Fast Gossip via Non-reversible Random Walk

Kyomin Jung
MIT
Email: kmjung@mit.edu

Devavrat Shah
MIT
Email: devavrat@mit.edu

***Abstract*— Distributed computation of average is essential for many tasks such as estimation, eigenvalue computation, scheduling in the context of wireless sensor and ad-hoc networks. The wireless communication imposes the *gossip* constraint: each node can communicate with at most one other node at a given time. Recent interest in emerging wireless sensor network has led to exciting developments in the context of gossip algorithms for averaging. Most of the known algorithms are iterative and based on certain reversible random walk on the network graph. Subsequently, the running time of algorithm is affected by the diffusive nature of reversible random walk. For example, they take $\Omega(n^2)$ time to compute average on a simple path or ring graph of n nodes. In contrast, an optimal (simple) centralized algorithm takes $\Theta(n)$ time to compute average in a path. This raises the following questions: is it possible for a distributed algorithm to compute average in $O(n)$ time for path graph? is it possible to improve over diffusive behavior of current algorithms in arbitrary graphs?**

In this paper, we answer the above questions in affirmative. To overcome the diffusive nature of algorithms, we utilize non-reversible random walks. Specifically, we design our algorithms by "projecting down" the "lifted" non-reversible random walks of Diaconis-Holmes-Neal (2000) and Chen-Lovasz-Pak (1999). The running time of our algorithm is square-root of the time taken by corresponding reversible random walk for a large class of graphs including path. As a sub-routine, our algorithm uses a simple distributed maximal matching algorithm that runs in $O(\log^2 n)$ time for arbitrary graph, which may be of separate interest.

I. Introduction

With the development of peer-to-peer, sensor, and wireless ad hoc networks, there has been a lot of recent interest in totally distributed algorithms for fault-tolerant computation. This is primarily due to dynamic nature of network, lack of infrastructure and limited computation and communication resources. The wireless communication imposes constraints on simultaneous exchanges. Motivated by popular interference model, we consider algorithms with the *gossip* constraints: (a) each node communicates with at most one other node at a given time, (b) nodes lack *unique* identity and (c) nodes can utilize only *local* information for computation. Thus, gossip algorithms are *totally* distributed. We will consider gossip algorithms for the question of computing average in a network. Distribution computation of an average is essential to many distributed tasks such as estimation, eigenvalue computation for clustering, etc. [9] and scheduling [10].

A. Setup

We are given an arbitrary connected network. Let its graph be $G = (V, E)$ with $|V| = n$ nodes. Initially, each node $i \in V$ begins with its value $x_i \in \mathbf{R}_+$. If $(i, j) \in E$ then nodes i and j can exchange messages. Time, denoted by $t \in \mathbf{Z}_+$, is assumed to be slotted and in a time-slot two nodes can transmit a number to each other. Let $\mathbf{x}(t) = [x_i(t)]^T$ be column-vector of values at n nodes at time t, with $\mathbf{x}(0) = [x_i]^T$ under some algorithm $\mathcal{A}$. The goal is to compute the average $\mathrm{x}_{\text{ave}} = \mathbf{x}(0)^T\mathbf{1}/n = \sum_{i=1}^n x_i/n = \|\mathbf{x}(0)\|_1/n$, at all nodes. We wish to design algorithm so as to minimize the computation time. Specifically, we define ϵ-averaging time $T_{\text{ave}}^{\mathcal{A}}(\epsilon)$ of an algorithm $\mathcal{A}$ as follows: let $\mathcal{S} = \{\mathbf{x}(0) \in \mathbf{R}_+^n : \|\mathbf{x}(0)\|_1 = 1\}$, then

$$T_{\text{ave}}^{\mathcal{A}}(\epsilon) = \sup_{\mathcal{S}} \inf\{t \in \mathbf{Z}_+ : \Pr\left(\|\mathbf{x}(s) - \mathrm{x}_{\text{ave}}\mathbf{1}\|_1 > \epsilon\right) < \epsilon, \quad \forall s \geq t\}.$$

where $\Pr(\cdot)$ denotes the probability induced by randomization of algorithm $\mathcal{A}$. Note that $\mathrm{x}_{\text{ave}} = 1/n$ for $\mathbf{x} \in \mathcal{S}$. Naturally, the above definition applies for deterministic algorithms as well. We remind ourselves that interest is in gossip algorithms, in which simultaneous transmissions in a time-slot form a matching in the network graph.

