Distributed Design of Finite-time Average Consensus Protocols

Thi-Minh Dung Tran, Alain Y. Kibangou

Distributed Design of Finite-time Average Consensus Protocols

Thi Minh Dung Tran Alain Y. Kibangou

Gipsa-Lab, CNRS, University Joseph Fourier,
11 rue des Mathématiques, Grenoble Campus, France.
(e-mail: thi-minh-dung.tran@gipsa-lab.fr, alain.kibangou@ujf-grenoble.fr).

Abstract: In this paper, we are interested in the finite-time average consensus problem for multi-agent systems or wireless sensor networks. This issue is formulated in a discrete-time framework by utilizing a linear iteration scheme, where each node repeatedly updates its value as a weighted linear combination of its own value and those of its neighbors. Unlike most of research in literature, this work deals with the foremost step, called configuration step, during which the consensus protocol is to be set up in each agent. Designing consensus protocols can be viewed as a matrix factorization problem. For connected undirected graphs, we propose a learning method for solving such matrix factorization problem in a distributed way. More precisely, we first show how solving this problem for the particular case of strongly regular graphs. Then, a distributed gradient back-propagation algorithm is derived for the general case. The performance of the proposed algorithm is evaluated by means of simulation results.

Keywords: Protocol design, finite-time average consensus, back-propagation algorithm, matrix factorization, self-configuration.

1. INTRODUCTION

The consensus issue in networks of autonomous agents has been widely investigated in various fields, including computer science and engineering. In such networks, according to an a priori specified rule, also called protocol, each agent updates its state based on the information received from its neighbors with the aim of reaching an agreement to a common value. When the common value corresponds to the average of the initial states, average consensus is to be achieved. Average consensus algorithms are commonly used as building block for distributed control, estimation or inference algorithms. In the recent literature, one can find average consensus algorithms embedded in the Distributed Kalman filter, Olfati-Saber (2007); Distributed Least Squares algorithm, Bolognani et al. (2008); Distributed Alternating Least Squares for tensors factorization, Kibangou and de Almeida (2010); Distributed Principal Component Analysis, Valcarel Macua et al. (2010); or distributed joint input and state estimation, Esna-Ashari et al. (2012), to cite few.

In order to run an average consensus algorithm, two main steps are required: the configuration step (also called design step) and the execution step. During the configuration step, the consensus protocol is to be uploaded in each agent. Such a task can be achieved through a self-configuration algorithm instead of resorting to a network manager. Self-configuration can include graph discovering and distributed decision on some parameters. For instance, if the protocol is the maximum-degree weights one, each agent first computes the number of its neighbors before running a max-consensus algorithm for computing the maximum degree of the underlying graph. In the case of the Metropolis-Hasting based protocol, also called local-degree weights, each agent compares its degree with that of its neighbors in order to compute the weights of the average consensus protocol. One commonly used protocol is the constant edge weights, or graph Laplacian based average consensus protocol, where a common step-size is used by all the agents. Asymptotic convergence is then guaranteed if the step-size is strictly positive and lower than $2/\lambda_1(\mathbf{L})$ where $\lambda_1(\mathbf{L})$ stands for the largest graph Laplacian eigenvalue. Even though there are some simple bounds that give choices for the step-size without requiring exact knowledge of the Laplacian spectrum, agents have to agree in an adequate step-size. To the best to our knowledge, there is no paper dealing with self-configuration protocols for the constant edge weights based average consensus protocol. It is worth noting that the step-size influences on the speed of convergence of the average consensus algorithm when using constant edge weights. That's why some recent works have been devoted to accelerating the speed of convergence of consensus algorithms by solving some optimization problems in a centralized way; the goal being to reduce the spectral gap between the matrix of weights and the average consensus matrix $\mathbf{J}_N = \frac{1}{N}\mathbf{1}\mathbf{1}^T$, where N stands for the number of agents of the network, Xiao and Boyd (2004); Kokiopoulou and Frossard (2009).

Since average consensus can be embedded in more sophisticated distributed algorithms, protocols that guarantee a minimal execution time are more appealing than those ensuring asymptotic convergence. For this purpose, several contributions dedicated to finite-time consensus have been recently published in the literature. In most of them, in particular those formulated in a continuous-time framework, the goal is to agree on a common value in finite-time without necessarily specifying such a common value, Cortes (2006); Wang and Xiao (2010); Shaofu Yang and Lu (2012). Finite-time average consensus protocols have been mainly proposed in a discrete-time framework. In Sundaram and Hadjicostis (2007), a finite-time consensus algorithm based on the minimal polynomial of the weight matrix was proposed. However, the design of this protocol requires a strong knowledge on the underlying

network topology. Therefore, a decentralized calculation of the minimal polynomial was proposed. An improvement of this method has been proposed in Yuan et al. (2013). The author has proposed algorithms to compute the consensus value using the minimal number of observations of an arbitrary chosen node in a network. However, the computation of the rank of a given matrix and that of its kernel are needed. Therefore, the computational cost is the weakness of these methods. In Ko (2010) and Georgopoulos (2011), the finite-time average consensus was formulated as a matrix factorization problem. The resulting solution yields a link scheduling on the complete graph to achieve finite time consensus. Such a scheduling is to be controlled by a central node. Following the idea of matrix factorization, Laplacian based joint diagonalizable matrices were suggested in Kibangou (2011, 2012). The proposed solutions make use of the graph Laplacian spectrum. However, the implementation of these protocols during the configuration step was not really discussed. In fact, for self-configuration of Laplacian based finite-time average consensus protocols, distributed estimation of Laplacian eigenvalues is required. Such a task can be carried out by means of distributed or decentralized algorithms such as those proposed in Tran and Kibangou (2013), Sahai et al. (2012), and Franceschelli et al. (2009).

