^T E_l u = -\frac{1}{2n}(u_i - u_j)^2, \quad l \sim (i,j)$$

where $|l| = 1, \ldots, m$. Observe that if each node i knows its own component u_i of the unit eigenvector, then this subgradient can be computed locally, using only local information.

The following is the projected subgradient method for (55).

- **Initialization**: Initialize p to some feasible vector, for example, p corresponding to the natural random walk. Set $k := 1$.
- **Repeat** for $k \geq 1$
  - — **Subgradient step.** Compute a subgradient $g^{(k)}$ at p, and set
  $$p := p - \nu_k g^{(k)}.$$
  - — **Projection onto feasible set.** At each node i, project $\boldsymbol{p_i}$ obtained from the subgradient step onto $\mathbf{1}^T q \leq 1$, $q \succeq 0$. This is achieved as follows:
    1) If
    $$\sum_{j=1}^{m_i} \max\{0, \boldsymbol{p_i}_j\} \leq 1$$
    then set $\boldsymbol{p_i} = \max\{0, \boldsymbol{p_i}\}$, stop.
    2) If not, then use bisection to find $x \geq 0$ such that
    $$\sum_{j=1}^{m_i} \max\{0, \boldsymbol{p_i}_j - x\} = 1$$
    then set $\boldsymbol{p_i} = \max\{0, \boldsymbol{p_i}_j - x\}$, stop.

In this algorithm, Step 1 moves p in the direction of the subgradient with stepsize ν_k. Step 2 projects the vector p onto the feasible set. Since the constraints at each node are separable, the variables $\boldsymbol{p_i}$ corresponding to nodes i are projected onto the feasible set separately.

The projection method is derived from the optimality conditions of the projection problem

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{m_i}(q_j - \boldsymbol{p_i}_j)^2 \\ \text{subject to} & \mathbf{1}^T q \leq 1, \quad q \succeq 0 \end{array} \tag{58}$$

as shown.

Introduce Lagrange multipliers $\lambda \in \boldsymbol{R}_i^m$ for the inequality $q \succeq 0$, and ν for $\mathbf{1}^T q - 1 \leq 0$. The Karush–Kuhn–Tucker (KKT) conditions for optimal primal and dual variables q^*, λ^*, ν^* are

$$q^* \succeq 0, \quad \mathbf{1}^T q^* \leq 1$$
$$\lambda^* \succeq 0, \quad \nu^* \geq 0$$
$$\nu^*\left(\mathbf{1}^T q^* - 1\right) = 0, \quad \lambda_j^* q_j^* = 0, \quad j = 1, \ldots, m_i$$
$$2\left(q_j^* - \boldsymbol{p_i}_j\right) + \nu^* - \lambda_j^* = 0, \quad j = 1, \ldots, m_i.$$

Eliminating the slack variables λ_j, we get the equivalent optimality conditions

$$q^* \succeq 0, \quad \mathbf{1}^T q^* \leq 1 \tag{59}$$
$$\nu^* \geq 0, \quad \nu^*\left(\mathbf{1}^T q^* - 1\right) = 0 \tag{60}$$
$$q_j^*\left(2(q_j^* - \boldsymbol{p_i}_j) + \nu^*\right) = 0, \quad j = 1, \ldots, m_i \tag{61}$$
$$2\left(q_j^* - \boldsymbol{p_i}_j\right) + \nu^* \geq 0, \quad j = 1, \ldots, m_i. \tag{62}$$

If $\nu^* < 2\boldsymbol{p_i}_j$, then from the last condition, necessarily $q_i^* > 0$. From (61), this gives us $q_j^* = \boldsymbol{p_i}_j - \nu^*/2$. If on the other hand, $\nu^* \geq 2\boldsymbol{p_i}_j$, then $\nu^* \geq 2\boldsymbol{p_i}_j - 2q_j^*$ as well since $q_j^* \geq 0$, and so to satisfy (61), we must have $q_j^* = 0$. Combining these gives us that

$$q_j^* = \max\left\{0, \boldsymbol{p_i}_j - \frac{\nu^*}{2}\right\}. \tag{63}$$

The q_j^* must satisfy $\mathbf{1}^T q^* \leq 1$, i.e., $\sum \max\{0, q_j - \nu^*/2\} \leq 1$. However, we must also satisfy the complementary slackness condition $\nu^*(\mathbf{1}^T q^* - 1) = 0$. These two conditions combined together lead to a unique solution for ν^*, obtained either at $\nu^* = 0$, or at the solution of $\sum \max\{0, q_j - \nu^*/2\} = 1$; from ν^* the q_j^* can be found from (63).

2) Decentralization: Now consider the issue of decentralization. Observe that in the above algorithm, g can be computed locally at each node if u, the unit eigenvector corresponding to $\lambda_2(W)$, is known; more precisely, if each node i is aware of its own component of u and that of its immediate neighbors. The projection step can be carried out exactly at each node using local information alone. The rest of the subsection proceeds as follows: first we will discuss approximate distributed computation of the eigenvector u of W, and then show that the subgradient method converges to a certain neighborhood of the optimal value in spite of the error incurred during the distributed computation of u at each iteration.

The problem of distributed computation of the top-k eigenvectors of a matrix on a graph is discussed in [28]. By distributed computation of an eigenvector u of a matrix W, we mean that each node i is aware of the ith row of W, and can only communicate with its immediate neighbors. Given these constraints, the distributed computation must ensure that each node holds its value u_i in the unit eigenvector u. In [28], the authors present a distributed implementation of orthogonal iterations, referred to as DECENTRALOI (for decentralized orthogonal iterations), along with an error analysis.

Since the matrix W is symmetric and stochastic (it is a convex combination of symmetric stochastic matrices), we know that the first eigenvector is $\mathbf{1}$. Therefore, orthogonal iterations takes a particularly simple form (in particular, we do not need any Cholesky factorization type of computations at the nodes). We describe orthogonal iterations for this problem as follows.

- DECENTRALOI: Initialize the process with some randomly chosen vector v_0; for $k \geq 1$, repeat
  - — Set $v_k = Wv_{k-1}$
  - — (Orthogonalize) $v_k = v_k - \left(\sum_{i=1}^{n} \frac{1}{n} v_{k_i}\right) \mathbf{1}$
  - — (Scale to unit norm) $v_k = v_k / \|v_k\|$

Here, the multiplication by W is distributed, since W respects the graph structure, i.e., $W_{ij} \neq 0$ only if (i, j) is an edge. So entry i of v_k can be found using only values of v_{k-1} corresponding to neighbors of node i, i.e., the computation is distributed. The orthogonalize and scale steps can be carried out in a distributed fashion using the gossip algorithm outlined in this paper, or just by distributed averaging as described in [48] and used in [28]. Observe that the very matrix W can be used for the distributed averaging step, since it is also a probability matrix. We state the following result (applied to our special case) from [28], which basically states that it is possible to compute the eigenvector up to an arbitrary accuracy.

Lemma 6: If DECENTRALOI is run for $\Omega\left(t\tau_{\text{mix}} \log(16/\epsilon)\right)$ iterations, producing orthogonal vector u, then

$$\|u - u_r\| \leq O\left(\left(\frac{\lambda_3}{\lambda_2}\right)^t n\right) + 3\epsilon^{4t} \tag{64}$$

where $\|u - u_r\|$ is the L_2 distance between u and the eigenspace of λ_2; u_r is the vector in the eigenspace achieving this distance; and τ_{mix} is the mixing time of the doubly stochastic matrix used in the averaging step in DECENTRALOI.

For the algorithm to be completely decentralized, a decentralized criterion for stopping when the eigenvector has been computed up to an accuracy ϵ is necessary. This is discussed in detail in [28]; we merely use the fact that it is possible for the nodes to compute the eigenvector, in a distributed fashion, up to a desired accuracy. Note also that the very matrix W being optimized is a doubly stochastic matrix, and can be used in the averaging step in DECENTRALOI. If this is done, as the iterations proceed, the averaging step becomes faster.

From the above discussion, it is clear we have a distributed algorithm that computes an approximate eigenvector, and therefore an approximate subgradient.

3) Convergence Analysis: It now remains to show that the subgradient method converges despite approximation errors in computation of the eigenvector, which spill over into computation of the subgradient. To show this, we will use a result from [24] on the convergence of *approximate* subgradient methods.

Given an optimization problem with objective function f and feasible set S, the approximate subgradient method generates a sequence $\{x^k\}_{k=1}^{\infty} \subset S$ such that

$$x^{k+1} = P_S(x^k - \nu_k g^k), \quad g^k \in \partial_{\epsilon_k} f_S(x^k) \tag{65}$$

where P_S is a projection onto the feasible set, $\nu_k > 0$ is a step size, and

$$\partial_{\epsilon_k} f_S(x^k) = \{g : f_S(x) \geq f_S(x^k) + \langle g, x - x^k \rangle - \epsilon_k \, \forall \, x\} \tag{66}$$

is the ϵ_k-subdifferential of the objective function f_S at x^k.

Let $\gamma_k = (1/2)|g_k|^2 \nu_k$ and $\delta_k = \gamma_k + \epsilon_k$. Then we have the following theorem from [24],

Lemma 7: If $\sum \nu_k = \infty$, then

$$\liminf_k f(x^k) \leq f^* + \delta$$

where $\delta = \limsup \delta_k$, and f^* is the optimal value of the objective function.

Consider the kth iteration of the subgradient method, with current iterate $p(k)$, and let $\sqrt{\epsilon}$ be the error in the (approximate) eigenvector u corresponding to $\lambda_2(W(p(k)))$. (By error in the eigenvector, we mean the L_2 distance between u and the (actual) eigenspace corresponding to λ_2). Again, denote by u_r the vector in the eigenspace minimizing the distance to u, and denote the exact subgradient computed from u_r by g_r.