B. Previous Results

The question of averaging has recently received a lot of attention. One of the earlier work on distributed averaging was by Tsitsiklis [13]. In that and follow-on work, many considered iterative algorithms where a node was allowed to exchange information with possibly all of its neighbor. Essentially, the $\mathbf{x}(t)$ evolved according to a linear-dynamics, where $\mathbf{x}(t+1) = P\mathbf{x}(t)$, with $P\mathbf{1} = \mathbf{1}$ and P is graph conformant, i.e. $P_{ij} \neq 0$ only if $(i, j) \in E$. Let $\lambda_2(P)$ denote the norm of second largest eigenvalue of P. Then, it is well-known that the ϵ-averaging time is of order $s(P)^{-1}\log\epsilon^{-1}$, where $s(P) = 1 - \lambda_2(P)$. In addition, if $P \geq [0]$ and be a probability transition matrix, then the ϵ-averaging time becomes the same as ϵ-mixing time, $T_{\text{mix}}^P(\epsilon)$ of P defined as follows: let $P\pi = \pi$, then

$$T_{\text{mix}}^P(\epsilon) = \sup_{\mathcal{S}} \inf\{t \in \mathbf{Z}_+ : \|\mathbf{x}(s) - \pi\|_1 < \epsilon,\ \forall s \geq t\}.$$

This previous work required each node to communicate with multiple nodes in a given time-slot and hence violating gossip constraints. The natural question was: is it possible to achieve ϵ-averaging time same as ϵ-mixing time under *gossip* constrained communications? Karp, Schindelhauer, Shenker and Vocking [6] showed that answer is negative by establishing that ϵ-averaging time $\Omega(\log n)$ for small enough but constant $\epsilon > 0$, while $P = [1/n]$ gives $T^P_{\text{mix}}(\epsilon) = O(1)$ for all $\epsilon > 0$! Kempe, Dobra and Gehrke [8] showed the existence of an averaging algorithm with ϵ-averaging time $O(\log n)$ for $\epsilon = \Omega(1/n^k)$ for any finite k in the context of complete graph. The question still remained: how is the ϵ-averaging time for arbitrary graph related to the ϵ-mixing time? In [2], Boyd, Ghosh, Prabhakar and Shah established that there are gossip algorithms with ϵ-averaging time $\Theta(\log n + T^P_{\text{mix}}(\epsilon))$ for given matrix P and $\epsilon = \Theta(1/n^k)$ for any positive finite k. Implicit in results of [2], authors also establish that the for any randomized gossip algorithm, the related matrix P is symmetric or corresponding random walk is reversible. This established equivalence between optimal randomized gossip averaging algorithm and fastest mixing reversible random walk on graph. Consequently, randomized gossip algorithms are only as fast as mixing of reversible random walks. Now, reversible random walks exhibit *diffusive* behavior. Precisely, for a large class of graph (e.g. bounded growth) the spectral gap $s(P)$ scales as $1/\Phi(P)^2$ where $\Phi(P)$ is the conductance defined as

$$\Phi(P) = \min_{S \subset V, |S| \leq n/2} \frac{\sum_{i \in S, j \notin S} \pi_i P_{ij}}{\pi(S)},$$

for P where $P\pi = \pi$ and $\pi(S) = \sum_{i \in S} \pi_i$. For example, in the context of path graph for any reversible P, $\Phi(P) = \Omega(1/n)$ and $s(P) = \Omega(1/n^2)$. In contrast, a simple centralized deterministic scheme will have ϵ-averaging time $\Theta(n)$. A natural question: is it possible to improve the behavior of gossip averaging algorithm beyond mixing time of reversible random walk?

C. Our Contribution

We answer the above question in affirmative. Specifically, given matrix P, we device deterministic gossip algorithm that have ϵ-averaging time scaling proportional to $1/\Phi(P)$. We state our precise result as follows.

