sensors

Article

In-Network Computation of the Optimal Weighting Matrix for Distributed Consensus on Wireless Sensor Networks

Xabier Insausti *, Jesús Gutiérrez-Gutiérrez, Marta Zárraga-Rodríguez and Pedro M. Crespo

Department of Biomedical Engineering and Sciences, Tecnun, University of Navarra, Manuel Lardizábal 13, 20018 San Sebastián, Spain; jgutierrez@tecnun.es (J.G.-G.); mzarraga@tecnun.es (M.Z.-R.); pcrespo@tecnun.es (P.M.C.)

* Correspondence: xinsausti@tecnun.es; Tel.: +34-943-219-877 (ext. 2867)

Received: 17 May 2017; Accepted: 21 July 2017; Published: 25 July 2017

Abstract: In a network, a distributed consensus algorithm is fully characterized by its weighting matrix. Although there exist numerical methods for obtaining the optimal weighting matrix, we have not found an in-network implementation of any of these methods that works for all network topologies. In this paper, we propose an in-network algorithm for finding such an optimal weighting matrix.

Keywords: consensus; distributed computation; networks

1. Introduction

A sensor is a device capable of measuring a certain physical property. Normally, in a wireless sensor network (WSN), each sensor or node can transmit and receive data wirelessly, and it has the ability of performing multiple tasks, which are usually based on simple mathematical operations such as additions and multiplications. Moreover, the sensors within a WSN are usually powered with batteries, leading to very limited energy resources.

For most tasks, it is required that each sensor computes a target value that depends on the values measured by other sensors of the WSN. Commonly, a WSN has a central entity, known as the central node, which collects the values measured by all the sensors, computes the target values, and sends each target value to the corresponding sensor. This strategy is known as centralized computation.

The main disadvantage of the centralized computation strategy is that it is extremely energy inefficient from the transmission point of view because, when a sensor is far away from the central node, it has to consume disproportionate amounts of energy, with respect to the energy provided by its battery, in order to transmit its measured value to the central node. An alternative strategy to overcome the energy inefficiency of the centralized computation is the distributed or in-network computation strategy. In distributed computation, which is a cooperative strategy, each sensor computes its target value by interchanging information with its neighbouring sensors.

In many recent signal processing applications of distributed computations (e.g., [1–4]), the average needs to be computed (i.e., each sensor seeks the arithmetic mean of the values measured by all the sensors of the WSN). The problem of obtaining that average in all the sensors of the WSN by using the distributed computation strategy is known as the distributed averaging problem, or the distributed average consensus problem. Moreover, the problem of obtaining the same value in all the sensors of the WSN by using the distributed computation strategy is known as the distributed consensus problem (see, for example, [5] for a review on this subject).

A common approach for solving the distributed averaging problem is to use a synchronous linear iterative algorithm that is characterized by a matrix, which is called the weighting matrix.

A well-known problem related to this topic is that of finding a symmetric weighting matrix that achieves consensus as fast as possible. This is the problem of finding the fastest symmetric distributed linear averaging (FSDLA) algorithm.

The FSDLA problem was solved in [6]. Specifically, in [6], the authors proved that solving the FSDLA problem is equivalent to solving a semidefinite program, and they used the subgradient method for efficiently solving such a problem to obtain the corresponding weighting matrix. Unfortunately, solving the FSDLA problem this way requires a central entity with full knowledge of the entire network. This central entity has to solve the FSDLA problem and then communicate the solution to each node of the network. This process has to be repeated each time the network topology changes due to, for example, a node failing, a node being added or removed (plug-and-play networks), or a node changing its location.

Moreover, WSNs may not have a central entity to compute the optimal weighting matrix. This paper proposes, for those networks without a central entity, an in-network algorithm for finding the optimal weighting matrix.

It is worth mentioning that in the literature, one can find other in-network algorithms that solve the FSDLA problem in a distributed way. In particular, in [7], the authors present an in-network algorithm that computes the fastest symmetric weighting matrix, but only with positive weights. As will be made more explicit in the next section, this matrix is not a solution of the FSDLA problem in general, as the latter might contain negative weights.