The purpose of this paper is to study a new algorithm for self-configuration protocols for finite-time average consensus where weighted matrices are not necessarily based on the graph Laplacian. The aim is to design protocols that allow achieving average consensus in the fastest possible time, possibly as fast as the diameter of the underlying graph. More precisely, we solve a matrix factorization problem in a distributed way.

The remainder of this paper is organized as follows: in Section 2, we give the preliminary views of the theory and formulate the problem under study. The particular case of strongly regular graphs is studied in Section 3. Then, a gradient backpropagation algorithms is derived in Section 4 for solving a matrix factorization problem in a distributed way. The performance of the proposed algorithm is evaluated in Section 5 by means of simulation results before concluding the paper.

2. PROBLEM STATEMENT

Through out this paper, we consider a connected undirected graph $G(V,E)$, where $V=\{v_1,v_2,\dots v_N\}$ is the set of vertices of graph G, and $E\subset V\times V$ is the set of edges. Vertices v_i are agents in a network connected according to E. Before formulating the problem, we first state some basic notations:

- $N_i=\{v_j\in V:(v_i,v_j)\in E\}$ stands for the set of neighbors of node v_i.
- Given two vertices v_i and v_j, the distance $dist(v_i,v_j)$ is the length of the shortest path between v_i and v_j.
- The eccentricity $\text{ecc}(v_i)$ of a vertex v_i is the greatest distance between v_i and any other vertex $v_j\in V$.
- The radius $r(G)$ of a graph is the minimum eccentricity of any vertex.
- The diameter $d(G)$ of a graph is the maximum eccentricity of any vertex in the graph, i.e. $d(G)=\max_{v_i,v_j} dist(v_i,v_j)$.
- We denote by $\mathbf{A}$ the adjacency matrix of the graph. Its entries $A_{i,j}$ being equal to one if $(v_i,v_j)\in E$ and zero elsewhere.
- The graph Laplacian $\mathbf{L}$ is defined as the matrix with entries l_{ij} given by: $l_{ij}=\begin{cases}\sum_{k=1,k\neq i}^{N} A_{ik}, & \text{if} \quad j=i\\ -A_{i,j} & \text{if} \quad j\neq i\end{cases}$.
- A regular graph is a graph where each vertex has the same number of neighbors; i.e. every vertex has the same degree or valency. A regular graph with vertices of degree k is called a $k-$regular graph.

Now, let us state the finite-time average consensus problem.

For each agent $v_i\in V$, let $x_i(t)$ be its state at time-step t. At each time-step each node updates its state as

$$x_i(t)=w_{ii}^t x_i(t-1)+\sum_{j\in N_i} w_{ij}^t x_j(t-1). \tag{1}$$

Defining the state of the network as:

$\mathbf{x}(t)=[x_1(t),x_2(t),\dots,x_N(t)]^T$, where N is the number of nodes in the network, the dynamics of the network is given as follows:

$$\mathbf{x}(t)=\mathbf{W}_t\mathbf{x}(t-1),\quad t=1,2,\dots, \tag{2}$$

where $\mathbf{W}_t$, with entries w_{ij}^t, is consistent with the graph topologies, i.e. $\mathbf{W}_t\in\mathscr{S}_G$ where $\mathscr{S}_G$ is the set of matrices that can be factorized as $\mathbf{W}_t=\mathbf{Q}_t\circ(\mathbf{I}_N+\mathbf{A})$ where $\mathbf{Q}_t$ stands for an arbitrary square matrix, $\mathbf{I}_N$ being the $N\times N$ identity matrix, whereas $\circ$ denotes the Hadamard matrix product that corresponds to an entry-wise matrix product.

Given any set of initial values $\mathbf{x}(0)$, we are interested in a finite sequence of weighted matrices, $\mathbf{W}_t\in\mathscr{S}_G$, that allows all agents to reach average consensus in a finite number of steps (or finite-time) D, i.e. $\mathbf{x}(D)=\frac{1}{N}\mathbf{1}\mathbf{1}^{\mathbf{T}}\mathbf{x}(0)=\mathbf{J}_N\mathbf{x}(0)$. Ultimately, we desire a finite sequence of matrices $\{\mathbf{W}_1,\mathbf{W}_2,\dots,\mathbf{W}_D\}$ such that

$$\mathbf{x}(D)=\prod_{t=D}^{1}\mathbf{W}_t\mathbf{x}(0)=\mathbf{J}_N\mathbf{x}(0)\quad \text{for all } \mathbf{x}(0)\in R^N, \tag{3}$$

meaning that

$$\prod_{t=D}^{1}\mathbf{W}_t=\mathbf{J}_N. \tag{4}$$

Within this framework the fundamental problems concern existence, minimality, and design issues: For a given graph G, does there exist a finite sequence of matrices $\mathbf{W}_t\in\mathscr{S}_G$ such that (4) is achieved ? If it exists, what it is the minimal value of D ? How can we carry out such a factorization?

The existence issue has been deeply considered in Ko (2010) and Georgopoulos (2011). It has been pointed out that no solution exists if the factor matrices $\mathbf{W}_t$ are all equal except if the graph is complete. For trees and graphs with minimum diameter spanning tree, solutions have been also provided. For more general graphs, solutions based on graph Laplacian have been recently introduced in closed-form, Kibangou (2011) and Kibangou (2012). The factors matrices are constrained to be equal to $\mathbf{W}_t=\alpha_t\mathbf{I}+\beta_t\mathbf{L}$ where α_t and β_t are parameters to be designed. More precisely, in Kibangou (2012), the solution was given by $\alpha_t=1$ and $\beta_t=-\frac{1}{\lambda_{t+1}(\mathbf{L})}$, $\lambda_t(\mathbf{L})$ being a nonzero Laplacian eigenvalue. Therefore, the number of factor matrices is equal to the number of distinct nonzero Laplacian eigenvalues. Intuitively, since the diameter of the graph characterizes the time necessary for a given information to reach all the agents in the network, the number of factor matrices cannot be lower than the diameter $d(G)$. According to the results in Kibangou (2012), the number D is upper bounded by $N-1$. For distance regular graphs, Brouwer et al. (1989), $D=d(G)$. In Hendrickx

et al. (2012), an upper bound is given by $2r(G)$ where $r(G)$ stands for the radius of the graph.

