A Random-Walk Model for Distributed Computation in Energy-Limited Networks

Murat Alanyali, Venkatesh Saligrama, Onur Savas
Department of Electrical and Computer Engineering
Boston University, Boston, MA 02215, USA
Email: {alanyali,srv,savas}@bu.edu

***Abstract*— We consider two distributed algorithms to compute functions that admit flexible decomposition in terms of pairwise computations. Under these algorithms a transmitting node becomes inactive and does not transmit further messages until it is reactivated by a message reception from another node. The algorithms thereby have sequential nature and bear a close relationship to random walks. We quantify their time and message complexities on the d-dimensional torus and establish substantial gains in message complexity with respect to gossip algorithms. The algorithms exhibit a favorable tradeoff between the two complexities in lower dimensions. In particular on the 2-dimensional torus with n nodes, time and per-node message complexities scale as $\Theta(n \log n)$ and $O((\log n)^2)$ respectively, whereas both complexities scale as $\Omega(n)$ for gossip algorithms.**

I. Introduction

Scalability of information dissemination arise in a variety of applications. In database synchronization, for instance, the main objective is to rapidly make available a scattered set of information items to a set of locations. Sensor applications have different paradigms in that they typically require determination of a certain feature, such as the average or the maximum, of the scattered information. Furthermore energy is often a scarce resource in such applications; in turn completing the computational task with minimal message transmissions is of importance.

The theme of this paper is a constructive investigation of the tradeoff between the time and the number of messages required to compute a certain class of decomposable functions. Several decentralized algorithms have been previously proposed for similar computational problems in [1], [3], [7], [8], [10]. These algorithms are based on parallel asynchronous transmissions of local estimates, which in turn lead to improved estimates at recipient nodes. On the one hand this operational philosophy has the favorable properties of robustness against message losses and unknown network topologies. On the other hand it has a fundamental disadvantage from the energy viewpoint, mainly because the liberal nature of individual messaging decisions results in substantial number of largely redundant message transmissions.

We consider two randomized algorithms that lead to striking gains in message complexity while enjoying similar operational advantages as previously studied algorithms. Under the algorithms studied here, a transmitting node becomes inactive and does not transmit further messages until it is reactivated by a message reception from another node. An active node generates messages at constant rate and sends each message to a randomly selected neighbor. Algorithm `SIMPLE-WALK` maintains a single active node, whose trajectory is a random walk on the communication graph. Local processing at each active node exploits the decomposability of the considered functions to guarantee that the value of interest is computed when each node becomes active at least once. Both the time and the message complexities of this algorithm are determined by the cover time of the random walk. Under algorithm `COALESCENT` all nodes are initially active and the computation is completed when a single active node remains in the network. This latter algorithm is closely related to coalescing random walks. We quantify the time and the message complexities of these algorithms on the lattice torus and provide a comparison with gossip algorithms.

The abstract setting considered in this paper is as follows: Let Λ be a set and let $f : \Lambda^2 \mapsto \Lambda$ be an operation on Λ. We shall assume that f is commutative and associative, and that Λ has an identity element with respect to f. That is, there exists $e \in \Lambda$ such that $f(\lambda, e) = \lambda$ for $\lambda \in \Lambda$. Let $F_2 = f$. For $n \geq 3$ and given a permutation π of $\{1, 2, \cdots, n\}$ define the mapping $F_n : \Lambda^n \mapsto \Lambda$ recursively by

$$F_n(\lambda_1, \lambda_2, \cdots, \lambda_n) = f(\lambda_{\pi(1)}, F_{n-1}(\lambda_{\pi(2)}, \lambda_{\pi(3)}, \cdots, \lambda_{\pi(n)})), \tag{1}$$

for $\lambda_1, \lambda_2, \cdots, \lambda_n \in \Lambda$. Note that the mapping F_n does not depend on π owing to the properties of f.

Example 1.1: Let $k \geq 1$ be an integer and let $M^{k\times k}$ be the set of k-dimensional positive definite matrices. For $i = 1, 2, \cdots, n$ let $x_i \in \mathbb{R}^k$ and let $w_i \in M^{k\times k}$. The weighted average $(\sum_{i=1}^n w_i)^{-1} \sum_{i=1}^n w_i x_i$ can be expressed in the form (1) by choosing $\Lambda = \{\mathbb{R}^k \times M^{k\times k}\} \cup \{e\}$ with $\lambda_i = (x_i, w_i)$, and by setting

$$f(\lambda, \lambda') = ((w + w')^{-1}(wx + w'x') \ , \ w + w'),$$

for $\lambda = (x, w), \lambda' = (x', w') \in \Lambda - \{e\}$.

