Distributed Consensus and Linear Functional Calculation in Networks: An Observability Perspective

Shreyas Sundaram
Coordinated Science Laboratory and
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, 61801
ssundarm@uiuc.edu

Christoforos N. Hadjicostis*
Coordinated Science Laboratory and
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, 61801
chadjic@uiuc.edu

ABSTRACT

We study the problem of performing sensor fusion and distributed consensus in networks, where the objective is to calculate some linear function of the initial sensor values at some or all of the sensors. We utilize a linear iteration where, at each time-step, each sensor updates its value to be a weighted average of its own previous value and those of its neighbors. We show that this approach can be viewed as a linear dynamical system, with dynamics that are given by the weight matrix for the linear iteration, and with outputs for each sensor that are captured by the subset of the state vector that is measurable by that sensor. We then cast the fusion and consensus problems as that of observing a linear functional of the initial state vector using only local measurements (that are available at each sensor). When the topology of the network is time-invariant, we show that the weight matrix can be chosen so that each sensor can calculate the desired function as a linear combination of its measurements over a finite number of time-steps.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms, Analysis of algorithms*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*Linear systems (direct and iterative methods)*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on matrices*

General Terms

Algorithms, Theory

Keywords

Distributed consensus, distributed fusion, networked control, in-network processing and aggregation

*Corresponding author.

IPSN'07, April 25-27, 2007, Cambridge, Massachusetts, USA.

1. INTRODUCTION

In distributed systems and networks, it is often necessary for some or all of the nodes to calculate some function of certain parameters. For example, the nodes might represent sensors that take measurements of their environment, and the average of the measurements must be calculated at a sink node or fusion center [6]. Another example is the case of multi-agent systems, where all agents communicate with each other to coordinate their speed and direction [14]. When all nodes calculate the same function of the initial values in the system, they are said to reach consensus. Such problems have received extensive attention in the computer science literature over the past few decades [10], leading to the development of various protocols. For instance, one simple method is to have each node share all of its knowledge about the other nodes' values with its neighbors. After repeating this process a sufficient number of times, all nodes will know all of the other nodes' values, and can therefore calculate any desired function of these values in order to obtain consensus (e.g., average of all the values, maximum of all the values, etc.). This process is known as *flooding* [10]. Such protocols have the benefit of being robust to node and link failures, but potentially require a large amount of data storage and communication.

Another approach to consensus proposed by the research community is to update the value of each node as a weighted linear combination of the values of itself and its neighbors. This strategy can be modeled via the linear iteration

$$x[k+1] = W[k]x[k] \ ,$$

where $x[k]$ is a column vector containing each of the node values at time-step k, and $W[k]$ is a weight matrix adhering to the underlying (possibly time-varying) graph topology for the distributed system (i.e., the (i, j) entry of $W[k]$ is zero if node j is not connected to node i at time-step k) [14, 18, 16, 2, 17, 13, 11, 8, 9]. The time-varying nature of the graph may arise, for example, from changes in the formation of mobile agents, or due to failures in the data links between nodes. Loosely speaking, the above works revealed that if the graph is connected in the long run, the weight matrices $W[k]$ can be chosen so that all of the nodes asymptotically converge to the same value.

One of the benefits of using linear iteration-based consensus schemes is that each node only has to store and transmit a single value to each of its neighbors at each time-step. Recently, schemes based on linear iterations have been applied to the sensor fusion and distributed consensus problems in

sensor networks [18, 12, 5]. However, almost all of the linear iteration schemes currently present in the literature only produce asymptotic convergence (i.e., exact consensus is not reached in a finite number of steps). In graphs with time-invariant topologies, Xiao and Boyd showed in [17] that the speed of convergence is determined by the second largest eigenvalue of the appropriately chosen weight matrix W. Furthermore, they showed that the optimal weight matrix (i.e., the matrix that maximizes the rate of convergence) can be obtained by solving a convex optimization problem. Nevertheless, even the optimal weight matrix will not produce finite-time convergence.

In this paper, we study the general problem of calculating linear functionals in networks, and treat the topic of distributed consensus as a special case of this problem. Our approach to the problem will be from the perspective of observability theory. Specifically, in graphs with time-invariant topologies, we show that the linear iteration-based scheme can be viewed as a dynamical system, where each node has access to some subset of the system state. We show that with an appropriate choice of the weight matrix, each node can immediately calculate the desired linear functional of the initial state after observing the evolution of its own value and those of its neighbors over a finite number of time-steps (specifically, upper bounded by the size of the network). The topic of finite-time convergence was briefly discussed by Kingston and Beard in [9]. However, the method described in that paper requires the graph to be fully-connected for at least one time-step, which is a very strict condition. In contrast, our method applies to graphs with arbitrary (connected) topologies. Finite-time consensus was also studied for continuous-time systems in [4]. The approach in that paper was to have the nodes evolve according to the gradient of a suitably defined function of their values. The resulting protocol, which is nonlinear, does not directly translate to discrete-time systems, and the paper only considered a restricted class of possible consensus values. In contrast, our method achieves finite-time consensus in discrete-time systems by running a simple linear iteration, and allows the consensus value to be a broad range of linear functions.

The rest of the paper is organized as follows. In Section 2, we provide a more detailed background on linear iteration-based consensus schemes. In Section 3, we review some concepts from observability theory that will be useful in solving the problem of linear functional calculation in networks. We then use these concepts in Section 4 to design weight matrices that allow the desired functions to be calculated at certain nodes. In Section 5, we show how each node can obtain the necessary information about the network in order to calculate the linear functional. We finish with our conclusions in Section 6, and suggest directions for future research.

In our development, we will use the notation e_i to indicate the column vector with a 1 in its i'th position and zeros elsewhere. The symbol $\mathbf{1}$ represents the column vector (of appropriate size) that has all entries equal to one. We will say that an eigenvalue of a matrix W is *simple* to indicate that it is *algebraically simple* (i.e., it appears only once in the spectrum of W) [7]. The notation A' indicates the transpose of matrix A. We will denote the rank of matrix A by $\rho(A)$.

