Finite-Time Distributed Consensus in Graphs with Time-Invariant Topologies

Shreyas Sundaram and Christoforos N. Hadjicostis

***Abstract*— We present a method for achieving consensus in distributed systems in a finite number of time-steps. Our scheme involves a linear iteration where, at each time-step, each node updates its value to be a weighted average of its own previous value and those of its neighbors. If D denotes the degree of the minimal polynomial of the weight matrix associated with the linear iteration, we show that each node can immediately calculate the consensus value as a linear combination of its own past values over at most D time-steps. We also show that each node can determine the coefficients for this linear combination in a decentralized manner. The proposed scheme has the potential to significantly reduce the time and communication required to reach consensus in distributed systems.**

I. Introduction

In distributed systems, it is often necessary for the various nodes to come to an agreement about certain parameters. For example, the nodes might be voting on a course of action, or attempting to balance their workload across all available computing resources. The problem of reaching consensus in such systems has received extensive attention in the computer science literature over the past few decades [1], leading to the development of various protocols. The topic of distributed consensus has also attracted the attention of the control community due to its applicability to tasks such as data fusion in sensor networks and coordination of multi-agent systems [2]. In these cases, the approach to consensus is to use a linear iteration, where each node repeatedly updates its value as a weighted linear combination of itself and its neighbors [3], [4], [5], [6], [7]. These works have revealed that if the network topology satisfies certain conditions, the weights for the linear iteration 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, at each time-step, each node only has to transmit a single value to each of its neighbors. 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 this paper, we present a method to address the above shortcoming of linear iteration-based consensus schemes. Specifically, in graphs with time-invariant topologies, we show that each node can immediately calculate the consensus value after observing the evolution of its own value over a finite number of time-steps. The topic of finite-time convergence was briefly discussed by Kingston and Beard in [8]. 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 [9]. 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.

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. 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) [10]. The notation $(\cdot)'$ indicates matrix transpose.

II. Background

The interaction constraints in distributed systems 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., there is a directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_j can transmit information to node x_i). 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 $\mathcal{N}_i$. Suppose that each node 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

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

This material is based upon work supported in part by the National Science Foundation under NSF Career Award 0092696 and NSF EPNES Award 0224729, 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.

The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA. E-mail {ssundarm, chadjic}@uiuc.edu

where the w_{ij}'s are a set of weights. 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 $(x_j, x_i) \notin \mathcal{E}$.

Definition 1 (Distributed Consensus): The system is said to achieve distributed consensus if

$$\lim_{k\to\infty} x[k] = \mathbf{1}\mathbf{c}'x[0] \ ,$$

where $\mathbf{1}$ is the column vector consisting of all 1's, and $\mathbf{c}'$ is some constant row vector. In other words, the values of all nodes converge to the same linear combination of the initial node values $\mathbf{c}'x[0]$. 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 following result from [5] characterizes the conditions under which the iteration (1) achieves consensus.

Theorem 1 ([5]): The iteration given by (1) achieves distributed 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 1.
2) The left and right eigenvectors of W corresponding to eigenvalue 1 are $\mathbf{c}'$ and $\mathbf{1}$, respectively.

□

Furthermore, [5] showed that the rate of convergence is determined by the second largest eigenvalue of W. Consequently, to maximize the rate of convergence, the authors of [5] formulated a convex optimization problem to obtain the weight matrix satisfying the above conditions while minimizing the magnitude of the second largest eigenvalue.

A salient feature of the analysis in [5], and of other linear iterations studied in the literature (e.g., see [2] and the references therein), is that they are concerned with asymptotic convergence. However, we show in the next section that if the matrix W satisfies the conditions in Theorem 1, then we can obtain consensus in a *finite* number of steps. We then extend our discussion to a larger class of weight matrices (including matrices with unstable eigenvalues).