We are concerned with distributed computation of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ in the case when each λ_i is known to a distinct agent. This computational effort is subject to communication constraints summarized by an undirected graph $G = (V, E)$ where each node in V denotes an agent (hence $|V| = n$) and an edge $(i, j) \in E$ indicates a bidirectional communication link between agents i and j. To avoid trivialities G is assumed to be connected.

Given a spanning tree of G, decomposability of F_n permits computation of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ via a sequence of pairwise operations on the edges of this tree. In particular, the value of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ can be obtained at a designated node at the expense of $n-1$ message transmissions in the network and in time proportional to the diameter of G. Time and energy aspects of optimal centralized algorithms beyond this simplistic view were considered by [5], [6]. Rather than such centralized algorithms our focus here is on algorithms that have local specifications, require no centralized coordination, and exhibit robustness to message losses and topological variations.

Section II gives formal definitions of studied algorithms. Time and message complexities of these algorithms on the d-dimensional lattice torus are quantified in Section III and compared to those of gossip algorithms in Section IV. The paper concludes with final remarks in Section V.

II. Algorithm and Complexity Definitions

a) Algorithms: We study two algorithms coined here as `SIMPLE-WALK` and `COALESCENT`. A pseudo-code for the two algorithms is given in Figure 1. The algorithms have identical dynamic specification, but differ in their initialization. Under each algorithm, each node maintains two variables `value` and `status`. Content of `value` takes values in Λ, and `status` is either *'active'* or *'idle'*. We identify these variables by $v_i(t)$ and $\xi_i(t)$ at node $i \in V$ at time $t \geq 0$; in particular

$$\begin{aligned} v_i(t) &= \text{the content of } \texttt{value} \text{ of node } i \text{ at time } t, \\ \xi_i(t) &= \begin{cases} 1 & \text{if } \texttt{status} \text{ of node } i \text{ is 'active' at time } t, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Initially $v_i(0) = \lambda_i$ for each node i. The initial value $\xi_i(0)$ (i.e. of `status`) depends on the particular algorithm in the following fashion: Under `SIMPLE-WALK` $\xi_i(0) = 1$ for exactly one node, say node i_o, whereas $\xi_i(0) = 1$ for all nodes i under `COALESCENT`. Variables `value, status` evolve according to the same rules under both algorithms: Namely, each node has an independent Poisson clock that ticks at unit rate. When the local clock of node i ticks, say at time t_o, the node does not take any action unless it is active (i.e. $\xi_i(t_o^-) = 1$). Otherwise, if $\xi_i(t_o^-) = 1$, the node chooses a neighbor at random, sends its current value $v_i(t_o^-)$ to that neighbor, and sets $(v_i(t_o), \xi_i(t_o)) = (e, 0)$. In particular it becomes idle and ceases message transmission until it becomes active again. The selected neighbor, say node j, sets $v_j(t_o) = f(v_j(t_o^-), v_i(t_o^-))$ and becomes active by setting $\xi_j(t_o) = 1$.

In both algorithms an active node can be interpreted to be holding a transmit token that moves with the transmitted message. Reflecting on the algorithm specification reveals that two such tokens coalesce into one if they meet at the same node. Clearly, `SIMPLE-WALK` maintains a single such token in the network. An illustrative scenario that depicts the evolution of token locations (equivalently of active nodes) under `COALESCENT` is given by Figure 2.

Variables: `status, value`.
Initialize: `value` $\leftarrow \lambda_i$;

$$\texttt{SIMPLE}-\texttt{WALK}: \quad \texttt{status} \leftarrow \begin{cases} \text{'active'} & \text{if } i = i_o; \\ \text{'idle'} & \text{else.} \end{cases}$$

$$\texttt{COALESCENT}: \quad \texttt{status} \leftarrow \text{'active'}.$$

Procedure **Send**()
 if(`status` == 'active') {
  choose neighbor;
  send(neighbor , `value`);
  `value` $\leftarrow e$;
  `status` $\leftarrow$ 'idle';
  }
Procedure **Receive**(message) {
 `value` $\leftarrow f($`value`, message$)$;
 `status` $\leftarrow$ 'active';
 }

Fig. 1. Pseudo-code for algorithms `SIMPLE-WALK` and `COALESCENT` at node i. Send() is activated by the local Poisson clock at the node, and Receive() is activated by message reception from another node. The two algorithms differ in the initialization of the variable `status`. That is, only one node starts out active in `SIMPLE-WALK` whereas all nodes are initially active in `COALESCENT`.