2. BACKGROUND

The interaction constraints in distributed systems and networks can be conveniently modeled via a directed graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is the set of nodes in the system and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of directed edges (i.e., directed edge $(x_i, x_j) \in \mathcal{E}$ if node x_i can receive information from node x_j). Note that undirected graphs can be readily handled by treating each undirected edge as two directed edges. All nodes that can transmit information to node x_i are said to be neighbors of node i, and are represented by the set $\mathcal{N}_i$. The number of neighbors of node i is called the in-degree of node i, and is denoted as $\deg_i$. Suppose that each node i has some initial value, given by $x_i[0]$. At each time-step k, all nodes update their values based on some strategy. For ease of analysis, the values of all nodes at time-step k will be aggregated into the value vector

$$x[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_N[k] \end{bmatrix}' .$$

The scheme that we study in this paper makes use of linear iterations; specifically, at each time-step, each node updates its value as

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{j \in \mathcal{N}_i} w_{ij}x_j[k]$$

where the w_{ij}'s are a set of weights. In other words, each node updates its value to be a linear combination of its previous value and the values of its neighbors. The update strategy for the entire system can then be represented as

$$x[k+1] = \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{W} x[k] \qquad (1)$$

for $k = 0, 1, \ldots$, where $w_{ij} = 0$ if $j \notin \mathcal{N}_i$ (i.e., if $(x_i, x_j) \notin \mathcal{E}$).

Definition 1 (Calculable Functional). *Let Q_i be a $\beta_i \times N$ real-valued matrix. If node i can obtain the (vector) value $Q_i x[0]$ after following some strategy for a sufficiently large number of time-steps, we say the linear functional $Q_i x[0]$ is calculable at node i.*

Definition 2 (Distributed Consensus). *Let $\mathbf{c}$ be a $N \times 1$ real-valued column vector. The system is said to achieve distributed consensus if all nodes in the system calculate the function $\mathbf{c}'x[0]$ after a sufficiently large number of time-steps. When $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$, the system is said to perform distributed averaging (i.e., the consensus value is the average of all initial node values). The system is said to reach asymptotic consensus if $\lim_{k\to\infty} x_i[k] = \mathbf{c}'x[0]$ for each i, or equivalently, $\lim_{k\to\infty} x[k] = \mathbf{1}\mathbf{c}'x[0]$.*

The following result by Xiao and Boyd from [17] characterizes the conditions under which the iteration (1) achieves asymptotic consensus.

Theorem 1 ([17]). *The iteration given by (1) achieves asymptotic consensus (under the technical condition that $\mathbf{c}$ is normalized so that $\mathbf{c}'\mathbf{1} = 1$) if and only if the weight matrix W satisfies the following conditions:*

1. *W has a simple eigenvalue at 1, and all other eigenvalues have magnitude strictly less than one.*

2. *The left and right eigenvectors of W corresponding to eigenvalue 1 are $\mathbf{c}'$ and $\mathbf{1}$, respectively:*

$$\mathbf{c}'W = \mathbf{c}', \;\; W\mathbf{1} = \mathbf{1} \;\; .$$

Furthermore, Xiao and Boyd showed that the rate of convergence is determined by the eigenvalue with second-largest magnitude. Consequently, to maximize the rate of convergence, they formulated a convex optimization problem to obtain the weight matrix satisfying the above conditions while minimizing the magnitude of the second largest eigenvalue.

As mentioned in the previous section, a salient feature of the analysis in [17], and of other linear iterations studied in the literature (e.g., see [14] and the references therein), is that the focus is on asymptotic convergence. Clearly, distributed consensus requires that the function $\mathbf{c}'x[0]$ be calculable at all nodes, and in the case of asymptotic consensus, the nodes take an infinite number of time-steps in order to calculate this function. In the next section, we show how nodes can perform function calculation in a finite number of time-steps by using techniques from observability theory. Furthermore, this approach allows us to obtain distributed consensus with a much larger class of weight matrices than permitted by Theorem 1. By focusing on the calculation of a linear functional rather than the more specific problem of distributed consensus, our results can treat a wide range of problems, including those where only a subset of the nodes have to calculate a given function.

3. OBSERVABILITY OF LINEAR FUNCTIONALS OF THE INITIAL STATE

Consider the linear iteration given by (1). At each time-step, each node i has access to its own value and those of its neighbors. The linear iteration can then be viewed as the dynamical system

$$\begin{aligned} x[k+1] &= Wx[k] \\ y_i[k] &= E_i x[k], \;\; 1 \leq i \leq N \;\; , \end{aligned} \tag{2}$$

where $y_i[k]$ denotes the outputs that are available to node i during the k'th time-step. Here, E_i is the $(\deg_i + 1) \times N$ matrix with a single 1 in each row denoting the positions of the state-vector $x[k]$ that are available to node i (i.e., these positions correspond to nodes that are neighbors of node i, along with node i itself). Since $x[k] = W^k x[0]$, we have $y_i[k] = E_i W^k x[0]$, and the set of all outputs seen by node i over $L+1$ time-steps is given by

$$\begin{bmatrix} y_i[0] \\ y_i[1] \\ y_i[2] \\ \vdots \\ y_i[L] \end{bmatrix} = \underbrace{\begin{bmatrix} E_i \\ E_i W \\ E_i W^2 \\ \vdots \\ E_i W^L \end{bmatrix}}_{\mathcal{O}_{i,L}} x[0] \;\; . \tag{3}$$

When $L = N-1$, the matrix $\mathcal{O}_{i,L}$ in the above equation is known as the observability matrix for the pair (W, E_i) [3]. The row-space of $\mathcal{O}_{i,L}$ characterizes the set of all calculable linear functionals for node i up to time-step L. We will now summarize some properties of this matrix, and show how it allows node i to calculate the desired linear functional of the initial state.

First, note that the rank of $\mathcal{O}_{i,L}$ is a nondecreasing function of L, and bounded above by N. Suppose ν_i is the first integer for which $\rho(\mathcal{O}_{i,\nu_i}) = \rho(\mathcal{O}_{i,\nu_i-1})$. This implies that there exists a matrix $\mathcal{K}_i$ such that

$$E_i W^{\nu_i} = \mathcal{K}_i \begin{bmatrix} E_i \\ E_i W \\ \vdots \\ E_i W^{\nu_i - 1} \end{bmatrix} \;\; .$$

In other words, the matrix $E_i W^{\nu_i}$ can be written as a linear combination of the rows in $\mathcal{O}_{i,\nu_i-1}$. In turn, this implies that

$$E_i W^{\nu_i+1} = E_i W^{\nu_i} W = \mathcal{K}_i \mathcal{O}_{i,\nu_i-1} W = \mathcal{K}_i \begin{bmatrix} E_i W \\ E_i W^2 \\ \vdots \\ E_i W^{\nu_i} \end{bmatrix} \;\; ,$$

and so the matrix $E_i W^{\nu_i+1}$ can be written as a linear combination of rows in $\mathcal{O}_{i,\nu_i}$. Continuing in this way, it is seen that the rank of $\mathcal{O}_{i,L}$ monotonically increases with L until $L = \nu_i - 1$, at which point it stops increasing, i.e.,

$$\begin{aligned} \rho(\mathcal{O}_{i,0}) < \rho(\mathcal{O}_{i,1}) < \cdots < \rho(\mathcal{O}_{i,\nu_i-1}) = \\ \rho(\mathcal{O}_{i,\nu_i}) = \rho(\mathcal{O}_{i,\nu_i+1}) = \cdots \;\; . \end{aligned}$$

This means that the outputs of the system

$$y_i[0], y_i[1], \ldots, y_i[\nu_i - 1]$$

contain the maximum amount of information that is possible to obtain about the initial state, and future outputs of the system do not provide any extra information. The integer ν_i is called the *observability index* of the pair (W, E_i), and can be upper bounded as

$$\nu_i \leq \min(D, N - \rho(E_i) + 1) \;\; ,$$

where D is the degree of the minimal polynomial of W [3]. Since E_i has rank $\deg_i + 1$, the above bound becomes

$$\nu_i \leq \min(D, N - \deg_i) \;\; . \tag{4}$$

Thus, if it is possible for node i to calculate the desired function $\mathbf{c}'x[0]$, it can do so in at most $N - \deg_i$ time-steps.