Now, for a given value of D lower bounded by the diameter of the graph and upper bounded by two times the radius of the graph, how could we compute the factor matrices without the Laplacian constraint? Strictly speaking, equation (4) gives rise to a system of multivariate polynomial equations, which can be solved using Groebner basis theory and Buchberger's algorithm, in Buchberger (1976). However, the computational complexity and centralized nature of such an approach is crippling.

In what follows, we first consider the particular case of strongly regular graphs where closed form solutions of the factorization problem can be obtained in a distributed way, Godsil and Royle (2001). Then, for the general case, we design a distributed method that makes use of learning sequences. Indeed, in most systems where communications are involved, learning sequences are used for communication channel identification or for synchronization. These sequences are used during the mandatory configuration step before transmitting informative data, i.e. running average consensus in our case. We assume that all the agents know the average of the learning sequence.

3. CLOSED FORM SOLUTION OF THE MATRIX FACTORIZATION PROBLEM FOR STRONGLY REGULAR GRAPHS

A graph G is said to be strongly regular, SRG(N,k,a,c), if it is neither complete nor empty and there are integers k, a, and c such that:

- G is regular with valency k;
- any two adjacent vertices have exactly a common neighbors;
- any two distinct non-adjacent vertices have exactly c common neighbors.

These parameters are linked as follows:

$$(N-k-1)c = k(k-a-1). \tag{5}$$

In addition, the following interesting property can be stated as in Godsil and Royle (2001):

$$\mathbf{A}^2 - (a-c)\mathbf{A} - (k-c)\mathbf{I}_N = Nc\mathbf{J}_N. \tag{6}$$

Examples of strongly regular graphs are: Petersen graph ($SRG(10,3,0,1)$), Clebsch graph ($SRG(16,5,0,2)$), Payley graphs ($SRG(q,\frac{q-1}{2},\frac{q-5}{4},\frac{q-1}{4})$,with q congruent 1 (mod 4)), two dimensional Hamming graphs ($SRG(n^2,2n-2,n-2,2)$, $n \geq 2$), a cycle with $N < 6$. The most obvious property of the strongly regular graphs is that the diameter is equal to two ($d(G) = 2$).

Let us state the following theorem:

Theorem 1. Let γ_2 be a nonzero free parameter. For a strongly regular graph SRG(N,k,a,c), average consensus can be reached in two steps with the weights given by the matrices $\mathbf{W}_i = w_i\mathbf{I}_N + \gamma_i\mathbf{A}$, $i = 1,2$ with $\gamma_1\gamma_2 = \frac{1}{cN}$, $w_1w_2 = \frac{1}{N}\left(1-\frac{k}{c}\right)$, and

$$w_2 = \frac{\gamma_2}{2}\left(c-a \pm \sqrt{(a-c)^2+4(k-c)}\right).$$

Proof: With matrices $\mathbf{W}_i = w_i\mathbf{I}_N + \gamma_i\mathbf{A}$, $i = 1,2$, we get: $\mathbf{W}_2\mathbf{W}_1 = w_2w_1\mathbf{I}_N + (w_2\gamma_1 + w_1\gamma_2)\mathbf{A} + \gamma_1\gamma_2\mathbf{A}^2$. For a strongly regular graph SRG(N,k,a,c), using (6), we get

$$\begin{aligned}\mathbf{W}_2\mathbf{W}_1 = {} & Nc\gamma_1\gamma_2\mathbf{J}_N + (w_2w_1 + (k-c)\gamma_1\gamma_2)\mathbf{I}_N \\ & + ((w_2\gamma_1 + w_1\gamma_2) + \gamma_1\gamma_2(a-c))\mathbf{A}.\end{aligned} \tag{7}$$

As a consequence, the matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ are factors of $\mathbf{J}_N$ if and only if $w_2w_1 + (k-c)\gamma_1\gamma_2 = 0$, $(w_2\gamma_1 + w_1\gamma_2) + \gamma_1\gamma_2(a-c) = 0$, and $Nc\gamma_1\gamma_2 = 1$. After few manipulations, we get the following equation to solve: $Nc\gamma_1w_2^2 + (a-c)w_2 - \gamma_2(k-c) = 0$, whose discriminant $\Delta = (a-c)^2 + 4(k-c)$ is strictly positive. Meaning that for any nonzero value of the free parameter γ_1 we can always get a factorization of $\mathbf{J}_N$ with factors in the form $\mathbf{W}_i = w_i\mathbf{I}_N + \gamma_i\mathbf{A}$. ■

Assuming that the structural property is known, the agents can determine the factor matrices in a distributed way. Indeed, the number a of common parameters can be easily computed by comparing the ID of neighbors then, using (5), parameter c can be deduced as $c = \frac{k(k-a-1)}{N-k-1}$. Agreement on γ_2, after a random selection, can be reached using a max or a min consensus [1] .