We have $\|u - u_r\|^2 \leq \epsilon$. First, we find ϵ_k in terms of ϵ as follows:

$$\begin{aligned} \lambda_2(W(p)) &\geq \lambda_2(W(p(k))) + \langle g_r, p - p(k) \rangle \\ &= \lambda_2(W(p(k))) + \langle g, p - p(k) \rangle \\ &\quad - \langle g - g_r, p - p(k) \rangle. \end{aligned}$$

Therefore,

$$\begin{aligned} \epsilon_k &= \sup_p \langle g - g_r, p - p(k) \rangle \\ &= c\|g - g_r\|^2 \end{aligned}$$

where c is a scaling constant.

Next, we will find $\|g - g_r\|^2$ in terms of ϵ as follows:

$$\begin{aligned} \|u - u_r\|^2 \leq \epsilon &\Rightarrow \sum_{i=1}^{n} (u_i - u_{r_i})^2 \leq \epsilon \\ &\Rightarrow (u_i - u_{r_i})^2 \leq \epsilon, \quad 1 \leq i \leq n. \end{aligned}$$

The lth component of $g - g_r$ is

$$\begin{aligned} (g - g_r)_l &= \frac{1}{2n} \left((u_i - u_j)^2 - (u_{r_i} - u_{r_j})^2\right) \\ &= \frac{1}{2n} ((u_i - u_{r_i}) - (u_j - u_{r_j})) \\ &\quad \cdot ((u_i - u_j) + (u_{r_i} - u_{r_j})). \end{aligned}$$

Combining the facts that

$$|u_i - u_{r_i}| \leq \sqrt{\epsilon}, \quad \forall \, i;$$

and (since $||u|| = 1$)

$$|u_i - u_j| \leq \sqrt{2}, \quad \forall\, i, j$$

we get

$$(g - g_r)_l^2 \leq \frac{1}{4n^2}(2\sqrt{\epsilon})^2(2\sqrt{2})^2 = 8\epsilon/n^2.$$

Summing over all m edges gives us $\|g - g_r\|^2 \leq 8m\epsilon/n^2$, i.e., $\epsilon_k \leq 8m\epsilon/n^2$.

Now choose $\nu_k = 1/k$. From (57), it can be seen that $\|g_k\|^2$ is bounded above by $\sqrt{m}/n$, and so γ_k in Theorem 7 converges to 0. Therefore, if in each iteration i, the eigenvector is computed to within an error of ϵ_i, and $\epsilon = \liminf \epsilon_i$, we have the claimed result.

Remark: The fact that each constraint in (55) is local is crucial to the existence of a distributed algorithm using the subgradient method. The proof of convergence of the subgradient method relies on the fact that the *distance* to the optimal set decreases at each iteration. This means that an exact projection needs to be computed at each step: if only an approximate projection can be computed, this crucial property of decreasing the distance to the optimal set cannot be verified.

Thus, for example, if the algorithm were formulated in terms of picking one of all possible edges at random at each step, the constraint would be $\sum_{(i,j)\in E} P_{ij} = 1$, which is not a local constraint. Although this algorithm has a larger feasible set than the optimization problem for the algorithm $\mathcal{A}$, it does not allow for a distributed computation of the optimal algorithm: though the projection can be computed approximately by distributed averaging, an exact projection cannot be computed, and the convergence of the subgradient method is not guaranteed.

V. Averaging Time and Mixing Time

In this section, we explore the relation between the averaging time of an algorithm $\mathcal{A}(P)$ with a symmetric probability matrix P, and the mixing time of the Markov chain with transition matrix P. Since we assume that P is symmetric, the Markov chain with transition matrix P has a uniform equilibrium distribution.

Definition 2 (Mixing Time): For a Markov chain with transition matrix P, let $\Delta_i(t) = \frac{1}{2}\sum_{j=1}^{n}\left|P_{ij}^t - \frac{1}{n}\right|$. Then, the ϵ-mixing time is defined as

$$T_{\text{mix}}(\epsilon) = \sup_i \inf\{t : \Delta_i(t') \leq \epsilon, \forall\, t' \geq t\}. \tag{67}$$

Recall also the following well-known bounds on the ϵ-mixing time for a Markov chain (see, for example, the survey [15]).

Lemma 8: The ϵ-mixing time of a Markov chain with doubly stochastic transition matrix P is bounded as

$$\frac{\lambda_{\max}(P)\log(2\epsilon)^{-1}}{2(1-\lambda_{\max}(P))} \leq T_{\text{mix}}(\epsilon) \leq \frac{\log n + \log \epsilon^{-1}}{1-\lambda_{\max}(P)}. \tag{68}$$

For $\epsilon = o(1/n)$, (68) becomes

$$\Omega\left(\frac{\lambda_{\max}(P)\log n}{2(1-\lambda_{\max}(P))}\right) \leq T_{\text{mix}}(\epsilon) \leq O\left(\frac{\log n}{1-\lambda_{\max}(P)}\right). \tag{69}$$

In the rest of the paper, if we do not specify ϵ, we mean $\epsilon = o(1/n)$; the corresponding mixing time $T_{\text{mix}}(\epsilon)$ is denoted simply as T_{mix}.

We use Lemma 8 and Theorem 3 to prove the following theorem.

Theorem 7: The averaging time of the gossip algorithm $\mathcal{A}(P)$ in absolute time is related to the mixing time of the Markov chain with transition matrix $\tilde{P} = \frac{1}{2}(I + P)$ as

$$T_{\text{ave}}(\epsilon) = \Theta(\log n + T_{\text{mix}}(\epsilon, \tilde{P})).$$

Proof: Let $\epsilon = 1/n^\delta$. It is shown in [29] that $T_{\text{ave}}(\epsilon) = \Omega(\log n)$ for $\epsilon < 1/2$. Since P is symmetric, we can use the result in Corollary 1, so that in absolute time, for $\epsilon = 1/n^\delta$

$$T_{\text{ave}}(\epsilon) = \Theta\left(\frac{\log n}{1-\lambda_2(P)}\right).$$

We will first show that $T_{\text{ave}}(\epsilon) = \Omega(\log n + T_{\text{mix}}(\epsilon, \tilde{P}))$. Using the result of [29] and Corollary 1, we already have that

$$T_{\text{ave}}(\epsilon) = \Omega\left(\log n + \frac{\log n}{1-\lambda_2(P)}\right). \tag{70}$$

Note that the eigenvalues of $\tilde{P}$ are all positive, so that $\lambda_{\max}(\tilde{P}) = \lambda_2(P)$. There are two cases to consider.

- $\lambda_2(\tilde{P}) \leq 1/4$: [4] In this case, by Lemma 8, $T_{\text{mix}}(\epsilon, \tilde{P}) = O(\log n)$. Further, $\frac{\log n}{1-\lambda_2(P)} = O(\log n)$. It follows that $T_{\text{ave}}(\epsilon) = \Omega(\log n + T_{\text{mix}}(\epsilon))$.
- $\lambda_2(\tilde{P}) > 1/4$: From Lemma 8, we get

$$T_{\text{mix}}(\epsilon, \tilde{P}) = \Theta\left(\frac{\log n}{1-\lambda_2(\tilde{P})}\right) \tag{71}$$

$$= \Theta\left(\frac{\log n}{\frac{1}{2}(1-\lambda_2(P))}\right). \tag{72}$$

Combining this with (70), we see that $T_{\text{ave}}(\epsilon) = \Omega(\log n + T_{\text{mix}}(\epsilon, \tilde{P}))$.

Now we will show that $T_{\text{ave}}(\epsilon) = O(\log n + T_{\text{mix}}(\epsilon, \tilde{P}))$, which will give us our result. Again we consider the same two cases.

- If $\lambda_2(\tilde{P}) < 1/4$, then by (1) and Lemma 8

$$T_{\text{ave}}(\epsilon) \leq \frac{3\delta \log n}{1-\lambda_2(P)} \tag{73}$$

$$\leq 3\delta\left(\log n + \frac{\lambda_2(P)\log n}{1-\lambda_2(P)}\right) \tag{74}$$

$$= \Theta\left(\log n + \frac{\lambda_2(\tilde{P})\log n}{1-\lambda_2(P)}\right) \tag{75}$$

$$= O(\log n + T_{\text{mix}}(\epsilon, \tilde{P})). \tag{76}$$

- If $\lambda_2(\tilde{P}) > 1/4$, then using Lemma 8, and $\lambda_2(\tilde{P}) = \frac{1}{2}(1 + \lambda_2(P)$

$$T_{\text{mix}}(\epsilon, \tilde{P}) = \Theta\left(\frac{\log n}{1-\lambda_2(P)}\right) \tag{77}$$

so that

$$T_{\text{ave}}(\epsilon) \leq \frac{3\delta \log n}{1-\lambda_2(P)} = O(\log n + T_{\text{mix}}(\epsilon, \tilde{P})).$$

[4] The specific value $\frac{1}{4}$ is not crucial; we could have chosen any $a > 0$ instead.

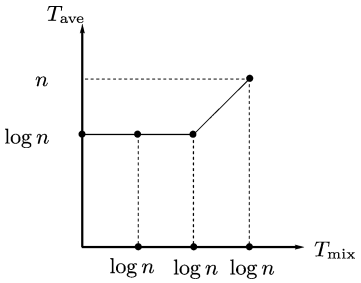

Fig. 2. Graphical interpretation of Theorem 7.

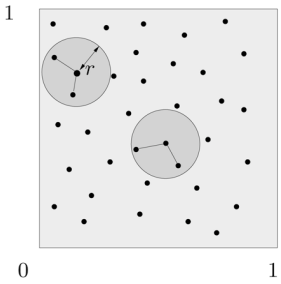

Fig. 3. An example of a Geometric Random Graph in two dimensions. A node is connected to all other nodes that are within the distance r of itself.