Note that if the recipient of a message has value e just before reception then the value at the originating node simply passes onto the recipient node. Otherwise, the value of the recipient node is set to the image of the two values under f, thereby executing a significant step towards computation of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$. Such a step occurs if the recipient node becomes active for the first time under `SIMPLE-WALK`, and if the recipient is already active under `COALESCENT`.

b) Correctness and complexity: The following sample-path property is useful in proving that both algorithms compute the exact value of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ in finite time:

Proposition 2.1: Under both `SIMPLE-WALK` and `COALESCENT`,

$$F_n(v_1(t), v_2(t), \cdots, v_n(t)) = F_n(\lambda_1, \lambda_2, \cdots, \lambda_n), \quad t > 0.$$

Let τ_S and τ_C be random times that are defined as follows:

$$\begin{aligned} \tau_S &= \inf\{t : \text{ each node becomes active by time } t \ \} \\ \tau_C &= \inf\{t : \sum_i \xi_i(t) = 1\}. \end{aligned}$$

In particular, under algorithm `COALESCENT`, a single active node remains in the network after time τ_C . Note also that for $t > \tau_C$ (and for $t > \tau_S$ under `SIMPLE-WALK`) $v_i(t) = e$ for all nodes i such that $\xi_i(t) = 0$; in turn Proposition 2.1 has the following corollary:

Corollary 2.1: Let $t \geq \tau_S$ (respectively $t \geq \tau_C$) and let $i(t)$ be the unique active node at time t under algorithm `SIMPLE-WALK` (resp. `COALESCENT`). For such t,

$$v_{i(t)}(t) = F_n(\lambda_1, \lambda_2, \cdots, \lambda_n).$$

We regard τ_S and τ_C as termination times of respectively `SIMPLE-WALK` and `COALESCENT` in the sense that the value of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ is known to the single active node from

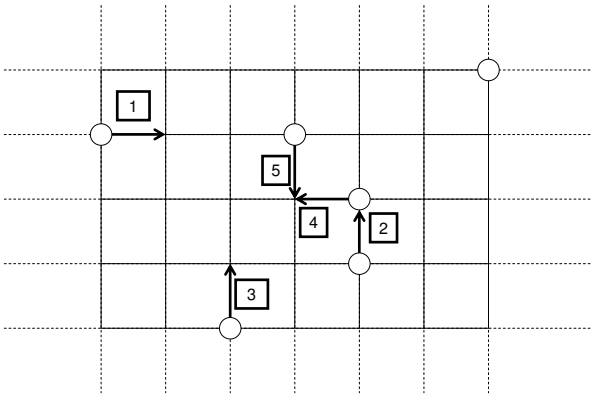


Fig. 2. An illustration of evolution of active nodes in algorithm COALESCENT. The circles in the figure represent nodes that are active at some arbitrary time t_o and arrows represent message transmissions after t_o, in the order indicated by their numbers. After message 1 the transmitting and receiving nodes exchange their status, and the value of the transmitting node passes onto the receiver. Message 2 results in one less active node in the network the transmitter becomes idle whereas the receiver maintain its active status. The number of active nodes do not change due to messages 3 or 4, but decrease again due to message 5.

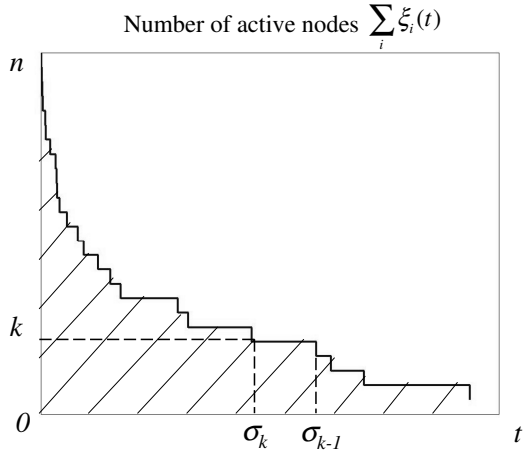


Fig. 3. A sample path of the number of active nodes under algorithm COALESCENT. σ_k denotes the first time that k active nodes remain in the network. This trajectory has qualitatively different statistics for small and large values of t/s_N. The mean of the shaded area is the mean aggregate number of transmitted messages in the network.

there on. If n is known in advance then the algorithms can be modified, namely by including a counter that moves with the transmit token(s), to allow recognition of the termination time by an active node.

