Fast Distributed Algorithms for Computing Separable Functions

Damon Mosk-Aoyama and Devavrat Shah, *Member, IEEE*

***Abstract*—The problem of computing functions of values at the nodes in a network in a fully distributed manner, where nodes do not have unique identities and make decisions based only on local information, has applications in sensor, peer-to-peer, and ad-hoc networks. The task of computing separable functions, which can be written as linear combinations of functions of individual variables, is studied in this context. Known iterative algorithms for averaging can be used to compute the normalized values of such functions, but these algorithms do not extend in general to the computation of the actual values of separable functions.**

The main contribution of this paper is the design of a distributed randomized algorithm for computing separable functions. The running time of the algorithm is shown to depend on the running time of a minimum computation algorithm used as a subroutine. Using a randomized gossip mechanism for minimum computation as the subroutine yields a complete fully distributed algorithm for computing separable functions. For a class of graphs with small spectral gap, such as grid graphs, the time used by the algorithm to compute averages is of a smaller order than the time required by a known iterative averaging scheme.

***Index Terms*—Data aggregation, distributed algorithms, gossip algorithms, randomized algorithms.**

I. Introduction

THE development of sensor, peer-to-peer, and ad hoc wireless networks has stimulated interest in distributed algorithms for data aggregation, in which nodes in a network compute a function of local values at the individual nodes. These networks typically do not have centralized agents that organize the computation and communication among the nodes. Furthermore, the nodes in such a network may not know the complete topology of the network, and the topology may change over time as nodes are added and other nodes fail. In light of the preceding considerations, distributed computation is of vital importance in these modern networks.

We consider the problem of computing separable functions in a distributed fashion in this paper. A separable function can be expressed as the sum of the values of individual functions. Given a network in which each node has a number, we seek a distributed protocol for computing the value of a separable function of the numbers at the nodes. Each node has its own estimate of the value of the function, which evolves as the protocol proceeds. Our goal is to minimize the amount of time required for all of these estimates to be close to the actual function value.

In this work, we are interested in *fully distributed* computations, in which nodes have a local view of the state of the network. Specifically, an individual node does not have information about nodes in the network other than its neighbors. To accurately estimate the value of a separable function that depends on the numbers at all of the nodes, each node must obtain information about the other nodes in the network. This is accomplished through communication between neighbors in the network. Over the course of the protocol, the global state of the network effectively diffuses to each individual node via local communication among neighbors.

More concretely, we assume that each node in the network knows only its neighbors in the network topology, and can contact any neighbor to initiate a communication. On the other hand, we assume that the nodes do not have unique identities (i.e., a node has no unique identifier that can be attached to its messages to identify the source of the messages). This constraint is natural in ad-hoc and mobile networks, where there is a lack of infrastructure (such as IP addresses or static GPS locations), and it limits the ability of a distributed algorithm to recreate the topology of the network at each node. In this sense, the constraint also provides a formal way to distinguish distributed algorithms that are truly local from algorithms that operate by gathering enormous amounts of global information at all the nodes.

The absence of identifiers for nodes makes it difficult, without global coordination, to simply transmit every node's value throughout the network so that each node can identify the values at all the nodes. As such, we develop an algorithm for computing separable functions that relies on an *order- and duplicate-insensitive* statistic [1] of a set of numbers, the minimum. The algorithm is based on properties of exponential random variables, and reduces the problem of computing the value of a separable function to the problem of determining the minimum of a collection of numbers, one for each node.

This reduction leads us to study the problem of *information spreading* or *information dissemination* in a network. In this problem, each node starts with a message, and the nodes must spread the messages throughout the network using local communication so that every node eventually has every message. Because the minimum of a collection of numbers is not affected by the order in which the numbers appear, nor by the presence of duplicates of an individual number, the minimum computation required by our algorithm

D. Mosk-Aoyama is with the Department of Computer Science, Stanford University, Stanford, CA, 94305 USA e-mail: damonma@cs.stanford.edu.

D. Shah is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 02139 USA e-mail: devavrat@mit.edu.

The material in this paper was presented at the 25th Annual ACM Symposium on Principles of Distributed Computing, Denver, CO, July 2006.

Manuscript received April 22, 2007; revised February 25, 2008.

0000–0000/00$00.00 © 2008 IEEE

for computing separable functions can be performed by any information spreading algorithm. Our analysis of the algorithm for computing separable functions establishes an upper bound on its running time in terms of the running time of the information spreading algorithm it uses as a subroutine.

In view of our goal of distributed computation, we analyze a *gossip* algorithm for information spreading. Gossip algorithms are a useful tool for achieving fault-tolerant and scalable distributed computations in large networks. In a gossip algorithm, each node repeatedly initiates communication with a small number of neighbors in the network, and exchanges information with those neighbors.

The gossip algorithm for information spreading that we study is randomized, with the communication partner of a node at any time determined by a simple probabilistic choice. We provide an upper bound on the running time of the algorithm in terms of the *conductance* of a stochastic matrix that governs how nodes choose communication partners. By using the gossip algorithm to compute minima in the algorithm for computing separable functions, we obtain an algorithm for computing separable functions whose performance on certain graphs compares favorably with that of known iterative distributed algorithms [2] for computing averages in a network.

A. Related work

In this section, we present a brief summary of related work. Algorithms for computing the number of distinct elements in a multiset or data stream [3], [4] can be adapted to compute separable functions using information spreading [5]. We are not aware, however, of a previous analysis of the amount of time required for these algorithms to achieve a certain accuracy in the estimates of the function value when the computation is fully distributed (i.e., when nodes do not have unique identities). These adapted algorithms require the nodes in the network to make use of a common hash function. In addition, the discreteness of the counting problem makes the resulting algorithms for computing separable functions suitable only for functions in which the terms in the sum are integers. Our algorithm is simpler than these algorithms, and can compute functions with non-integer terms.

There has been a lot of work on the distributed computation of averages, a special case of the problem of reaching agreement or consensus among processors via a distributed computation. Distributed algorithms for reaching consensus under appropriate conditions have been known since the classical work of Tsitsiklis [6] and Tsitsiklis, Bertsekas, and Athans [7] (see also the book by Bertsekas and Tsitsiklis [8]). Averaging algorithms compute the ratio of the sum of the input numbers to n, the number of nodes in the network, and not the exact value of the sum. Thus, such algorithms cannot be extended in general to compute arbitrary separable functions. On the other hand, an algorithm for computing separable functions can be used to compute averages by separately computing the sum of the input numbers, and the number of nodes in the graph (using one as the input at each node).

Recently, Kempe, Dobra, and Gehrke showed the existence of a randomized iterative gossip algorithm for averaging with the optimal averaging time [9]. This result was restricted to complete graphs. The algorithm requires that the nodes begin the computation in an asymmetric initial state in order to compute separable functions, a requirement that may not be convenient for large networks that do not have centralized agents for global coordination. Furthermore, the algorithm suffers from the possibility of oscillation throughout its execution.

In a more recent paper, Boyd, Ghosh, Prabhakar, and Shah presented a simpler iterative gossip algorithm for averaging that addresses some of the limitations of the Kempe et al. algorithm [2]. Specifically, the algorithm and analysis are applicable to arbitrary graph topologies. Boyd et al. showed a connection between the averaging time of the algorithm and the mixing time (a property that is related to the conductance, but is not the same) of an appropriate random walk on the graph representing the network. They also found an optimal averaging algorithm as a solution to a semi-definite program.

For completeness, we contrast our results for the problem of averaging with known results. As we shall see, iterative averaging, which has been a common approach in the previous work, is an order slower than our algorithm for many graphs, including ring and grid graphs. In this sense, our algorithm is quite different than (and has advantages in comparison with) the known averaging algorithms.

On the topic of information spreading, gossip algorithms for disseminating a message to all nodes in a complete graph in which communication partners are chosen uniformly at random have been studied for some time [10]–[12]. Karp, Schindelhauer, Shenker, and Vöcking presented a *push and pull* gossip algorithm, in which communicating nodes both send and receive messages, that disseminates a message to all n nodes in a graph in $O(\log n)$ time with high probability [13]. In this work, we have provided an analysis of the time required for a gossip algorithm to disseminate n messages to n nodes for the more general setting of arbitrary graphs and non-uniform random choices of communication partners. For other related results, we refer the reader to [14]–[16]. We take note of the similar (independent) recent work of Ganesh, Massoulié, and Towsley [17], and Berger, Borgs, Chayes, and Saberi [18], on the spread of epidemics in a network.