In [8], the FSDLA problem is solved in a centralized way when the communication among nodes is noisy. Closed-form expressions for the optimal weights for certain network topologies (paths, cycles, grids, stars, and hypercubes) are also provided. However, unless the considered network topology is one of these five, an in-network solution to the FSDLA is not provided.

Finally, in [9], an in-network algorithm for solving the FSDLA problem is provided. However, as the authors claim, the algorithm breaks down when the second- and third-largest eigenvalues of the weighting matrix become similar or equal.

Unlike the approaches found in the literature, the in-network algorithm presented in this paper is proved to always converge to the solution of the FSDLA problem, irrespective of the considered network topology.

2. Preliminaries

2.1. The Distributed Average Consensus Problem

We consider a network composed of n nodes. The network can be viewed as an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. An edge $e = \{i, j\} \in \mathcal{E}$ means that nodes $i, j \in \mathcal{V}$ are connected and can therefore interchange information. Conversely, if $\{i, j\} \notin \mathcal{E}$, this means that nodes $i, j \in \mathcal{V}$ are not connected and cannot interchange information. We let K be the cardinal of $\mathcal{E}$, i.e., K is the number of edges in the graph G. For simplicity, we enumerate the edges in the graph G as $\mathcal{E} = \{e_1, e_2, \ldots, e_K\}$, where $e_k = \{i_k, j_k\}$ for all $k \in \{1, 2, \ldots, K\}$.

We assume that each node $i \in \mathcal{V}$ has an initial value $x_i(0) \in \mathbb{R}$, where $\mathbb{R}$ denotes the set of (finite) real numbers. Accordingly, in this paper, $\mathbb{R}^{m \times n}$ denotes the set of $m \times n$ real matrices. We consider that all the nodes are interested in obtaining the arithmetic mean (average) x_{ave} of the initial values of the nodes, that is,

$$x_{\text{ave}} := \frac{1}{n} \sum_{i=1}^{n} x_i(0),$$

using a distributed algorithm. This problem is commonly known as the distributed averaging problem, or the distributed average consensus problem.

The approach that will be considered here for solving the distributed averaging problem is to use a linear iterative algorithm of the form

$$x_i(t+1) = w_{i,i}x_i(t) + \sum_{j\in\mathcal{V}:\,\{i,j\}\in\mathcal{E}} w_{i,j}x_j(t), \qquad i \in \mathcal{V} \tag{1}$$

where time t is assumed to be discrete (namely, $t \in \{0, 1, 2, \ldots\}$) and $w_{i,j} \in \mathbb{R}$ are the weights that need to be set so that

$$\lim_{t\to\infty} x_i(t) = x_{\text{ave}} \tag{2}$$

for all $i \in \mathcal{V}$ and for all $x_1(0), x_2(0), \ldots, x_n(0) \in \mathbb{R}$. From the point of view of communication protocols, there exist efficient ways of implementing synchronous consensus algorithms of the form of Equation (1) (e.g., [10]).

We observe that Equation (1) can be written in matrix form as

$$\boldsymbol{x}(t+1) = \boldsymbol{W}\boldsymbol{x}(t) \tag{3}$$

where $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))^\top \in \mathbb{R}^{n\times 1}$, and $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is called the *weighting matrix*, which is such that its entry at the ith row and jth column, $[\boldsymbol{W}]_{i,j}$, is given by

$$[\boldsymbol{W}]_{i,j} = \begin{cases} 0 & \text{if } i \neq j \text{ and } \{i,j\} \notin \mathcal{E}, \\ w_{i,j} & \text{otherwise.} \end{cases} \qquad i,j \in \{1, 2, \ldots, n\} \tag{4}$$

Therefore, Equation (2) can be rewritten as

$$\lim_{t\to\infty} \boldsymbol{W}^t = \boldsymbol{P}_n \tag{5}$$

where $\boldsymbol{P}_n := \frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top$, and $\boldsymbol{1}_n$ is the $n \times 1$ matrix of ones.