Definition 2.1: *Average time complexity* of algorithm SIMPLE-WALK (resp. COALESCENT) refers to $E[\tau_S]$ (resp. $E[\tau_C]$).

In adopting a measure of messaging complexity, let $\eta_S(t)$ and $\eta_C(t)$ be the total number of transmitted messages in the network by time t under algorithms SIMPLE-WALK and COALESCENT respectively:

Definition 2.2: *Average per-node message complexity* of algorithm SIMPLE-WALK (resp. COALESCENT) refers to $n^{-1}E[\eta_S(\tau_S)]$ (resp. $n^{-1}E[\eta_C(\tau_C)]$).

For each time t define $\xi(t) \triangleq (\xi_1(t), \xi_2(t), \cdots, \xi_n(t))$. Note that $(\xi(t) : t \geq 0)$ is a time-homogeneous Markov process under both algorithms. More precisely, $(\xi(t) : t \geq 0)$ is a random walk on G under SIMPLE-WALK, and a coalescing random walk on G under COALESCENT. In particular the average time complexity of SIMPLE-WALK is the mean cover time of G. Average message complexities of the algorithms are characterized by the following lemma:

Lemma 2.1:

$$E[\eta_S(\tau_S)] = E\left[\int_0^{\tau_S} \sum_i \xi_i(t)dt\right] = E[\tau_S],$$

$$E[\eta_C(\tau_C)] = E\left[\int_0^{\tau_C} \sum_i \xi_i(t)dt\right].$$

III. Time and Message Complexities on the Torus

We specialize to lattice tori for which a substantial literature on random walks sheds light on the complexity analysis of the present algorithms. In the scope of this section G is thus a d-dimensional lattice torus, $d \geq 1$, with N^d nodes. In particular $n = N^d$.

Let

$$m_N = \begin{cases} N^2 & \text{if } d = 1 \\ N^2(\log N)^2 & \text{if } d = 2 \\ N^d \log N & \text{if } d \geq 3. \end{cases}$$

The following theorem determines the growth rate of the time as well as the message complexities of SIMPLE-WALK:

Theorem 3.1: ([2], [11])

$$\limsup_{N\to\infty} E[\tau_S]/m_N < \infty,$$
$$\liminf_{N\to\infty} E[\tau_S]/m_N > 0.$$

We next turn to COALESCENT and determine its average time complexity via previously obtained results on the coalescing random walk. Let

$$s_N = \begin{cases} N^2 & \text{if } d = 1 \\ N^2 \log N & \text{if } d = 2 \\ N^d & \text{if } d \geq 3. \end{cases}$$

Theorem 3.2: [4, Theorem 6]

$$0 < \liminf_{N\to\infty} E[\tau_C]/s_N = \limsup_{N\to\infty} E[\tau_C]/s_N < \infty.$$

Obtaining the message complexity of COALESCENT requires a handle on the number of active nodes in the network. A typical sample path of the process $(\sum_i \xi_i(t) : t \geq 0)$ is illustrated in Figure 3. This process has two qualitatively different phases as illustrated by the figure: For small values of t (more precisely for $t = o(s_N)$) there are plenty of active nodes in the network, but their number diminish very quickly due to their high density. For $t = \Omega(s_N)$ only a bounded number of carrier nodes remain. The former phase is short but it involves a high rate of message transmissions, whereas the latter phase lasts long but fewer messages are transmitted

per unit time. The following theorem provides an upper bound for the message complexity based on a bound on $E[\sum_i \xi(t)]$ for $t = o(s_N)$ and on a shaper characterization of the process $(\sum_i \xi_i(t) : t = \Omega(s_N))$ obtained in [4].