Combining the two results gives us the theorem. □

Fig. 2 is a pictorial description of Theorem 7. The x-axis denotes mixing time and the y-axis denotes averaging time. The scale on the axis is in order notation. As shown in the figure, for P such that $T_{\text{mix}}(\tilde{P}) = o(\log n)$, $T_{\text{ave}}\left(\frac{1}{n}, P\right) = \Theta(\log n)$; for P such that $T_{\text{mix}}(\tilde{P}) = \Omega(\log n)$, $T_{\text{ave}}\left(\frac{1}{n}, P\right) = \Theta(T_{\text{mix}})$. Thus, the mixing time of the random walk essentially characterizes the averaging time of the corresponding averaging algorithm on the graph.

VI. Applications

In this section, we briefly discuss applications of our results in the context of wireless *ad hoc* networks and the Internet.

A. Wireless Networks

The Geometric Random Graph, introduced by Gupta and Kumar [13], has been used successfully to model *ad hoc* wireless networks. A d-dimensional Geometric Random Graph on n nodes, denoted $G^d(n,r)$, models a wireless *ad hoc* network of n nodes with wireless transmission radius r. It is obtained as follows: place n nodes on a d–dimensional unit cube uniformly at random and connect any two nodes that are within distance r of each other. An example of a two-dimensional graph, $G^2(n,r)$ is shown in Fig. 3. The following is a well-known result about the connectivity of $G^d(n,r)$ (for a proof, see [13], [14], [40]).

Lemma 9: For $nr^d \geq 2\log n$, the $G(n,r)$ is connected with probability at least $1 - 1/n^2$.

We have the following results for averaging algorithms on a wireless sensor network, which are stated at the end of this section as Theorem 9. (We will prove these by evaluating the mixing times for the natural and optimal random walks on geometric random graphs, and then using Theorem 7, which relates averaging times and mixing times.)

- On the Geometric Random Graph, $G^d(n,r)$, the absolute $1/n^\alpha$-averaging time $\alpha > 0$ of the optimal averaging algorithm is $\Theta\left(\frac{\log n}{r^2}\right)$.

Thus, in wireless sensor networks with a small radius of communication, distributed computing is necessarily slow, since the fastest averaging algorithm is itself slow. However, consider the natural averaging algorithm, based on the natural random walk, which can be described as follows: each node, when it becomes active, chooses one of its neighbors uniformly at random and averages its value with the chosen neighbor.

We have noted before that, in general, the performance of such an algorithm can be far worse than the optimal algorithm. Interestingly, in the case of $G^d(n,r)$, the performances of the natural averaging algorithm and the optimal averaging algorithm are comparable (i.e., they have averaging times of the same order). We will show the following result for the natural averaging algorithm on geometric random graphs.

- In the Geometric Random Graph, $G^d(n,r)$, the absolute $1/n^\alpha$-averaging time $\alpha > 0$ of the natural averaging algorithm is of the same order as the optimal averaging algorithm, i.e., $\Theta\left(\frac{\log n}{r^2}\right)$.

We now prove the following theorem about the mixing times of the optimal and natural random walks on $G^d(n,r)$.

Theorem 8: For $G^d(n,r)$ with $r = \omega(r_c(d))$, with high probability

a) the mixing time of the optimal reversible random walk with uniform stationary distribution is $\Theta\left(r^{-2}\log n\right)$; and
b) the mixing time of the modified natural random walk, where a node jumps to any of its neighbors (other than itself) with equal probability, and has a self-loop of probability $1/2$, is also $\Theta\left(r^{-2}\log n\right)$.

The outline of the proof is as follows. To prove a), we will start by showing that with high probability, the geometric random graph is a regular graph. We bound the mixing rate of the optimal random walk on the corresponding regular graph, and then relate the mixing times of the optimal random walks on this regular graph and the $G^d(n,r)$ graph. The proof of b) uses a modification of the path counting argument of Diaconis and Stroock to upper-bound the second largest eigenvalue of the natural random walk on the $G^d(n,r)$ graph.

We start with evaluating the mixing time of the optimal random walk on $G^d(n,r)$.

1) Regularity of $G^d(n,r)$: In this subsection, we prove a regularity property of $G^d(n,r)$, which allows a simpler analysis of the mixing time of random walks.

Lemma 10: For $G^d(n,r)$ with $r = \omega(r_c(d))$, the degree of every node is $\alpha_d n r^d(1 + o(1))$ with high probability, where $\alpha_d = \frac{\pi^{d/2}}{\Gamma(1+d/2)}$.

Proof: Let nodes be numbered $i = 1, \ldots, n$. Consider a particular node, say 1. Let random variable X_j be 1 if node j is within distance r of node 1 and 0 otherwise. The X_j's are i.i.d.

Bernoulli with probability $p_d = \alpha_d r^d$ of success (the volume of a d-dimensional sphere with radius r is $\alpha_d r^d$). The degree of node 1 is

$$d_1 = \sum_{j=2}^{n} X_j. \tag{78}$$

By application of the Chernoff bound we obtain

$$P\left(\left|\sum_{j=2}^{n} X_j - (n-1)p_d\right| \geq \delta(n-1)p_d\right) \leq 2\exp\left(-\frac{\delta^2(n-1)p_d}{2}\right). \tag{79}$$

If we choose $\delta = \frac{\sqrt{2c\log n}}{\sqrt{p_d(n-1)}}$, then the right-hand side in (79) becomes $2\exp(-c\log n) = 2/n^c$. So, for $p_d = \omega(\log n/n)$, node 1 has degree

$$\begin{aligned} d_1 &= (n-1)p_d \pm \sqrt{2c(n-1)p_d \log n} \\ &\simeq np_d(1 \pm o(1)), \quad \text{w.p.} \geq 1 - \frac{2}{n^c}. \end{aligned} \tag{80}$$

Using the union bound, we see that

$$\begin{aligned} P(\text{any node has degree} \neq np_d(1 \pm o(1))) &\leq n\frac{2}{n^c} \\ &= \frac{2}{n^{c-1}}. \end{aligned} \tag{81}$$

So for large n, *w.h.p.* (with high probability), all nodes in the d-dimensional $G(n,r)$ have degree $np_d(1 \pm o(1))$. □

2) Proof of Theorem 8 a): Optimal Random Walk on $G^d(n,r)$: In this subsection, we characterize the scaling of the optimal random walk on $G^d(n,r)$. We first consider the case of $d = 1$, i.e., $G^1(n,r)$. This is much easier than the higher dimensional $G^d(n,r)$ with $d \geq 2$. We completely characterize $G^1(n,r)$ with the help of one-dimensional regular graphs. For $G^d(n,r)$ with $d \geq 2$, we obtain a lower bound on the fastes mixing reversible random walk. Note that since we are interested in reversible random walks with uniform stationary distribution, the transition matrix corresponding to the random walk must be symmetric. (An upper bound of the same order is implied by the natural random walk as in Theorem 8 b).) The remainder of the section is a proof of Theorem 8 a).

Optimal random walk on $G^1(n,r)$

Let G_k denote the regular graph on n nodes with every node of degree $2k$; it is constructed by placing the n nodes on the circumference of a circle, and connecting every node to k neighbors on the left, and k on the right. From the regularity lemma, we have that *w.h.p.*, every node in $G(n,r)$ has degree $2nr(1 \pm o(1))$. Also, observe that the same technique can be used to show that *w.h.p.* the number of neighbors to the right (ditto left) is $nr(1 \pm o(1))$.

In this one-dimensional case, it is clear that *w.h.p.*, the $G(n,r)$ is a subgraph of G_k for $k = 4nr$, since for any mapping of the nodes of $G(n,r)$ to G_k, an edge between nodes i and j in $G(n,r)$ is also present in G_k. Similarly, $G(n,r)$ also contains G_l, for $l = (1/2)nr$. Given this, we can now study the problem of finding the optimal random walk on G_k with uniform stationary distribution. We have the following lemma.

Lemma 11: For k, n such that $k \leq n/4$, the mixing rate of the fastest mixing symmetric random walk on G_k cannot be smaller than $\cos(2\pi k/n)$.

Proof: It can be shown using symmetry arguments [41] that the fastest mixing Markov chain on G_k with uniform stationary distribution will have a symmetric and circulant transition matrix. (For this simple graph, this can be easily seen using convexity of the second eigenvalue). So we can restrict our attention to the (circulant symmetric) transition matrices given in (82) at the bottom of the page. The eigenvalues of this matrix are

$$\begin{aligned} \mu_m &= \sum_{j=0}^{k} p_j e^{-2\pi i jm/n} + \sum_{j=1}^{k} p_j e^{-2\pi i(n-j)m/n} \\ &= p_0 + 2\sum_{j=1}^{k} p_j \cos\left(\frac{2\pi jm}{n}\right), \quad m = 0, \ldots, n-1. \end{aligned}$$

For $m = 0$, $\mu_m = 1$, which is the largest eigenvalue. Let $\boldsymbol{p} = (p_0, p_1, \ldots, p_k, p_k, \ldots, p_1)$. We are interested in the smallest possible second largest eigenvalue in absolute value, i.e., in

$$\begin{aligned} \text{minimize}_{\boldsymbol{p}} \quad & \max_{m=\{1,\ldots,n-1\}} |\mu_m| \\ \text{subject to} \quad & \mathbf{1}^T\boldsymbol{p} = 1 \\ & \boldsymbol{p} \succeq 0. \end{aligned} \tag{83}$$

We can obtain a lower bound for the optimal value of (83). Now

$$\begin{aligned} \mu_1 &\leq \max_{m=\{1,\ldots,n-1\}} |(\mu_m)| \\ \Rightarrow \min_{\boldsymbol{p}} \mu_1 &\leq \min_{\boldsymbol{p}} \max_{m=\{1,\ldots,n-1\}} |(\mu_m)|. \end{aligned} \tag{84}$$