III. Convergence in a Finite Number of Steps

To see how one can obtain consensus in a finite number of time-steps, we use the concept of the *minimal polynomial* of a matrix. Specifically, the minimal polynomial of the matrix W is the unique monic polynomial $q(t)$ of smallest degree such that $q(W) = 0$. If the distinct eigenvalues of W are denoted by $\lambda_1, \lambda_2, \ldots, \lambda_m$, then the minimal polynomial can be explicitly calculated as

$$q(t) = \prod_{i=1}^{m} (t - \lambda_i)^{r_i} \ , \qquad (2)$$

where r_i is the size of the largest Jordan block of W corresponding to eigenvalue λ_i [10]. Note that the minimal polynomial will have degree at most N (since the sizes of all Jordan blocks of W have to add up to N). Suppose the minimal polynomial of W has degree $D+1$ ($D+1 \leq N$), and expand (2) as $q(t) = t^{D+1} + \alpha_D t^D + \cdots + \alpha_1 t + \alpha_0$ for some constants $\alpha_0, \alpha_1, \ldots, \alpha_D$. Since $q(W) = 0$, we obtain the identity

$$W^{D+1} + \alpha_D W^D + \cdots + \alpha_1 W + \alpha_0 I = 0 \ . \qquad (3)$$

Examining the linear iteration in (1), we note that $x[k] = W^k x[0]$ for all k. However, from (3), we see that the value $x[D+1]$ can equivalently be obtained as

$$\begin{aligned} x[D+1] &= W^{D+1}x[0] \\ &= -(\alpha_D W^D + \cdots + \alpha_1 W + \alpha_0 I)x[0] \\ &= -\alpha_D x[D] - \cdots - \alpha_1 x[1] - \alpha_0 x[0] \ . \end{aligned}$$

Therefore for all $k \geq 0$ and for all $1 \leq i \leq N$, $x_i[k]$ satisfies a linear difference equation of the form

$$x_i[k+D+1] + \alpha_D x_i[k+D] + \cdots + \alpha_1 x_i[k+1] + \alpha_0 x_i[k] = 0 \ .$$

In particular, after time-step $D+1$, each node only needs to know its own $D+1$ previous values in order to compute the linear iteration (1), and does not need to receive further information from its neighbors. Taking the Z-transform of the above expression, we obtain

$$\begin{aligned} &(z^{D+1} + \alpha_D z^D + \cdots + \alpha_1 z + \alpha_0) X_i(z) = \\ &\sum_{j=0}^{D} x_i[j] z^{D+1-j} + \alpha_D \sum_{j=0}^{D-1} x_i[j] z^{D-j} + \cdots + \alpha_1 z x_i[0] \ . \end{aligned} \qquad (4)$$

The expression on the right side of the above equation represents the initial conditions of the linear difference equation, and arises from the Z-transform (e.g., the Z-transform of $x_i[k+1]$ is $zX_i(z) - zx_i[0]$, and so forth) [11]. The term multiplying $X_i(z)$ on the left side of the equation is $q(z)$, the minimal polynomial of W. In particular, if W satisfies the conditions in Theorem 1, the minimal polynomial will have a single root at $z = 1$, and all other roots will have magnitude less than 1 (since the roots of the minimal polynomial are the eigenvalues of W). We can therefore factor $q(z)$ as

$$z^{D+1} + \alpha_D z^D + \cdots + \alpha_1 z + \alpha_0 = (z-1)p(z) \ ,$$

where

$$\begin{aligned} p(z) = z^D + (1+\alpha_D) z^{D-1} + (1 + \alpha_D + \alpha_{D-1}) z^{D-2} \\ + \cdots + (1 + \sum_{j=2}^{D} \alpha_j) z + (1 + \sum_{j=1}^{D} \alpha_j) \ . \end{aligned} \qquad (5)$$

Since the roots of $p(z)$ have magnitude strictly less than 1, we can now use (4) and (5) in the final value theorem (e.g., see [11]) to obtain

$$\begin{aligned} \lim_{k\to\infty} x_i[k] &= \lim_{z\to 1} (z-1) X_i(z) \\ &= \frac{\begin{bmatrix} x_i[D] & x_i[D-1] & \cdots & x_i[0] \end{bmatrix} S}{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} S} \ , \end{aligned} \qquad (6)$$

where

$$S = \begin{bmatrix} 1 \\ 1+\alpha_D \\ 1+\alpha_{D-1}+\alpha_D \\ \vdots \\ 1+\sum_{j=1}^{D}\alpha_j \end{bmatrix} . \quad (7)$$

Note that (6) is completely decentralized, in the sense that the calculation of the final value of each component of x does not depend on any other component of x. The only requirement is that the linear difference equation be initialized with a sufficiently large number of values. During this initialization phase, each node obtains enough information about the values of the other nodes to calculate $\mathbf{c}'x[0]$.