B. Organization

The rest of the paper is organized as follows. Section II presents the distributed computation problems we study and an overview of our results. In Section III, we develop and analyze an algorithm for computing separable functions in a distributed manner. Section IV contains an analysis of a simple randomized gossip algorithm for information spreading, which can be used as a subroutine in the algorithm for computing separable functions. In Section V, we discuss applications of our results to particular types of graphs, and compare our results to previous results for computing averages. Finally, we present conclusions and future directions in Section VI.

II. Preliminaries and Results

We consider an arbitrary connected network, represented by an undirected graph $G = (V, E)$, with $|V| = n$ nodes.

For notational purposes, we assume that the nodes in V are numbered arbitrarily so that $V = \{1, \ldots, n\}$. A node, however, does not have a unique identity that can be used in a computation. Two nodes i and j can communicate with each other if (and only if) $(i, j) \in E$.

To capture some of the resource constraints in the networks in which we are interested, we impose a *transmitter gossip* constraint on node communication. Each node is allowed to contact at most one other node at a given time for communication. However, a node can be contacted by multiple nodes simultaneously.

Let 2^V denote the power set of the vertex set V (the set of all subsets of V). For an n-dimensional vector $\vec{x} \in \mathbf{R}^n$, let $x_1, \ldots, x_n$ be the components of $\vec{x}$.

Definition 1: We say that a function $f : \mathbf{R}^n \times 2^V \to \mathbf{R}$ is *separable* if there exist functions $f_1, \ldots, f_n$ such that, for all $\vec{x} \in \mathbf{R}^n$ and $S \subseteq V$,

$$f(\vec{x}, S) = \sum_{i \in S} f_i(x_i). \tag{1}$$

Goal. Let $\mathcal{F}$ be the class of separable functions f for which $f_i(x) \geq 1$ for all $x \in \mathbf{R}$ and $i = 1, \ldots, n$. Given a function $f \in \mathcal{F}$, and a vector $\vec{x}$ containing initial values x_i for all the nodes, the nodes in the network are to compute the value $f(\vec{x}, V)$ by a distributed computation, using repeated communication between nodes.

Note 1: Consider a function g for which there exist functions $g_1, \ldots, g_n$ satisfying, for all $S \subseteq V$, the condition $g(\vec{x}, S) = \prod_{i \in S} g_i(x_i)$ in lieu of (1). Then, g is *logarithmic separable*, i.e., $f = \log_b g$ is separable. Our algorithm for computing separable functions can be used to compute the function $f = \log_b g$. The condition $f_i(x) \geq 1$ corresponds to $g_i(x) \geq b$ in this case. This lower bound of 1 on $f_i(x)$ is arbitrary, although our algorithm does require the terms $f_i(x_i)$ in the sum to be positive.

Before proceeding further, we list some practical situations where the distributed computation of separable functions arises naturally. By definition, the sum of a set of numbers is a separable function.

1) *Summation.* Let the value at each node be $x_i = 1$. Then, the sum of the values is the number of nodes in the network.
2) *Averaging.* According to Definition 1, the average of a set of numbers is not a separable function. However, the nodes can estimate the separable function $\sum_{i=1}^{n} x_i$ and n separately, and use the ratio between these two estimates as an estimate of the mean of the numbers.

   Suppose the values at the nodes are measurements of a quantity of interest. Then, the average provides an unbiased maximum likelihood estimate of the measured quantity. For example, if the nodes are temperature sensors, then the average of the sensed values at the nodes gives a good estimate of the ambient temperature.

For more sophisticated applications of a distributed averaging algorithm, we refer the reader to [19] and [20]. Averaging is used for the distributed computation of the top k eigenvectors of a graph in [19], while in [20] averaging is used in a throughput-optimal distributed scheduling algorithm in a wireless network.

Time model. In a distributed computation, a time model determines when nodes communicate with each other. We consider two time models, one synchronous and the other asynchronous, in this paper. The two models are described as follows.

1) *Synchronous time model:* Time is slotted commonly across all nodes in the network. In any time slot, each node may contact one of its neighbors according to a random choice that is independent of the choices made by the other nodes. The simultaneous communication between the nodes satisfies the transmitter gossip constraint.
2) *Asynchronous time model:* Each node has a clock that ticks at the times of a rate 1 Poisson process. Equivalently, a common clock ticks according to a rate n Poisson process at times $C_k, k \geq 1$, where $\{C_{k+1} - C_k\}$ are i.i.d. exponential random variables of rate n. On clock tick k, one of the n nodes, say I_k, is chosen uniformly at random. We consider this global clock tick to be a tick of the clock at node I_k. When a node's clock ticks, it contacts one of its neighbors at random. In this model, time is discretized according to clock ticks. On average, there are n clock ticks per one unit of absolute time.

In this paper, we measure the running times of algorithms in absolute time, which is the number of time slots in the synchronous model, and is (on average) the number of clock ticks divided by n in the asynchronous model. To obtain a precise relationship between clock ticks and absolute time in the asynchronous model, we appeal to tail bounds on the probability that the sample mean of i.i.d. exponential random variables is far from its expected value. In particular, we make use of the following lemma, which also plays a role in the analysis of the accuracy of our algorithm for computing separable functions.

Lemma 1: For any $k \geq 1$, let $Y_1, \ldots, Y_k$ be i.i.d. exponential random variables with rate λ. Let $R_k = \frac{1}{k} \sum_{i=1}^{k} Y_i$. Then, for any $\varepsilon \in (0, 1/2)$,

$$\Pr\left(\left|R_k - \frac{1}{\lambda}\right| \geq \frac{\varepsilon}{\lambda}\right) \leq 2 \exp\left(-\frac{\varepsilon^2 k}{3}\right). \tag{2}$$

Proof: By definition, $E[R_k] = \frac{1}{k} \sum_{i=1}^{k} \lambda^{-1} = \lambda^{-1}$. The inequality in (2) follows directly from Cramér's Theorem (see [21], pp. 30, 35) and properties of exponential random variables. ■

A direct implication of Lemma 1 is the following corollary, which bounds the probability that the absolute time C_k at which clock tick k occurs is far from its expected value.

Corollary 1: For $k \geq 1$, $E[C_k] = k/n$. Further, for any $\varepsilon \in (0, 1/2)$,

$$\Pr\left(\left|C_k - \frac{k}{n}\right| \geq \frac{\varepsilon k}{n}\right) \leq 2 \exp\left(-\frac{\varepsilon^2 k}{3}\right).$$

Our algorithm for computing separable functions is randomized, and is not guaranteed to compute the exact quantity

$f(\vec{x}, V) = \sum_{i=1}^n f_i(x_i)$ at each node in the network. To study the accuracy of the algorithm's estimates, we analyze the probability that the estimate of $f(\vec{x}, V)$ at every node is within a $(1 \pm \varepsilon)$ multiplicative factor of the true value $f(\vec{x}, V)$ after the algorithm has run for some period of time. In this sense, the error in the estimates of the algorithm is relative to the magnitude of $f(\vec{x}, V)$.