Suppose that $\rho(\mathcal{O}_{i,\nu_i-1}) = N_i$. If $N_i = N$, then node i can uniquely determine the entire initial state $x[0]$ from the outputs of the system $y_i[0], y_i[1], \ldots, y_i[\nu_i - 1]$. In this case, the pair (W, E_i) is said to be *observable*. However, if $N_i < N$, then the matrix $\mathcal{O}_{i,\nu_i-1}$ will have a nullspace of dimension greater than zero. Any initial state $x[0]$ in the nullspace of $\mathcal{O}_{i,\nu_i-1}$ will produce an output $y_i[k] = 0$ for all k, and will be indistinguishable from the all-zero initial state $x[0] = 0$. Such initial states are said to be *unobservable*, and the nullspace of the observability matrix is called the *unobservable subspace*.

Based on the above discussion, we see that node i can calculate the linear functional $\mathbf{c}'x[0]$ if and only if the vector $\mathbf{c}'$ is in the row-space of the matrix $\mathcal{O}_{i,\nu_i-1}$. Therefore, if the objective in the system is for some subset of the nodes to calculate the linear functional $\mathbf{c}'x[0]$, one has to choose the weight matrix W so that $\mathbf{c}'$ is in the row-space of the observability matrices for all those nodes; we will later describe a systematic method to achieve this. We will first

require a deeper characterization of the row-spaces of the observability matrices in terms of the eigenvalues of W.

Suppose that the distinct eigenvalues of W are given by $\lambda_1, \lambda_2, \ldots, \lambda_m$, where $1 \leq m \leq N$. Eigenvalue λ_j is said to be an observable eigenvalue of the pair (W, E_i) if

$$\rho\left(\begin{bmatrix}\lambda_j I - W \\ E_i\end{bmatrix}\right) = N \ ,$$

and it is called an unobservable eigenvalue otherwise [1]. An alternative characterization of the observability of eigenvalues can be obtained from the Jordan form of the pair (W, E_i). We summarize this characterization here; a more detailed treatment can be found in [3].

Let the Jordan normal form of the pair (W, E_i) be given by

$$T^{-1}WT = \begin{bmatrix} J_1 & 0 & \cdots & 0 \\ 0 & J_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_m \end{bmatrix} \ , \qquad (5)$$

$$E_i T = \begin{bmatrix} E_{i,1} & E_{i,2} & \cdots & E_{i,m} \end{bmatrix} \ ,$$

where T is the matrix whose columns are the right eigenvectors of W (grouped by eigenvalue), each J_j is the set of all Jordan blocks corresponding to eigenvalue λ_j, and each $E_{i,j}$ is the set of columns in $E_i T$ corresponding to J_j [7]. Decompose each of the pairs $(J_j, E_{i,j})$ as

$$J_j = \begin{bmatrix} J_{j,1} & 0 & \cdots & 0 \\ 0 & J_{j,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_{j,r(j)} \end{bmatrix} \ ,$$

$$E_{i,j} = \begin{bmatrix} E_{i,j,1} & E_{i,j,2} & \cdots & E_{i,j,r(j)} \end{bmatrix} \ ,$$

where each $J_{j,l}$ is a Jordan block for eigenvalue λ_j, $r(j)$ is the number of Jordan blocks for eigenvalue λ_j, and each $E_{i,j,l}$ is the set of columns in $E_{i,j}$ corresponding to the Jordan block $J_{j,l}$. Let $\widehat{E}_{i,j}$ be the $(\deg_i + 1) \times r(j)$ matrix formed by taking the first column from each of the blocks $E_{i,j,1}, E_{i,j,2}, \ldots, E_{i,j,r(j)}$. Then λ_j is an observable eigenvalue of the pair (W, E_i) if and only if the columns of $\widehat{E}_{i,j}$ are linearly independent (i.e., $\widehat{E}_{i,j}$ must be full column rank) [3].

With the notion of observable eigenvalues in hand, we can propose the following theorem to obtain a deeper characterization of the row-spaces of the observability matrices. We will later use this theorem and the above test for observable eigenvalues to design the weight matrix so that a particular linear functional is calculable by all of the nodes.

THEOREM 2. *The row-space of $\mathcal{O}_{i,\nu_i - 1}$ contains all left eigenvectors of W corresponding to observable eigenvalues of the pair (W, E_i).*

PROOF. Let Q_i be a $N_i \times N$ matrix whose rows form a basis for the row-space of $\mathcal{O}_{i,\nu_i-1}$. Thus Q_i represents all linear functions of the state $x[0]$ that can be obtained from the output of the system (i.e., it completely characterizes all linear functionals that are calculable by node i). Define the matrix

$$P_i = \begin{bmatrix} Q_i \\ R_i \end{bmatrix} \ , \qquad (6)$$

where R_i is chosen to make P_i square and invertible. If we use P_i to perform a similarity transformation on the system (2) with output $y_i[k] = E_i x[k]$, we obtain the *observable canonical form*

$$\bar{W} \equiv P_i W P_i^{-1} = \begin{bmatrix} W_{i,o} & 0 \\ W_{i,21} & W_{i,\bar{o}} \end{bmatrix} \ , \qquad (7)$$

$$\bar{E}_i \equiv E_i P_i^{-1} = \begin{bmatrix} E_{i,o} & 0 \end{bmatrix} \ .$$

In this form, the pair $(W_{i,o}, E_{i,o})$ is observable, and in particular, all observable eigenvalues of the pair (W, E_i) are contained in the submatrix $W_{i,o}$ [3, 1].