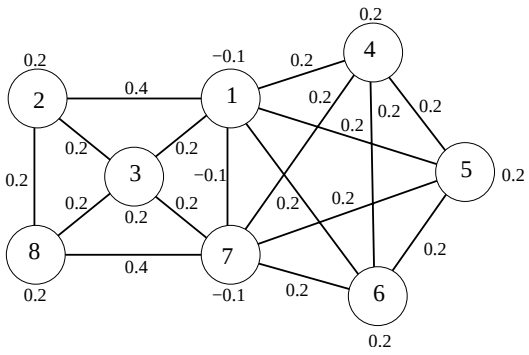

Fig. 1. Graph with optimal weights to maximize convergence rate [5].

Example 1: Consider the undirected graph from [5] shown in Fig. 1. The objective in this system is for the nodes to perform distributed averaging via linear iterations (i.e., $x[k] \to \frac{1}{N}\mathbf{1}\mathbf{1}'x[0]$). The weights on the nodes and edges indicate the optimal weights to maximize the rate of convergence for the system (or equivalently, to minimize the second largest eigenvalue of W), as calculated in [5]. The corresponding weight matrix W is symmetric and has eigenvalues $\text{eig}(W) = \{1, 0.6, 0.4, 0, 0, 0, -0.4, -0.6\}$. The eigenvalue 1 is simple, and has $\mathbf{1}$ as a right eigenvector and $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$ as a left eigenvector. Since W is symmetric, the largest Jordan block for each eigenvalue will have size 1. Therefore, from (2), the minimal polynomial will be

$$\begin{aligned} q(t) &= (t-1)(t-0.6)(t-0.4)(t)(t+0.4)(t+0.6) \\ &= t^6 - t^5 - 0.52t^4 + 0.52t^3 + 0.0576t^2 - 0.0576t \\ &\equiv t^6 + \alpha_5 t^5 + \alpha_4 t^4 + \alpha_3 t^3 + \alpha_2 t^2 + \alpha_1 t + \alpha_0 \ , \end{aligned}$$

with degree $D+1 = 6$. Using the above values for the α_i's, the vector S from (7) is

$$S = \begin{bmatrix} 1 & 0 & -0.52 & 0 & 0.0576 & 0 \end{bmatrix}' .$$

Suppose the initial values of the nodes are given by

$$x[0] = \begin{bmatrix} 1.3889 & 2.0277 & 1.9872 & 6.0379 & 2.7219 \\ & & & & 1.9881 & 0.1527 & 7.4679 \end{bmatrix}' ,$$

which has a mean of 2.9715. Running iteration (1) for $D+1 = 6$ time-steps, we get

$$\begin{bmatrix} x[5] & x[4] & x[3] & x[2] & x[1] & x[0] \end{bmatrix} = \begin{bmatrix} 3.1148 & 2.7022 & 3.2728 & 2.2782 & 3.2039 & 1.3889 \\ 2.9303 & 3.0128 & 2.8791 & 2.8445 & 2.8521 & 2.0277 \\ 2.9240 & 3.0825 & 2.8395 & 3.2797 & 2.6049 & 1.9872 \\ 2.9050 & 3.0507 & 2.7866 & 3.1915 & 2.4579 & 6.0379 \\ 2.9050 & 3.0507 & 2.7866 & 3.1915 & 2.4579 & 2.7219 \\ 2.9050 & 3.0507 & 2.7866 & 3.1915 & 2.4579 & 1.9881 \\ 3.1705 & 2.6705 & 3.6210 & 2.0804 & 5.3800 & 0.1527 \\ 2.9177 & 3.1521 & 2.8000 & 3.7149 & 2.3576 & 7.4679 \end{bmatrix} .$$

Each node can now obtain the consensus value by using (6) to get

$$\frac{\begin{bmatrix} x_i[5] & x_i[4] & x_i[3] & x_i[2] & x_i[1] & x_i[0] \end{bmatrix} S}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} S} = 2.9715$$

for $1 \le i \le 8$. □

IV. Finite-Time Convergence under More General Conditions

We now discuss an extension of the above ideas to a more general framework. We showed in the last section that it is possible to obtain finite-time consensus with any weight matrix that satisfies the conditions for asymptotic consensus. In this section, we will show that one does not need to restrict attention to such matrices. To accomplish this, we use (6) as a guide to ask the question: when is it possible for node i to calculate the consensus value as a linear combination of the values $x_i[0], x_i[1], \ldots, x_i[D_i]$, for some D_i? Mathematically, we would like to find a coefficient vector $\Gamma_i \equiv \begin{bmatrix} \gamma_{i,D_i} & \gamma_{i,D_i-1} & \cdots & \gamma_{i,0} \end{bmatrix}'$ for each node i such that

$$\begin{bmatrix} x_i[D_i] & x_i[D_i-1] & \cdots & x_i[0] \end{bmatrix} \Gamma_i = \mathbf{c}'x[0] \ . \quad (8)$$