We only consider algorithms of the form of Equation (3), for which the weighting matrix $\boldsymbol{W}$ is symmetric. If $\boldsymbol{W}$ is symmetric, it is shown in [6] (Theorem 1) that Equation (5) holds if and only if $\boldsymbol{W}\boldsymbol{1}_n = \boldsymbol{1}_n$ and $\|\boldsymbol{W} - \boldsymbol{P}_n\|_2 < 1$, where $\|\cdot\|_2$ denotes the spectral norm. For the reader's convenience, we here recall that if $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is symmetric, then $\|\boldsymbol{A}\|_2 = |\lambda_1(\boldsymbol{A})|$, where $\lambda_l(\boldsymbol{A})$, $l \in \{1, 2, \ldots, n\}$, denote the eigenvalues of $\boldsymbol{A}$, which, in this paper, are arranged such that $|\lambda_1(\boldsymbol{A})| \geq |\lambda_2(\boldsymbol{A})| \geq \ldots \geq |\lambda_n(\boldsymbol{A})|$ (e.g., [11] (pp. 350, 603)).

We observe that Equation (10) can be computed in a distributed way if each node $i \in \mathcal{V}$ is able to know y_i. The following result provides a means of computing such a unit eigenvector $\boldsymbol{y}$ of $\boldsymbol{W}(\boldsymbol{w})$ in a distributed way.

2.2. Considered Minimization Problem: FSDLA Problem

We denote with $\mathcal{W}(G)$ the set of all the $n \times n$ real symmetric matrices that satisfy Equation (4) and $\boldsymbol{W}\boldsymbol{1}_n = \boldsymbol{1}_n$ simultaneously, that is,

$$\mathcal{W}(G) := \Big\{\boldsymbol{W} \in \mathbb{R}^{n\times n},\ [\boldsymbol{W}]_{i,j} = 0 \text{ if } i \neq j \text{ and } \{i,j\} \notin \mathcal{E}, \\ \boldsymbol{W} = \boldsymbol{W}^\top, \quad \boldsymbol{W}\boldsymbol{1}_n = \boldsymbol{1}_n\Big\}.$$

In [6], the convergence time of an algorithm of the form of Equation (3) with symmetric weighting matrix $\boldsymbol{W}$ is defined as

$$\tau(\boldsymbol{W}) := \frac{-1}{\log\|\boldsymbol{W} - \boldsymbol{P}_n\|_2} \tag{6}$$

This convergence time is a mathematical measure of the convergence speed of the algorithm.

According to the previous, we call the FSDLA problem to find a weighting matrix $\boldsymbol{W}_{\text{opt}} \in \mathcal{W}(G)$ such that

$$\|\boldsymbol{W}_{\text{opt}} - \boldsymbol{P}_n\|_2 \leq \|\boldsymbol{W} - \boldsymbol{P}_n\|_2 \qquad \forall \boldsymbol{W} \in \mathcal{W}(G) \tag{7}$$

We observe that in this definition the meaning of *fastest* is in terms of convergence time.

It is shown in [6] that the FSDLA problem of Equation (7) is a constrained convex minimization problem that can be efficiently solved. In fact, in [6], it is shown that the FSDLA problem of Equation (7) can be expressed as a semidefinite program, and semidefinite programs can be efficiently solved [12]. However, to the best of our knowledge, there are yet no approaches for solving this FSDLA problem in a distributed (in-network) way. The contribution of this paper is to solve the FSDLA problem of Equation (7) in a distributed way. To do so, we develop a distributed subgradient method.