Let $\bar{C}_{i,o}$ be the matrix whose rows contain all left eigenvectors of $W_{i,o}$ corresponding to the observable eigenvalues of the pair (W, E_i). Thus, we have

$$\bar{C}_{i,o} W_{i,o} = J_{i,o} \bar{C}_{i,o} \ ,$$

where $J_{i,o}$ is the set of all Jordan blocks corresponding to observable eigenvalues of the pair (W, E_i) [3, 7]. Now note that

$$\begin{bmatrix} \bar{C}_{i,o} & 0 \end{bmatrix} \begin{bmatrix} W_{i,o} & 0 \\ W_{i,21} & W_{i,\bar{o}} \end{bmatrix} = J_{i,o} \begin{bmatrix} \bar{C}_{i,o} & 0 \end{bmatrix} \ .$$

From the definition of $\bar{W}$ in (7), we have

$$\begin{bmatrix} \bar{C}_{i,o} & 0 \end{bmatrix} P_i W = J_{i,o} \begin{bmatrix} \bar{C}_{i,o} & 0 \end{bmatrix} P_i \ ,$$

and using (6), we get

$$\bar{C}_{i,o} Q_i W = J_{i,o} \bar{C}_{i,o} Q_i \ .$$

This implies that the rows of the matrix $\bar{C}_{i,o} Q_i$ are the left eigenvectors of W corresponding to observable eigenvalues of the pair (W, E_i). Since the rows of Q_i form a basis for the row-space of $\mathcal{O}_{i,\nu_i-1}$, we see that all left eigenvectors of the matrix W corresponding to observable eigenvalues of the pair (W, E_i) can be obtained as a linear combination of rows in $\mathcal{O}_{i,\nu_i-1}$, thereby completing the proof. □

Note that the above theorem only characterizes the eigenvectors of observable eigenvalues. It may be the case that some eigenvectors of unobservable eigenvalues also appear in the row-space of the observability matrix for node i. However, it is difficult to provide a general characterization of exactly which of these eigenvectors are included in the observability matrix, and so we focus on the eigenvectors of observable eigenvalues in the rest of the paper.

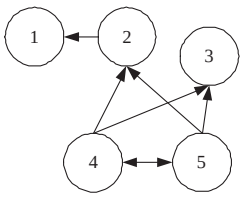

Figure 1: Directed graph with all edge weights equal to 1.

Example 1. Consider the graph shown in Fig. 1. The weights on all the edges and nodes are taken to be 1, which

produces the weight matrix

$$W = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} .$$

The matrix T of right eigenvectors and the Jordan form in (5) are given by

$$T = \left[\begin{array}{ccc|c|c} 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right] ,$$

$$T^{-1}WT = \left[\begin{array}{cc|c|c|c} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline\hline 0 & 0 & 0 & 2 & 0 \\ \hline\hline 0 & 0 & 0 & 0 & 0 \end{array}\right] .$$

Thus, W has distinct eigenvalues $\lambda_1 = 1$ with two Jordan blocks, $\lambda_2 = 2$ with one Jordan block, and $\lambda_3 = 0$ with one Jordan block. Suppose we wish to determine the set of observable eigenvalues for node 1. Since node 1 has access to its own value and the value from node 2, we have

$$E_1 T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} T = \left[\begin{array}{cc|c|c} 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 & 0 \end{array}\right] .$$

Consider eigenvalue $\lambda_1 = 1$. To test observability of this eigenvalue, we examine the columns from $E_1 T$ corresponding to the first column of each Jordan block for λ_1. These columns are used to form $\widehat{E}_{i,1} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, which is not full column rank; therefore, eigenvalue $\lambda_1 = 1$ is not observable from node 1. Now consider eigenvalue $\lambda_2 = 2$. Since there is only one Jordan block for λ_2, we only need to examine the fourth column in $E_i T$ to determine observability of this eigenvalue. This column is $\widehat{E}_{i,2} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, and since this matrix has full column rank, λ_2 is observable from node 1. Finally, we have $\widehat{E}_{i,3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, and since this is not full column rank, $\lambda_3 = 0$ is not observable from node 1. Since $\mathbf{c}' = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ is the left eigenvector of W corresponding to λ_2, we see from Theorem 2 that node 1 is guaranteed to be able to calculate the linear functional $\mathbf{c}'x[0]$.

In the following section, we apply the above concepts to design the weight matrix W to facilitate linear functional calculation in the network.

4. DESIGNING THE WEIGHT MATRIX

Suppose we wish to design the weight matrix so that some or all of the nodes can calculate the function $\mathbf{c}'x[0]$, for some vector $\mathbf{c}$. Based on our discussion in the previous section, we know that this function will be guaranteed to be calculable if $\mathbf{c}'$ is a left eigenvector of an observable eigenvalue for all nodes. The strategy will be to choose the weights so that this is the case.

As described in the previous section, the observability of eigenvalue j at node i can be determined by examining the appropriate entries of the right eigenvectors of W. In particular, if eigenvalue λ_j is simple with right eigenvector $\mathbf{d}$, then λ_j is observable if and only if at least one of the entries in $\mathbf{d}$ indexed by E_i is nonzero (so that the matrix $\widehat{E}_{i,j}$ is full column rank). This implies that for any weight matrix W that satisfies the conditions for asymptotic consensus given in Theorem 1, the eigenvalue 1 will be observable by all nodes (since all entries in the eigenvector $\mathbf{1}$ are nonzero), and the function $\mathbf{c}'x[0]$ will be calculable by all nodes. Therefore, one strategy for picking W is to choose the weights to satisfy the conditions in Theorem 1. For example, if the graph is undirected and if each node knows N (or an upper bound for N), it was shown in [18] that each node i can independently choose its weights as

$$w_{ij} = \begin{cases} \frac{1}{N}, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{if } j \notin \mathcal{N}_i \\ 1 - \sum_{j \in \mathcal{N}_i} w_{ij}, & \text{if } i = j. \end{cases} \quad (8)$$

The resulting weight matrix is symmetric and satisfies the conditions of Theorem 1 with $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$. Many other choices of weights exist that provide asymptotic consensus (e.g., see [14, 2, 8, 9]), and any of these weights can be used in conjunction with our approach.

However, since the observability approach allows the nodes to calculate the linear functional in a finite number of timesteps, we do not need to restrict attention to matrices that satisfy the conditions in Theorem 1. For instance, consider the weight matrix that is given by

$$w_{ij} = \begin{cases} 1, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{if } j \notin \mathcal{N}_i \\ S - \deg_i, & \text{if } i = j \end{cases} , \quad (9)$$

where S is some integer. These weights are obtained by multiplying the weights in (8) by N, and then offsetting each of the diagonal weights by a constant amount $S - N$. The corresponding weight matrix has a simple eigenvalue at $\mu = S$, with right eigenvector $\mathbf{d} = \mathbf{1}$ and left eigenvector $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$. This choice of weight matrix is appealing because all of its entries are integers, which could potentially simplify the implementation of the consensus protocol. This is a valid weight matrix in our scheme, but it is not allowed in asymptotic consensus protocols since it will typically have eigenvalues with magnitude larger than 1.