Note that we are not requiring all nodes to have the same value of D_i; we will comment more on this later in the section. Since $x_i[k] \equiv e_i'x[k] = e_i'W^k x[0]$, and since equation (8) has to hold for any $x[0]$, we need to solve the equivalent problem of finding $\gamma_{i,0}, \gamma_{i,1}, \ldots, \gamma_{i,D_i}$ such that

$$e_i'\left(\gamma_{i,D_i}W^{D_i} + \gamma_{i,D_i-1}W^{D_i-1} + \cdots + \gamma_{i,0}I\right) = \mathbf{c}' \ . \quad (9)$$

The following theorem provides a method to determine the coefficients $\gamma_{i,j}$ satisfying equation (9). Note that the conditions on W required by the theorem are less restrictive than those required for asymptotic convergence (as given in Theorem 1): the only constraint on W is that it have a simple eigenvalue with a right eigenvector that has all entries nonzero. However, the magnitudes of the eigenvalues are completely unconstrained (e.g., they can be unstable, if desired).

Theorem 2: Let W be the weight matrix for a graph, with the constraint that $w_{ij} = 0$ if there is no edge from node j to node i. Suppose W has a simple eigenvalue at μ, with

$$W\mathbf{d} = \mu\mathbf{d}, \ \ \mathbf{c}'W = \mu\mathbf{c}' \ ,$$

for some column vectors $\mathbf{c}$ and $\mathbf{d}$, where all entries of $\mathbf{d}$ are nonzero. Let $\hat{q}_i(t)$ be any monic polynomial with a single root at $t = \mu$ that satisfies $e_i'\hat{q}_i(W) = 0$, and let $p_i(t) \equiv \frac{\hat{q}_i(t)}{t-\mu}$. Then

$$e_i' \frac{\mathbf{c}'\mathbf{d}}{p_i(\mu)e_i'\mathbf{d}} p_i(W) = \mathbf{c}' \ . \tag{10}$$

□

Proof: Since $e_i'\hat{q}_i(W) = 0$ and $\hat{q}_i(t) = p_i(t)(t-\mu)$, we have

$$e_i' p_i(W)\,(W - \mu I) = 0 \ .$$

This means that $e_i'p_i(W)W = \mu e_i'p_i(W)$, which indicates that $e_i'p_i(W)$ is a left eigenvector of W corresponding to eigenvalue μ. Since μ is a simple eigenvalue of W with left eigenvector $\mathbf{c}'$, we have

$$e_i'p_i(W) = \nu_i \mathbf{c}' \tag{11}$$

for some constant ν_i. Now, since $\mathbf{d}$ is a right eigenvector of W corresponding to eigenvalue μ, we have $e_i'W\mathbf{d} = e_i'\mu\mathbf{d} = \mu e_i'\mathbf{d}$. Right-multiplying (11) by $\mathbf{d}$ we get

$$\nu_i \mathbf{c}'\mathbf{d} = e_i'p_i(W)\mathbf{d} = e_i'p_i(\mu)\mathbf{d} = p_i(\mu)e_i'\mathbf{d} \ ,$$

from which we obtain $\nu_i = \frac{p_i(\mu)e_i'\mathbf{d}}{\mathbf{c}'\mathbf{d}}$. Substituting this into (11), we get the desired equation (10). ■