To measure the amount of time required for an algorithm's estimates to achieve a specified accuracy with a specified probability, we define the following quantity. For an algorithm $\mathcal{C}$ that estimates $f(\vec{x}, V)$, let $\hat{y}_i(t)$ be the estimate of $f(\vec{x}, V)$ at node i at time t. Furthermore, for notational convenience, given $\varepsilon > 0$, let $A_i^\varepsilon(t)$ be the following event.

$$A_i^\varepsilon(t) = \{\hat{y}_i(t) \notin [(1-\varepsilon) f(\vec{x}, V), (1+\varepsilon) f(\vec{x}, V)]\}$$

Definition 2: For any $\varepsilon > 0$ and $\delta \in (0,1)$, the (ε, δ)-computing time of $\mathcal{C}$, denoted $T_{\mathcal{C}}^{\text{cmp}}(\varepsilon, \delta)$, is

$$\begin{aligned} & T_{\mathcal{C}}^{\text{cmp}}(\varepsilon, \delta) \\ & \quad = \sup_{f \in \mathcal{F}} \sup_{\vec{x} \in \mathbf{R}^n} \inf \Big\{ \tau : \forall t \geq \tau, \Pr\left(\cup_{i=1}^n A_i^\varepsilon(t) \right) \leq \delta \Big\}. \end{aligned}$$

Intuitively, the significance of this definition of the (ε, δ)-computing time of an algorithm $\mathcal{C}$ is that, if $\mathcal{C}$ runs for an amount of time that is at least $T_{\mathcal{C}}^{\text{cmp}}(\varepsilon, \delta)$, then the probability that the estimates of $f(\vec{x}, V)$ at the nodes are all within a $(1 \pm \varepsilon)$ factor of the actual value of the function is at least $1 - \delta$.

As noted before, our algorithm for computing separable functions is based on a reduction to the problem of information spreading, which is described as follows. Suppose that, for $i = 1, \ldots, n$, node i has the one message m_i. The task of information spreading is to disseminate all n messages to all n nodes via a sequence of local communications between neighbors in the graph. In any single communication between two nodes, each node can transmit to its communication partner any of the messages that it currently holds. We assume that the data transmitted in a communication must be a set of messages, and therefore cannot be arbitrary information.

Consider an information spreading algorithm $\mathcal{D}$, which specifies how nodes communicate. For each node $i \in V$, let $S_i(t)$ denote the set of nodes that have the message m_i at time t. While nodes can gain messages during communication, we assume that they do not lose messages, so that $S_i(t_1) \subseteq S_i(t_2)$ if $t_1 \leq t_2$. Analogous to the (ε, δ)-computing time, we define a quantity that measures the amount of time required for an information spreading algorithm to disseminate all the messages m_i to all the nodes in the network.

Definition 3: For $\delta \in (0,1)$, the δ-information-spreading time of the algorithm $\mathcal{D}$, denoted $T_{\mathcal{D}}^{\text{spr}}(\delta)$, is

$$T_{\mathcal{D}}^{\text{spr}}(\delta) = \inf \left\{ t : \Pr\left(\cup_{i=1}^n \{ S_i(t) \neq V \} \right) \leq \delta \right\}.$$

In our analysis of the gossip algorithm for information spreading, we assume that when two nodes communicate, each node can send all of its messages to the other in a single communication. This rather unrealistic assumption of *infinite* link capacity is merely for convenience, as it provides a simpler analytical characterization of $T_{\mathcal{C}}^{\text{cmp}}(\varepsilon, \delta)$ in terms of $T_{\mathcal{D}}^{\text{spr}}(\delta)$. Our algorithm for computing separable functions requires only links of unit capacity.

A. Our contribution

The main contribution of this paper is the design of a distributed algorithm to compute separable functions of node values in an arbitrary connected network. Our algorithm is randomized, and in particular uses exponential random variables. This usage of exponential random variables is analogous to that in an algorithm by Cohen[1] for estimating the sizes of sets in a graph [22]. The basis for our algorithm is the following property of the exponential distribution.

Property 1: Let $W_1, \ldots, W_n$ be n independent random variables such that, for $i = 1, \ldots, n$, the distribution of W_i is exponential with rate λ_i. Let $\bar{W}$ be the minimum of $W_1, \ldots, W_n$. Then, $\bar{W}$ is distributed as an exponential random variable of rate $\lambda = \sum_{i=1}^n \lambda_i$.

Proof: For an exponential random variable W with rate λ, for any $z \in \mathbf{R}_+$,

$$\Pr(W > z) = \exp(-\lambda z).$$

Using this fact and the independence of the random variables W_i, we compute $\Pr(\bar{W} > z)$ for any $z \in \mathbf{R}_+$.

$$\begin{aligned} \Pr(\bar{W} > z) &= \Pr\left(\cap_{i=1}^n \{ W_i > z \} \right) \\ &= \prod_{i=1}^n \Pr(W_i > z) \\ &= \prod_{i=1}^n \exp(-\lambda_i z) \\ &= \exp\left(-z \sum_{i=1}^n \lambda_i \right) \end{aligned}$$

This establishes the property stated above. ∎

Our algorithm uses an information spreading algorithm as a subroutine, and as a result its running time is a function of the running time of the information spreading algorithm it uses. The faster the information spreading algorithm is, the better our algorithm performs. Specifically, the following result provides an upper bound on the (ε, δ)-computing time of the algorithm.

Theorem 1: Given an information spreading algorithm $\mathcal{D}$ with δ-spreading time $T_{\mathcal{D}}^{\text{spr}}(\delta)$ for $\delta \in (0,1)$, there exists an algorithm $\mathcal{A}$ for computing separable functions $f \in \mathcal{F}$ such that, for any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$,

$$T_{\mathcal{A}}^{\text{cmp}}(\varepsilon, \delta) \leq 18 \varepsilon^{-2} \left(1 + \ln \delta^{-1} \right) T_{\mathcal{D}}^{\text{spr}}(\delta/2).$$

Motivated by our interest in decentralized algorithms, we analyze a simple randomized gossip algorithm for information spreading. When node i initiates a communication, it contacts each node $j \neq i$ with probability P_{ij}. With probability P_{ii}, it does not contact another node. The $n \times n$ matrix $P = [P_{ij}]$ characterizes the algorithm; each matrix P gives rise to an information spreading algorithm $\mathcal{P}$. We assume that P is stochastic, and that $P_{ij} = 0$ if $i \neq j$ and $(i,j) \notin E$, as nodes

[1] We thank Dahlia Malkhi for pointing this reference out to us.

that are not neighbors in the graph cannot communicate with each other. Section IV describes the data transmitted between two nodes when they communicate.

We obtain an upper bound on the δ-information-spreading time of this gossip algorithm in terms of the *conductance* of the matrix P, which is defined as follows.

Definition 4: For a stochastic matrix P, the conductance of P, denoted $\Phi(P)$, is

$$\Phi(P) = \min_{S \subset V,\, 0<|S|\le n/2} \frac{\sum_{i\in S, j\notin S} P_{ij}}{|S|}.$$

In general, the above definition of conductance is not the same as the classical definition [23]. However, we restrict our attention in this paper to doubly stochastic matrices P. When P is doubly stochastic, these two definitions are equivalent.

Note that the definition of conductance implies that $\Phi(P) \le 1$. Throughout the remainder of the paper, we assume that $n \ge 3$ and $\Phi(P) > 0$. Without these assumptions, each node in the network would have at most one neighbor to communicate with, or the network would contain a non-empty node subset $S \subset V$ such that no node in S could contact a node outside of S.

Theorem 2: Consider any doubly stochastic matrix P such that if $i \neq j$ and $(i,j) \notin E$, then $P_{ij} = 0$. There exists an information dissemination algorithm $\mathcal{P}$ such that, for any $\delta \in (0,1)$,

$$T_{\mathcal{P}}^{\mathrm{spr}}(\delta) \le \frac{62\left(\ln n + \ln \delta^{-1}\right)}{\Phi(P)}.$$

Note 2: The results of Theorems 1 and 2 hold for both the synchronous and asynchronous time models. Recall that the quantities $T_{\mathcal{C}}^{\mathrm{cmp}}(\varepsilon,\delta)$ and $T_{\mathcal{D}}^{\mathrm{spr}}(\delta)$ are defined with respect to absolute time in both models.

A comparison. Theorems 1 and 2 imply that, given a doubly stochastic matrix P, the time required for our algorithm to obtain a $(1 \pm \varepsilon)$ approximation with probability at least $1-\delta$ is, up to constant factors, at most $\frac{\varepsilon^{-2}(1+\ln\delta^{-1})(\ln n+\ln\delta^{-1})}{\Phi(P)}$. When the network size n and the accuracy parameters ε and δ are fixed, the running time scales in proportion to $1/\Phi(P)$, a factor that captures the dependence of the algorithm on the matrix P. Our algorithm can be used to compute the average of a set of numbers. For iterative averaging algorithms such as the ones in [6] and [2], the convergence time largely depends on the mixing time of P, which is lower bounded by $\Omega(1/\Phi(P))$ (see [23], for example). Thus, our algorithm is (up to a $\ln n$ factor) no slower than the fastest iterative algorithm based on time-invariant linear dynamics.