Theorem 3.3: [9, Theorem 7]

$$\limsup_{N\to\infty} E[\eta_C(\tau_C)]/m_N \ < \ \infty.$$

Remark 3.1: Note that $E[\eta_C(\tau_C)] \ > \ E[\tau_C]$ since $\sum_i \xi_i(t) \geq 1$ for all t, and therefore Theorem 3.2 provides a lower bound on the growth rate of $E[\eta_C(\tau_C)]$. This bound is order-wise tight for $d = 1$ and is off by at most a factor of $\log N$ in higher dimensions. Simulations suggest that the growth rate of $E[\eta_C(\tau_C)]$ is strictly larger than s_N and strictly smaller than m_N.

In comparing the two algorithms, we note that on the one hand COALESCENT entails higher instantaneous rate of aggregate message transmissions in the network since $\sum_i \xi_i(t) > 1$ for $t < \tau_C$. On the other hand it typically terminates much faster than SIMPLE-WALK. Simulation results in Figure 4 indicate that this compromise settles in favor of COALESCENT on the 2-dimensional torus.

IV. Comparison with Gossip Algorithms

We further specialize to distributed computation of averages, and compare time and message complexities of SIMPLE-WALK and COALESCENT with those of gossip algorithms. Namely, the task here is to obtain the algebraic average of n numbers $x_1, x_2, \cdots, x_n$ each of which is known to a distinct node of G. As noted by Example 1.1 this task can be accomplished by algorithms SIMPLE-WALK or COALESCENT by proper choice of Λ, f and λ_i (in particular by choosing $\lambda_i = (x_i, 1)$ for node i).

In broad terms gossip algorithms refer to distributed randomized algorithms that are based on pairwise relaxations between randomly chosen node pairs. In the present context a pairwise relaxation refers to averaging of two values available at distinct nodes. For completeness we next give a full description of this algorithm as studied in [3]. In what follows a stochastic matrix $P = [P_{ij}]_{n\times n}$ is called *admissible* for G if $P_{ij} = 0$ unless nodes i and j are neighbors in G. The algorithm is parameterized by such P:

Algorithm GOSSIP-AVE(P): Each node i maintains a real valued variable with initial value $z_i(0) = x_i$. At the tick of a local Poisson clock, say at time t_o, node i chooses a neighbor j with respect to the distribution $(P_{ij} : j = 1, 2, \cdots, n)$ and both nodes update their internal variables as $z_i(t_o) = z_j(t_o) = (z_i(t_o^-) + z_j(t_o^-))/2$.

Let $\bar{x}$ denote the average of $x_1, x_2, \cdots, x_n$, let $z(t)$ denote the vector $(z_1(t), z_2(t), \cdots, z_n(t))$ of node values at time t, and $\mathbf{1}$ denote the vector of all 1s. Define τ_k as the kth time instant such that some local clock ticks and thereby triggers messaging in the network. For $\varepsilon > 0$ let the deterministic quantity $K(\varepsilon, P)$ be defined by

$$K(\varepsilon, P) = \sup_{z(0)} \inf \left\{ k \ : \ Pr\left(\frac{\|z(\tau_k) - \bar{x}\mathbf{1}\|_2}{\|z(0)\|_2} \geqslant \varepsilon \right) \leqslant \varepsilon \right\}.$$

In [3] $K(\varepsilon, P)$ is considered as a termination time for Algorithm GOSSIP-AVE(P) and minimization of $K(\varepsilon, P)$ is sought by proper choice of P. Here we adopt the same interpretation for comparison purposes. It should perhaps be noted here that this is a fairly weak stopping criterion as $\|z(\tau_{K(\varepsilon,P)}) - \bar{x}\mathbf{1})\|_\infty/|\bar{x}|$ may be much larger than ε. The following theorem provides a lower bound for $K(\varepsilon, P)$ that applies uniformly for all P on the d-dimensional lattice torus:

Theorem 4.1: [9, Theorem 8] Let G be d-dimensional lattice torus with N^d nodes. Given $\varepsilon > 0$, $K(\varepsilon, P) = \Omega(N^{d+2})$ for all admissible P.

We note that $K(\varepsilon, P)$ is a termination criterion in terms of the transmitted messages in the network. Specifically, $K(\varepsilon, P)$ represents half of the aggregate number of messages transmitted in the network before Algorithm GOSSIP-AVE(P) terminates (to be precise, every time a clock ticks two messages are transmitted in opposite directions on some edge). This observation translates to a termination-time estimate since the point process $(\tau_1, \tau_2, \tau_3, \cdots)$ is Poisson with rate n and it takes roughly $K(\varepsilon, P)/n$ time units to produce a total of $2K(\varepsilon, P)$ messages. Theorem 4.1 thus has the following corollary:

Corollary 4.1: On the d-dimensional lattice torus with N^d nodes, both the time and the per-node message complexities of GOSSIP-AVE(P) are $\Omega(N^2)$ for any admissible P.