The result in Theorem 2 applies to any monic polynomial $\hat{q}_i(t)$ with a single root at $t = \mu$ satisfying $e_i'\hat{q}_i(W) = 0$. For any such polynomial, we can obtain the coefficients $\gamma_{i,j}$ in (9) as follows. First, find $p_i(t) = \frac{\hat{q}_i(t)}{t-\mu} \equiv t^{D_i} + \beta_{i,D_i-1}t^{D_i-1} + \cdots + \beta_{i,0}$. Comparing (10) to (9), we can then obtain $\Gamma_i \equiv \begin{bmatrix}\gamma_{i,D_i} & \gamma_{i,D_i-1} & \cdots & \gamma_{i,0}\end{bmatrix}'$ as

$$\gamma_{i,j} = \frac{\mathbf{c}'\mathbf{d}}{p_i(\mu)e_i'\mathbf{d}}\beta_{i,j} \ , \tag{12}$$

taking $\beta_{i,D_i} = 1$. Node i can then use (8) to calculate the consensus value after $D_i + 1$ time-steps. Note that node i does not necessarily have to store all of the values $x_i[0], \ldots, x_i[D_i]$ in order to calculate $\mathbf{c}'x[0]$. Instead, node i only requires a single additional register, denoted $\bar{x}_i[k]$, initialized with $\bar{x}_i[0] = \gamma_{i,0}x_i[0]$. At each time-step, after node i calculates $x_i[k]$, it updates this additional register as $\bar{x}_i[k] = \bar{x}_i[k-1] + \gamma_{i,k}x_i[k]$. In this way, $\bar{x}_i[D_i]$ will contain the consensus value given by (8).

One choice of $\hat{q}_i(t)$ satisfying the required conditions is $q(t)$, the minimal polynomial of W. With this choice, the coefficient vectors Γ_i will be the same for all nodes, and all nodes will take $D+1$ time-steps to calculate the coefficient value (i.e., $D_i = D$ for all i). Note that when $\mu = 1$, $\mathbf{d} = \mathbf{1}$, $\mathbf{c}'\mathbf{1} = 1$, and $\hat{q}_i(t) = q(t)$, equation (8) reduces to equation (6). We now develop another choice of polynomials that can be used to obtain finite-time consensus. We introduce the notion of the *minimal polynomial of node i*, and characterize some properties of this polynomial.

Definition 2 (Minimal Polynomial of Node i): The *minimal polynomial of node i*, denoted $q_i(t)$, is the monic polynomial of smallest degree such that $e_i'q_i(W) = 0$. □

Theorem 3: The minimal polynomial $q_i(t)$ of node i divides the minimal polynomial $q(t)$ of W for all $1 \le i \le N$. □

Proof: First, note that since $q(W) = 0$, we automatically have $e_i'q(W) = 0$ for all i. Therefore, the degree of $q_i(t)$ is upper bounded by the degree of $q(t)$. Now suppose $q_i(t)$ does not divide $q(t)$. Then we can write $q(t) = q_i(t)h_i(t) + r_i(t)$ for some quotient polynomial $h_i(t)$ and some remainder polynomial $r_i(t)$ (of smaller degree than $q_i(t)$). Therefore, we have $e_i'q(W) = e_i'q_i(W)h_i(W) + e_i'r_i(W)$, and since both $e_i'q(W)$ and $e_i'q_i(W)$ are zero, we must have $e_i'r_i(W) = 0$. However, this means that $q_i(t)$ is not the polynomial of smallest degree such that $e_i'q_i(W) = 0$, which contradicts our assumption. Therefore $r_i(t)$ must be identically zero, and so $q_i(t)$ must divide $q(t)$. ■

Lemma 1: Suppose the weight matrix W has a simple eigenvalue at μ with right eigenvector $\mathbf{d}$, where all entries of $\mathbf{d}$ are nonzero. Then $q_i(t)$, the minimal polynomial of node i, has a single root at $t = \mu$ for all $1 \le i \le N$.

Proof: Let the Jordan decomposition of W be given by

$$W = T \begin{bmatrix}\mu & 0 \\ 0 & Z\end{bmatrix} T^{-1} \ ,$$

where Z is the set of Jordan blocks of all eigenvalues not equal to μ, and T is a nonsingular matrix whose columns are the (generalized) eigenvectors of W [10]. By the definition of $q_i(t)$, we have $e_i'q_i(W) = 0$, or equivalently,

$$e_i'T \begin{bmatrix}q_i(\mu) & 0 \\ 0 & q_i(Z)\end{bmatrix} T^{-1} = 0 \ .$$

Since $e_i'T$ is the i'th row of T, and since the first column of T is $\mathbf{d}$, the above equation becomes

$$\begin{bmatrix}d_i & T_{i,2:N}\end{bmatrix} \begin{bmatrix}q_i(\mu) & 0 \\ 0 & q_i(Z)\end{bmatrix} = 0 \ ,$$