More generally, if the matrix W is nonnegative and irreducible (i.e., the underlying graph is strongly connected), then the largest eigenvalue will be simple, and the corresponding left and right eigenvectors will have all entries positive [7]. Suppose we want the weight matrix to have a left eigenvector $\mathbf{c}' = \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix}$ that is observable by all nodes, with $c_i \neq 0$ for all i. One way to ensure that the weight matrix has this property is to use the following procedure.

1. Choose any sets of weights so that the weight matrix W is nonnegative and irreducible. For example, choosing a weight of 1 for all edges, and zero for all self-weights, will cause W to be the adjacency matrix of the graph, which will be irreducible if the graph is strongly connected [7]. Another option is to choose one of the asymptotic consensus weights, such as the constant weights in (8). Suppose the largest eigen-

value in the chosen W matrix is μ, with left and right eigenvectors

$$\bar{\mathbf{c}} = \begin{bmatrix}\bar{c}_1 & \bar{c}_2 & \cdots & \bar{c}_N\end{bmatrix}'$$
$$\bar{\mathbf{d}} = \begin{bmatrix}\bar{d}_1 & \bar{d}_2 & \cdots & \bar{d}_N\end{bmatrix}' \ ,$$

respectively. As noted above, all entries in these vectors will be positive.

2. Perform the similarity transformation

$$\widehat{W} = \begin{bmatrix} \frac{\bar{c}_1}{c_1} & 0 & \cdots & 0 \\ 0 & \frac{\bar{c}_2}{c_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\bar{c}_N}{c_N} \end{bmatrix} W \begin{bmatrix} \frac{c_1}{\bar{c}_1} & 0 & \cdots & 0 \\ 0 & \frac{c_2}{\bar{c}_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{c_N}{\bar{c}_N} \end{bmatrix} \ .$$

The matrix $\widehat{W}$ has zeros in the same locations as W, and therefore represents the same underlying graph. Furthermore, the eigenvalues of W are unchanged by this transformation, and one can readily verify that the left and right eigenvectors of the eigenvalue μ for the matrix $\widehat{W}$ are given by $\mathbf{c}' = \begin{bmatrix}c_1 & c_2 & \cdots & c_N\end{bmatrix}$ and

$$\mathbf{d} = \begin{bmatrix} \frac{\bar{c}_1}{c_1} & 0 & \cdots & 0 \\ 0 & \frac{\bar{c}_2}{c_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\bar{c}_N}{c_N} \end{bmatrix} \bar{\mathbf{d}} \ ,$$

respectively. Since all c_i's and $\bar{c}_i$'s are nonzero, all entries in the eigenvector $\mathbf{d}$ will also be nonzero. Thus, the weight matrix $\widehat{W}$ is a valid matrix for the given graph, with a left eigenvector $\mathbf{c}'$ corresponding to an observable eigenvalue for all nodes.

Once the weight matrix is chosen so that the function $\mathbf{c}'x[0]$ is calculable by node i, we can find the smallest L_i for which the vector $\mathbf{c}'$ is in the row-space of the matrix $\mathcal{O}_{i,L_i}$ in (3), i.e., find the smallest L_i for which

$$\rho\left(\begin{bmatrix}\mathcal{O}_{i,L_i} \\ \mathbf{c}'\end{bmatrix}\right) = \rho(\mathcal{O}_{i,L_i}) \ . \tag{10}$$

The largest value of L_i for which this can happen is given by the bounds in (4). One can then find a vector $\Gamma_i = \begin{bmatrix}\Gamma_{i,0} & \Gamma_{i,1} & \cdots & \Gamma_{i,L_i}\end{bmatrix}$ such that

$$\Gamma_i \mathcal{O}_{i,L_i} = \mathbf{c}' \ . \tag{11}$$

Thus, after running the linear iteration (1) for $L_i + 1$ time-steps, node i can immediately calculate the desired function as a linear combination of the outputs of the system over those time steps, i.e.,

$$\Gamma_i \begin{bmatrix} y_i[0] \\ y_i[1] \\ \vdots \\ y_i[L_i] \end{bmatrix} = \Gamma_i \mathcal{O}_{i,L_i} x[0] = \mathbf{c}'x[0] \ .$$

Note that node i does not necessarily have to store the entire set of values $y_i[0], y_i[1], \ldots, y_i[L_i]$ in order to calculate this quantity. Instead, suppose node i has a single extra register, denoted $\bar{x}_i[k]$, initialized with $\bar{x}_i[0] = \Gamma_{i,0} y_i[0]$. At each time-step k, node i updates this register as

$$\bar{x}_i[k] = \bar{x}_i[k-1] + \Gamma_{i,k} y_i[k] \ . \tag{12}$$

After time-step $k = L_i$, this register will hold the value of $\mathbf{c}'x[0]$.

Note that if multiple nodes have to calculate the same linear functional of the initial values (as in distributed consensus), it may be the case that some nodes calculate the value faster than others. In this case, once a node has calculated the value of the function, it can send this value to its neighbors with a flag indicating that it is the final value, and not simply the next step in the iteration. In this way, slower neighbors will receive the function value at most one time-step after a faster neighbor calculates it.

Also note that the ability to reach consensus in this scheme is completely independent of the magnitude of the eigenvalues in W. It is sufficient that the eigenvalue associated with the eigenvector $\mathbf{c}'$ be observable by some or all of the nodes.

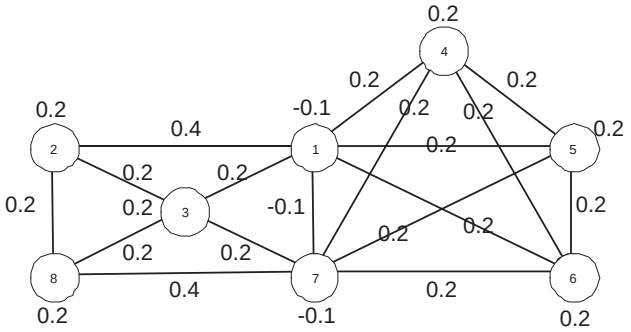

Figure 2: Graph with optimal weights to maximize asymptotic convergence rate [17].