III. Function Computation

In this section, we describe our algorithm for computing the value $y = f(\vec{x}, V) = \sum_{i=1}^n f_i(x_i)$ of the separable function f, where $f_i(x_i) \ge 1$. For simplicity of notation, let $y_i = f_i(x_i)$. Given x_i, each node can compute y_i on its own. Next, the nodes use the algorithm shown in Fig. 1, which we refer to as COMP, to compute estimates $\hat{y}_i$ of $y = \sum_{i=1}^n y_i$. The quantity r is a parameter to be chosen later.

We describe how the minimum is computed as required by step **2** of the algorithm in Section III-A. The running time of the algorithm COMP depends on the running time of the algorithm used to compute the minimum.

```
Algorithm COMP
0. Initially, for $i = 1, \dots, n$, node $i$ has the value $y_i \ge 1$.
1. Each node $i$ generates $r$ independent random numbers
   $W_1^i, \dots, W_r^i$, where the distribution of each $W_\ell^i$ is
   exponential with rate $y_i$ (i.e., with mean $1/y_i$).
2. Each node $i$ computes, for $\ell = 1, \dots, r$, an estimate $\hat{W}_\ell^i$
   of the minimum $\bar{W}_\ell = \min_{i=1}^n W_\ell^i$. This computation
   can be done using an information spreading algorithm
   as described below.
3. Each node $i$ computes $\hat{y}_i = \frac{r}{\sum_{\ell=1}^r \hat{W}_\ell^i}$ as its estimate of
   $\sum_{i=1}^n y_i$.
```

Fig. 1. An algorithm for computing separable functions.

Now, we show that COMP effectively estimates the function value y when the estimates $\hat{W}_\ell^i$ are all correct by providing a lower bound on the conditional probability that the estimates produced by COMP are all within a $(1 \pm \varepsilon)$ factor of y.

Lemma 2: Let $y_1, \dots, y_n$ be real numbers (with $y_i \ge 1$ for $i = 1, \dots, n$), $y = \sum_{i=1}^n y_i$, and $\bar{W} = (\bar{W}_1, \dots, \bar{W}_r)$, where the $\bar{W}_\ell$ are as defined in the algorithm COMP. For any node i, let $\hat{W}^i = (\hat{W}_1^i, \dots, \hat{W}_r^i)$, and let $\hat{y}_i$ be the estimate of y obtained by node i in COMP. For any $\varepsilon \in (0, 1/2)$,

$$\Pr\left(\cup_{i=1}^n \{|\hat{y}_i - y| > 2\varepsilon y\} \mid \forall i \in V,\ \hat{W}^i = \bar{W}\right) \le 2\exp\left(-\frac{\varepsilon^2 r}{3}\right).$$

Proof: Observe that the estimate $\hat{y}_i$ of y at node i is a function of r and $\hat{W}^i$. Under the hypothesis that $\hat{W}^i = \bar{W}$ for all nodes $i \in V$, all nodes produce the same estimate $\hat{y} = \hat{y}_i$ of y. This estimate is $\hat{y} = r\left(\sum_{\ell=1}^r \bar{W}_\ell\right)^{-1}$, and so $\hat{y}^{-1} = \left(\sum_{\ell=1}^r \bar{W}_\ell\right) r^{-1}$.

Property 1 implies that each of the n random variables $\bar{W}_1, \dots, \bar{W}_r$ has an exponential distribution with rate y. From Lemma 1, it follows that for any $\varepsilon \in (0, 1/2)$,

$$\Pr\left(\left|\hat{y}^{-1} - \frac{1}{y}\right| > \frac{\varepsilon}{y} \;\middle|\; \forall i \in V,\ \hat{W}^i = \bar{W}\right) \le 2\exp\left(-\frac{\varepsilon^2 r}{3}\right). \tag{3}$$

This inequality bounds the conditional probability of the event $\{\hat{y}^{-1} \notin [(1-\varepsilon)y^{-1}, (1+\varepsilon)y^{-1}]\}$, which is equivalent to the event $\{\hat{y} \notin [(1+\varepsilon)^{-1}y, (1-\varepsilon)^{-1}y]\}$. Now, for $\varepsilon \in (0, 1/2)$,

$$(1-\varepsilon)^{-1} \in [1+\varepsilon, 1+2\varepsilon] \tag{4}$$

and

$$(1+\varepsilon)^{-1} \in [1-\varepsilon, 1-2\varepsilon/3]. \tag{5}$$

Applying the inequalities in (3), (4), and (5), we conclude that for $\varepsilon \in (0, 1/2)$,

$$\Pr\left(|\hat{y} - y| > 2\varepsilon y \mid \forall i \in V,\ \hat{W}^i = \bar{W}\right) \le 2\exp\left(-\frac{\varepsilon^2 r}{3}\right).$$

Noting that the event $\cup_{i=1}^n \{|\hat{y}_i - y| > 2\varepsilon y\}$ is equivalent to the event $\{|\hat{y} - y| > 2\varepsilon y\}$ when $\hat{W}^i = \bar{W}$ for all nodes i completes the proof of Lemma 2. ∎

A. Using information spreading to compute minima

We now elaborate on step **2** of the algorithm COMP. Each node i in the graph starts this step with a vector $W^i = (W_1^i, \ldots, W_r^i)$, and the nodes seek the vector $\bar{W} = (\bar{W}_1, \ldots, \bar{W}_r)$, where $\bar{W}_\ell = \min_{i=1}^n W_\ell^i$. In the information spreading problem, each node i has a message m_i, and the nodes are to transmit messages across the links until every node has every message.

If all link capacities are infinite (i.e., in one time unit, a node can send an arbitrary amount of information to another node), then an information spreading algorithm $\mathcal{D}$ can be used directly to compute the minimum vector $\bar{W}$. To see this, let the message m_i at the node i be the vector W^i, and then apply the information spreading algorithm to disseminate the vectors. Once every node has every message (vector), each node can compute $\bar{W}$ as the component-wise minimum of all the vectors. This implies that the running time of the resulting algorithm for computing $\bar{W}$ is the same as that of the information spreading algorithm.

The assumption of infinite link capacities allows a node to transmit an arbitrary number of vectors W^i to a neighbor in one time unit. A simple modification to the information spreading algorithm, however, yields an algorithm for computing the minimum vector $\bar{W}$ using links of capacity r. To this end, each node i maintains a single r-dimensional vector $w^i(t)$ that evolves in time, starting with $w^i(0) = W^i$.

Suppose that, in the information dissemination algorithm, node j transmits the messages (vectors) $W^{i_1}, \ldots, W^{i_c}$ to node i at time t. Then, in the minimum computation algorithm, j sends to i the r quantities $w_1, \ldots, w_r$, where $w_\ell = \min_{u=1}^c W_\ell^{i_u}$. The node i sets $w_\ell^i(t^+) = \min(w_\ell^i(t^-), w_\ell)$ for $\ell = 1, \ldots, r$, where t^- and t^+ denote the times immediately before and after, respectively, the communication. At any time t, we will have $w^i(t) = \bar{W}$ for all nodes $i \in V$ if, in the information spreading algorithm, every node i has all the vectors $W^1, \ldots, W^n$ at the same time t. In this way, we obtain an algorithm for computing the minimum vector $\bar{W}$ that uses links of capacity r and runs in the same amount of time as the information spreading algorithm.

An alternative to using links of capacity r in the computation of $\bar{W}$ is to make the time slot r times larger, and impose a unit capacity on all the links. Now, a node transmits the numbers $w_1, \ldots, w_r$ to its communication partner over a period of r time slots, and as a result the running time of the algorithm for computing $\bar{W}$ becomes greater than the running time of the information spreading algorithm by a factor of r. The preceding discussion, combined with the fact that nodes only gain messages as an information spreading algorithm executes, leads to the following lemma.