where d_i is the i'th entry of $\mathbf{d}$, and $T_{i,2:N}$ represents the last $N-1$ entries in the i'th row of T. This means that

$$\begin{bmatrix}d_i q_i(\mu) & T_{i,2:N} q_i(Z)\end{bmatrix} = 0 \ ,$$

from which we have that μ is a root of the polynomial $q_i(t)$ (since d_i is nonzero). To show that it is a single root, note that $q_i(t)$ divides the minimal polynomial $q(t)$ of W (from Theorem 3). Since μ is a simple eigenvalue of W, $q(t)$ will have a single root at $t = \mu$ (from (2)), and so $q_i(t)$ also has a single root at $t = \mu$. ■

Based on the above properties of $q_i(t)$ (the minimal polynomial of node i), we see that if one chooses $\hat{q}_i(t) = q_i(t)$ in Theorem 2, then node i could potentially calculate the consensus value faster than if $\hat{q}_i(t) = q(t)$ (specifically after $D_i + 1$ time-steps, rather than after $D+1$ time-steps). Note that the values of D_i might not be the same for all nodes, and so some nodes can calculate the consensus value faster than others (as we will see in a later example). In fact, once a node has calculated the consensus value, it can then send that value to its neighbors with a flag indicating that it is the consensus value, and not simply the next step in the linear iteration. In this way, slower neighbors of node i can also

obtain the consensus value at most one time-step after node i obtains the value.

The cost for obtaining faster consensus is that the polynomials $q_i(t)$ have to be calculated for each node, and consequently, the coefficients Γ_i may also be different for each node. Given the matrix W, we can calculate $q_i(t)$ for each i as follows. Suppose $q_i(t) = t^{D_i+1} + \alpha_{i,D_i} t^{D_i} + \cdots + \alpha_{i,0}$. Since $e_i' q_i(W) = 0$, we have

$$e_i' \left(W^{D_i+1} + \alpha_{i,D_i} W^{D_i} + \cdots + \alpha_{i,0} I \right) = 0 \ , \qquad (13)$$

or equivalently,

$$\begin{bmatrix} \alpha_{i,0} & \cdots & \alpha_{i,D_i} & 1 \end{bmatrix} \underbrace{\begin{bmatrix} e_i' \\ e_i' W \\ \vdots \\ e_i' W^{D_i+1} \end{bmatrix}}_{\Theta_i} = 0 \ .$$

The matrix Θ_i in the above expression has the form of the observability matrix for the pair (W, e_i') (e.g., see [11]). Thus, $q_i(t)$ can be found by forming the matrix Θ_i and increasing D_i until Θ_i loses rank. The coefficients of $q_i(t)$ can then be obtained from the left nullspace of Θ_i. In the next section, we show how each node can calculate its minimal polynomial (and hence its coefficient vector Γ_i) in a decentralized fashion.

V. Decentralized Calculation of the Minimal Polynomial

If the global topology of the graph and all of the weights are known *a priori* to some centralized source, then the minimal polynomials and the Γ_i's can be calculated from the weight matrix W and conveyed to the nodes, which could then use these parameters to achieve distributed consensus. However, in practice, it may the case that there is no centralized decision maker to assign the weights and calculate the coefficient vectors, and therefore it will be necessary for the nodes to calculate these parameters using only local information. For example, if the graph is undirected and if each node knows N (or an upper bound for N), each node i can independently choose its weights as

$$w_{ij} = \begin{cases} \frac{1}{N}, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{if } j \notin \mathcal{N}_i \\ 1 - \sum_{l \in \mathcal{N}_i} w_{il}, & \text{if } i = j. \end{cases} \qquad (14)$$

The resulting weight matrix is symmetric and satisfies the conditions of Theorem 1 with $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$ [12]. Many other choices of weights exist that provide asymptotic consensus (e.g., see [2], [4], [7], [8]). In this section, we show how the nodes can distributively calculate their own minimal polynomials (and, correspondingly, the coefficient vectors) once they choose their weights. To accomplish this, we will assume that the nodes know N (or an upper bound for N).