Example 2. Consider the undirected graph from [17] in Fig. 2. The objective in this system is for the nodes to reach consensus on the average of the initial values (i.e., each node must calculate $\frac{1}{N}\mathbf{1}'x[0]$). The weights on the nodes and edges indicate the optimal weights to maximize the asymptotic rate of convergence for the system (or equivalently, to minimize the second largest eigenvalue of W), as calculated in [17]. This choice of weights satisfies the conditions in Theorem 1, and so the eigenvalue 1 is simple, with $\mathbf{1}$ as a right eigenvector and $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$ as a left eigenvector. From our discussion so far, this implies that the function $\frac{1}{N}\mathbf{1}'x[0]$ is calculable by all nodes. We will use these weights to demonstrate our scheme for distributed consensus, but as we have already discussed, one does not have to restrict attention to the optimal asymptotic weights when using our method.

To find the set of coefficients Γ_i for each node i, we first have to find the smallest integer for which the rank condition in (10) holds with $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$. For example, consider node 2 in Fig. 2. The matrix E_2 for this node is

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \ .$$

With $L_2 = 0$, we have $\mathcal{O}_{2,0} = E_2$, and equation (10) does

not hold. For $L_2 = 1$, we have

$$\mathcal{O}_{2,1} = \begin{bmatrix} E_2 \\ E_2 W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -0.1 & 0.4 & 0.2 & 0.2 & 0.2 & 0.2 & -0.1 & 0 \\ 0.4 & 0.2 & 0.2 & 0 & 0 & 0 & 0 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0 & 0 & 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 & 0 & 0 & 0 & 0.4 & 0.2 \end{bmatrix},$$

and we find that (10) is satisfied. Therefore node 2 can calculate the average of the initial values after two time-steps (i.e., after it sees the outputs $y_2[0]$ and $y_2[1]$). The coefficients Γ_2 in (11) are given by

$$\Gamma_2 = \begin{bmatrix} -0.2679 & -0.4464 & -0.3214 & -0.1964 \\ 0.6250 & -0.2098 & 2.6965 & -0.8795 \end{bmatrix}.$$

The coefficients Γ_i for all of the other nodes are found in a similar manner. For this example, it is found that $L_i = 1$ for all i (i.e., all nodes can calculate the average after two time-steps). We omit the explicit values of the coefficients in the interest of space.

Suppose the initial values of the nodes are given by

$$x[0] = \begin{bmatrix} -1.3256 & -13.6558 & 4.2533 & 5.8768 \\ -8.4647 & 14.9092 & 14.8916 & 2.6237 \end{bmatrix}',$$

which has a mean of 2.3885. Running iteration (1) for 2 time-steps, we get

$$\begin{bmatrix} x[0] & x[1] \end{bmatrix} = \begin{bmatrix} -1.3256 & -3.5040 \\ -13.6558 & -1.8860 \\ 4.2533 & 1.3574 \\ 5.8768 & 5.1774 \\ -8.4647 & 5.1774 \\ 14.9092 & 5.1774 \\ 14.8916 & 3.0078 \\ 2.6237 & 4.6009 \end{bmatrix}.$$

Node 2 has access to the values $y_2[0] = E_2 x[0]$ and $y_2[1] = E_2 x[1]$. It can therefore calculate the average as

$$\Gamma_2 \begin{bmatrix} y_2[0] \\ y_2[1] \end{bmatrix} = \Gamma_2 \begin{bmatrix} -1.3256 \\ -13.6558 \\ 4.2533 \\ 2.6237 \\ -3.5040 \\ -1.8860 \\ 1.3574 \\ 4.6009 \end{bmatrix} = 2.3885 .$$

All other nodes also obtain the average by multiplying their outputs by their own coefficients Γ_i, and so consensus is reached in the system after two time-steps.

Remark 1. Note that both the radius and the diameter of the graph in Fig. 2 are equal to 2. In other words, for every node i, there is a node j at distance 2 from i. Therefore no algorithm can achieve consensus in fewer than two time-steps (assuming it takes one time-step for a message to propagate along an edge), and this lower bound is achieved by our scheme (for this example). In contrast, suppose we utilize a naive approach where all nodes send their values to a single node, which then calculates the desired function and broadcasts it back to all nodes. In the above graph, this approach would require at least four time-steps (two time-steps for the central node to receive all values, and another two time-steps for the result to propagate back to all nodes).

Remark 2. Note also that the weights in the above example are the optimal weights for maximizing the rate of asymptotic convergence. Loosely speaking, we see from $x[1]$ that the nodes are far from reaching consensus after one iteration, even when the optimal weights are used. Since our scheme makes use of the history of values seen by each node, instead of only the values at the current time-step, we are able to immediately calculate the consensus value.

Remark 3. In this example, the optimal weights for asymptotic consensus also provided convergence in a minimal number of time-steps when used in conjunction with our scheme. However, it is not clear whether this will be true in general graphs. In fact, there exist weight matrices for this example that have worse asymptotic performance than the optimal weights shown in Fig. 2, but that still allow all nodes to reach consensus in two time-steps when used in our scheme; one such weight matrix is

$$W = \frac{1}{8} \begin{bmatrix} 3 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 5 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 5 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 4 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 4 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 4 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 5 \end{bmatrix}.$$

A characterization of the optimal weights for finite-time convergence, and their relationship to the optimal weights for asymptotic convergence, will be the subject of future research.

5. ONLINE CALCULATION OF THE COEFFICIENTS

From our development in the last section, we see that each node only requires the coefficient vector Γ_i and the values of its neighbors and itself over the first $L_i + 1$ time-steps in order to calculate the desired function of the initial values. Furthermore, the calculation of Γ_i requires knowledge of the matrix $\mathcal{O}_{i,L_i}$. If the global topology of the graph and all of the weights are known *a priori*, then Γ_i can be calculated from the matrix $\mathcal{O}_{i,L_i}$ and conveyed to node i. Note that in graphs with time-invariant topologies, Γ_i only has to be computed once for each i. However, in practice, it may the case that there is no opportunity to calculate the coefficients and the weight matrix W *a priori*, and therefore it will be necessary for the nodes to calculate these parameters using only local information. In this section, we show how the nodes can distributively calculate their own coefficients once the weights in matrix W are chosen. To accomplish this, we will assume that the nodes know N (or an upper bound for N). We will also assume that each node has a unique identifier, and that the nodes know their position in an appropriate ordering of the identifiers (e.g., the node with the j'th smallest identifier takes its position to be j). We assume without loss of generality that the vector $x[k]$ is

consistent with this ordering (i.e., $x_i[k]$ is the value of the node whose position is i'th in the ordering).

As noted earlier, there are various ways for the nodes to choose their weights so that W has an eigenvalue μ that is observable by all nodes, with a left eigenvector $\mathbf{c}'$ such that $\mathbf{c}'x[0]$ is calculable by all nodes. For example, the integer weights in (9) produce a weight matrix with $\mu = S$ and $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$. Other sets of weights that produce similar results are described in [14, 2, 8, 9], and in the previous section.