Theorem 1: Let Δ be maximum vertex degree of network-graph $G = (V, E)$. Given a graph conformant P, such that $P\mathbf{1} = \mathbf{1}$, we obtain a deterministic averaging algorithm $\mathcal{A}$ such that for $\epsilon = \Theta(1/n^k)$ for any finite k,

$$T^{\mathcal{A}}_{\text{ave}}(\epsilon) = O\left(\frac{\Delta \log^2 n}{\Phi(P)}\right).$$

An immediate Corollary of Theorem 1 is as follows:

Corollary 2: If G is a path graph of n nodes, then there exists a deterministic gossip averaging algorithm $\mathcal{A}$ such that for any $\epsilon = \Theta(1/n^k)$ with positive constant k, $T^{\mathcal{A}}_{\text{ave}}(\epsilon) = O\left(n \log^2 n\right)$.

Note. A recent gossip algorithm by Mosk-Aoyama and Shah [11] for computing separable function can be used to compute average with ϵ-averaging time $O(\epsilon^{-2}\Phi^{-1}(P))$. While this is the best known for finite ϵ, it becomes very large for ϵ scaling down with n (e.g. $\epsilon = 1/n$). In such regime, the algorithm of this paper will be the most effective. Such high-precision algorithms are necessary for many tasks such as wireless scheduling [10].

D. Organization

The rest of the paper is organized as follows. In Section II we prove Theorem 1 by presenting gossip algorithm based on Non-reversible random walks (RW) of Diaconis-Holmes-Neal [4] and Chen-Lovasz-Pak [3]. We will describe [3] for arbitrary graph and specialize it to path graph, which is similar to [4]. We note that results of [3] are generalization of [4]. In Section III, we describe a distributed maximal matching algorithm. We use this algorithm as a sub-routine in the algorithm of Section II.

II. Averaging via Non-reversible RW

Given graph G conformant matrix P such that $P\mathbf{1} = \mathbf{1}$, we describe a gossip algorithm that satisfied claim of Theorem 1.

A. Non-reversible Q via P

Chen-Lovasz-Pak [3] presented construction of a matrix Q (not G conformant) by *lifting* the graph G to a larger graph $\hat{G}$, that has mixing time linearly scaling in $1/\Phi(P)$. First, we describe this construction and then show how it can be *projected down* on G to obtain a gossip algorithm with running time proportional to mixing time of Q.

Construction of [3]. We describe lifting of P to Q as in [3], but somewhat differently for ease of exposition (for more details, we refer interested to [3]). The graph G of n nodes is lifted to graph $\hat{G} = (\hat{V}, \hat{E})$ of upto n^3 nodes by making $L(i) \leq n^2$ copies of node $i, 1 \leq i \leq n$, denoted as $i_1, \ldots, i_{L(i)}$. The edges $\hat{E}$ are such that: (1) $(i_1, j_1) \in \hat{E}$ iff $(i, j) \in E$; (2) for any $1 \leq p < q \leq L(i)$, $(i_p, i_q) \in \hat{E}$ only if $p = 1$; (3) for $p \leq L(i), q \leq L(j)$, $(i_p, j_q) \in \hat{E}$ only if $(i, j) \in E$ and if $(i_p, j_q) \in \hat{E}$ then $(i_p, j_r) \notin \hat{E}$ for all $1 \leq r \leq L(j), r \neq q$. In [3], a $\hat{G}$ conformant probability matrix Q is defined that has the following property.

Lemma 3: Q admits a unique stationary distribution on $\hat{G}$. Let $Q\sigma = \sigma$, then for $\epsilon = \Theta(1/n^k)$ with finite k,

$$\sum_{p=1}^{L(i)} \sigma_{i_p} = \frac{1}{n} \quad \text{and} \quad T^Q_{\text{mix}}(\epsilon) = O\left(\frac{\log^2 n}{\Phi(Q)}\right). \tag{1}$$

Lemma 3 is proved in [3]. Now, we use the Q to compute average over G. Let $m = |\hat{V}|$ and $\mathbf{y}(t) \in \mathbf{R}^m_+$ be vector of values at nodes of $\hat{G}$ under the following dynamics: $\mathbf{y}(0) = [y_v]$ where $y_v = x_i$ if $v = i_1$ and 0 otherwise. For $t \in \mathbf{Z}_+$,

$$\mathbf{y}(t+1) \quad = \quad Q\mathbf{y}(t). \tag{2}$$

Define corresponding vector of values of nodes in G, $\mathbf{x}(t) = [x_i(t)] \in \mathbf{R}^n_+, t \in \mathbf{Z}_+$ as

$$x_i(t) \quad = \quad \sum_{p=1}^{L(i)} y_{i_p}(t). \tag{3}$$

Note that $\mathbf{x}(0) = [x_i]$ as defined by (3).