Lemma 3: Suppose that the COMP algorithm is implemented using an information spreading algorithm $\mathcal{D}$ as described above. Let $\hat{W}^i(t)$ denote the estimate of $\bar{W}$ at node i at time t. For any $\delta \in (0,1)$, let $t_m = rT_{\mathcal{D}}^{\text{spr}}(\delta)$. Then, for any time $t \geq t_m$, with probability at least $1 - \delta$, $\hat{W}^i(t) = \bar{W}$ for all nodes $i \in V$.

Note that the amount of data communicated between nodes during the algorithm COMP depends on the values of the exponential random variables generated by the nodes. Since the nodes compute minima of these variables, we are interested in a probabilistic lower bound on the values of these variables (for example, suppose that the nodes transmit the values $1/W_\ell^i$ when computing the minimum $\bar{W}_\ell = 1/\max_{i=1}^n\{1/W_\ell^i\}$). To this end, we use the fact that each $\bar{W}_\ell$ is an exponential random variable with rate y to obtain that, for any constant $c > 1$, the probability that any of the minimum values $\bar{W}_\ell$ is less than $1/B$ (i.e., any of the inverse values $1/W_\ell^i$ is greater than B) is at most δ/c, where B is proportional to cry/δ.

B. Proof of Theorem 1

Now, we are ready to prove Theorem 1. In particular, we will show that the COMP algorithm has the properties claimed in Theorem 1. To this end, consider using an information spreading algorithm $\mathcal{D}$ with δ-spreading time $T_{\mathcal{D}}^{\text{spr}}(\delta)$ for $\delta \in (0,1)$ as the subroutine in the COMP algorithm. For any $\delta \in (0,1)$, let $\tau_m = rT_{\mathcal{D}}^{\text{spr}}(\delta/2)$. By Lemma 3, for any time $t \geq \tau_m$, the probability that $\hat{W}^i \neq \bar{W}$ for any node i at time t is at most $\delta/2$.

On the other hand, suppose that $\hat{W}^i = \bar{W}$ for all nodes i at time $t \geq \tau_m$. For any $\varepsilon \in (0,1)$, by choosing $r = \lceil 12\varepsilon^{-2}\ln(4\delta^{-1})\rceil$, we obtain from Lemma 2 that

$$\Pr\left(\cup_{i=1}^n \{\hat{y}_i \notin [(1-\varepsilon)y, (1+\varepsilon)y]\} \mid \forall i \in V,\, \hat{W}^i = \bar{W}\right) \\ \leq \delta/2. \qquad (6)$$

Note that, because $\varepsilon \in (0,1)$, $r \leq 12\varepsilon^{-2}\ln(4\delta^{-1}) + 1 \leq 18\varepsilon^{-2}(1 + \ln\delta^{-1})$.

Recall that $T_{COMP}^{\text{cmp}}(\varepsilon, \delta)$ is the smallest time τ such that, under the algorithm COMP, at any time $t \geq \tau$, all the nodes have an estimate of the function value y within a multiplicative factor of $(1\pm\varepsilon)$ with probability at least $1-\delta$. By a straightforward union bound of events and (6), we conclude that, for any time $t \geq \tau_m$,

$$\Pr\left(\cup_{i=1}^n \{\hat{y}_i \notin [(1-\varepsilon)y, (1+\varepsilon)y]\}\right) \leq \delta.$$

For any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$, we now have, by the definition of (ε, δ)-computing time,

$$\begin{aligned} T_{COMP}^{\text{cmp}}(\varepsilon, \delta) &\leq \tau_m \\ &\leq 18\varepsilon^{-2}\left(1 + \ln\delta^{-1}\right) T_{\mathcal{D}}^{\text{spr}}(\delta/2). \end{aligned}$$

This completes the proof of Theorem 1.

IV. Information spreading

In this section, we analyze a randomized gossip algorithm for information spreading. The method by which nodes choose partners to contact when initiating a communication and the data transmitted during the communication are the same for both time models defined in Section II. These models differ in the times at which nodes contact each other: in the asynchronous model, only one node can start a communication

Algorithm SPREAD(P)

When a node i initiates a communication at time t:

1. Node i chooses a node u at random, and contacts u. The choice of the communication partner u is made independently of all other random choices, and the probability that node i chooses any node j is P_{ij}.
2. Nodes u and i exchange all of their messages, so that
$$M_i(t^+) = M_u(t^+) = M_i(t^-) \cup M_u(t^-).$$

Fig. 2. A gossip algorithm for information spreading.

at any time, while in the synchronous model all the nodes can communicate in each time slot.

The information spreading algorithm that we study is presented in Fig. 2, which makes use of the following notation. Let $M_i(t)$ denote the set of messages node i has at time t. Initially, $M_i(0) = \{m_i\}$ for all $i \in V$. For a communication that occurs at time t, let t^- and t^+ denote the times immediately before and after, respectively, the communication occurs.

As mentioned in Section II-A, the nodes choose communication partners according to the probability distribution defined by an $n \times n$ matrix P. The matrix P is non-negative and stochastic, and satisfies $P_{ij} = 0$ for any pair of nodes $i \neq j$ such that $(i, j) \notin E$. For each such matrix P, there is an instance of the information spreading algorithm, which we refer to as SPREAD(P).

We note that the data transmitted between two communicating nodes in SPREAD conform to the *push and pull mechanism*. That is, when node i contacts node u at time t, both nodes u and i exchange all of their information with each other. We also note that the description in the algorithm assumes that the communication links in the network have infinite capacity. As discussed in Section III-A, however, an information spreading algorithm that uses links of infinite capacity can be used to compute minima using links of unit capacity.

This algorithm is simple, distributed, and satisfies the transmitter gossip constraint. We now present analysis of the information spreading time of SPREAD(P) for doubly stochastic matrices P in the two time models. The goal of the analysis is to prove Theorem 2. To this end, for any $i \in V$, let $S_i(t) \subseteq V$ denote the set of nodes that have the message m_i after any communication events that occur at absolute time t (communication events occur on a global clock tick in the asynchronous time model, and in each time slot in the synchronous time model). At the start of the algorithm, $S_i(0) = \{i\}$.

A. Asynchronous model

As described in Section II, in the asynchronous time model the global clock ticks according to a Poisson process of rate n, and on a tick one of the n nodes is chosen uniformly at random. This node initiates a communication, so the times at which the communication events occur correspond to the ticks of the clock. On any clock tick, at most one pair of nodes can exchange messages by communicating with each other.

Let $k \geq 0$ denote the index of a clock tick. Initially, $k = 0$, and the corresponding absolute time is 0. For simplicity of notation, we identify the time at which a clock tick occurs with its index, so that $S_i(k)$ denotes the set of nodes that have the message m_i at the end of clock tick k. The following lemma provides a bound on the number of clock ticks required for every node to receive every message.

Lemma 4: For any $\delta \in (0, 1)$, define
$$K(\delta) = \inf\{k \geq 0 : \Pr(\cup_{i=1}^n \{S_i(k) \neq V\}) \leq \delta\}.$$
Then,
$$K(\delta) \leq n\left(\frac{14\ln n + 5\ln \delta^{-1}}{\Phi(P)}\right).$$

Proof: Fix any node $v \in V$. We study the evolution of the size of the set $S_v(k)$. For simplicity of notation, we drop the subscript v, and write $S(k)$ to denote $S_v(k)$.

Note that $|S(k)|$ is monotonically non-decreasing over the course of the algorithm, with the initial condition $|S(0)| = 1$. For the purpose of analysis, we divide the execution of the algorithm into two phases based on the size of the set $S(k)$. In the first phase, $|S(k)| \leq n/2$, and in the second phase $|S(k)| > n/2$.