Once an appropriate set of weights is chosen, suppose the nodes perform N runs of the linear iteration, each for $N-1$ time-steps. During the j'th run, node j sets its initial condition to be 1, and all other nodes set their initial conditions to be zero. In other words, if $x_{*,j}[k]$ denotes the vector of node values during the k'th time-step of the j'th run, the nodes calculate $x_{*,j}[k+1] = Wx_{*,j}[k]$, $0 \leq k \leq N-2$, $1 \leq j \leq N$, where $x_{*,j}[0] = e_j$. Since the structure of $x_{*,j}[0]$ is predetermined, each node also knows the value of the function $\mathbf{c}'x_{*,j}[0]$.

Suppose that for each of the N runs, node i stores the outputs of the system over the first $N - \deg_i$ time-steps, along with the function value $\mathbf{c}'x_{*,j}[0]$. Each node i then has access to the matrix

$$\Psi_{i,L_i} = \begin{bmatrix} y_{i,1}[0] & y_{i,2}[0] & \cdots & y_{i,N}[0] \\ y_{i,1}[1] & y_{i,2}[1] & \cdots & y_{i,N}[1] \\ \vdots & \vdots & \ddots & \vdots \\ y_{i,1}[L_i] & y_{i,2}[L_i] & \cdots & y_{i,N}[L_i] \end{bmatrix}$$

for any $0 \leq L_i \leq N - \deg_i - 1$, where $y_{i,j}[k] = E_i x_{*,j}[k] = E_i W^k x_{*,j}[0]$. Each node can also form the matrix

$$\Theta = \begin{bmatrix} \mathbf{c}'x_{*,1}[0] & \mathbf{c}'x_{*,2}[0] & \cdots & \mathbf{c}'x_{*,N}[0] \end{bmatrix} \ .$$

Denoting $\mathcal{R} = \begin{bmatrix} x_{*,1}[0] & x_{*,2}[0] & \cdots & x_{*,N}[0] \end{bmatrix}$, we note that

$$\Psi_{i,L_i} = \mathcal{O}_{i,L_i}\mathcal{R} \ , \tag{13}$$

$$\Theta = \mathbf{c}'\mathcal{R} \ . \tag{14}$$

Given the matrices Ψ_{i,L_i} and Θ, each node i finds the smallest integer L_i for which

$$\rho\left(\begin{bmatrix} \Psi_{i,L_i} \\ \Theta \end{bmatrix}\right) = \rho(\Psi_{i,L_i}) \ , \tag{15}$$

and it can then find a vector Γ_i such that

$$\Gamma_i \Psi_{i,L_i} = \Theta \ . \tag{16}$$

Using (13) and (14), this is equivalent to

$$\Gamma_i \mathcal{O}_{i,L_i}\mathcal{R} = \mathbf{c}'\mathcal{R} \ .$$

The matrix $\mathcal{R}$ is the identity matrix (due to the structure of the initial conditions $x_{*,1}[0], \ldots, x_{*,N}[0]$), and the above expression is equivalent to

$$\Gamma_i \mathcal{O}_{i,L_i} = \mathbf{c}' \ .$$

Thus, Γ_i is precisely the set of coefficients that node i requires in order to calculate the function $\mathbf{c}'x[0]$ for any $x[0]$.

Remark 4. Note that calculating Γ_i from (16) actually only requires that the matrix $\mathcal{R}$ be nonsingular and that each node have knowledge of the final values of each run (given by the matrix Θ). Thus, an alternative method of calculating Γ_i would be as follows. Suppose the nodes choose the weights in order to obtain asymptotic consensus (i.e., the W matrix satisfies the conditions in Theorem 1). Therefore, for any initial condition $x_{*,j}[0]$, each node can find the value $\mathbf{c}'x_{*,j}[0]$ (by running the asymptotic consensus protocol), and can form the matrix Θ. Furthermore, if all initial conditions are taken to be independent, identically distributed random variables with a Gaussian distribution, then the vectors $x_{*,1}[0], \ldots, x_{*,N}[0]$ will almost surely be linearly independent (e.g., [15]), and the matrix $\mathcal{R}$ will be nonsingular. Equation (16) can then be used to calculate the coefficients Γ_i. The benefit of this method is that the nodes do not need to have identifiers in order to specify the initial conditions appropriately. However, obtaining the values in Θ requires the nodes to run each linear iteration for a large number of time-steps in order to calculate a sufficiently accurate approximation to the final function values $\mathbf{c}'x_{*,j}[0]$ (using the asymptotic consensus protocol). In contrast, when the initial conditions for each run are deterministic, each linear iteration only has to be performed for $N-1$ time-steps (in order to obtain the matrix Ψ_{i,L_i}); this, however, requires nodes to be associated with unique identifiers, and ordered.

Once node i has calculated its coefficient vector Γ_i, it can calculate the linear functional $\mathbf{c}'x[0]$ for future initial conditions $x[0]$ by running the linear iteration (1) for $L_i + 1$ time-steps to obtain the values

$$y_i[0], y_i[1], \ldots, y_i[L_i] \ ,$$

or until it receives the function value from a neighbor. If $L_i + 1$ time-steps pass without receiving the function value, node i can then immediately calculate the value as

$$\mathbf{c}'x[0] = \Gamma_i \begin{bmatrix} y_i[0] \\ \vdots \\ y_i[L_i] \end{bmatrix} \ .$$

As noted in the previous section, node i does not have to store the outputs $y_i[0], \ldots, y_i[L_i]$ in order to calculate the above quantity, but can instead utilize a single additional register updated according to equation (12). Once node i obtains the function value, it then transmits this value to its neighbors with a flag indicating that it is the final value, and the algorithm terminates (for node i).

Example 3. Consider the graph shown in Fig. 3. Node 3 in the graph is the fusion center, and it has to calculate the average of all the initial node values. None of the nodes know the entire graph topology, and node 3 does not have *a priori* knowledge of the coefficient vector Γ_3. However, suppose that all nodes know that there are $N = 6$ nodes in the system.

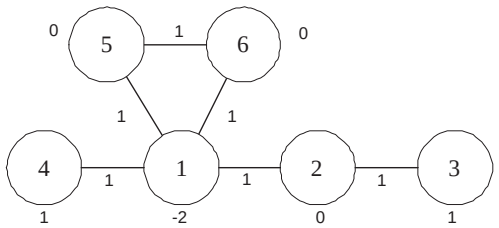

Figure 3: Graph with integer weights. The edge weights are set to 1, and the self weights are chosen so that each row of W sums to $S = 2$.

The first step is for the nodes to choose their weights. For instance, the nodes can choose the integer weights in (9), with S chosen (somewhat arbitrarily) as $S = 2$. This produces a weight matrix W with $\mu = 2$, $W\mathbf{1} = \mu\mathbf{1}$ and $\frac{1}{6}\mathbf{1}'W = \mu\frac{1}{6}\mathbf{1}'$ (i.e., the functional $\frac{1}{6}\mathbf{1}'x[0]$ is calculable by all nodes).