Finally, it should be mentioned that in [7], the authors solved, in a distributed way, a related problem: they find the fastest mixing Markov chain (FMMC). The FMMC problem is devoted to finding a matrix $\boldsymbol{W}_{\text{opt}}^{+} \in \mathcal{W}(G) \cap \{\boldsymbol{W} \in \mathbb{R}^{n\times n} : [\boldsymbol{W}]_{i,j} \geq 0, \ \forall i,j \in \{1,\ldots,n\}\}$ such that $\|\boldsymbol{W}_{\text{opt}}^{+} - \boldsymbol{P}_n\|_2 \leq \|\boldsymbol{W} - \boldsymbol{P}_n\|_2$ for all $\boldsymbol{W} \in \mathcal{W}(G) \cap \{\boldsymbol{W} \in \mathbb{R}^{n\times n} : [\boldsymbol{W}]_{i,j} \geq 0, \ \forall i,j \in \{1,\ldots,n\}\}$. We observe that $\|\boldsymbol{W}_{\text{opt}} - \boldsymbol{P}_n\|_2 \leq \|\boldsymbol{W}_{\text{opt}}^{+} - \boldsymbol{P}_n\|_2$, i.e., the solution of the FSDLA problem is faster than, or is at least as fast as, the solution of the FMMC problem.

2.3. FSDLA as an Unconstrained Convex Minimization Problem

In order to use a distributed subgradient method (the classical reference on subgradient methods is [13]), we first need to convert the FSDLA problem into an unconstrained convex minimization problem. We observe that if $\boldsymbol{W} \in \mathcal{W}(G)$, it is clear that $\boldsymbol{W}$ depends on $w_{e_k} := w_{i_k,j_k}$ for all $k \in \{1,2,\ldots,K\}$. We notice that w_{e_k} is well defined because $\boldsymbol{W}$ is symmetric. In fact, as it was stated in [6], given the vector $\boldsymbol{w} = (w_{e_1}, w_{e_2}, \ldots, w_{e_K})^\top \in \mathbb{R}^{K\times 1}$, there exists a unique $\boldsymbol{W} \in \mathcal{W}(G)$ such that $[\boldsymbol{W}]_{i_k,j_k} = w_{e_k}$ for all $k \in \{1,2,\ldots,K\}$, namely

$$\boldsymbol{W}(\boldsymbol{w}) = \boldsymbol{I}_n + \sum_{k=1}^{K} w_{e_k} \boldsymbol{A}_k \tag{8}$$

where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix and $\boldsymbol{A}_k \in \mathbb{R}^{n\times n}$ is defined as

$$[\boldsymbol{A}_k]_{i,j} := \begin{cases} 1 & \{i,j\} = \{i_k, j_k\}, \\ -1 & i = j = i_k \text{ or } i = j = j_k, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall k \in \{1,2,\ldots,K\}$$

In other words, the function $\boldsymbol{W} : \mathbb{R}^{K\times 1} \mapsto \mathcal{W}(G)$ defined in Equation (8) is a bijection. We define the function $f : \mathbb{R}^{K\times 1} \mapsto [0,\infty)$ as $f(\boldsymbol{w}) := \|\boldsymbol{W}(\boldsymbol{w}) - \boldsymbol{P}_n\|_2$. We observe that the FSDLA problem of Equation (7) can now be expressed as an unconstrained minimization of the function f.

In the sequel, we denote with $\widehat{\boldsymbol{w}}$ a solution of the FSDLA problem, that is,

$$f(\widehat{\boldsymbol{w}}) \leq f(\boldsymbol{w}) \qquad \forall \boldsymbol{w} \in \mathbb{R}^{K\times 1}.$$

It is easy to show that f has a bounded set of minimum points $\widehat{\boldsymbol{w}}$. In the sequel, we will refer to the function f as the cost function of the FSDLA problem. We finish the section with Lemma 1 which will be useful in the derivation of the algorithm.

Lemma 1. *If $\boldsymbol{w} \in \mathbb{R}^{K\times 1}$, then*

$$f(\boldsymbol{w}) = \begin{cases} |\lambda_1(\boldsymbol{W}(\boldsymbol{w}))| & \textit{if } |\lambda_2(\boldsymbol{W}(\boldsymbol{w}))| \geq 1, \\ |\lambda_2(\boldsymbol{W}(\boldsymbol{w}))| & \textit{if } |\lambda_2(\boldsymbol{W}(\boldsymbol{w}))| < 1. \end{cases}$$