Example: Considering a Petersen Graph ($SRG(10,3,0,1)$). Using (6), we get the equation: $\frac{1}{10}\mathbf{A}^2 + \frac{1}{10}\mathbf{A} - \frac{1}{5}\mathbf{I}_N = \mathbf{J}_N$. With an arbitrary γ_2, by Theorem 1 , we can compute:

$$w_2 = \frac{\gamma_2}{2}(1 \pm 3); w_1 = -\frac{1}{5w_2}; \gamma_1 = \frac{1}{10\gamma_2}.$$

Then, these results give us $\mathbf{W}_1, \mathbf{W}_2$ as follows:

For $w_2 = 2\gamma_2$:

$$\mathbf{W}_1 = -\frac{1}{10\gamma_2}\mathbf{I}_N + \frac{1}{10\gamma_2}\mathbf{A}; \mathbf{W}_2 = 2\gamma_2\mathbf{I}_N + \gamma_2\mathbf{A}.$$

For $w_2 = -\gamma_2$:

$$\mathbf{W}_1 = \frac{1}{5\gamma_2}\mathbf{I}_N + \frac{1}{10\gamma_2}\mathbf{A}; \mathbf{W}_2 = -\gamma_2\mathbf{I}_N + \gamma_2\mathbf{A}.$$

In both cases, we can easily check that:

$$\mathbf{W}_2\mathbf{W}_1 = -\frac{1}{5}\mathbf{I}_N + \frac{1}{10}\mathbf{A} + \frac{1}{10}\mathbf{A}^2 = \mathbf{J}_N.$$

Deriving closed form solutions is not always possible. Therefore, in next section, we derive a distributed solution of the matrix factorization problem for more general graphs.

4. DISTRIBUTED SOLUTION OF THE MATRIX FACTORIZATION PROBLEM

Let $\{x_{i,p}(0), y_{i,p}\}$, $i = 1,\cdots,N$, $p = 1,\cdots,P$, be the input-output signals defining the learning sequences, with $y_{i,p} = y_p = \frac{1}{N}\sum_{i=1}^{N} x_{i,p}(0)$. Our aim is to estimate the factor matrices $\mathbf{W}_t$, $t = 1,\cdots,D$, by minimizing the quadratic error

$$E(\mathbf{W}_1,\cdots,\mathbf{W}_D) = \frac{1}{2}\sum_{i=1}^{N}\sum_{p=1}^{P}(x_{i,p}(D) - y_p)^2, \tag{8}$$

with $x_{i,p}(t) = \sum_{j\in N_i\cup\{i\}} w_{ij}^t x_{j,p}(t-1)$, w_{ij}^t being the entries of matrices $\mathbf{W}_t$. We can rewrite the cost function (8) as

$$E(\mathbf{W}_1,\cdots,\mathbf{W}_D) = \frac{1}{2}\left\|\prod_{t=D}^{1}\mathbf{W}_t\mathbf{X}(0) - \mathbf{Y}\right\|_F^2, \tag{9}$$

where $\|.\|_F$ stands for the Frobenius norm, $\mathbf{Y} = \mathbf{J}_N\mathbf{X}(0)$, $\mathbf{Y}$ and $\mathbf{X}(0)$ being $N \times P$ matrices with $y_{i,p}$ and $x_{i,p}$ as entries, respectively. We assume that $\mathbf{X}(0)\mathbf{X}(0)^T = \mathbf{I}_N$ which means

[1] Max and Min consensus can be reached in a finite number of steps, Olfati-Saber and Murray (2004)

that the input vector is orthogonal. For instance, vectors of the canonical basis of $\Re^N$ can be used as inputs. Hence, we can note that $E(\mathbf{W}_1,\cdots,\mathbf{W}_D)=\frac{1}{2}\left\|\prod_{t=D}^{1}\mathbf{W}_t-\mathbf{J}_N\right\|_F^2$, meaning that minimizing (9) is equivalent to solving the factorization problem (4):

$$\{\mathbf{W}_t^*\}|_{t=1,\ldots,D}=\arg\min_{\{\mathbf{W}_t\}_{t=1,\ldots,D}}\frac{1}{2}\sum_{p=1}^{P}tr\left(\varepsilon_p(\mathbf{W})\varepsilon_p^T(\mathbf{W})\right), \quad (10)$$

with

$$\varepsilon_p(\mathbf{W})=\prod_{t=D}^{1}\mathbf{W}_t\mathbf{x}_p(0)-\mathbf{y}_p, \quad (11)$$

where $tr(.)$ denotes the trace operator and $\mathbf{y}_p=\mathbf{J}_N\mathbf{x}_p(0)$.

Denoting $E_p(\mathbf{W})=\frac{1}{2}tr\left(\varepsilon_p(\mathbf{W})\varepsilon_p^T(\mathbf{W})\right)$, the solution of this optimization problem can then be obtained iteratively by means of a gradient descent method:

$$\mathbf{W}_t:=\mathbf{W}_t-\alpha\sum_{p=1}^{P}\frac{\partial E_p(\mathbf{W})}{\partial\mathbf{W}_t}$$

or a stochastic gradient one:

$$\mathbf{W}_t:=\mathbf{W}_t-\alpha\frac{\partial E_p(\mathbf{W})}{\partial\mathbf{W}_t},$$

where the gradient of the cost function is approximated by the gradient at a single input-output sequence. In what follows, we resort to a stochastic gradient method. For this purpose, we first state the following technical lemma:

Lemma 1. The derivatives of the cost function $E_p(\mathbf{W})=\frac{1}{2}tr\left(\varepsilon_p(\mathbf{W})\varepsilon_p^T(\mathbf{W})\right)$ with $\varepsilon_p(\mathbf{W})$ defined in (11) can be computed as follows:

$$\frac{\partial E_p(\mathbf{W})}{\partial\mathbf{W}_t}=\boldsymbol{\delta}_{t,p}\mathbf{x}_p^T(t-1), t=1,\ldots,D \quad (12)$$

$$\frac{\partial E_p(\mathbf{W})}{\partial\mathbf{W}_D}=\boldsymbol{\delta}_{D,p}\mathbf{x}_p^T(D-1) \quad (13)$$

where $\boldsymbol{\delta}_{D,p}=\mathbf{x}_p(D)-\bar{\mathbf{x}}_p$ is the difference between the actual output and the desired output with $\bar{\mathbf{x}}_p=\mathbf{y_p}=\frac{1}{N}\sum_{i=1}^{N}x_{i,p}(0)$; and

$$\boldsymbol{\delta}_{t-1,p}=\mathbf{W}_t^T\boldsymbol{\delta}_{t,p}, t=1,\ldots,D. \quad (14)$$