Lemma 4: For $t \geq T^Q_{\text{mix}}(\epsilon)$,

$$\frac{\|\mathbf{x}(t) - \mathrm{x}_{\text{ave}}\mathbf{1}\|_1}{n\mathrm{x}_{\text{ave}}} < \epsilon.$$

Proof: Proof sketch is as follows: without loss of generality, assume $\|\mathbf{x}(0)\|_1 = 1$. Using Lemma 3 and triangular inequality for $|\cdot|$, we obtain

$$\begin{aligned}\frac{\|\mathbf{x}(t) - \mathrm{x}_{\text{ave}}\mathbf{1}\|_1}{n\mathrm{x}_{\text{ave}}} &= \|\mathbf{x}(t) - \frac{1}{n}\mathbf{1}\|_1 \\ &= \sum_{i=1}^n |x_i(t) - 1/n| = \sum_{i=1}^n \left|\sum_{p=1}^{L(i)} y_{i_p} - \sigma_{i_p}\right| \\ &\leq \|\mathbf{y}(t) - \sigma\|_1. \end{aligned} \tag{4}$$

Definition of $T^Q_{\text{mix}}(\epsilon)$, (4) imply the Lemma 4. ∎

Lemma 4 suggests that it is sufficient to *simulate* the dynamics of (2) in graph G via gossip algorithm. We do that next.

Gossip simulation of (2). We will show that a $\mathbf{y}(t+1)$ can be computed via a gossip algorithm given $\mathbf{y}(t)$ in $O(\Delta)$ steps. For this, decompose edges E of G into collection of matchings with disjoint edges as follows. Initially, set $E_0 = E$, $G_0 = G$ and iteration $k = 0$. In iteration k, run distributed maximal matching algorithm RMA (described in Section III-A) to obtain matching Π_k. Set $E_{k+1} = E_k - \Pi_k$, $k = k+1$. If $E_k = \emptyset$ stop, else repeat.

Lemma 5: Due to property of RMA that it finds maximal matching, the algorithm stops within 2Δ iterations.

Proof: We sketch the proof in interest of space: if Lemma is not true then there is an $\mathbf{e} \in E$, that is not chosen by 2Δ iteration. Then due to property of maximal matching there must be 2Δ other edges chosen in first 2Δ iterations that shared a vertex with $\mathbf{e}$. But maximum degree being Δ, this is not possible. ∎

Now, we have $E = \cup_{k=1}^{2\Delta} \Pi_k$, where all edges of Π_k are different (and possibly some $\Pi_k = \emptyset$). This takes $O(\Delta \log^2 n)$ iterations w.h.p. as established in Section III-A.

Now, for simulation each node $i \in V$ creates virtual nodes $i_1, \ldots, i_{L(i)}$ of $\hat{G}$ and maintains $y_{i_p}(t), 1 \leq p \leq L(i)$. For $t = 0$, each i knows $y_{i_1}(0) = x_i$ and $y_{i_p}(0) = 0, p \neq 1$. Inductively, we assume that each node $i \in V$ knows $y_{i_p}(t), 1 \leq p \leq L(i)$. Now we'll show how each $i \in V$ can compute $y_{i_p}(t+1), 1 \leq p \leq L(i)$ via a gossip algorithm in $2\Delta + 1$ time-slots, $0 \leq k \leq 2\Delta$. In $k = 0^{th}$ slot, each i_p sends $y_{i_p}(t)Q_{i_q i_p}$ to node i_q, $1 \leq p, q \leq L(i)$. This can be done at node i itself as all i_p are maintained at i itself. In k^{th} time-slot, $1 \leq k \leq 2\Delta$, communications are done according to matching Π_k (thus satisfying gossip constraint of G). If $(i,j) \in \Pi_k$, then for all $(i_p, j_q) \in \hat{E}$ send $y_{i_p}(t)Q_{j_q i_p}$ from i_p to node j_q and send $y_{j_q}(t)Q_{i_p j_q}$ from j_q to i_p. All these transmissions can be done simultaneously along edge (i,j) as nodes i and j have access to required information as well as these can be packaged together in a single message[1]. At the end of all $2\Delta + 1$ time-slots, assign $y_{i_p}(t+1)$ as the sum of all received values at node i_p. By definition,

$$y_{i_p}(t+1) \quad = \quad \sum_{j=1}^{n} \sum_{q=1}^{L(j)} Q_{i_p j_q} y_{j_q}(t). \tag{5}$$

The (5) is identical to (2). Thus, we have showed how to simulate one-step of (2) via a gossip algorithm in $O(\Delta)$ time-steps. Putting all the above discussion together, we obtain the following (which implies Theorem 1).