Remark 4.1: ([9, Lemma 15]) The two complexities of GOSSIP-AVE(P) scale as $\Theta(N^2)$ when P reflects uniform choice among neighbors in the torus.

In Figure 5 simulation results for the number of messages transmitted per node under algorithm COALESCENT are compared to an analytical lower bound for those of GOSSIP-AVE obtained via [3, Theorem 3]. Although the average time complexity of COALESCENT is larger, the per-node message transmission rate under COALESCENT tends to n^{-1} with time and the overall effect is a substantial asymptotic gain in message complexity with respect to GOSSIP-AVE.

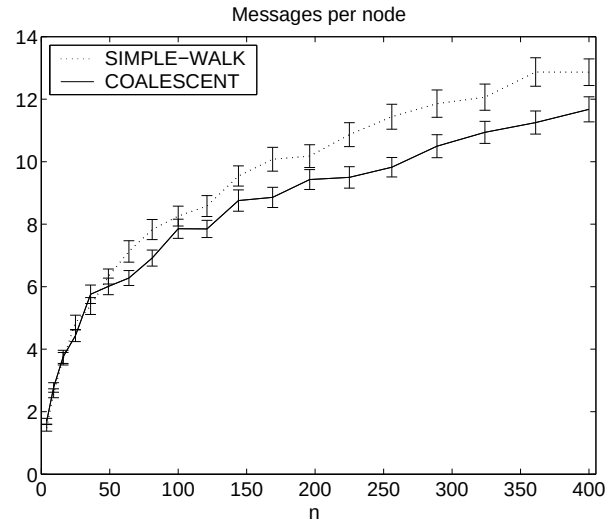
Complexity of GOSSIP-AVE(P) is inherently related to the spectral gap of the stochastic matrix P [3], and thereby to mixing times of random walks on the torus. Conclusions of the present section are thus expected to hold for other distributed algorithms such as those of [7], [8], [10] that are based on powers of the incidence matrix of an underlying connectivity graph, provided that a similar stopping criteria is adopted.

V. Conclusion

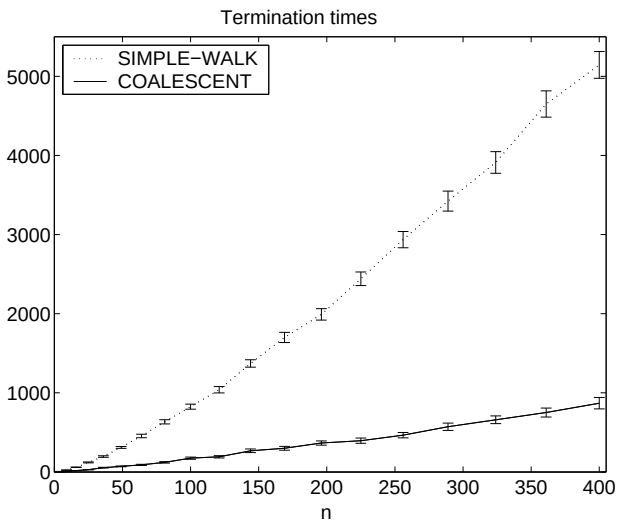
Main conclusions of the paper are summarized in Table I. On the lattice torus the algorithms introduced here lead to substantial gains in message complexity compared to gossip algorithms. These gains entail some sacrifice in termination time but the tradeoff appears favorable, particularly in lower dimensions. The algorithms are parsimonious in message transmission due to their sequential nature and thereby appear suitable for energy-critical applications.

Acknowledgment

This research was supported by Presidential Early Career Award N00014-02-100362, NSF CAREER Programs ANI-



(a) Per-node message complexities



(b) Time complexities

Fig. 4. Empirical values for the message and the time complexities on the 2-dimensional torus. Error bars indicate 95% confidence intervals.