Proof. Observe that as $\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{w})$ is symmetric and $\boldsymbol{W}\mathbf{1}_n = \mathbf{1}_n$, there exists an eigenvalue decomposition of $\boldsymbol{W}$, $\boldsymbol{W} = \boldsymbol{U}\mathrm{diag}_n\left(1, \lambda_2(\boldsymbol{W}), \ldots, \lambda_n(\boldsymbol{W})\right)\boldsymbol{U}^\top$, where $\boldsymbol{U}$ is a real $n \times n$ orthogonal matrix such that $[\boldsymbol{U}]_{i,1} = \frac{1}{\sqrt{n}}$ for all $i \in \{1, 2, \ldots, n\}$ and $|\lambda_2(\boldsymbol{W})| \geq |\lambda_3(\boldsymbol{W})| \geq \ldots \geq |\lambda_n(\boldsymbol{W})|$. As $\boldsymbol{P}_n = \boldsymbol{U}\mathrm{diag}_n\left(1, 0, \ldots, 0\right)\boldsymbol{U}^\top$, we have

$$\begin{aligned} f(\boldsymbol{w}) &= \|\boldsymbol{W} - \boldsymbol{P}_n\|_2 = \|\boldsymbol{U}\mathrm{diag}_n\left(0, \lambda_2(\boldsymbol{W}), \ldots, \lambda_n(\boldsymbol{W})\right)\boldsymbol{U}^\top\|_2 \\ &= \|\mathrm{diag}_n\left(0, \lambda_2(\boldsymbol{W}), \ldots, \lambda_n(\boldsymbol{W})\right)\|_2 = |\lambda_2(\boldsymbol{W})|. \end{aligned}$$

□

3. Algorithm for the In-Network Solution of the FSDLA Problem

We here derive the algorithm that solves the FSDLA problem in a distributed way (Algorithm 1). To this end, we assume that n is known by all the nodes of the network. The task of counting nodes can be performed in a distributed way (see [14]). The algorithm is a distributed implementation of a subgradient method. More specifically, each pair of nodes $\{i_k, j_k\}$ will update their weight w_{i_k, j_k} according to the following iterative formula:

$$\boldsymbol{w}_{p+1} = \boldsymbol{w}_p - \eta_{p+1}\widetilde{\nabla} f(\boldsymbol{w}_p) \tag{9}$$

where $\boldsymbol{w}_p \in \mathbb{R}^{K \times 1}$ is the vector of weights at the pth step, $\eta_p \in \mathbb{R}$ is the stepsize, and $\widetilde{\nabla} f(\boldsymbol{w})$ is a subgradient of f at $\boldsymbol{w}$. We recall here that a vector $\widetilde{\nabla} f(\boldsymbol{w}) \in \mathbb{R}^{K \times 1}$ is a subgradient of $f : \mathbb{R}^{K \times 1} \mapsto \mathbb{R}$ at $\boldsymbol{w} \in \mathbb{R}^{K \times 1}$ if $f(\boldsymbol{v}) \geq f(\boldsymbol{w}) + (\widetilde{\nabla} f(\boldsymbol{w}))^\top(\boldsymbol{v} - \boldsymbol{w})$ for all $\boldsymbol{v} \in \mathbb{R}^{K \times 1}$.

Theorem 1. *If $\boldsymbol{w} \in \mathbb{R}^{K \times 1}$ such that $0 < f(\boldsymbol{w}) < 1$, and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\top \in \mathbb{R}^{n \times 1}$ is such that $\|\boldsymbol{y}\| = 1$ and $\boldsymbol{W}(\boldsymbol{w})\boldsymbol{y} = (-1)^s|\lambda_2(\boldsymbol{W}(\boldsymbol{w}))|\boldsymbol{y}$ for some $s \in \{1, 2\}$, then a subgradient of f at $\boldsymbol{w}$ is*

$$\widetilde{\nabla} f(\boldsymbol{w}) = (-1)^{s+1}\begin{pmatrix} (y_{i_1} - y_{j_1})^2 \\ \vdots \\ (y_{i_K} - y_{j_K})^2 \end{pmatrix} \tag{10}$$

We observe that Equation (10) can be computed in a distributed way if each node $i \in \mathcal{V}$ is able to know y_i. The following result provides a means of computing such a unit eigenvector $\boldsymbol{y}$ of $\boldsymbol{W}(\boldsymbol{w})$ in a distributed way.