Under the gossip algorithm, after clock tick $k+1$, we have either $|S(k+1)| = |S(k)|$ or $|S(k+1)| = |S(k)|+1$. Further, the size increases if a node $i \in S(k)$ contacts a node $j \notin S(k)$, as in this case i will push the message m_v to j. For each such pair of nodes i, j, the probability that this occurs on clock tick $k+1$ is P_{ij}/n. Since only one node is active on each clock tick,
$$E[|S(k+1)| - |S(k)| \mid S(k)] \geq \sum_{i\in S(k), j\notin S(k)} \frac{P_{ij}}{n}. \qquad (7)$$
When $|S(k)| \leq n/2$, it follows from (7) and the definition of the conductance $\Phi(P)$ of P that
$$\begin{aligned} &E[|S(k+1)| - |S(k)| \mid S(k)] \\ &\quad\geq \frac{|S(k)|}{n} \frac{\sum_{i\in S(k), j\notin S(k)} P_{ij}}{|S(k)|} \\ &\quad\geq \frac{|S(k)|}{n} \min_{S\subset V,\, 0<|S|\leq n/2} \frac{\sum_{i\in S, j\notin S} P_{ij}}{|S|} \\ &\quad= \frac{|S(k)|}{n}\Phi(P). \end{aligned}$$
Let $\hat{\Phi} = \frac{\Phi(P)}{n}$, so that
$$E[|S(k+1)| - |S(k)| \mid S(k)] \geq |S(k)|\hat{\Phi}. \qquad (8)$$

We seek an upper bound on the duration of the first phase. To this end, let
$$Z(k) = \frac{\exp\left(\frac{\hat{\Phi}}{4}k\right)}{|S(k)|}.$$
Define the stopping time $L = \inf\{k : |S(k)| > n/2\}$, and $L \wedge k = \min(L, k)$. If $|S(k)| > n/2$, then $L \wedge (k+1) = L \wedge k$, and thus $E[Z(L \wedge (k+1)) \mid S(L \wedge k)] = Z(L \wedge k)$.

Now, suppose that $|S(k)| \leq n/2$, in which case $L \wedge (k+1) = (L \wedge k) + 1$. The function $g(z) = 1/z$ is convex for $z > 0$, which implies that, for $z_1, z_2 > 0$,
$$g(z_2) \geq g(z_1) + g'(z_1)(z_2 - z_1). \qquad (9)$$

Applying (9) with $z_1 = |S(k+1)|$ and $z_2 = |S(k)|$ yields

$$\frac{1}{|S(k+1)|} \le \frac{1}{|S(k)|} - \frac{1}{|S(k+1)|^2}(|S(k+1)| - |S(k)|).$$

Since $|S(k+1)| \le |S(k)| + 1 \le 2|S(k)|$, it follows that

$$\frac{1}{|S(k+1)|} \le \frac{1}{|S(k)|} - \frac{1}{4|S(k)|^2}(|S(k+1)| - |S(k)|). \quad (10)$$

Combining (8) and (10), and using the fact that $1 - z \le \exp(-z)$ for $z \ge 0$, we obtain that, if $|S(k)| \le n/2$, then

$$\begin{aligned} E\left[\frac{1}{|S(k+1)|} \;\middle|\; S(k)\right] &\le \frac{1}{|S(k)|}\left(1 - \frac{\hat{\Phi}}{4}\right) \\ &\le \frac{1}{|S(k)|}\exp\left(-\frac{\hat{\Phi}}{4}\right). \end{aligned}$$

This implies that

$$\begin{aligned} &E[Z(L \wedge (k+1)) \mid S(L \wedge k)] \\ &= E\left[\frac{\exp\left(\frac{\hat{\Phi}}{4}(L \wedge (k+1))\right)}{|S(L \wedge (k+1))|} \;\middle|\; S(L \wedge k)\right] \\ &= \exp\left(\frac{\hat{\Phi}}{4}(L \wedge k)\right)\exp\left(\frac{\hat{\Phi}}{4}\right) \\ &\quad \times E\left[\frac{1}{|S((L \wedge k)+1)|} \;\middle|\; S(L \wedge k)\right] \\ &\le \exp\left(\frac{\hat{\Phi}}{4}(L \wedge k)\right)\exp\left(\frac{\hat{\Phi}}{4}\right)\exp\left(-\frac{\hat{\Phi}}{4}\right)\frac{1}{|S(L \wedge k)|} \\ &= Z(L \wedge k). \end{aligned}$$

Therefore, $Z(L \wedge k)$ is a supermartingale.

Since $Z(L \wedge k)$ is a supermartingale, we have the inequality $E[Z(L \wedge k)] \le E[Z(L \wedge 0)] = 1$ for any $k > 0$, as $Z(L \wedge 0) = Z(0) = 1$. The fact that the set $S(k)$ can contain at most the n nodes in the graph implies that

$$\begin{aligned} Z(L \wedge k) &= \frac{\exp\left(\frac{\hat{\Phi}}{4}(L \wedge k)\right)}{|S(L \wedge k)|} \\ &\ge \frac{1}{n}\exp\left(\frac{\hat{\Phi}}{4}(L \wedge k)\right). \end{aligned} \quad (11)$$

Taking expectations on both sides of (11) yields

$$\begin{aligned} E\left[\exp\left(\frac{\hat{\Phi}}{4}(L \wedge k)\right)\right] &\le nE[Z(L \wedge k)] \\ &\le n. \end{aligned}$$

Because $\exp(\hat{\Phi}(L \wedge k)/4) \uparrow \exp(\hat{\Phi}L/4)$ as $k \to \infty$, the monotone convergence theorem implies that

$$E\left[\exp\left(\frac{\hat{\Phi}L}{4}\right)\right] \le n.$$

Applying Markov's inequality, we obtain that, for $k_1 = 4(\ln 2 + 2\ln n + \ln \delta^{-1})/\hat{\Phi}$,

$$\begin{aligned} \Pr(L > k_1) &= \Pr\left(\exp\left(\frac{\hat{\Phi}L}{4}\right) > \frac{2n^2}{\delta}\right) \\ &< \frac{\delta}{2n}. \end{aligned}$$

For the second phase of the algorithm, when $|S(k)| > n/2$, we study the evolution of the size of the set of nodes that do not have the message, $|S(k)^c|$. This quantity will decrease as the message spreads from nodes in $S(k)$ to nodes in $S(k)^c$. For simplicity, let us consider restarting the process from clock tick 0 after L (i.e., when more than half the nodes in the graph have the message), so that we have $|S(0)^c| \le n/2$.

In clock tick $k+1$, a node $j \in S(k)^c$ will receive the message if it contacts a node $i \in S(k)$ and pulls the message from i. As such,

$$E[|S(k)^c| - |S(k+1)^c| \mid S(k)^c] \ge \sum_{j \in S(k)^c, i \notin S(k)^c} \frac{P_{ji}}{n}.$$

Thus, we have

$$\begin{aligned} &E[|S(k+1)^c| \mid S(k)^c] \\ &\quad \le |S(k)^c| - \frac{\sum_{j \in S(k)^c, i \notin S(k)^c} P_{ji}}{n} \\ &\quad = |S(k)^c|\left(1 - \frac{\sum_{j \in S(k)^c, i \notin S(k)^c} P_{ji}}{n|S(k)^c|}\right) \\ &\quad \le |S(k)^c|\left(1 - \hat{\Phi}\right). \end{aligned} \quad (12)$$

We note that this inequality holds even when $|S(k)^c| = 0$, and as a result it is valid for all clock ticks k in the second phase. Repeated application of (12) yields

$$\begin{aligned} E[|S(k)^c|] &= E[E[|S(k)^c| \mid S(k-1)^c]] \\ &\le \left(1 - \hat{\Phi}\right)E[|S(k-1)^c|] \\ &\le \left(1 - \hat{\Phi}\right)^k E[|S(0)^c|] \\ &\le \exp\left(-\hat{\Phi}k\right)\left(\frac{n}{2}\right). \end{aligned}$$

For $k_2 = \ln(n^2/\delta)/\hat{\Phi} = (2\ln n + \ln \delta^{-1})/\hat{\Phi}$, we have $E[|S(k_2)^c|] \le \delta/(2n)$. Markov's inequality now implies the following upper bound on the probability that not all of the nodes have the message at the end of clock tick k_2 in the second phase.

$$\begin{aligned} \Pr(|S(k_2)^c| > 0) &= \Pr(|S(k_2)^c| \ge 1) \\ &\le E[|S(k_2)^c|] \\ &\le \frac{\delta}{2n} \end{aligned}$$