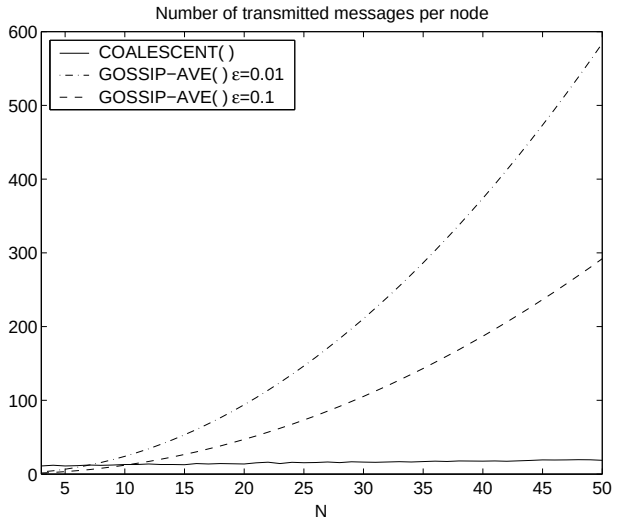


Fig. 5. Number of messages per node on the 2-dimensional torus with N^2 nodes. The solid curve represents simulation results for `COALESCENT` whereas the dotted curves represent lower bounds for `GOSSIP-AVE`(P) based on a lower bound for $K(\varepsilon, P)$. Note also that exact value of $F_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ is obtained at the termination of `COALESCENT` whereas no such claim can be made for `GOSSIP-AVE`(P).

	SIMPLE-WALK	COALESCENT	GOSSIP-AVE
$d = 1$	$\Theta(n)$	$\Theta(n)$	$\Omega(n^2)$
$d = 2$	$\Theta((\log n)^2)$	$O((\log n)^2)$	$\Omega(n)$
$d \geq 3$	$\Theta(\log n)$	$O(\log n)$	$\Omega(n^{2/d})$

(a) Per-node message complexity.

	SIMPLE-WALK	COALESCENT	GOSSIP-AVE
$d = 1$	$\Theta(n^2)$	$\Theta(n^2)$	$\Omega(n^2)$
$d = 2$	$\Theta(n(\log n)^2)$	$\Theta(n \log n)$	$\Omega(n)$
$d \geq 3$	$\Theta(n \log n)$	$\Theta(n)$	$\Omega(n^{2/d})$

(b) Time complexity.

TABLE I

AVERAGE TIME AND PER-NODE MESSAGE COMPLEXITIES ON THE d-DIMENSIONAL TORUS WITH n NODES.

0238397, ECS-0449194 and NSF programs CCF-0430983, and CNS-0435353.

REFERENCES

[1] M. Alanyali, S. Venkatesh, O. Savas, and S. Aeron. Distributed bayesian hypothesis testing in sensor networks. *American Control Conference*, Boston, MA, July 2004.

[2] D.J. Aldous. On the time taken by random walks on finite groups to visit every state, *Z. Wahrsch. Verw. Gebiete*, vol. 62, pp. 361–374, 1983.

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

[4] J.T. Cox. Coalescing random walks and voter model consensus times on the torus. *The Annals of Probability*, 17(4):1333-1366, 1989.

[5] A. Giridhar and P.R. Kumar, Computing and communicating functions over sensor networks. *IEEE Journal on Selected Areas in Communications.* pp. 755–764, vol. 23, no. 4, April 2005.

[6] N. Khude, A. Kumar and A. Karnik, Time and Energy Complexity of Distributed Computation in Wireless Sensor Networks, *IEEE INFOCOM 2005*, 2005.

[7] R. Olfati-Saber and R. M. Murray. Consensus Problems in Networks of Agents with Switching Topology and Time-Delays. *IEEE Transactions On Automatic Control*, vol. 49, pp. 1520-1533, Sep., 2004.

[8] V. Saligrama, M. Alanyali and O. Savas. Distributed detection in sensor networks with packet losses and finite capacity links. *IEEE Transactions on Signal Processing*, to appear, 2005.

[9] O. Savas, M.Alanyali, and V. Saligrama, Efficient In-Network Processing through Local Ad-hoc Information Coalescence. Technical Report. Boston University, Dept. of Electrical and Computer Engineering. 2006.

[10] D.S. Scherber and H.C. Papadopoulos. Distributed computation of averages over ad hoc networks. *IEEE Journal on Selected Areas in Communications*, vol. 23(4), 2005.

[11] D. Zuckerman, A technique for lower bounding the cover time, *SIAM Journal on Discrete Mathematics*, no. 5, pp. 81-87, 1992.