Theorem 6: The above described simulation of (2) and (3) gives a deterministic gossip algorithm $\mathcal{A}$, such that for $\epsilon = \Theta(1/n^k)$ for finite k,

$$T^{\mathcal{A}}_{\text{ave}}(\epsilon) = O\left(\frac{\Delta \log^2 n}{\Phi(P)}\right).$$

Non-reversible Q for path. We explicitly describe non-reversible RW of [4] for path of n nodes: for each node $1 \leq i \leq n$, create two copies $(i,+)$ and $(i,-)$. Intuitively, $+$ is right direction and $-$ is left direction. The transition matrix Q on these $2n$ nodes is defined as follows: (1) $Q_{(i,+)(i+1,+)} = 1 - 1/n$, for $1 \leq i < n$, (2) $Q_{(i,+)(i+1,-)} = 1/n$, for $1 \leq i < n$, (3) $Q_{(i,-)(i-1,-)} = 1 - 1/n$, for $1 < i \leq n$, (4) $Q_{(i,-)(i-1,+)} = 1/n$,

[1]It is contentious to assume the possibility of all upto $\max\{L(i), L(j)\}$ numbers transmitted together as one message. We allow this in our model. If computed separately, the complexity can increase at most by $\max_i L(i)$. For path graph, the construction of [4] has $L(i) = O(1)$.

for $1 < i \leq n$, (5) $Q_{(n,+)(n,-)} = 1 - 1/n$, (6) $Q_{(n,+)(n,+)} = 1/n$, (7) $Q_{(1,-)(1,+)} = 1 - 1/n$, and (8) $Q_{(1,-)(1,-)} = 1/n$. As shown in [4], the $T^Q_{\text{mix}}(1/n^k) = O(n \log n)$ for finite k.

III. Distributed Maximal Matching

In this section, we describe and analyze a randomized distributed maximal matching algorithm. We believe that it will be of great interest in many other applications including other gossip algorithm and wireless scheduling. The algorithm, we believe, is certainly well known in some version. However, the analysis of this algorithm is new. Matchings are extremely well-studied combinatorial objects and there are a large number of different algorithms to find different types of matching. Here, we recall some of the well-known distributed matching algorithms. Karp, Upfal, Wigderson [7] and Mulmuley, Vazirani, Vazirani [12] gave randomized distributed algorithms to find maximum size matching that take $O(poly(\log n))$ running time with $O(n^{3.5})$ processors. In contrast, our model has $O(n)$ processors (i.e. n nodes of graph). In the context of maximum weight matching, Bertsekas [1] gave distributed *auction* algorithm, which may take upto $O(n)$ iterations to converge. For maximal matching, there is a well known (and rather obvious) distributed algorithm: pick edges one by one in arbitrary fashion maintaining matching structure till one can. Though very simple, best known bound on the performance of such algorithm is $O(n)$. A randomized algorithm (somewhat more complicated that ours) by Israeli and Itai [5] was shown to take $O(\log^2 n)$ iterations on average. However, it does not have such performance with probability $1 - O(1/n^k)$ for any finite k. We need such high probability guarantee, which will be proved for algorithm described next.

A. Algorithm and Analysis

We consider randomized version of the above stated naive maximal matching algorithm with $O(n)$ processors. In contrast to the above results, we find that the algorithm finds maximal matching in $O(\log^2 n)$ time w. h. p. for arbitrary graph. First, we state algorithm.

Algorithm RMA.

---

(1) Initially, iteration $i = 1$ and all vertices are *unmatched*.

(2) In iteration i, do the following:

  (i) Each unmatched vertex having at least one unmatched neighbor decided to be *left* or *right* with probability $1/2$ independently.