The right-hand side is the solution of the following linear program with a single total sum constraint:

$$\begin{aligned} \text{minimize}_{\boldsymbol{p}} \quad & p_0 + 2\sum\nolimits_{j=1}^{k} p_j \cos(2\pi j/n) \\ \text{subject to} \quad & \mathbf{1}^T\boldsymbol{p} = 1 \\ & \boldsymbol{p} \succeq 0. \end{aligned} \tag{85}$$

For k such that each of the coefficients $\cos(2\pi j/n)$ is positive, i.e., for $k \leq n/4$, the smallest coefficient is $\cos(2\pi k/n)$, and so for all such k and n, the minimum value is $\cos(2\pi k/n)$, obtained

$$P = \begin{bmatrix} p_0 & p_1 & \ldots & p_k & 0 & 0 & \ldots & 0 & p_k & \ldots & p_2 & p_1 \\ p_1 & p_0 & p_1 & \ldots & p_k & 0 & \ldots & 0 & 0 & p_k & \ldots & p_2 \\ & & & \vdots & & & & & & \vdots & & \\ & & & \vdots & & & & & & \vdots & & \\ p_1 & \ldots & p_k & 0 & 0 & \ldots & 0 & p_k & \ldots & p_2 & p_1 & p_0 \end{bmatrix}. \tag{82}$$

at $p_k = 1/2$, $p_j = 0$ for all other j.[5] So the fastest mixing random walk on this graph cannot have a mixing rate smaller than $\cos(2\pi k/n)$. □

The preceding result was proved for all $k \leq n/4$; however, we will be interested only in those cases where $k = o(n)$, i.e., the graph is not too well connected. For such k, the following lemma allows us to find a "nearly optimal" transition matrix.

Lemma 12: For $k = o(n)$, there is a random walk on G_k for which the mixing rate is $\lambda_{\max} = \cos(2\pi k/n) + \Theta(k^4/n^4)$.

Proof: For simplicity, let us assume that $2k$ divides n; it is not difficult to obtain the same results when this is not the case.

Consider the Markov chain with transition probabilities $p_0 = 0$, $p_i = \delta, i = 1, \ldots, k-1$, $p_k = 1/2 - (k-1)\delta$. We will show that for a certain δ, small enough, μ_1 is indeed $\lambda_{\max}$, and is away from $\cos(2\pi k/n)$ by $\Theta(k^4/n^4)$.

For the transition matrix P^* corresponding to these probabilities, the eigenvalues are, for $m = 0, \ldots, n-1$

$$\begin{aligned}\mu_m =& 2\sum_{i=1}^{k-1} \delta \cos\left(\frac{2\pi i m}{n}\right) \\ &+ 2\left(\frac{1}{2} - (k-1)\delta\right)\cos\left(\frac{2\pi k m}{n}\right) \qquad (86)\\ =& \cos\left(\frac{2\pi k m}{n}\right) \\ &+ 2\delta \sum_{i=1}^{k-1}\left(\cos\left(\frac{2\pi i m}{n}\right) - \cos\left(\frac{2\pi k m}{n}\right)\right). \qquad (87)\end{aligned}$$

We want to find the smallest positive δ such that μ_1 is $\lambda_{\max}$ (this is not true, for example, for $\delta = 0$). However, we need δ to be small enough so that the residual term, $2\delta\sum_{i=1}^{k-1}(\cos(2\pi i/n) - \cos(2\pi k/n))$, is small compared to $\cos(2\pi k/n)$.

Since $k = o(n)$ and we hope that δ is small $(o(1))$, we see that the values of m for which $|\mu_m|$ is comparable to μ_1 are those values of m for which $|\cos(2\pi k m/n)| = 1$. This happens for $m = \frac{n}{2k}, \frac{n}{k}, \frac{3n}{2k}, \ldots, \frac{n}{2}$. (We only need consider values of m until $n/2$, since $\lambda_i = \lambda_{n-i}$.) At all odd multiples of $n/2k$, $\cos(2\pi k m/n) = -1$, and for the even multiples, $\cos(2\pi k m/n) = 1$. For δ to satisfy $|\mu_m| \leq \mu_1$, we must have for m an even multiple of $n/2k$

$$\begin{aligned}1 + 2\delta\sum_{i=1}^{k-1}\left(\cos\left(\frac{2\pi i m}{n}\right) - 1\right) \leq& \cos\left(\frac{2\pi k}{n}\right) \\ &+ 2\delta\sum_{i=1}^{k-1}\left(\cos\frac{2\pi i}{n} - \cos\frac{2\pi k}{n}\right); \qquad (88)\end{aligned}$$

and for m an odd multiple of $n/2k$

$$\begin{aligned}\left|-1 + 2\delta\sum_{i=1}^{k-1}\left(\cos\left(\frac{2\pi i m}{n}\right) + 1\right)\right| \leq& \cos\left(\frac{2\pi k}{n}\right) \\ &+ 2\delta\sum_{i=1}^{k-1}\left(\cos\frac{2\pi i}{n} - \cos\frac{2\pi k}{n}\right)\end{aligned}$$

that is,

$$\begin{aligned}1 - 2\delta\sum_{i=1}^{k-1}\left(\cos\left(\frac{2\pi i m}{n}\right) + 1\right) \leq& \cos\left(\frac{2\pi k}{n}\right) \\ &+ 2\delta\sum_{i=1}^{k-1}\left(\cos\frac{2\pi i}{n} - \cos\frac{2\pi k}{n}\right). \qquad (89)\end{aligned}$$

From (88), we see that δ must be greater or equal to

$$\frac{\frac{1}{2}\left(1 - \cos\left(\frac{2\pi k}{n}\right)\right)}{(k-1)\left(1 - \cos\left(\frac{2\pi k}{n}\right)\right) + \sum_{i=1}^{k-1}\cos\left(\frac{2\pi i}{n}\right) + \cos\left(\frac{2\pi i m}{n}\right)} \qquad (90)$$

for m an odd multiple of $n/2k$, and from (89), δ must be less or equal to

$$\frac{\frac{1}{2}\left(1 - \cos\left(\frac{2\pi k}{n}\right)\right)}{(k-1)\left(1 - \cos\left(\frac{2\pi k}{n}\right)\right) + \sum_{i=1}^{k-1}\cos\left(\frac{2\pi i}{n}\right) - \cos\left(\frac{2\pi i m}{n}\right)} \qquad (91)$$

for m a multiple of n/k. So δ can be only as small as the maximum over the specified m of all of these right-hand sides.

Note that the only term dependent on m in each of these expressions is $\sum_{i=1}^{k-1}\cos(2\pi i m/n)$. For $m = pn/2k$, p odd

$$\sum_{i=1}^{k-1}\cos(2\pi i m/n) = \sum_{i=1}^{k-1}\cos(\pi i p/k) = 0 \qquad (92)$$

since $\cos(\pi i p/k) = -\cos(\pi(k-i)p/k)$ for odd p, and if k is even, $\cos(\pi k p/2k) = 0$ also. For $m = qn/k$

$$\sum_{i=1}^{k-1}\cos(2\pi i m/n) = \sum_{i=1}^{k}\cos(2\pi i q/k) - 1 = -1 \qquad (93)$$

since $\sum_{i=1}^{k}\cos(2\pi i q/k) = 0$ (sum of real parts of the kth roots of unity).

So $\delta = \Theta(k/n^2)$, and returning to (86), we see that the residual term in μ_1 is of order $(k/n^2)(k^3/n^2)$, i.e., k^4/n^4, while $\cos(2\pi k/n) \approx 1 - 2\pi^2 k^2/n^2$. So the difference between $\lambda_{\max}$ and $\cos(2\pi k/n)$ is $\Theta(k^4/n^4)$. □

Optimal walk on $G^2(n, r)$

We present the lower bound on the fastest mixing reversible random walk on $G^2(n, r)$ in this section. The same method can be easily extended to $d \geq 3$. First we characterize the fastest mixing reversible random walk on a two-dimensional regular graph G_{kk} defined as follows: form a lattice on the unit torus, where lattice points are located at $(i/\sqrt{n}, j/\sqrt{n})$, $-\sqrt{n}/2 \leq i, j \leq +\sqrt{n}/2$, and place the n nodes at these points. An edge between two vertices exists if the L_∞ distance between them is at most $k/\sqrt{n}$. For such G_{kk} the fastest mixing time scales as follows.

Lemma 13: The mixing rate of the optimal reversible random walk on G_{kk} is no smaller than $\cos^2(2\pi k/\sqrt{n})$.

Proof: As in the one-dimensional case, by symmetry, the optimal transition probability between nodes i and j will depend only on the distance between these nodes. Using this, we can write the transition matrix corresponding to such a symmetric random walk on G_{kk} as the Kronecker (or tensor) product $P_k \otimes$

[5]Note that this is only a lower bound: for this $\boldsymbol{p}$, if k divides n, the second largest eigenvalue is also 1, attained at $m = n/k$.

P_k, where $P_k \in \boldsymbol{R}^{n\times n}$ is as in (82). This is not difficult to visualize: for $i,j=0,\ldots,n-1$, $a,b=1,\ldots,n$

$$(P\otimes P)_{ni+a,nj+b} = P_{i+1,j+1}P_{ab}. \tag{94}$$

Now the eigenvalues of $A\otimes B$ are all products of eigenvalues of A and B, so that for $0\le i,j\le n-1$

$$\begin{aligned}\lambda_{ij}(P\otimes P) &= \lambda_i(P)\lambda_j(P)\\ &= \left(p_0+2\sum_{m=1}^{k} p_m\cos\left(2\pi\frac{im}{\sqrt{n}}\right)\right)\\ &\quad\cdot\left(p_0+2\sum_{m=1}^{k} p_m\cos\left(2\pi\frac{jm}{\sqrt{n}}\right)\right).\end{aligned}$$

The eigenvalue 1 is obtained by setting $i=j=0$; all other eigenvalues will have absolute value less **(or ?)** equal **(to ?)** 1. We want to find a lower bound for the second largest eigenvalue in absolute value, call it $\lambda^{\star}_{\max}$.