Proof. See Appendix.■

Applying the results of Lemma 1, the updating scheme of the optimization algorithm is as follows:

$$\begin{aligned}\mathbf{W}_t[m+1]&=\mathbf{W}_t[m]-\alpha\frac{\partial E_{p(m)}(\mathbf{W})}{\partial\mathbf{W}_t}\\&=\mathbf{W}_t[m]-\alpha\boldsymbol{\delta}_{t,p(m)}\mathbf{x}_{p(m)}^T(t-1),\end{aligned} \quad (15)$$

where $p(m)\in\{1,2,\cdots,P\}$, and m stands for the m^{th} iteration of the optimization process.

The gradient descent algorithm (15) acts by alternating the following steps: the learning sequence is first propagated forward. Then the error between the targeted output and $\mathbf{x}(D)$ is computed and then propagated . This mechanism is similar to the gradient backpropagation algorithm. Its convergence has been well studied in the literature, see Chong and Zak (2012) and Mangasarian and Solodov (1994) for instance. Convergence towards a local minimum is guaranteed if the step-size α is appropriately chosen ($0<\alpha<1$). Such a step-size also influences on speed of convergence of the algorithm. Several rules have been proposed for accelerating the convergence speed. However, these techniques are not consistent with the distributed framework of the proposed algorithm. One trick to speed-up the convergence is to add a regularization term in the cost function to be minimized.

$$\begin{aligned}\{\mathbf{W}_t^*\}|_{t=1,\ldots,D}=\arg\min_{\{\mathbf{W}_t\}_{t=1,\ldots,D}}&\sum_{p=1}^{P}E_{p(m)}(\mathbf{W})\\&+\frac{1}{2}\sum_{t=1}^{D}\beta\|\mathbf{W}_t[m]-\mathbf{W}_t[m-1]\|^2.\end{aligned} \quad (16)$$

By minimizing such a cost function, the udpate equation is given by:

$$\begin{aligned}\mathbf{W}_t[m+1]=\mathbf{W}_t[m]&-\alpha\boldsymbol{\delta}_{t,p(m)}\mathbf{x}_{p(m)}^T(t-1)\\&+\beta(\mathbf{W}_t[m]-\mathbf{W}_t[m-1]).\end{aligned}$$

Entry-wise, each agent updates its entries as follows:

$$w_{ij}^t:=w_{ij}^t-\alpha\delta_{i,t}x_j(t-1)+\beta(w_{ij}^t-w_{ij}^{(t-1)}),$$

where $\delta_{i,t}$ is the ith entry of $\boldsymbol{\delta}_t$ and $x_j(t-1)$ the j-th entry of $\mathbf{x}(t-1)$. α and β are learning rate and momentum rate respectively.

The distributed algorithm is then described as follows:

(1) *Initialization*:
- Number of steps D, number of patterns P
- Learning sequence $\{x_{i,p}(0),y_p\}$, where $i=1,\cdots,N$, $p=1,\cdots,P$, with $y_p=\frac{1}{N}\sum_{i=1}^{N}x_{i,p}(0)$.
- Random initial weighted matrices $\mathbf{W}_t[0], t=1,\ldots,D$, and $\mathbf{W}_t[-1]=0$
- Learning rate: $0<\alpha<1$;
- Momentum terms: $0<\beta$<1;
- Select a threshold γ
- Set $m=0$

(2) Set $p=0$;
- (a) Set $p:=p+1$,
- (b) Select the corresponding input-output sequence $x_i(0)=x_{i,p}(0)$, $\bar{x}=y_p$.
- (c) Learning sequence propagation:
$$x_i(t)=\sum_{j\in N_i\cup\{i\}}w_{ij}^t[m]x_j(t-1),\quad t=1,\cdots,D.$$
- (d) Error computation:
$$\delta_{i,D}=x_i(D)-\bar{x};\quad e_{i,p}=\delta_{i,D}^2.$$
- (e) Error propagation:
$$\delta_{i,t-1}=\sum_{j\in N_i\cup\{i\}}w_{ji}^t[m]\delta_{j,t},\quad t=D,\cdots,2.$$
- (f) Matrices updating: for $t=1,\cdots,D$, $i=1,\cdots,N$, and $j\in N_i\cup\{i\}$,
$$\begin{aligned}w_{ij}^t[m+1]=w_{ij}^t[m]&-\alpha\delta_{i,t}x_j(t-1)\\&+\beta(w_{ij}^t[m]-w_{ij}^t[m-1]).\end{aligned}$$
- (g) Increment m.
- (h) If $p=P$, compute the mean square error $E_i=\frac{1}{N}\sum_{p=1}^{P}e_{i,p}$, else return to 2a.

(i) If $E_i < \gamma$ stop the learning process else return to 2.

5. SIMULATION RESULTS

In this section, we consider two examples of different network topologies to evaluate the developed algorithm. The learning sequence is constituted with the vectors of the canonical basis of $\Re^N$.