  (ii) If a vertex, say v_l, becomes *left*, it requests to one of its unmatched neighbor uniformly at random.

  (iii) If a vertex, say v_r, becomes *right*, on receiving requests from one or more *left* neighbors, it chooses one of them uniformly at random, say u. Set vertices v_r and u as matched and they inform all of their unmatched neighbors about it.

(3) Set $i = i + 1$. Repeat from (2) till no more edge can be added.

Theorem 7: For any graph G, the RMA algorithm finds a maximal matching in $O(\log^2 n)$ iterations with probability at least $1 - O(1/n^\ell)$, for any finite ℓ. For complete graph with n nodes, the algorithm RMA takes $\Omega(\log n)$ iterations to compute maximal matching with probability $1 - O(1/n^2)$.

Proof: We present the proof of Upper bound. The lower bound of the algorithm follows by studying its behavior for complete graph. We will skip the proof of lower bound in the interest of space.

Upper bound. Given ℓ, let C be a constant satisfying $(\frac{1}{2} + \frac{1}{2}e^{-\frac{1}{4}})^C < e^{-\ell-2}$. We show that for any graph G, the RMA finds maximal matching in $O(\log^2 n)$ iterations with probability at least $1 - O(\log n/n)$. To prove this, we divide the first $C(\log n)(1 + \log n)$ iterations of RMA into $1 + \log n$ stages, each stage consisting of $C \log n$ iterations. Thus, iterations $\{(k-1)(C \log n) + 1, \ldots, k(C \log n)\}$ correspond to stage $1 \leq k \leq 1 + \log n$. Let $i_k^- \triangleq (k-1)(C \log n)+1$ and $i_k^+ \triangleq k(C \log n)$ During the execution of algorithm RMA, the degree of unmatched vertices decrease as the matched vertices and edges incident on them are subsequently removed. Also, if a vertex is matched, we say that the vertex has degree 0. In this setup, let A_k denote the event that at the end of stage k, all vertices of G have degree at most $\frac{n}{2^k}$. We claim the following.

Claim 8: For $k = 1, 2, \ldots, 1 + \log n$,

$$\Pr[A_k | A_{k-1}] > 1 - \frac{1}{n^{\ell+1}}.$$

Proof: Note that $\Pr[A_0] = 1$. Now, consider $k \geq 1$. We wish to evaluate $\Pr[A_k | A_{k-1}]$. Given A_{k-1}, all vertices have degree $\leq \frac{n}{2^{k-1}}$ for $i \geq i_k^-$. Let v be a vertex with $\frac{n}{2^k} < \deg(v) \leq \frac{n}{2^{k-1}}$ at the beginning of the stage k, where $\deg(v)$ denotes the degree of vertex v. For $(k-1)(C \log n)+1 \leq i \leq k(C \log n)$, let $B_{v,i}$ be the event that after iteration i, $\deg(v) \geq \frac{n}{2^k}$. Now, by definition

$A_k^c \subseteq \cup_v \cap_{i_k^- \le i \le i_k^+} B_{v,i}$. Hence,

$$\begin{aligned}
\Pr[A_k^c|A_{k-1}] &\le \Pr\left[\cup_v \cap_{i_k^- \le i \le i_k^+} B_{v,i}|A_{k-1}\right] \\
&\le \sum_v \Pr\left[\cap_{i_k^- \le i \le i_k^+} B_{v,i}|A_{k-1}\right] \\
&= \sum_v \Pr(B_{v,i_k^-}|A_{k-1}) \prod_{i_k^- < i \le i_k^+} \Pr\left[B_{v,i}|B_{v,i-1}; A_{k-1}\right] \\
&\le \sum_v \prod_{i_k^- < i \le i_k^+} \Pr\left[B_{v,i}|B_{v,i-1}; A_{k-1}\right] \\
&\le \sum_v \left(\frac{1+e^{-\frac{1}{4}}}{2}\right)^{C\log n} \qquad (6) \\
&\le n \times e^{-(\ell+2)\log n} \le \frac{1}{n^{\ell+1}} \;=\; \frac{1}{n}, \qquad (7)
\end{aligned}$$

where justification for (6) is provided next. Note that, (7) completes the proof of Claim 8. Now, if vertex v gets matched in iteration i, then $B_{v,i}$ does not hold. Next, we show that given $B_{v,i-1} \cap A_{k-1}$, v gets matched with probability $1 - \frac{1+e^{-\frac{1}{4}}}{2}$. This will imply the bound used in (6).