As before, choose $i=j=1$. Then

$$\begin{aligned}\lambda_{11} &\le \max_{i,j\neq 0}|\lambda_{ij}|\\ \Rightarrow \min_{\boldsymbol{p}}\ \lambda_{11} &\le \min_{\boldsymbol{p}}\ \max_{i,j\neq 0}|\lambda_{ij}|\end{aligned}$$

so that $\min_{\boldsymbol{p}}\ \lambda_{11}$ is a lower bound for $\lambda^{\star}_{\max}$. Making the assumption again that $k\le\sqrt{n}/4$, the minimizing $\boldsymbol{p}$ is the one with $p_k=1/2$ and $p_i=0, i\neq k$ (which corresponds to transition probabilities of $1/4$ for each of the four farthest diagonal nodes, and 0 everywhere else). The value of λ_{11} corresponding to this distribution is $\cos^2\left(2\pi\frac{k}{\sqrt{n}}\right)$. This is of order $1-\Theta\left(\frac{k^2}{n}\right)$, since[6]

$$\cos^2\left(2\pi\frac{k}{\sqrt{n}}\right)=\frac{1}{2}+\frac{1}{2}\cos\left(2\frac{2\pi k}{\sqrt{n}}\right)=1-\Theta\left(\frac{k^2}{n}\right).\ \square$$

The G_{kk} graph was constructed using the L_∞ distance between vertices. Therefore, the graph formed by placing edges between vertices based on distance measured in *any* L_p norm (for the same k) is a subgraph of G_{kk}, and has a mixing time lower-bounded by the mixing time of G_{kk}. Thus, our bounds will be valid for the $G(n,r)$ graph constructed according to *any* L_p norm.

Now we will use the bound on the fastest mixing walk on G_{11} to obtain a bound for $G^2(n,r)$. First, we create a new graph $\tilde{G}^2(n,r)$ as follows: place a square grid with squares of side r on the unit torus. By Lemma 10, each square of area r^2 contains $nr^2(1+o(1))$ nodes. For each such square, connect every node in this square to all the nodes in the neighboring squares, as well as the nodes in the same square. Thus, each node is connected to $9nr^2(1+o(1))$ nodes in $\tilde{G}^2(n,r)$. By definition, all edges in $G^2(n,r)$ are present in $\tilde{G}^2(n,r)$ and therefore, the fastest mixing random walk on $\tilde{G}^2(n,r)$ is at least as fast as that of $G^2(n,r)$. Thus, lower-bounding the mixing time of the fastest mixing random walk on $\tilde{G}^2(n,r)$ is sufficient.

Construct a graph G of r^{-2} nodes as follows: corresponding to each square in the square grid used in $\tilde{G}^2(n,r)$, create a node in G. Thus, G has r^{-2} nodes. Two nodes are connected in G if the corresponding squares in the grid are adjacent. Thus, each node is connected to eight other nodes. Thus, G is a regular graph G_{11} with r^{-2} nodes. In order to use this bound as a lower bound on $\tilde{G}^2(n,r)$, we need to show that the fastest mixing symmetric random walk on $\tilde{G}^2(n,r)$ induces a time-homogeneous reversible random walk on G. This will be implied by the following lemma.

Lemma 14: There exists a fastest mixing symmetric random walk on $\tilde{G}^2(n,r)$, whose transition matrix P has the following property: for any two nodes i and j belonging to the same square, $P_{ik}=P_{jk}$ for $k\neq i,j$, and $P_{ii}=P_{jj}$.

Proof: We prove this by contradiction. Suppose the claimed statement is not true, i.e., there is no transition matrix achieving the smallest $\lambda_{\max}$ with the above property. Since the optimal value of $\lambda_{\max}$ must be attained ([1]), consider such an optimizing P_1, and let i and j be two nodes in the same square for which the above property is not true.

Let A be the permutation matrix with $A_{ij}=A_{ji}=1$, $A_{ii}=A_{jj}=0$, and all other diagonal entries 1 and all other nondiagonal entries 0. Note that A is a symmetric permutation matrix, and therefore $A^{-1}=A^T=A$. Consider the matrix $P_2=AP_1A$; since $A=A^{-1}$, P_1 and P_2 are similar, and so have the same eigenvalues. Note that since i and j belong to the same square in $\tilde{G}$, they have exactly the same neighbors, and therefore P_2 also respects the graph structure (i.e., $P_{2_{ij}}\neq 0$ only if i and j have an edge between them).

Now, $\lambda_{\max}(P)$ is a convex function of P for symmetric stochastic P ([1]), so

$$\begin{aligned}\lambda_{\max}\left(\frac{P_1+P_2}{2}\right) &\le \frac{1}{2}\lambda_{\max}(P_1)+\frac{1}{2}\lambda_{\max}(P_2)\\ &= \lambda_{\max}(P_1).\end{aligned}\tag{95}$$

But $P=(P_2+P_1)/2$ has the property claimed in the lemma for nodes i and j: $P_{ik}=P_{jk}$ for all $k\neq i,j$, $P_{ii}=P_{jj}=(P_{1_{ii}}+P_{1_{jj}})/2$, and $\lambda_{\max}(P)\le\lambda_{\max}(P_1)$. We can apply the above procedure recursively (even for multiple rows) to construct a matrix P^* with smallest $\lambda_{\max}$ and the property claimed in the lemma. This contradicts our assumption and completes the proof. $\square$

From Lemma 14, we see that under the fastest mixing random walk, the probability of transiting from a node in a square, say A, to some neighboring square, say B, is the same for all nodes in A and B. Thus, essentially we can view the random walk as evolving over squares. That is, the fastest random walk on $\tilde{G}^2(n,r)$ induces a random walk on the graph G. By definition of mixing time, the mixing time for this induced random walk on G (with the induced equilibrium distribution) certainly lower-bounds the mixing time for the random walk on $\tilde{G}^2(n,r)$. Further, the induced random walk is reversible as the random walk was symmetric on $\tilde{G}^2(n,r)$. Therefore, we see that the lower bound on mixing time for the fastest mixing random walk on G implies a lower bound on the mixing time for the fastest mixing random walk on $\tilde{G}^2(n,r)$. From Lemma 13, we have a lower bound of $\Omega(r^{-2}\log n)$ on the mixing time of the fastest mixing symmetric random walk (i.e., with uniform stationary distribution). From Lemma 15, given below, this in turn implies a lower bound of $\Omega(r^{-2}\log n)$ on the mixing time of the fastest

[6]It is easy to see that a result similar to Lemma 12 can be obtained for $d\ge 2$ using the same method.

mixing reversible random walk on $G^2(n,r)$. This completes the proof of 2 a) **(please define "2 a")** for $G^2(n,r)$. It is easy to see that the arguments presented above can readily be extended to the case of $d \geq 3$.

Lemma 15: Consider a connected graph

$$G = (\{1,\ldots,n\}, E)$$

with diameter $\Omega(\log n)$. Let $T^*_{\text{mix}}(\pi)$ be the mixing time of the fastest mixing reversible random walk on G with stationary distribution $\pi = [\pi(1)\ldots\pi(n)]^T$. Let $\beta(\pi) = \max_{i,j} \frac{\pi(i)}{\pi(j)} \leq C$, where C is a constant. Then

$$T^*_{\text{mix}}(\pi) = \Omega\left(T^*_{\text{mix}}\left(\frac{1}{n}\mathbf{1}\right)\right) \tag{96}$$

i.e., the fastest mixing time for π is no faster than that of the uniform distribution.

Proof: Consider a reversible random walk with stationary distribution π on G and let its transition matrix be R. We will prove the following claim, which in turn implies the statement of the lemma.

Claim I: There exists a symmetric random walk on graph G with transition matrix S such that

$$T_{\text{mix}}(S) = O(T_{\text{mix}}(R)).$$

Proof of Claim I: For a reversible matrix R, by definition

$$\pi(i)R(i,j) = \pi(j)R(j,i), \quad \forall\, i,j.$$

Define matrix $P = [P(i,j)]$, where for $i \neq j$

$$P(i,j) = \begin{cases} R(i,j), & \text{if } \pi(i) \geq \pi(j) \\ \pi(i)R(j,i)/\pi(j), & \text{if } \pi(i) < \pi(j) \end{cases}$$

and $P(i,i) = 1 - \sum_{j\neq i} P(i,j)$.