Combining the analysis of the two phases, we obtain that, for $k' = k_1 + k_2$, $\Pr(S_v(k') \ne V) \le \delta/n$. By applying the union bound over all the nodes in the graph, using the fact that $n \ge 2$, and recalling that $\hat{\Phi} = \Phi(P)/n$, we conclude that

$$\begin{aligned} &K(\delta) \le k' \\ &\quad = \frac{4\left(\ln 2 + 2\ln n + \ln \delta^{-1}\right) + \left(2\ln n + \ln \delta^{-1}\right)}{\hat{\Phi}} \\ &\quad \le n\left(\frac{14\ln n + 5\ln \delta^{-1}}{\Phi(P)}\right). \end{aligned}$$

This completes the proof of Lemma 4. ■

To extend the bound in Lemma 4 to absolute time, observe that Corollary 1 implies that the probability that $\kappa = K(\delta/3) + 27\ln(3/\delta)$ clock ticks do not occur in absolute time $(4/3)\kappa/n$

is at most $2\delta/3$. Applying the union bound now yields $T^{\text{spr}}_{SPREAD(P)}(\delta) \leq (4/3)\kappa/n \leq 62(\ln n + \ln \delta^{-1})/\Phi(P)$, where the last inequality follows from the inequalities $\Phi(P) \leq 1$ and $n \geq 3$. This establishes the upper bound in Theorem 2 for the asynchronous time model.

B. Synchronous model

In the synchronous time model, in each time slot every node contacts a neighbor to exchange messages. Thus, n communication events may occur simultaneously. Recall that absolute time is measured in rounds or time slots in the synchronous model.

The analysis of the randomized gossip algorithm for information spreading in the synchronous model is similar to the analysis for the asynchronous model. However, we need additional analytical arguments to reach analogous conclusions due to the technical challenges presented by multiple simultaneous transmissions.

In this section, we sketch a proof of the time bound in Theorem 2, $T^{\text{spr}}_{SPREAD(P)}(\delta) \leq 62(\ln n + \ln \delta^{-1})/\Phi(P)$, for the synchronous time model. Since the proof follows a similar structure as the proof of Lemma 4, we only point out the significant differences.

As before, we fix a node $v \in V$, and study the evolution of the size of the set $S(t) = S_v(t)$. Again, we divide the execution of the algorithm into two phases based on the evolution of $S(t)$: in the first phase $|S(t)| \leq n/2$, and in the second phase $|S(t)| > n/2$. In the first phase, we analyze the increase in $|S(t)|$, while in the second we study the decrease in $|S(t)^c|$. For the purpose of analysis, in the first phase we ignore the effect of the increase in $|S(t)|$ due to the *pull* aspect of protocol: that is, when node i contacts node j, we assume (for the purpose of analysis) that i sends the messages it has to j, but that j does not send any messages to i. Clearly, an upper bound obtained on the time required for every node to receive every message under this restriction is also an upper bound for the actual algorithm.

Consider a time slot $t+1$ in the first phase. For $j \notin S(t)$, let X_j be an indicator random variable that is 1 if node j receives the message m_v via a push from some node $i \in S(t)$ in time slot $t+1$, and is 0 otherwise. The probability that j does not receive m_v via a push is the probability that no node $i \in S(t)$ contacts j, and so

$$\begin{aligned} E[X_j \mid S(t)] &= 1 - \Pr(X_j = 0 \mid S(t)) \\ &= 1 - \prod_{i \in S(t)} (1 - P_{ij}) \\ &\geq 1 - \prod_{i \in S(t)} \exp(-P_{ij}) \\ &= 1 - \exp\left(-\sum_{i \in S(t)} P_{ij}\right). \end{aligned} \tag{13}$$

The Taylor series expansion of $\exp(-z)$ about $z = 0$ implies that, if $0 \leq z \leq 1$, then

$$\exp(-z) \leq 1 - z + z^2/2 \leq 1 - z + z/2 = 1 - z/2. \tag{14}$$

For a doubly stochastic matrix P, we have $0 \leq \sum_{i \in S(t)} P_{ij} \leq 1$, and so we can combine (13) and (14) to obtain

$$E[X_j \mid S(t)] \geq \frac{1}{2} \sum_{i \in S(t)} P_{ij}.$$

By linearity of expectation,

$$\begin{aligned} E[|S(t+1)| - |S(t)| \mid S(t)] &= \sum_{j \notin S(t)} E[X_j \mid S(t)] \\ &\geq \frac{1}{2} \sum_{i \in S(t), j \notin S(t)} P_{ij} \\ &= \frac{|S(t)|}{2} \frac{\sum_{i \in S(t), j \notin S(t)} P_{ij}}{|S(t)|}. \end{aligned}$$

When $|S(t)| \leq n/2$, we have

$$E[|S(t+1)| - |S(t)| \mid S(t)] \geq |S(t)| \frac{\Phi(P)}{2}. \tag{15}$$

Inequality (15) is analogous to inequality (8) for the asynchronous time model, with $\Phi(P)/2$ in the place of $\hat{\Phi}$. We now proceed as in the proof of Lemma 4 for the asynchronous model. Note that $|S(t+1)| \leq 2|S(t)|$ here in the synchronous model because of the restriction in the analysis to only consider the push aspect of the protocol in the first phase, as each node in $S(t)$ can push a message to at most one other node in a single time slot. Repeating the analysis from the asynchronous model leads to the conclusion that the first phase of the algorithm ends in at most $8(\ln 2 + 2\ln n + \ln \delta^{-1})/\Phi(P)$ rounds with probability at least $1 - \delta/2n$.

The analysis of the second phase is the same as that presented for the asynchronous time model, with $\hat{\Phi}$ replaced by Φ, and thus the second phase requires at most $(2\ln n + \ln \delta^{-1})/\Phi(P)$ rounds with probability at least $1 - \delta/2n$. Combining these two bounds, we conclude that it takes at most $26(\ln n + \ln \delta^{-1})/\Phi(P)$ rounds for the algorithm to spread all the messages to all the nodes with probability at least $1 - \delta$. The constant here is smaller than the corresponding one for the asynchronous model because absolute time is measured in rounds in the synchronous model, and as a consequence there is no need here to convert between clock ticks and absolute time as in the asynchronous model. This completes the proof of Theorem 2 for the synchronous time model.

V. Applications

We study here the application of our preceding results to several types of graphs. In particular, we consider complete graphs, constant-degree expander graphs, and grid graphs. We use grid graphs as an example to compare the performance of our algorithm for computing separable functions with that of a known iterative averaging algorithm.

For each class of graphs, we are interested in the δ-information-spreading time $T^{\text{spr}}_{SPREAD(P)}(\delta)$, where P is a doubly stochastic matrix that assigns equal probability to each of the neighbors of any node. Specifically, the probability P_{ij} that a node i contacts a node $j \neq i$ when i becomes active is $1/\Delta$, where Δ is the maximum degree of the graph, and $P_{ii} = 1 - d_i/\Delta$, where d_i is the degree of i. Recall from Theorem 1 that the information dissemination algorithm

SPREAD(P) can be used as a subroutine in an algorithm for computing separable functions, with the running time of the resulting algorithm being a function of $T^{\mathrm{spr}}_{SPREAD(P)}(\delta)$. We consider how this running time scales with the number of nodes n for different graphs in each class.

A. Complete graph

On a complete graph, the communication matrix P has $P_{ii} = 0$ for $i = 1, \ldots, n$, and $P_{ij} = 1/(n-1)$ for $j \neq i$. This regular structure allows us to directly evaluate the conductance of P, which is $\Phi(P) \approx 1/2$. This implies that the (ε, δ)-computing time of the algorithm for computing separable functions based on SPREAD(P) is, up to constant factors, at most $\varepsilon^{-2}(1 + \ln \delta^{-1})(\ln n + \ln \delta^{-1})$. Thus, for a constant $\varepsilon \in (0,1)$ and $\delta = 1/n$, the computation time scales as $O(\log^2 n)$ as n increases.