Let $x_{*,1}[0], x_{*,2}[0], \ldots, x_{*,N}[0]$ denote a set of N different initial conditions. For each of these initial conditions, suppose that the nodes run the linear iteration (1) for $N+1$ time-steps (i.e., they calculate $x_{*,j}[k+1] = W x_{*,j}[k]$ for $j = 1, \ldots, N$ and $0 \leq k \leq N-1$). Each node i then has access to the $N \times (D_i + 2)$ matrix

$$\mathbf{X}_{i,D_i} \equiv \begin{bmatrix} x_{i,1}[D_i+1] & x_{i,1}[D_i] & \cdots & x_{i,1}[0] \\ x_{i,2}[D_i+1] & x_{i,2}[D_i] & \cdots & x_{i,2}[0] \\ \vdots & \vdots & \ddots & \vdots \\ x_{i,N}[D_i+1] & x_{i,N}[D_i] & \cdots & x_{i,N}[0] \end{bmatrix} \qquad (15)$$

for any $0 \leq D_i \leq N-1$. The value of the i'th node at time-step k for the j'th initial condition can be written as

$$x_{i,j}[k] = e_i' x_{*,j}[k] = e_i' W^k x_{*,j}[0] \ . \qquad (16)$$

Suppose D_i is the smallest integer for which the matrix $\mathbf{X}_{i,D_i}$ in (15) is not full column rank. Then there exists a vector $\mathbf{a}_i \equiv \begin{bmatrix} 1 & \alpha_{i,D_i} & \alpha_{i,D_i-1} & \cdots & \alpha_{i,0} \end{bmatrix}'$ such that $\mathbf{X}_{i,D_i} \mathbf{a}_i = 0$, which means that

$$x_{i,j}[D_i+1] + \alpha_{i,D_i} x_{i,j}[D_i] + \cdots + \alpha_{i,0} x_{i,j}[0] = 0$$

for $j = 1, \ldots, N$. Substituting (16) into the above expression, we get

$$e_i' \left(W^{D_i+1} + \alpha_{i,D_i} W^{D_i} + \cdots + \alpha_{i,0} I \right) x_{*,j}[0] = 0$$

for $j = 1, 2, \ldots, N$. Now, if the set of initial conditions $x_{*,1}[0], \ldots, x_{*,N}[0]$ are linearly independent, the above expression is equivalent to (13). In other words, the degree of the minimal polynomial of node i can be determined from the smallest integer D_i for which the matrix in (15) loses rank, and the nullspace of the corresponding matrix provides the coefficients of $q_i(t)$.

Remark 1: There are various ways to ensure the linear independence of the N different initial conditions. For example, 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., [13]). Alternatively, if the nodes have unique identifiers, one can artificially choose the initial conditions to impose linear independence. For instance, during the j'th run, the node with the j'th smallest identifier can choose its initial condition to be 1, and all other nodes set their initial conditions to be zero. □

Based on the above analysis, we see that each node can learn its own minimal polynomial after running N different linear iterations, each for at least $N+1$ time-steps. For example, one can incorporate this learning phase into asymptotic consensus protocols as follows. For the first N runs, the nodes simply run the linear iteration (1) in order to obtain asymptotic convergence. However, each node stores its own values over the first $N+1$ time-steps of each run. Assuming that the initial conditions for the N asymptotic consensus runs were linearly independent, each node i can obtain its minimal polynomial $q_i(t)$ as described in this section. It then uses $q_i(t)$ to calculate $p_i(t) \equiv \frac{q_i(t)}{t-1}$, and obtains the vector Γ_i, whose components are given by equation (12). For future runs of the distributed consensus algorithm, node i runs the linear iteration for $D_i + 1$ time-steps, or until it receives the consensus value from a neighbor. If $D_i + 1$ time-steps pass without receiving the consensus value, node i can then

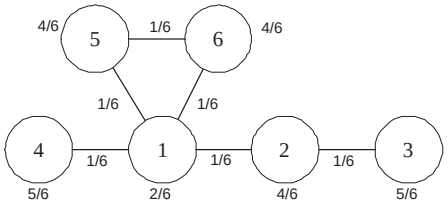

Fig. 2. Graph with constant edge weights of $\frac{1}{N}$.

immediately calculate the consensus value from (8). It then transmits this value to its neighbors with a flag indicating that it is the consensus value, and the algorithm terminates (for node i).