By definition and reversibility of R, P is a symmetric doubly stochastic matrix. Further, for $i \neq j$, $P(i,j) > 0$ if and only if $R(i,j) > 0$. Hence, P can be viewed as a transition matrix of a symmetric random walk on G, whose stationary distribution is uniform. Define $Q^R = [Q^R(i,j)]$, where

$$Q^R(i,j) = \pi(i)R(i,j) = \pi(j)R(j,i).$$

Similarly, define $Q^P = \frac{1}{n}P$. Let $\phi : \{1,\ldots,n\} \to \boldsymbol{R}$ be a nonconstant function. Define two quadratic forms, $\mathcal{E}^R$ and $\mathcal{E}^P$, of ϕ, as

$$\mathcal{E}^R(\phi,\phi) = \frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2 Q^R(i,j)$$
$$\mathcal{E}^P(\phi,\phi) = \frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2 Q^P(i,j).$$

Let the variance of ϕ with respect to these two random walks be

$$V^R(\phi) = \frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2\pi(i)\pi(j)$$
$$V^P(\phi) = \frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2\frac{1}{n^2}.$$

Let $\lambda_2(P)$ and $\lambda_2(R)$ denote the second largest eigenvalue of matrices P and R, respectively. The minimax characterization of eigenvalues ([18, p. 176]), gives a bound on the second largest eigenvalue of a reversible matrix $X(=P,R)$ as

$$(1-\lambda_2(X)) = \inf\left\{\frac{\mathcal{E}^X(\phi,\phi)}{V^X(\phi)} \mid \phi \text{ a nonconstant}\right\}. \tag{97}$$

For any π, $\sum_i \pi(i) = 1$, hence, $\max_i \pi(i) \geq 1/n$ and $\min_j \pi(j) \leq 1/n$. Further, by the property of π

$$\max_{i,j}\frac{\pi(i)}{\pi(j)} = \frac{\max_i \pi(i)}{\min_j \pi(j)} < C.$$

Hence, for any k

$$\pi(k) \geq \min_i \pi(i) \geq \frac{\max_i \pi(i)}{C} \geq \frac{1}{nC} \quad \text{and}$$
$$\pi(k) \leq \max_i \pi(i) \leq C \min_j \pi(j) \leq \frac{C}{n}.$$

Thus, for any k

$$\frac{\pi(k)}{1/n} \in \left(\frac{1}{C}, C\right).$$

This implies that

$$\frac{\mathcal{E}^R(\phi)}{\mathcal{E}^P(\phi)} \in \left(\frac{1}{C^2}, C^2\right); \quad \frac{V^R(\phi)}{V^P(\phi)} \in \left(\frac{1}{C^2}, C^2\right).$$

Hence, from (97) we obtain

$$(1-\lambda_2(P)) = \Theta(1-\lambda_2(R)).$$

Since the diameter of G is $\Omega(\log n)$, it is easy to see that the mixing time of all random walks on G is lower-bounded by $\Omega(\log n)$. Hence, from Lemma 8

$$T_{\text{mix}}(R) = \Theta\left(\frac{\log n}{1-\lambda_{\max}(R)}\right).$$

By definition, $(1-\lambda_{\max}(R)) \leq (1-\lambda_2(R))$. Hence,

$$T_{\text{mix}}(R) = \Omega\left(\frac{\log n}{1-\lambda_2(R)}\right).$$

It is easy to see that the random walk on G with symmetric transition matrix $S = (I+P)/2$ has mixing time given by

$$T_{\text{mix}}(S) = \Theta\left(\frac{\log n}{1-\lambda_2(P)}\right).$$

Thus, $T_{\text{mix}}(S) = O(T_{\text{mix}}(R))$. This completes the proof of Claim I and the proof of Lemma 15. □

Remark: In fact, a stronger result can be proved, which is

$$T^*_{\text{mix}}(\pi) = \Theta\left(T^*_{\text{mix}}\left(\frac{1}{n}\mathbf{1}\right)\right).$$

One part of this has already been proved in the lemma. The reverse direction is obtained similarly, as follows. Consider any symmetric random walk with transition matrix P, and suppose a stationary distribution π is specified, satisfying $\beta(\pi) = \max_{i,j}\frac{\pi(i)}{\pi(j)} \leq C$, where C is some constant. Then there is a reversible random walk $\bar{R}$ with stationary distribution

π, such that $T_{\text{mix}}(\bar{R}) = O(T_{\text{mix}}(P))$. $\bar{R}$ is obtained as follows. Construct a matrix R from P as

$$R(i,j) = \begin{cases} P(i,j), & \text{if } \pi(i) \geq \pi(j) \\ \pi(i)P(i,j)/\pi(j), & \text{if } \pi(i) < \pi(j) \end{cases}$$

for $i \neq j$, and $R_{ii} = 1 - \sum_{j \neq i} R_{ij}$. R is a stochastic reversible matrix, with stationary distribution π, since $\pi(i)R(i,j) = \pi(j)R(j,i)$. Following the same steps as above, we can conclude that

$$1 - \lambda_2(R) = \Theta(1 - \lambda_2(P)).$$

The matrix $\bar{R} = (I+R)/2$ has the same eigenvectors as R and, therefore, the same stationary distribution π. The eigenvalues are $(1 + \lambda_2(R))/2$. Therefore, since the diameter of the graph is $\Omega(\log n)$

$$T_{\text{mix}}(\bar{R}) = \Theta\left(\frac{\log n}{1 - \lambda_2(R)}\right).$$

As before

$$T_{\text{mix}}(P) = \Theta\left(\frac{\log n}{1 - \lambda_{\max}(P)}\right) = \Omega\left(\frac{\log n}{1 - \lambda_2(P)}\right).$$

Therefore, $T_{\text{mix}}(\bar{R}) = O(T_{\text{mix}}(P))$, and we have the stronger result claimed in the Remark.

3) Proof of Theorem 8 b): Natural Random Walk on $G^d(n,r)$: In this subsection, we study the mixing properties of the natural random walk on $G^d(n,r)$. Recall that under the natural random walk, the next node is equally likely to be any of the neighboring nodes. It is well known that under the stationary distribution, the probability of the walk being at node i is proportional to the degree of node i. By Lemma 10, all nodes have almost equal degree. Hence, the stationary distribution is almost uniform (it is uniform asymptotically). The rest of this section is the proof of Theorem 8 b).

We use a modification of a method developed by Diaconis–Stroock [9] to obtain bounds on the second largest eigenvalue using the geometry of the $G^d(n,r)$.

Note that for $d = 1$, the proof is rather straightforward. The difficulty arises in the case of $d \geq 2$. For ease of exposition in the rest of the section, we consider $d = 2$. Exactly the same argument can be used for $d > 2$. We begin with some initial setup and notation.

Square Grid: Divide the unit torus into a square grid where each square is of area $r^2/16$, i.e., of side length $r/4$. Consider a node in a square. By definition of $G(n,r)$, this node is connected to all nodes in the same square and all neighboring squares.

Paths and Distribution: A path between two nodes i and j, denoted by γ_{ij}, is a sequence of nodes $(i, v_1, \ldots, v_{l-1}, j)$, $l \geq 1$, such that $(i,v_1), \ldots, (v_{l-1}, j)$ are edges in $G^2(n,r)$. Let $\gamma = (\gamma_{ij})_{1 \leq i \neq j \leq n}$ denote a collection of paths for all $\binom{n}{2}$ node pairs. Let Γ be the collection of all possible γ. Consider the probability distribution induced on Γ by selecting paths between all node pairs as described below.

- Paths are chosen independently for different node pairs.
- Consider a particular node pair (i,j). Let i belong to square C_0 and j belong to square C_l.
  - — If $C_0 = C_l$ or i and j are in neighboring cells then the path between i and j is (i,j).
  - — Else, let $C_1, \ldots, C_{l-1}, l \geq 2$ be other squares lying on the straight line joining i and j. Select a node $v_k \in C_k$, $k = 1, \ldots, l-1$ uniformly at random. Then the path between i and j is $(i, v_1, \ldots, v_{l-1}, j)$.

Under the above setup, we claim the following lemma.

Lemma 16: Under the probability distribution on Γ as described above, the average number of paths passing through an edge is $O(1/r^3)$ *w.h.p.*, where $r = \omega(r_c(d))$.

Proof: We will compute the average load in order notation. Similar to the arguments of Lemma 10, it can be shown that each of the $16/r^2$ squares contains $\frac{nr^2(1+o(1))}{16}$ nodes and each node has degree $nr^2(1+o(1))$ *w.h.p.* We restrict our consideration to such instances of $G^2(n,r)$.

Now the total number of paths are $\Theta(n^2)$ since there are $\binom{n}{2}$ node pairs. Each path contains $O(1/r)$ edges, as $O(1/r)$ squares can be lying on a straight line joining two nodes. The total number of squares is $\Theta(1/r^2)$. Hence, by symmetry and regularity, the number of paths passing through each square is $\Theta(n^2 r)$. Consider a particular square C. For C, at least $1 - \Theta(r^2)(\approx 1)$ fraction of paths passing through it have endpoints lying in squares other than C. That is, most of the paths passing through C have C as an intermediate square, and not an originating square. Such paths are equally likely to select any of the nodes in C. Hence, the average number of paths containing a node, say 1, in C, is $\Theta(n^2 r/nr^2) = \Theta(n/r)$. The number of edges between 1 and neighboring squares is $\Theta(nr^2)$. By symmetry, the average load on an edge incident on 1 will be $\Theta(1/r^3)$. This is true for all nodes. Hence, the average load on an edge is at most $O(1/r^3)$. □

Next we will use this setup and Lemma 16 to obtain a bound on the second largest eigenvalue using a modified version of Poincare's inequality stated as follows.

Lemma 17: Consider the natural random walk on a graph $G = (\{1, \ldots, n\}, E)$ with Γ the set of all possible paths on all node pairs. Let γ_* be the maximum path length (among all paths and over all node pairs), d_* be the maximum node degree, and $|E|$ be the total number of edges. Let, according to some probability distribution on Γ, the maximum average load on any of the edges be b, i.e., on average no edge belongs to more than b paths. Then, the second largest eigenvalue, λ_2, is bounded above as

$$\lambda_2 \leq 1 - \left(\frac{2|E|}{d_*^2 \gamma_* b}\right). \tag{98}$$

Proof: The proof follows from a modification of Poincare's inequality ([9, Proposition 1]). Before proceeding to the proof, we introduce some notation.