B. Expander graph

Expander graphs have been used for numerous applications, and explicit constructions are known for constant-degree expanders [24]. We consider here undirected graphs in which the maximum degree of any vertex, Δ, is a constant. For a set of vertices $S \subseteq V$ in a graph $G = (V, E)$, let $F(S, S^c)$ be the set of edges with one endpoint in S and the other endpoint in S^c. The edge expansion of the graph is denoted by $\alpha(G)$ and defined as

$$\alpha(G) = \min_{S \subset V,\ 0 < |S| \leq n/2} \frac{|F(S, S^c)|}{|S|}.$$

In a family of expander graphs of various different sizes, $G_1, G_2, \ldots$, the edge expansion is bounded from below by $\alpha(G_\ell) \geq \alpha$ for each graph G_ℓ, where α is a positive constant. For a graph in such a family, the communication matrix P satisfies $P_{ij} = 1/\Delta$ for all $i \neq j$ such that $(i,j) \in E$, from which we obtain $\Phi(P) \geq \alpha/\Delta$. When α and Δ are constants, this leads to a similar conclusion as in the case of the complete graph: for any constant $\varepsilon \in (0,1)$ and $\delta = 1/n$, the computation time is $O(\log^2 n)$.

C. Grid

We now consider a d-dimensional grid graph on n nodes, where $c = n^{1/d}$ is an integer. Each node in the grid can be represented as a d-dimensional vector $a = (a_i)$, where $a_i \in \{1, \ldots, c\}$ for $1 \leq i \leq d$. There is one node for each distinct vector of this type, and so the total number of nodes in the graph is $c^d = (n^{1/d})^d = n$. For any two nodes a and b, there is an edge (a, b) in the graph if and only if, for some $i \in \{1, \ldots, d\}$, $|a_i - b_i| = 1$, and $a_j = b_j$ for all $j \neq i$.

In [25], it is shown that the edge expansion of this grid graph is

$$\min_{S \subset V,\ 0 < |S| \leq n/2} \frac{|F(S, S^c)|}{|S|} = \Theta\left(\frac{1}{c}\right) = \Theta\left(\frac{1}{n^{1/d}}\right).$$

By the definition of the edge set, the maximum degree of a node in the graph is $2d$. This means that $P_{ij} = 1/(2d)$ for all $i \neq j$ such that $(i,j) \in E$, and it follows that $\Phi(P) = \Omega\left(\frac{1}{dn^{1/d}}\right)$. Hence, for any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$, the (ε, δ)-computing time of the algorithm for computing separable functions is $O(\varepsilon^{-2}(1 + \log \delta^{-1})(\log n + \log \delta^{-1}) dn^{1/d})$.

D. Comparison with Iterative Averaging

We briefly contrast the performance of our algorithm for computing separable functions with that of the iterative averaging algorithms in [2], [6]. As noted earlier, the dependence of the performance of our algorithm on the communication matrix P is in proportion to $1/\Phi(P)$, which is a lower bound for the iterative algorithms based on a stochastic matrix P.

In particular, when our algorithm is used to compute the average of a set of numbers (by estimating the sum of the numbers and the number of nodes in the graph) on a d-dimensional grid graph, it follows from the analysis in Section V-C that the amount of time required to ensure the estimate is within a $(1 \pm \varepsilon)$ factor of the average with probability at least $1 - \delta$ is, up to constant factors, at most $\varepsilon^{-2}(1 + \ln \delta^{-1})(\ln n + \ln \delta^{-1}) dn^{1/d}$ for any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$. So, for a constant $\varepsilon \in (0,1)$ and $\delta = 1/n$, the computation time scales as $O(dn^{1/d} \log^2 n)$ with the size of the graph, n. The algorithm in [2] requires $\Omega(n^{2/d} \log n)$ time for this computation. Hence, the running time of our algorithm is (for fixed d, and up to logarithmic factors) the *square root* of the runnning time of the iterative algorithm! This relationship holds on other graphs for which the spectral gap is proportional to the square of the conductance.

VI. Conclusions and Future Work

In this paper, we presented a novel algorithm for computing separable functions in a fully distributed manner. The algorithm is based on properties of exponential random variables, and the fact that the minimum of a collection of numbers is an order- and duplicate-insensitive statistic.

Operationally, our algorithm makes use of an information spreading mechanism as a subroutine. This led us to the analysis of a randomized gossip mechanism for information spreading. We obtained an upper bound on the information spreading time of this algorithm in terms of the conductance of a matrix that characterizes the algorithm.

In addition to computing separable functions, our algorithm improves the computation time for the canonical task of averaging. For example, on graphs such as paths, rings, and grids, the performance of our algorithm is of a smaller order than that of a known iterative algorithm.

We believe that our algorithm will lead to the following fully distributed computations: (1) an approximation algorithm for convex minimization with linear constraints; and (2) a "packet marking" mechanism in the Internet. These areas, in which summation is a key subroutine, will be topics of our future research.

Acknowledgment

We thank Ashish Goel for a useful discussion and providing suggestions, based on previous work [26], when we started this work.

References

[1] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 250–262.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *Proceedings of IEEE INFOCOM 2005*, 2005, pp. 1653–1664.

[3] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, 1985.

[4] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *Proceedings of RANDOM 2002*, 2002, pp. 1–10.

[5] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proceedings of the 20th International Conference on Data Engineering*, 2004, pp. 449–460.

[6] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1984.

[7] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

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

[9] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 482–491.

[10] A. M. Frieze and G. R. Grimmett, "The shortest-path problem for graphs with random arc-lengths," *Discrete Applied Mathematics*, vol. 10, pp. 57–77, 1985.

[11] B. Pittel, "On spreading a rumor," *SIAM Journal of Applied Mathematics*, vol. 47, no. 1, pp. 213–223, 1987.

[12] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, 1987, pp. 1–12.

[13] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking, "Randomized rumor spreading," in *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000, pp. 565–574.

[14] R. Ravi, "Rapid rumor ramification: Approximating the minimum broadcast time," in *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, 1994, pp. 202–213.

[15] D. Kempe and J. Kleinberg, "Protocols and impossibility results for gossip-based communication mechanisms," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 471–480.

[16] D. Kempe, J. Kleinberg, and A. Demers, "Spatial gossip and resource location protocols," in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 2001, pp. 163–172.

[17] A. Ganesh, L. Massoulié, and D. Towsley, "The effect of network topology on the spread of epidemics," in *Proceedings of IEEE INFOCOM 2005*, 2005, pp. 1455–1466.

[18] N. Berger, C. Borgs, J. T. Chayes, and A. Saberi, "On the spread of viruses on the internet," in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 301–310.

[19] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analaysis," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, 2004, pp. 561–568.

[20] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," in *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, 2006, pp. 27–38.

[21] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*, 2nd ed. Springer, 1998.

[22] E. Cohen, "Size-estimation framework with applications to transitive closure and reachability," *Journal of Computer and System Sciences*, vol. 55, no. 3, pp. 441–453, 1997.

[23] A. Sinclair, *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Boston: Birkhäuser, 1993.

[24] O. Reingold, S. Vadhan, and A. Wigderson, "Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors," in *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000, pp. 3–13.

[25] M. C. Azizoğlu and Ö. Eğecioğlu, "The isoperimetric number of d-dimensional k-ary arrays," *International Journal of Foundations of Computer Science*, vol. 10, no. 3, pp. 289–300, 1999.

[26] M. Enachescu, A. Goel, R. Govindan, and R. Motwani, "Scale free aggregation in sensor networks," in *International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

Damon Mosk-Aoyama received the S.B. degree in computer science and engineering and the M.Eng. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2002. He is currently working toward the Ph.D. degree in the Department of Computer Science at Stanford University, Stanford, CA. His research interests are in theoretical computer science, primarily in algorithms.

Devavrat Shah received the B.Tech. in computer science and engineering from IIT-Bombay, India, and the Ph.D. in computer science from Stanford University. He is currently an assistant professor with the Department of EECS, MIT, where he has been since Fall 2005. He was co-awarded the IEEE INFOCOM best paper award in 2004 and the ACM SIGMETRICS/Performance best paper award in 2006. He received the 2005 George B. Dantzig best disseration award from the INFORMS. He received the NSF CAREER award in 2006. His research interests include network algorithms, stochastic networks, network information theory, and statistical inference.