The performance of designed protocols are evaluated by means of the MSE:

$$MSE = \frac{1}{NP}\sum_{i=1}^{N}\sum_{p=1}^{P}(x_{i,p}(D) - y_p)^2.$$

5.1 Example 1:

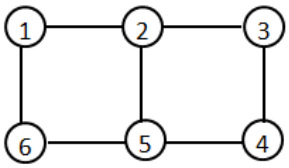


Fig. 1. 6-node graph

Let us consider the graph depicted in Figure 1. Its diameter is $d(G) = 3$ while the radius is $r(G) = 2$. The eigenvalues of the corresponding Laplacian matrix $\mathbf{L}$ are: 0, 1, 2, 3 and 5. According to Kibangou (2012), the factorization of the average consensus matrix by means of graph Laplacian based consensus matrices needs D factor matrices, D being the number of nonzero distinct eigenvalues of the Laplacian matrix, this number being, in general, greater or equal to the diameter of the graph. Therefore, we get the following factorization:

$$(\mathbf{I}_N - \mathbf{L})(\mathbf{I}_N - \frac{1}{2}\mathbf{L})(\mathbf{I}_N - \frac{1}{3}\mathbf{L})(\mathbf{I}_N - \frac{1}{5}\mathbf{L}) = \mathbf{J}_6,$$

meaning that consensus is achieved in 4 steps. Since the optimal number of steps is not a priori known, we run the algorithm for different values of D taken in the interval of $[d(G), 2r(G)]$. Here, the two possible values of D are 3 and 4.

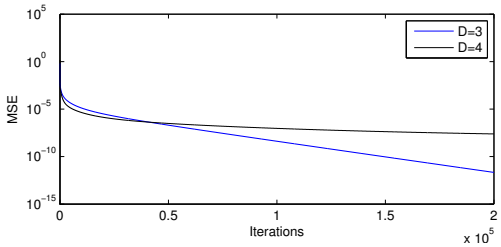


Fig. 2. MSE comparison for the two possible numbers of factors

Figures 2 shows that the best solution is obtained for $D = 3$. With the proposed algorithm for design protocol we get a number of step lower than that given by Laplacian based matrices proposed in Kibangou (2011, 2012). The weighted matrices are:

$$\mathbf{W}_1 = \begin{pmatrix} -0.4028 & 0.0415 & 0 & 0 & 0 & 0 \\ 0.4418 & -0.0455 & 0 & 0 & 0 & 0 \\ 0.5749 & 0 & 0.0537 & 0.0537 & 0.9219 & 0 \\ 0 & 0.4053 & 0.3675 & 0.3675 & 0 & 0.3902 \\ 0 & 0 & 0 & 0 & -1.3034 & 0.0806 \\ 0 & 0 & 0 & 0 & 0.1122 & -0.0069 \end{pmatrix}$$

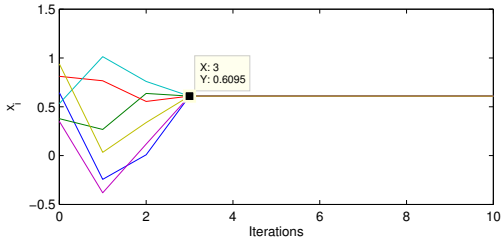
$$\mathbf{W}_2 = \begin{pmatrix} 0.7767 & 0.2508 & 0.1706 & 0 & 0 & 0 \\ 0.0201 & 0.6826 & 0 & 0.4527 & 0 & 0 \\ 0.6941 & 0 & 0.7680 & 0.2985 & 0.4385 & 0 \\ 0 & 0.2560 & 0.1578 & 0.5424 & 0 & 0.6013 \\ 0 & 0 & 0.3756 & 0 & 0.5232 & 0.8876 \\ 0 & 0 & 0 & 0.2684 & -0.1270 & 0.5548 \end{pmatrix}$$

$$\mathbf{W}_3 = \begin{pmatrix} 0.7429 & 0.6296 & 0.3653 & 0 & 0 & 0 \\ 0.2881 & 0.2741 & 0 & 0.5699 & 0 & 0 \\ 0.7055 & 0 & 0.6403 & 0.2695 & 0.3771 & 0 \\ 0 & 0.4246 & 0.0716 & 0.1496 & 0 & 0.5491 \\ 0 & 0 & 0.4218 & 0 & 0.4556 & 0.9515 \\ 0 & 0 & 0 & 0.4958 & 0.3039 & 0.5832 \end{pmatrix}$$

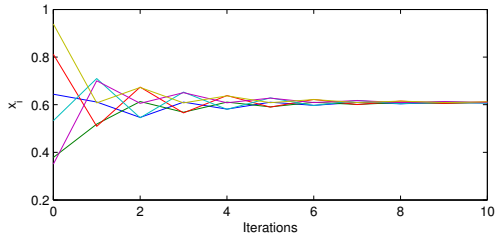
We can easily check that:

$$\mathbf{W}_3\mathbf{W}_2\mathbf{W}_1 = \mathbf{J}_6.$$

Now, let us consider the finite-time consensus protocol in its execution step, i.e. after the configuration step. For arbitrary initial values, the trajectories of the state of the network are depicted in Figure 3(a). Exact average consensus is achieved in 3 steps. When using the optimal constant edge weights protocol $\mathbf{W} = \mathbf{I}_N - \alpha\mathbf{L}$ where $\alpha = \frac{2}{\lambda_1(\mathbf{L})+\lambda_{N-1}(\mathbf{L})}$ proposed in Xiao and Boyd (2004), we get the trajectories in Figure 3(b). Showing that more iterations are actually needed to reach consensus.



(a)



(b)

Fig. 3. (a)Trajectory of a finite-time consensus protocol
(b)Trajectory of an asymptotic consensus protocol by using optimal constant edge weights protocol

5.2 Example 2:

Now considering the network in Figure 4, we can see that diameter of this graph equals to 2, and the radius of the graph equals to 2. Running the proposed algorithm, the performance in Figure 5 tells us that this network cannot achieve the average consensus in 2 steps, which is diameter of a graph. Such a result has been demonstrated analytically by Hendrickx et al. (2012). However, by running the algorithm as well as varying the value of D in the interval of $[d(G), 2r(G)]$, we can pick the optimal D in this case. The final MSE was obtained by considering a sufficient large number of iterations.