Let $\phi : \{1, \ldots, n\} \to \boldsymbol{R}$ be a real-valued function on the n nodes. Let $\pi = (\pi(i))_{\{1 \leq i \leq n\}}$ denote the equilibrium distribution of the random walk. Let d_i be the degree of node i, then it is well known that $\pi(i) = \frac{d_i}{2|E|} \leq \frac{d_*}{2|E|}$. For node pair (i,j), let

$$Q(i,j) = \pi(i)P_{ij} = \pi(j)P_{ji} = 1/2|E|.$$

Define the quadratic form of ϕ as

$$\mathcal{E}(\phi,\phi)=\frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2Q(i,j).$$

Let the variance of ϕ with respect to π be

$$V(\phi)=\frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2\pi(i)\pi(j).$$

For a directed edge e from $i\to j$, define $\phi(e)=\phi(i)-\phi(j)$ and $Q(e)=Q(i,j)$. First, consider one collection of paths $\gamma=(\gamma_{ij})$. Define

$$|\gamma_{ij}|_Q=\sum_{e\in\gamma_{ij}}Q(e)^{-1}.$$

Then, under the natural random walk

$$|\gamma_{ij}|_Q=|\gamma_{ij}|(2|E|) \tag{99}$$

where $|\gamma_{ij}|$ is the length of the path γ_{ij}

$$\begin{aligned}V(\phi)&=\frac{1}{2}\sum_{i,j}(\phi(i)-\phi(j))^2\pi(i)\pi(j)\\&\overset{(a)}{=}\frac{1}{2}\sum_{i,j}\left(\sum_{e\in\gamma_{ij}}\left(\frac{Q(e)}{Q(e)}\right)^{1/2}\phi(e)\right)^2\pi(i)\pi(j)\\&\overset{(b)}{\le}\frac{1}{2}\sum_{i,j}|\gamma_{ij}|_Q\pi(i)\pi(j)\sum_{e\in\gamma_{ij}}Q(e)\phi(e)^2\\&\le\left(\frac{d_*}{2|E|}\right)^2\frac{1}{2}\sum_e Q(e)\phi(e)^2\sum_{\gamma_{ij}\ni e}|\gamma_{ij}|_Q\\&\overset{(c)}{=}\left(\frac{d_*^2}{2|E|}\right)\frac{1}{2}\sum_e Q(e)\phi(e)^2\sum_{\gamma_{ij}\ni e}|\gamma_{ij}|\\&\overset{(d)}{\le}\left(\frac{d_*^2}{2|E|}\right)\gamma_*\frac{1}{2}\sum_e Q(e)\phi(e)^2b(\gamma,e)\end{aligned} \tag{100}$$

where $b(\gamma,e)$ denotes the number of paths passing through edge e under $\gamma=(\gamma_{ij})$. (a) follows by using $\pi(i)\approx 1/n$ for all i, and adding and subtracting values of ϕ on nodes of the path γ_{ij} for all node pairs (i,j) for a given path-set $\gamma=(\gamma_{ij})$. (b) follows from the Cauchy–Schwartz inequality. (c) follows from (99), and (d) follows from the fact that all path lengths are smaller than γ_*.

Note that in (100), $b(\gamma,e)$ is the only path-dependent term. So under a probability distribution on Γ (the set of all paths) in (100), $b(\gamma,e)$ can be replaced by $b(e)$ where

$$b(e)=\sum_{\gamma\in\Gamma}\Pr(\gamma)b(\gamma,e).$$

Let $b=\max_e b(e)$. Then

$$V(\phi)\le\left(\frac{d_*^2}{2|E|}\right)\gamma_*\frac{1}{2}\sum_e Q(e)\phi(e)^2b \tag{101}$$

$$=\left(\frac{d_*^2\gamma_*b}{2|E|}\right)\mathcal{E}(\phi,\phi). \tag{102}$$

The minimax characterization of eigenvalues [18, p. 176] gives a bound on the second largest eigenvalue as

$$\lambda_2=\sup\left\{1-\frac{\mathcal{E}(\phi,\phi)}{V(\phi)}\mid\phi\text{ a nonconstant}\right\}. \tag{103}$$

From (102) and (103), the statement of the lemma follows. □

From Lemmas 10, 16, and 17, and the fact that all paths are of length at most $\Theta(1/r)$, we obtain that the second largest eigenvalue corresponding to the natural random walk on $G^2(n,r)$ is bounded above as

$$\begin{aligned}\lambda_2&\le 1-\Theta\left(\frac{n^2r^2}{n^2r^4r^{-4}}\right)\\&=1-\Theta(r^2).\end{aligned} \tag{104}$$

We would like to note that, for mixing time, we need to show that the smallest eigenvalue (which can be negative), is also $\Theta(r^2)$ away from -1. One well-known way to avoid this difficulty is the following: modify transition probabilities as $Q=\frac{1}{2}(I+P)$. Q and P have the same stationary distribution. By definition, Q has all nonnegative eigenvalues, and $\lambda_2(Q)=\frac{1}{2}(1+\lambda_2(P))$. Thus, the mixing time of the random walk corresponding to Q is governed by $\lambda_2(P)$, and is therefore $\Theta(r^{-2}\log n)$. This random walk Q is the modified natural random walk in Theorem 8 b).

Thus, from Lemma 8 and (104), the proof of Theorem 8 b) for $G^2(n,r)$ follows. In general, the above argument can be carried out similarly for $d>2$ completing the proof of Theorem 8 b).

Averaging in $G^d(n,r)$: The natural averaging algorithm, based on the natural random walk, can be described as follows: when a node becomes active, it chooses one of its neighbors uniformly at random and averages with this neighbor. As noted before, in general, the performance of such an algorithm can be far worse than the optimal algorithm. Interestingly, in the case of $G^d(n,r)$, the performances of the natural averaging algorithm and the optimal averaging algorithm are comparable (i.e., they have averaging time of the same order). We state the following theorem.

Theorem 9: On the Geometric Random Graph $G^d(n,r)$, the absolute $1/n^\alpha$-averaging time, $\alpha>0$, of the natural averaging algorithm as well as of the optimal averaging algorithm is of order $\Theta\left(\frac{\log n}{r^2}\right)$.

Proof: We showed in Theorem 8 that for $\epsilon=1/n^\alpha,\alpha>0$, the ϵ-mixing times for the fastest mixing random walk and the natural random walk on $G^d(n,r)$ are of order $\Theta\left(\frac{\log n}{r^2}\right)$. Using this in Theorem 7, we have our result. □

Implication. In a wireless sensor network, Theorem 9 suggests that for a small radius of transmission, even the fastest averaging algorithm converges slowly, i.e., computing in a distributed fashion is slow. However, the good news is that the natural averaging algorithm, based only on local information, scales just as well as the fastest averaging algorithm. Thus, at least in the order sense, it is not necessary to optimize for the fastest averaging algorithm in a wireless sensor network.

B. Expander Graphs

An expander graph can be characterized as follows: let the transition matrix corresponding to the natural random walk on the graph be P. Then, there exists $\delta > 0$ such that

$$\delta \leq (1 - \lambda_{\max}(P)) \leq 1 \tag{105}$$

where $\lambda_{\max}(P)$ is the second largest eigenvalue of P in magnitude, i.e., the spectral gap is bounded away from zero by a constant.

Let P^* be the transition matrix corresponding to the fastest mixing random walk on an expander. The random walk corresponding to P^* must mix at least as fast as the natural one, and therefore,

$$\delta \leq (1 - \lambda_{\max}(P^*)) \leq 1. \tag{106}$$

It is easy to argue that there exists an optimal P^* that is symmetric: given any optimal P_0, the matrix $1/2\left(P_0 + P_0^T\right)$ is symmetric, and leads to the same $E[W]$ as P_0, since

$$E[W] = \sum_{i,j} \frac{1}{n} P_{ij} W_{ij} \tag{107}$$

and the W_{ij} are symmetric matrices.

Therefore, we are able to use the result relating the mixing time for P and the averaging time for $\mathcal{A}(P)$ for a symmetric P. From (105), (106), Theorem 3, and Corollary 2, we see that the optimal averaging algorithm on any expander graph has ϵ-averaging time $T_{\text{ave}}(\epsilon) = \Theta\left(\log \epsilon^{-1}\right)$.

The Preferential Connectivity (PC) model [35] is one of the popular models for the Internet. In [35], it is shown that the Internet is an expander under the PC model. Using the conclusion above, we obtain the following result for averaging on the Internet.

Theorem 10: Under the PC model, the optimal averaging algorithm on the Internet has an absolute ϵ-averaging time $T_{\text{ave}}(\epsilon) = \Theta\left(\log \epsilon^{-1}\right)$.

Implication. The absolute time for distributed computation on any expander graph is independent of the size of the network, and depends only on the desired accuracy of the computation. Assuming that the PC model is a good model for Internet, then this immediately suggests that the absolute computation time depends only on the desired accuracy.[7] One implication is that exchanging information on the Internet via peer-to-peer network built on top of it is extremely fast!

Remark: Let $d_{\max}$ be the maximum node degree of the graph G. For any family of graphs of bounded degree, the averaging time of the maximum-degree random walk ($P_{ij} = 1/d_{\max}$ if $(i,j) \in E, i \neq j$), and the fastest mixing random walk are of the same order.[8] This follows from an observation in [1], which says that the spectral gap for the fastest mixing Markov chain on a graph can be at most a factor $d_{\max}$ smaller than the maximum-degree chain. Thus, if P^* is the optimal transition matrix, i.e., the one with the smallest possible $\lambda_2(P)$, and P_{md} is the transition matrix for the maximum-degree chain, then

$$1 - \lambda_2(P^*) \leq d_{\max}(1 - \lambda_2(P_{\text{md}})). \tag{108}$$

Thus, the averaging times for both random walks are of the same order, and differ by a factor of atmost $d_{\max}$.

For example, the social network [27] is a regular graph with $d_{\max} = 5$, which is the degree of each node in the graph. For the social network, therefore, the natural random walk (which is the same as the maximum degree chain) leads to an averaging time of the same order as the optimal; and in fact, the averaging times differ by a factor of at most 5.

C. Information Exchange

Define $T_{\text{infex}}(\epsilon)$ to be the smallest time at which each node has information from all the other nodes with a probability greater than or equal to $1 - \epsilon$. The averaging time provides an upper bound for the information exchange time, as made precise in the following theorem.