The matrix E_3 for node 3 is given by

$$E_3 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} .$$

Since node 3 does not know the coefficients Γ_3 yet, the nodes simply use the protocol described in this section to perform N runs with initial conditions $x_{*,j}[0] = e_j$, $1 \leq j \leq 6$. The function value for each run is given by $\mathbf{c}'x_{*,j}[0] = \frac{1}{6}$, $1 \leq j \leq 6$. For each of these runs, node 3 stores the output of the system over the first $N - \deg_3 = 5$ time-steps. At this point, node 3 has access to the matrix

$$\begin{bmatrix} y_{3,1}[0] & y_{3,2}[0] & y_{3,3}[0] & y_{3,4}[0] & y_{3,5}[0] & y_{3,6}[0] \\ y_{3,1}[1] & y_{3,2}[1] & y_{3,3}[1] & y_{3,4}[1] & y_{3,5}[1] & y_{3,6}[1] \\ y_{3,1}[2] & y_{3,2}[2] & y_{3,3}[2] & y_{3,4}[2] & y_{3,5}[2] & y_{3,6}[2] \\ y_{3,1}[3] & y_{3,2}[3] & y_{3,3}[3] & y_{3,4}[3] & y_{3,5}[3] & y_{3,6}[3] \\ y_{3,1}[4] & y_{3,2}[4] & y_{3,3}[4] & y_{3,4}[4] & y_{3,5}[4] & y_{3,6}[4] \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ \hline -2 & 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 0 & 0 & 0 \\ \hline 9 & -1 & 3 & -1 & -1 & -1 \\ -1 & 3 & 3 & 1 & 1 & 1 \\ \hline -22 & 12 & 2 & 8 & 8 & 8 \\ 8 & 2 & 6 & 0 & 0 & 0 \end{bmatrix} ,$$

where $y_{3,j}[k]$ is the output seen by node 3 at time-step k of the j'th run. It then finds the smallest L_3 for which the rank condition in (15) holds. In this example, the condition holds for $L_3 = 2$, and node 3 calculates the coefficient vector Γ_3 from (16) as

$$\Gamma_3 = \frac{1}{48}\begin{bmatrix} 1 & -15 & 19 & -14 & 8 & 5 \end{bmatrix} .$$

On the next run, suppose that the initial conditions on the nodes are given by

$$x[0] = \begin{bmatrix} 7 & 6 & 2 & 11 & -2 & 26 \end{bmatrix}' ,$$

which has a mean of 8.3333. After running the linear iteration for three time-steps, the values of the nodes over those time-steps are given by

$$\begin{bmatrix} x[0] & x[1] & x[2] \end{bmatrix} = \begin{bmatrix} 7 & 27 & 11 \\ 6 & 9 & 35 \\ 2 & 8 & 17 \\ 11 & 18 & 45 \\ -2 & 33 & 32 \\ 26 & 5 & 60 \end{bmatrix} .$$

Using the outputs of the system $y_3[0] = E_3x[0], y_3[1] = E_3x[1]$ and $y_3[2] = E_3x[2]$, node 3 can now calculate the average as

$$\Gamma_3 \begin{bmatrix} y_3[0] \\ y_3[1] \\ y_3[2] \end{bmatrix} = \Gamma_3 \begin{bmatrix} 6 \\ 2 \\ 9 \\ 8 \\ 35 \\ 17 \end{bmatrix} = 8.3333 .$$

6. DISCUSSION AND FUTURE WORK

As we have seen from our development, once the matrix W is chosen so that a desired set of functions are calculable (i.e., they are in the row-space of the observability matrix), the nodes can obtain the function in a finite number of time-steps. We showed that all left eigenvectors of the matrix W corresponding to observable eigenvalues for a given node will be calculable by that node. Furthermore, we discussed how to choose the weight matrix so that certain functions will be calculable by all nodes. Our development does not depend on the magnitude of the eigenvalues of the weight matrix, unlike the asymptotic schemes currently studied in the literature. Furthermore, we showed that it is possible for the nodes to learn the necessary information for calculating the desired function after running the linear iteration with several different initial conditions.

There are a number of interesting directions for future research.

1. How does one choose the weight matrix to minimize the number of time-steps required to calculate the desired function by a certain node? The cost incurred by using our scheme is that node i has to store the coefficients Γ_i. The size of the coefficient vector Γ_i is proportional to the number of time-steps required to calculate the function, and so fewer coefficients will have to be stored if the time is minimized.

2. How does one choose the weight matrix so that a certain node can calculate multiple different functions at the same time? Along the same lines, can the weight matrix be designed so that only a subset of the nodes can calculate a certain function, but other nodes cannot?

3. Are there other (perhaps more efficient) methods for the nodes to distributively determine their coefficients Γ_i? The method that we have described in this paper requires the nodes to know N, and to potentially store a large amount of data (if N is large) in order to calculate the coefficients. Is there a way to reduce the storage requirements of the algorithm?

4. How does one incorporate robustness to errors in the nodes and links? More generally, how does one handle time-varying graphs?

We are actively pursuing these and other ideas in our research.

7. ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation under NSF Career Award 0092696 and NSF ITR Award 0426831, and in part by the Air Force Office of Scientific Research under URI Award No

F49620-01-1-0365URI. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or AFOSR.

8. REFERENCES

[1] D. S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005.

[2] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 2996–3000, 2005.

[3] C.-T. Chen. *Linear System Theory and Design*. Holt, Rinehart and Winston, 1984.

[4] J. Cortés. Finite-time convergent gradient flows with applications to network consensus. *Automatica*, 42(11):1993–2000, Nov. 2006.

[5] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright. Geographic gossip: Efficient aggregation for sensor networks. In *Proceedings of the 5th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 69–76, 2006.

[6] A. Giridhar and P. R. Kumar. Computing and communicating functions over sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):755–764, Apr. 2005.

[7] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[8] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[9] D. B. Kingston and R. W. Beard. Discrete-time average-consensus under switching network topologies. In *Proceedings of the American Control Conference*, pages 3551–3556, 2006.

[10] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[11] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, Sep. 2004.

[12] R. Olfati-Saber and J. S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6698–6703, 2005.

[13] W. Ren and R. W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, May 2005.

[14] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the American Control Conference*, pages 1859–1864, 2005.

[15] M. Rudelson. Norm of the inverse of a random matrix. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 487–496, 2006.

[16] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, Massachusetts Institute of Technology, 1984.

[17] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, Sep. 2004.

[18] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 63–70, 2005.