For this topology, $D = 3$ gives us the best solution for finite-time average consensus, see in Figure 5. Also, the trajectory of

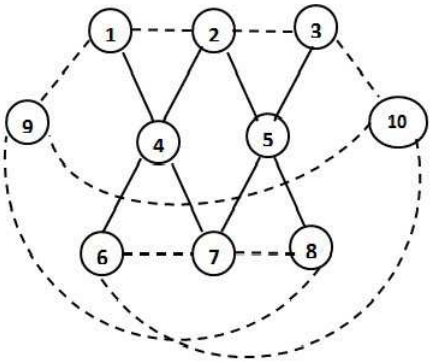


Fig. 4. 10-node graph

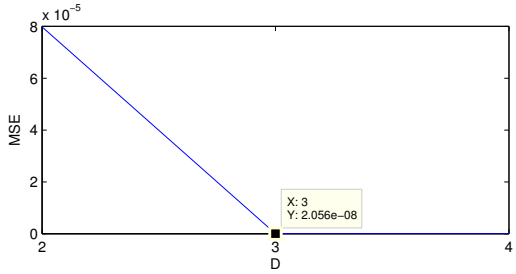


Fig. 5. Final MSE comparison for different values of D

an arbitrary vector of initial value is depicted in Figure 6 after getting the sequence of weighted matrices.

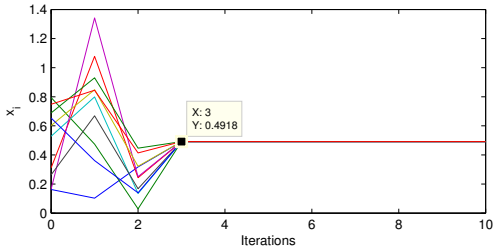


Fig. 6. Trajectory of an arbitrary vector for a 10-node network

6. CONCLUSION

In this paper, we have proposed a way for distributively designing the finite-time average consensus protocol. By using a learning sequence, we have shown how to solve a matrix factorization problem (4) in a fully distributed way. The method is based on the gradient back-propagation method. The factorization gives rise to factor matrices that are not necessarily symmetric or stochastic. Given the diameter and the radius of the graph, we can find out the optimal value of the number of steps necessary for reaching average consensus. However, the speed of convergence of the proposed algorithm is still an actual issue. For this purpose, future works encompass optimal step-size for gradient descent based method and other optimization methods with a particular focus to large size graphs. In addition, the dependency of the convergence speed on the learning sequence is to be studied. Hence, another interesting issue will be the design of optimal learning sequences. Beside that, the robustness of the finite-time average consensus should be definitely paid a great attention.

APPENDIX

The consensus network being a linear system we know that $\mathbf{x}_p(t) = \mathbf{W}_t\mathbf{x}_p(t-1)$, therefore we can explicitly write the output according to the weighting matrix of interest, i.e. $\mathbf{x}_p(D) = \mathbf{W}_D\mathbf{x}_p(D-1)$ and $\mathbf{x}_p(D) = \prod_{j=D}^{t+1}\mathbf{W}_j\mathbf{W}_t\mathbf{x}_p(t-1)$, $t = 1,\cdots,D-1$. Equivalently, by defining $\mathbf{Z}_{t+1} = \prod_{j=D}^{t+1}\mathbf{W}_j$, we get $\mathbf{x}_p(D) = \mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1)$. The cost function can be written as

$$E_p(\mathbf{W}) = \frac{1}{2}tr((\mathbf{W}_D\mathbf{x}_p(D-1) - \bar{\mathbf{x}}_p)(\mathbf{W}_D\mathbf{x}_p(D-1) - \bar{\mathbf{x}}_p)^T).$$

We can easily deduce that $\frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_D} = \boldsymbol{\delta}_{D,p}\mathbf{x}_p^T(D-1)$ with $\boldsymbol{\delta}_{D,p} = \mathbf{x}_p(D) - \bar{\mathbf{x}}_p$.

Now, we can express the cost function according to any matrix $\mathbf{W}_t$, $t = 1,\cdots,D-1$ as

$$E_p(\mathbf{W}) = \frac{1}{2}tr((\mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1) - \bar{\mathbf{x}}_p)(\mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1) - \bar{\mathbf{x}}_p)^T).$$

Expanding the above expression and taking into account the linearity of the trace operator yield

$$\begin{aligned} E_p(\mathbf{W}) = \frac{1}{2}[&tr(\mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1)\mathbf{x}_p(t-1)^T\mathbf{W}_t^T\mathbf{Z}_{t+1}^T) \\ &- tr(\mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1)\bar{\mathbf{x}}_p^T) \\ &- tr(\bar{\mathbf{x}}_p\mathbf{x}_p(t-1)^T\mathbf{W}_t^T\mathbf{Z}_{t+1}^T) \\ &- tr(\bar{\mathbf{x}}_p\bar{\mathbf{x}}_p^T)]. \end{aligned}$$