Theorem 11: For a gossip algorithm specified by a matrix P and $\epsilon = O\left(\frac{1}{n}\right)$

$$T_{\text{infex}}(\epsilon, P) \leq T_{\text{ave}}\left(\frac{\epsilon}{n}, P\right).$$

Proof: Consider first a single node i, and set $x_i(0) = 1$ and $x_j(0) = 0$ for all $j \neq i$. By the definition of averaging time, for all $t > T_{\text{ave}}\left(\frac{\epsilon}{n}, P\right)$, the probability that $x_i(j) > 0$ is greater than or equal to $1 - \frac{\epsilon}{n}$, since by the definition of T_{ave}, for all $t \geq T_{\text{ave}}\left(\frac{\epsilon}{n}, P\right)$

$$\Pr\left(\frac{\|x(t) - \frac{1}{n}\mathbf{1}\|^2}{\|x(0)\|^2} \leq \frac{\epsilon}{n}\right) \geq 1 - \frac{\epsilon}{n}. \tag{109}$$

Note that

$$\|x(0)\|^2 = 1$$

and

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

If any $x_i = 0$ (each x_i must be positive), then that term contributes $\frac{1}{n^2}$ to the sum, and thus the sum cannot be less that $\frac{\epsilon}{n}$ for $\epsilon = O\left(\frac{1}{n}\right)$.

Thus, for all $\epsilon = O\left(\frac{1}{n}\right)$, the probability that all of the $x_i > 0$ are greater **(or ?)** equal **(to ?)** $1 - \frac{\epsilon}{n}$. But this is exactly the probability that all nodes receive the message from node i. Using the union bound and summing for n nodes, we conclude that the probability of all nodes receiving information from all other nodes is greater **(or ?)** equal **(to ?)** $1 - n\frac{\epsilon}{n}$, and so $T_{\text{infex}}(\epsilon, P) \leq T_{\text{ave}}\left(\frac{\epsilon}{n}, P\right)$. □

VII. Conclusion

We presented a framework for the design and analysis of a randomized distributed averaging algorithm on an arbitrary connected network. We characterized the performance of the algorithm precisely in the terms of second largest eigenvalue of an appropriate doubly stochastic matrix. This allowed us to find the

[7] Although that the asymmetry of the P matrix for the natural random walk on the Internet prevents us from exactly quantifying the averaging time, we believe that averaging will be fast even under the natural random walk, since the spectral gap for this random walk is bounded away from 1 by a constant.

[8] The reason for using the maximum degree chain rather than the natural random walk is because the natural random walk need not be symmetric for an arbitrary graph. (Note that for a regular graph, the maximum degree chain and the natural random walk are exactly the same.) An alternative symmetric random walk with locally computable weights is the Metropolis–Hastings random walk with $P_{\text{mh}_{ij}} = \min\{1/d_i, 1/d_j\}$ for $(i,j) \in E, i \neq j$, for which a similar result holds.

fastest averaging algorithm of this class of algorithms, by establishing the corresponding optimization problem to be convex. We established a tight relation between the averaging time of the algorithm and the mixing time of an associated random walk, and utilized this connection to design fast averaging algorithms for two popular and well-studied networks: Wireless Sensor Networks (modeled as Geometric Random Graphs), and the Internet graph (under the so-called Preferential Connectivity Model).

In general, solving SDPs in a distributed manner is not possible. However, we utilized the structure of the problem in order to solve the SDP (corresponding to determining the optimal averaging algorithm) in a distributed fashion using the subgradient method. This allows for self-tuning weights: that is, the network can start out with some arbitrary averaging matrix, say, one derived from the natural random walk, and then locally, without any central coordination, converge to the optimal weights corresponding to the fastest averaging algorithm.

The framework developed in this paper is general and can be utilized for the purpose of design and analysis of distributed algorithms in many other settings.

ACKNOWLEDGMENT

Devavrat Shah wishes to thank Robert Gallager for a careful reading and suggestions that led to an improvement in the readability of the final paper. The authors would also like to thank an anonymous reviewer for several useful suggestions regarding the presentation of this paper.

REFERENCES

[1] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Rev., Problems and Techniques Section*, vol. 46, no. 4, pp. 667–689, 2004.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Analysis and optimization of randomized gossip algorithms," in *Proc. IEEE Conf. Decision and Control*, Nassau, Bahamas, Dec. 2004, pp. 5310–5315.

[3] ——, "Gossip algorithms: Design, analysis, and applications," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005.

[4] ——, "Mixing times of random walks on geometric random graphs," in *Proc. SIAM ANALCO*, Vancouver, BC, Canada, Jan. 2005.

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

[6] R. W. Beard and V. Stepanyan, "Synchronization of information in distributed multiple vehicle coordinated control," in *Proc. IEEE Conf. Decision and Control*, Dec. 2003, pp. 2029–2034.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge Univ. Press, 2004. Available [Online] at http://www.stanford.edu/~boyd/cvxbook.html.

[8] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Philadelphia, PA: SIAM, 1990.

[9] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of Markov chains," *Ann. Appl. Probab.*, vol. 1, no. 1, pp. 36–61, 1991.

[10] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*. New York: Springer-Verlag, 1999.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proc. 5th Int. Conf. Mobile Computing and Networking*, 1999, pp. 263–270.

[12] L. Elsner, I. Koltracht, and M. Neumann, "On the convergence of asynchronous paracontractions with applications to tomographic reconstruction from incomplete data," *Linear Algebra Appl.*, vol. 130, pp. 65–82, 1990.

[13] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[14] A. El Gamal, J. Mammen, B. Prabhakar, and D. Shah, "Throughput-delay trade-off in wireless networks," in *Proc. IEEE INFOCOM*, 2004.

[15] V. Guruswami. (2000) Rapidly Mixing Markov Chains: A Comparison of Techniques. [Online]. Available: cs.washington.edu/homes/venkat/pubs/papers.html

[16] I. Gupta, R. van Renesse, and K. Birman, "Scalable fault-tolerant aggregation in large process groups," in *Proc. Conf. Dependable Systems and Networks*, 2001, pp. 442–433.

[17] S. Hedetniemi, S. Hedetniemi, and A. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, pp. 319–349, 1988.

[18] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.

[19] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*. Berlin, Germany: Springer-Verlag, 1993.

[20] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proc. Int. Conf. Distributed Computing Systems*, Jul. 2002.

[21] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[22] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. Conf. Foundations of Computer Science*, 2003, pp. 482–491.

[23] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. Int. Workshop of Distributed Event Based Systems*, Jul. 2002.

[24] K. Kiwiel, "Convergence of approximate and incremental subgradient methods for convex optimization," *SIAM J. Optimization*, vol. 14, no. 3, pp. 807–840, 2004.

[25] D. Kempe and J. Kleinberg, "Protocols and impossibility results for gossip-based communication mechanisms," in *Proc. 43st IEEE Symp. Foundations of Computer Science*, 2002, pp. 471–480.

[26] D. Kempe, J. Kleinberg, and A. Demers, "Spatial gossip and resource location protocols," in *Proc. 33rd ACM Symp. Theory of Computing*, 2001, pp. 163–172.

[27] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. Symp. Theory of Computing*, 2000, pp. 163–170.

[28] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," in *Proc. Symp. Theory of Computing*, 2004, pp. 561–568.

[29] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking, "Randomized rumor spreading," in *Proc. Symp. Foundations of Computer Science*, 2000, pp. 564–574.

[30] Z. Lin, M. Brouke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Trans. Autom. Control*, vol. 49, no. 4, pp. 622–629, Apr. 2004.

[31] A. S. Lewis, "Convex analysis on the Hermitian matrices," *SIAM J. Optimization*, vol. 6, pp. 164–177, 1996.

[32] ——, "Nonsmooth analysis of eigenvalues," *Math. Programming*, vol. 84, pp. 1–24, 1999.

[33] A. S. Lewis and M. L. Overton, "Eigenvalue optimization," *Acta Numer.*, vol. 5, pp. 149–190, 1996.

[34] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, 2002.

[35] M. Mihail, C. Papadimitriou, and A. Saberi, "On certain connectivity properties of the internet topology," in *Proc. Conf. on Foundations of Computer Science*, 2003, pp. 28–35.

[36] L. Mureau, "Leaderless coordination via bidirectional and unidirectional time-dependent communication," in *Proc. IEEE Conf. Decision and Control*, Dec. 2003.

[37] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[38] M. L. Overton, "Large-scale optimization of eigenvalues," *SIAM J. Optimiz.*, vol. 2, pp. 88–120, 1992.

[39] M. L. Overton and R. S. Womersley, "Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices," *Math. Programming*, vol. 62, pp. 321–357, 1993.

[40] M. Penrose, "Random geometric graphs," in *Oxford Studies in Probability*. Oxford, U.K.: Oxford Univ. Press, 2003.

[41] P. A. Parrilo, L. Xiao, S. Boyd, and P. Diaconis, "Symmetry analysis of reversible Markov chains," *Internet Math.*, vol. 2, no. 1, pp. 31–71, 2003.

[42] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM SIGCOMM Conf.*, 2001.

[43] Y. Rabani, A. Sinclair, and R. Wanka, "Local divergence of Markov chains and the analysis of iterative load-balancing schemes," in *Proc. Conf. Foundations of Computer Science*, 1998, pp. 694–703.

[44] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM SIGCOMM Conf.*, 2001, pp. 149–160.

[45] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Lab. Information and Decision Systems, MIT, Cambridge, MA, 1984.

[46] R. van Renesse, "Scalable and secure resource location," in *Proc. 33rd Hawaii Int. Conf. System Sciences*, vol. 4, 2000, pp. 4012–4012.

[47] H. Wolkowicz, R. Saigal, and L. Vengerghe, Eds., *Handbook of Semidefinite Programming, Theory, Algorithms, and Applications*. Norwell, MA: Kluwer Academic, 2000.

[48] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *Proc. 2003 Conf. Decision and Control*, Dec. 2003, pp. 4997–5002.