The rest of the section is devoted to proving that Equation (9) can be computed in a distributed way (Theorems 1–3), and to proving that Equation (9) actually converges to $\widehat{\boldsymbol{w}}$ (Theorem 4).

In order to compute Equation (9) in a distributed way, we need to compute a subgradient of f in a distributed way. With this in mind, we review a result given in [6].

Theorem 2. *If $\boldsymbol{w} \in \mathbb{R}^{K \times 1}$ is such that $0 < f(\boldsymbol{w}) < 1$, then for all $\boldsymbol{x}(0) \in \mathbb{R}^{n \times 1}$,*

$$\boldsymbol{W}(\boldsymbol{w})\boldsymbol{y}_s = (-1)^s|\lambda_2(\boldsymbol{W}(\boldsymbol{w}))|\boldsymbol{y}_s \qquad s \in \{1, 2\} \tag{11}$$

where

$$\boldsymbol{y}_s := \lim_{t \to \infty}\left(\frac{\boldsymbol{x}(t) - \boldsymbol{x}(t-2)}{((-1)^s f(\boldsymbol{w}))^t} + \frac{\boldsymbol{x}(t-1) - \boldsymbol{x}(t-3)}{((-1)^s f(\boldsymbol{w}))^{t-1}}\right) \tag{12}$$

and $\boldsymbol{x}(t) = (\boldsymbol{W}(\boldsymbol{w}))^t\boldsymbol{x}(0)$ for all $t \in \{0, 1, 2, \ldots\}$. Furthermore, given $s \in \{1, 2\}$, for almost every $\boldsymbol{x}(0) \in \mathbb{R}^{n \times 1}$, the following assertions are equivalent:

(a) $\boldsymbol{y}_s \neq \mathbf{0}_{n \times 1}$*, where $\mathbf{0}_{n \times 1}$ is the $n \times 1$ zero matrix.*

(b) $(-1)^s|\lambda_2(\boldsymbol{W}(\boldsymbol{w}))|$ *is an eigenvalue of* $\boldsymbol{W}(\boldsymbol{w})$.