Example 2: Consider the graph shown in Fig. 2. None of the nodes know the entire graph topology, and there is no centralized decision maker to assign the edge weights or calculate the coefficient vectors. However, suppose each node knows that there are $N = 6$ nodes in the system. With this information, each node can select the constant edge weights in (14), and that produces a weight matrix W with $W\mathbf{1} = \mathbf{1}$ and $\frac{1}{6}\mathbf{1}'W = \frac{1}{6}\mathbf{1}'$ (i.e., this choice of weights performs distributed averaging). Since the nodes do not know their minimal polynomials yet, they simply use these weights to obtain asymptotic consensus for the first $N = 6$ runs. For each of these runs, each node stores its own value over the first $N + 1 = 7$ time-steps. It then finds the smallest D_i for which the matrix in (15) loses rank, and determines the coefficients of $q_i(t)$ from the nullspace of that matrix. In this way, the minimal polynomials for each i are found to be

$$\begin{aligned} q_1(t) &= q_2(t) = q_3(t) \\ &= t^4 - 2.6667t^3 + 2.4444t^2 - 0.8611t + 0.0833 \\ q_4(t) &= t^5 - 3.5t^4 + 4.667t^3 - 2.898t^2 + 0.801t - 0.0694 \\ q_5(t) &= q_6(t) = t^6 - 4t^5 + 6.4167t^4 - 5.2315t^3 + 2.25t^2 \\ &\quad - 0.4699t + 0.0347 \ . \end{aligned}$$

Thus, nodes 1, 2 and 3 can calculate the consensus value after running the linear iteration for four time-steps, node 4 can calculate the value after five time-steps, and nodes 5 and 6 can calculate it after six time-steps. Once the nodes have calculated their minimal polynomials, they calculate $p_i(t) = \frac{q_i(t)}{t-1}$, and use (12) with $\mu = 1$, $\mathbf{c} = \frac{1}{6}\mathbf{1}$ and $\mathbf{d} = \mathbf{1}$ to obtain

$$\begin{aligned} \Gamma_1 = \Gamma_2 = \Gamma_3 &= \begin{bmatrix} 36 & -60 & 28 & -3 \end{bmatrix}' \\ \Gamma_4 &= \begin{bmatrix} 216 & -540 & 468 & -158 & 15 \end{bmatrix}' \\ \Gamma_5 = \Gamma_6 &= \begin{bmatrix} 432 & -1296 & 1476 & -784 & 188 & -15 \end{bmatrix}' \ . \end{aligned}$$

At this point, the nodes can obtain the average of any initial conditions in the same way as in Example 1. After running the linear iteration for four time-steps, the first three nodes (i.e., $i \in \{1, 2, 3\}$) can immediately calculate the consensus value via (8). During the next time-step, node 1 would provide this value to nodes 4, 5 and 6, and so all nodes reach a consensus on the average of the initial values after five time-steps. □

VI. Discussion and Future Work

As we have seen from our development, once the matrix W is chosen to satisfy the conditions in Theorem 2, the nodes can obtain consensus after a finite number of time-steps (specifically, upper bounded by the degree of the minimal polynomial of W). This suggests that the optimality of a weight matrix W should be determined by the degree of its minimal polynomial, rather than the magnitude of its eigenvalues. However, finding the weight matrix with minimal polynomial of smallest degree for a given graph is an open problem [14]. Other interesting directions for future research would be to find more efficient methods for the nodes to distributively determine their minimal polynomials, and to extend our results to handle time-varying graphs. More generally, how does one incorporate robustness to errors in the nodes and links? We are actively pursuing these and other ideas in our research.

Finally, it is worth noting that the results in this paper can also be applied to continuous-time systems. If the evolution of the nodes is given by $\frac{d}{dt}x(t) = Wx(t)$, then for any W satisfying the conditions in Theorem 2, the nodes can obtain consensus in an arbitrarily short period of time by replacing $x_i[k]$ with $\frac{d^k}{dt^k}x_i(0)$ (the k'th derivative of x_i, evaluated at $t = 0$) in (8).

References

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

[2] 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*, 2005, pp. 1859–1864.

[3] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[4] 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 2005 European Control Conference*, 2005, pp. 2996–3000.

[5] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

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

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

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

[9] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, Nov. 2006.

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

[11] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd ed. Addison Wesley Longman, Inc., Menlo Park, CA., 1998.

[12] 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)*, 2005, pp. 63–70.

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

[14] F. Barioli and S. M. Fallat, "On two conjectures regarding an inverse eigenvalue problem for acyclic symmetric matrices," *Electronic Journal of Linear Algebra*, vol. 11, pp. 41–50, 2004.