Given $B_{v,i-1} \cap A_{k-1}$, i becomes *right* with probability $1/2$. When it becomes *right*, the (conditional on event $B_{v,i-1} \cap A_{k-1}$) probability that it does not get matched, denoted by $\bar{P}_{v,i}$, is upper bounded as

$$\begin{aligned}
\bar{P}_{v,i} &\le \sum_{j=0}^{\deg(v)} \binom{\deg(v)}{j} \left(\frac{1}{2}\right)^{\deg(v)} \left(1 - \frac{2^{k-1}}{n}\right)^j \\
&= \left(1 - \frac{2^{k-2}}{n}\right)^{\deg(v)} < e^{-\frac{1}{4}}. \qquad (8)
\end{aligned}$$

From above discussion and (8) the bound in (6) follows. ∎

Next, we use the Claim 8 to complete the proof of Theorem 7 as follows. $A_{1+\log n}$ implies that all nodes have degree < 1, that is, 0. Thus algorithm finds a maximal matching of G if $A_{1+\log n}$ holds. Now,

$$\Pr[A_{1+\log n}] \;\ge\; \Pr\left[A_{1+\log n}|A_{\log n}\right] \Pr\left[A_{\log n}\right]. \qquad (9)$$

Using the above argument repeatedly, we obtain

$$\begin{aligned}
\Pr[A_{1+\log n}] &\ge \left(\prod_{k=1}^{1+\log n} \Pr[A_k|A_{k-1}]\right) \times \Pr\left[A_0\right] \\
&> \left(1 - \frac{1}{n^{\ell+1}}\right)^{\log n} \;=\; 1 - O(n^{-\ell}), \qquad (10)
\end{aligned}$$

where we used the fact that $\Pr[A_0] = 1$. This completes the proof of upper bound. We note (rather straightforward) that probability of *bad* event happening can be bounded above by any $1/poly(n)$ by selecting appropriate constant C. ∎

IV. Conclusion

Motivated by applications peer-to-peer, wireless sensor and ad-hoc networks, we study gossip algorithms for averaging. Most of the previously known iterative algorithms suffered from the *diffusive* nature of reversible random walk. The algorithm of [11], based on property of exponential distributions, improves upon these algorithms but has poor scaling in the error-parameter. To overcome this, we presented deterministic gossip algorithm that utilizes non-reversible random walk. As a result, for a large class of graphs the time to compute average becomes square-root of the time taken by algorithms based on reversible random walk. For example, the time to average on a path graph becomes $O(n \log^2 n)$ instead of $O(n^2 \log n)$ for algorithm based on reversible random walk.

As a sub-routine of our algorithm, we developed a new distributed maximal matching algorithm which finds maximal matching in any graph in $O(\log^2 n)$ time. This algorithm will be of separate interest in the context of scheduling and other gossip algorithms.

References

[1] D. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of operations research*, 1988.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings of IEEE INFOCOM 2005*, 2005.

[3] F. Chen, L. Lovasz, and I. Pak. Lifting markov chains to speed up mixing. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 275–281, 1999.

[4] P. Diaconis, S. Holmes, and R. Neal. Analysis of a nonreversible markov chain sampler. *The Annals of Applied Probability*, 10(3):726–752, 2000.

[5] A. Israeli and A. Itai. A fast randomized parallel algorithm for maximal matching. *Inform. Process. Lett.*, 22(2):77–80, 1986.

[6] R. Karp, C. Schindelhauer, S. Shenker, and B. Vcking. Randomized rumor spreading. In *Proc. Symposium on Foundations of Computer Science*. IEEE, 2000.

[7] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. In *In Proceedings of the seventeenth annual ACM symposium on Theory of computing*, 1985.

[8] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. Conference on Foundations of Computer Science*. IEEE, 2003.

[9] D. Kempe and F. McSherry. A decentralized algorithm for spectral analaysis. In *Symposium on Theory of Computing*. ACM, 2004.

[10] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossip. *Submitted*, 2005.

[11] D. Mosk-Aoyama and D. Shah. Distributed computation of separable function. *Submitted*, 2006.

[12] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, 1987.

[13] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, MIT, 1984.