Algorithm 1 In-network solution of the FSDLA problem.

```
1:  $p \leftarrow 0$
2:  for all pair of nodes $e_k = \{i_k, j_k\}$ do
3:      $[\boldsymbol{w}_0]_k \leftarrow 1/\max(d_{i_k}, d_{j_k})$
4:  end for
5:  $p \leftarrow p + 1$
6:  for all nodes $i \in \mathcal{V}$ do
7:      $[\boldsymbol{x}]_i \leftarrow \text{rand}()$                                  ▷ An arbitrary value
8:      $[\gamma_1]_i \leftarrow \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0)\right]_i - \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 1)\right]_i$
9:      $[\gamma_2]_i \leftarrow \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 1)\right]_i - \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 2)\right]_i$
10:     $f(\boldsymbol{w}_p) = \|\boldsymbol{W}(\boldsymbol{w}_p) - \boldsymbol{P}_n\|_2 \leftarrow \sqrt{\frac{\left[\text{ave}_{\boldsymbol{w}_p}\left(([\gamma_1]_1^2, \ldots, [\gamma_1]_n^2)^\top, t_0\right)\right]_i}{\left[\text{ave}_{\boldsymbol{w}_p}\left(([\gamma_2]_1^2, \ldots, [\gamma_2]_n^2)^\top, t_0\right)\right]_i}}$
11:     if $f(\boldsymbol{w}_p) \geq 1$ then
12:         $\boldsymbol{w}_p \leftarrow \boldsymbol{w}_{p-1}$
13:     end if
14:     $[\boldsymbol{y}_1]_i \leftarrow \frac{\left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0)\right]_i - \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 2)\right]_i}{(-f(\boldsymbol{w}_p))^{t_0}} + \frac{\left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 1)\right]_i - \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 3)\right]_i}{(-f(\boldsymbol{w}_p))^{t_0 - 1}}$
15:     $[\boldsymbol{y}_2]_i \leftarrow \frac{\left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0)\right]_i - \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 2)\right]_i}{(f(\boldsymbol{w}_p))^{t_0}} + \frac{\left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 1)\right]_i - \left[\text{ave}_{\boldsymbol{w}_p}(\boldsymbol{x}, t_0 - 3)\right]_i}{(f(\boldsymbol{w}_p))^{t_0 - 1}}$
16:     $\|\boldsymbol{y}_1\| \leftarrow \sqrt{\left[\text{ave}_{\boldsymbol{w}_p}\left(([\boldsymbol{y}_1]_1^2, \ldots, [\boldsymbol{y}_1]_n^2)^\top, t_0\right)\right]_i}$
17:     if $\|\boldsymbol{y}_1\| \neq 0$ then
18:         $[\boldsymbol{y}]_i \leftarrow [\boldsymbol{y}_1]_i / \|\boldsymbol{y}_1\|$
19:         $s \leftarrow 1$
20:     else
21:         $\|\boldsymbol{y}_2\| \leftarrow \sqrt{\left[\text{ave}_{\boldsymbol{w}_p}\left(([\boldsymbol{y}_2]_1^2, \ldots, [\boldsymbol{y}_2]_n^2)^\top, t_0\right)\right]_i}$
22:         $[\boldsymbol{y}]_i \leftarrow [\boldsymbol{y}_2]_i / \|\boldsymbol{y}_2\|$
23:         $s \leftarrow 2$
24:     end if
25: end for
26: for all pair of nodes $e_k = \{i_k, j_k\}$ do
27:     $[\widetilde{\nabla} f(\boldsymbol{w}_p)]_k \leftarrow (-1)^{s+1}\left([\boldsymbol{y}]_{i_k} - [\boldsymbol{y}]_{j_k}\right)^2$
28:     $[\boldsymbol{w}_p]_k \leftarrow [\boldsymbol{w}_{p-1}]_k - \beta_p [\widetilde{\nabla} f(\boldsymbol{w}_p)]_k$
29: end for
30: if $p < p_{\max}$ go to 5
```

Proof. Let $\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{w}) = \boldsymbol{U}\mathrm{diag}_n\left(1, \lambda_2(\boldsymbol{W}), \ldots, \lambda_n(\boldsymbol{W})\right)\boldsymbol{U}^\top$ be as in the proof of Lemma 1, with $\boldsymbol{U} = (\boldsymbol{u}_1|\boldsymbol{u}_2|\ldots|\boldsymbol{u}_n)$. Observe that $\lambda_2(\boldsymbol{W}) \neq 0$, as $|\lambda_2(\boldsymbol{W})| = f(\boldsymbol{w}) \neq 0$.

If $(-1)^{s-1}|\lambda_2(\boldsymbol{W})|$ is an eigenvalue of $\boldsymbol{W}$ for some $s \in \{1,2\}$, then we denote by L_s its algebraic multiplicity. Otherwise we set $L_s = 0$. From Lemma 1, $f(\boldsymbol{w}) = |\lambda_2(\boldsymbol{W})|$ and consequently L_1 and L_2 cannot be simultaneously zero. Moreover, without loss of generality we can assume that $\lambda_2(\boldsymbol{W}) \geq \ldots \geq \lambda_{L_1+L_2+1}(\boldsymbol{W})$.