Now, computing the derivative, we get:

$$\begin{aligned} \frac{\partial E_p(\mathbf{W})}{\partial \mathbf{W}_t} &= \frac{1}{2}[2\times\mathbf{Z}_{t+1}^T\mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1)\mathbf{x}_p(t-1)^T \\ &\quad - 2\times\mathbf{Z}_{t+1}^T\bar{\mathbf{x}}_p\mathbf{x}_p(t-1)^T] \\ &= \mathbf{Z}_{t+1}^T(\mathbf{Z}_{t+1}\mathbf{W}_t\mathbf{x}_p(t-1) - \bar{\mathbf{x}})\mathbf{x}_p(t-1)^T \\ &= \mathbf{Z}_{t+1}^T\underbrace{(\mathbf{x}_p(D) - \bar{\mathbf{x}}_p)}_{\boldsymbol{\delta}_{D,p}}\mathbf{x}_p(t-1)^T \\ &= \mathbf{Z}_{t+1}^T\boldsymbol{\delta}_{D,p}\mathbf{x}_p(t-1)^T \\ &= \mathbf{W}_{t+1}^T\mathbf{W}_{t+2}^T\cdots\underbrace{\mathbf{W}_D^T\boldsymbol{\delta}_D}_{\boldsymbol{\delta}_{D-1,p}}\mathbf{x}_p(t-1)^T \\ &= \mathbf{W}_{t+1}^T\boldsymbol{\delta}_{t+1,p}\mathbf{x}_p(t-1)^T \\ &= \boldsymbol{\delta}_{t,p}\mathbf{x}(t-1)^T. \end{aligned}$$

REFERENCES

Bolognani, S., Del Favero, S., Schenato, L., and Varagnolo, D. (2008). Distributed sensor calibration and least-squares parameter identification in WSNs using consensus algorithms. In *Proc. of 46th annual Allerton Conference*, 1191–1198. Allerton House, UIUC, Illinois, USA.

Brouwer, A.E., Cohen, A.M., and Neumaier, A. (1989). *Distance-Regular Graphs*. Springer.

Buchberger, B. (1976). Some properties of Groebner-bases for polynomial ideals. *ACM SIGSAM Bulletin*, 10, 19–24.

Chong, E.K.P. and Zak, S.H. (2012). *An Introduction to Optimization*. John Wiley & Sons, third edition.

Cortes, J. (2006). Finite-time convergent gradient flows with application to network consensus. *Automatica*, 42(11), 1993–2000.

Esna-Ashari, A., Kibangou, A., and Garin, F. (2012). Distributed input and state estimation for linear discrete-time systems. In *Proc. 51st IEEE Conf. on Decision and Control (CDC)*. Maui, Hawaii, USA.

Franceschelli, M., Gasparri, A., Giua, A., and Seatzu, C. (2009). Decentralized Laplacian eigenvalues estimation for networked multi-agent systems. In *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, 2717–2722. Shanghai, P.R. China.

Georgopoulos, L. (2011). *Definitive consensus for distributed data inference*. Ph.D. thesis, EPFL, Lausanne.

Godsil, C. and Royle, G. (2001). *Algebraic graph theory*. Springer.

Hendrickx, J., Jungers, R., Olshevsky, A., and Vankeerberghen, G. (2012). Diameter, optimal-time consensus, and graph eigenvalues. Eprint arXiv:1211.6324.

Kibangou, A. (2011). Finite-time average consensus based protocol for distributed estimation over awgn channels. In *Proc. of the 50th IEEE Conference on Decision and Control (CDC)*, 261–265. Orlando, Fl, USA.

Kibangou, A. (2012). Graph Laplacian based matrix design for finite-time distributed average consensus. In *Proc. of the American Conference on Control(ACC)*, 261–265. Montréal, Canada.

Kibangou, A. and de Almeida, A. (2010). Distributed PARAFAC based DS-CDMA blind receiver for wireless sensor networks. In *Proc. of the IEEE Workshop SPAWC*. Marrakech, Morocco.

Ko, C.K. (2010). *On matrix factorization and scheduling for finite-time average consensus*. Ph.D. thesis, California Institue of Technology, Pasadena, California, USA.

Kokiopoulou, E. and Frossard, P. (2009). Polynomial filtering for fast convergence in distributed consensus. *IEEE Trans. on Signal Processing*, 57, 342–354.

Mangasarian, O.L. and Solodov, M.V. (1994). Serial and parallel backpropagation convergence via nonmonotone perturbed minimization (1994). *Optimzation Methods and Software*, 4, 103–116.

Olfati-Saber, R. (2007). Distributed kalman filtering for sensor networks. In *Proc. of the 46th IEEE Conf. on Decision and Control*, 5492–5498. New Orleans, LA, USA.

Olfati-Saber, R. and Murray, R. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Automatic Control*, 49, 1520–1533.

Sahai, T., Speranzon, A., and Banaszuk, A. (2012). Hearing the clusters of a graph: A distributed algorithm. *Automatica*, 48(1), 15–24.

Shaofu Yang, J.C. and Lu, J. (2012). A new protocol for finite-time consensus of detail-balanced multi-agent networks. *CHAOS*, 22.

Sundaram, S. and Hadjicostis, C. (2007). Finite-time distributed consensus in graphs with time-invariant topologies. In *Proc. of American Control Conference (ACC)*. New York City, USA.

Tran, T.M.D. and Kibangou, A. (2013). Consensus-based distributed estimation of laplacian eigenvalues of undirected graphs. In *Proc. of the European Control Conference*. Zurich, Switzerland.

Valcarel Macua, S., Belanovic, P., and Zazo, S. (2010). Consensus-based distributed principal component analysis in wireless sensor networks. In *Proc. of the IEEE Workshop SPAWC*. Marrakech, Morocco.

Wang, L. and Xiao, F. (2010). Finite-time consensus problems for networks of dynamic agents. *IEEE Trans. on Automatic Control*, 55(4), 950–955.

Xiao, L. and Boyd, S. (2004). Fast linear iterations for distributed averaging. *Systems Control Lett.*, 53, 65–78.

Yuan, Y., Stan, G., Shi, L., Barahona, M., and Goncalves, J. (2013). Decentralized minimim-time consensus. *Automatica 49*, 1227–1235.