Then, we have that

$$\begin{aligned}
\boldsymbol{x}(t) &= \boldsymbol{W}^t\boldsymbol{x}(0) = \boldsymbol{W}^t\left(\sum_{l=1}^{n}\alpha_l\boldsymbol{u}_l\right) = \sum_{l=1}^{n}\alpha_l\boldsymbol{W}^t\boldsymbol{u}_l \\
&= \alpha_1\boldsymbol{W}^t\boldsymbol{u}_1 + \sum_{l=2}^{L_1+1}\alpha_l\boldsymbol{W}^t\boldsymbol{u}_l + \sum_{l=L_1+2}^{L_1+L_2+1}\alpha_l\boldsymbol{W}^t\boldsymbol{u}_l + \sum_{l=L_1+L_2+2}^{n}\alpha_l\boldsymbol{W}^t\boldsymbol{u}_l \\
&= \alpha_1\boldsymbol{u}_1 + \sum_{l=2}^{L_1+1}\alpha_l|\lambda_2(\boldsymbol{W})|^t\boldsymbol{u}_l + \sum_{l=L_1+2}^{L_1+L_2+1}\alpha_l(-1)^t|\lambda_2(\boldsymbol{W})|^t\boldsymbol{u}_l + \sum_{l=L_1+L_2+2}^{n}\alpha_l\lambda_l(\boldsymbol{W})^t\boldsymbol{u}_l \qquad (13)\\
&= \alpha_1\boldsymbol{u}_1 + |\lambda_2(\boldsymbol{W})|^t\left(\sum_{l=2}^{L_1+1}\alpha_l\boldsymbol{u}_l + (-1)^t\sum_{l=L_1+2}^{L_1+L_2+1}\alpha_l\boldsymbol{u}_l\right) + \sum_{l=L_1+L_2+2}^{n}\alpha_l\lambda_l(\boldsymbol{W})^t\boldsymbol{u}_l \\
&= \alpha_1\boldsymbol{u}_1 + |\lambda_2(\boldsymbol{W})|^t\left((-1)^t\boldsymbol{a}_1 + \boldsymbol{a}_2\right) + \boldsymbol{r}(t) \qquad \forall t \in \{0,1,2,\ldots\}
\end{aligned}$$

where $\alpha_l = (\boldsymbol{x}(0))^\top\boldsymbol{u}_l$ for all $l \in \{1,2,\ldots,n\}$,

$$\boldsymbol{a}_1 := \sum_{l=L_1+2}^{L_1+L_2+1}\alpha_l\boldsymbol{u}_l,$$

$$\boldsymbol{a}_2 := \sum_{l=2}^{L_1+1}\alpha_l\boldsymbol{u}_l,$$

and

$$\boldsymbol{r}(t) := \sum_{l=L_1+L_2+2}^{n}\alpha_l\lambda_l(\boldsymbol{W})^t\boldsymbol{u}_l.$$

Observe that

$$\boldsymbol{W}\boldsymbol{a}_s = (-1)^s|\lambda_2(\boldsymbol{W})|\boldsymbol{a}_s \qquad \forall s \in \{1,2\} \qquad (14)$$

On the one hand, from Equation (13), we obtain

$$\begin{aligned}
\frac{\boldsymbol{x}(t)-\boldsymbol{x}(t-2)}{|\lambda_2(\boldsymbol{W})|^t} &= \frac{|\lambda_2(\boldsymbol{W})|^t\left((-1)^t\boldsymbol{a}_1+\boldsymbol{a}_2\right)+\boldsymbol{r}(t)}{|\lambda_2(\boldsymbol{W})|^t} - \frac{|\lambda_2(\boldsymbol{W})|^{t-2}\left((-1)^{t-2}\boldsymbol{a}_1+\boldsymbol{a}_2\right)-\boldsymbol{r}(t-2)}{|\lambda_2(\boldsymbol{W})|^t} \\
&= \left(1-|\lambda_2(\boldsymbol{W})|^{-2}\right)\left((-1)^t\boldsymbol{a}_1+\boldsymbol{a}_2\right) + \frac{\boldsymbol{r}(t)-\boldsymbol{r}(t-2)}{|\lambda_2(\boldsymbol{W})|^t}
\end{aligned}$$

for all $t \in \{2,3,\ldots\}$. On the other hand, as $|\lambda_l(\boldsymbol{W})| < |\lambda_2(\boldsymbol{W})|$ for all $l \in \{L_1+L_2+2,\ldots,n\}$, we have that

$$\lim_{t\to\infty}\frac{\boldsymbol{r}(t)}{|\lambda_2(\boldsymbol{W})|^t} = \lim_{t\to\infty}\sum_{l=L_1+L_2+2}^{n}\alpha_l\frac{\lambda_l(\boldsymbol{W})^t}{|\lambda_2(\boldsymbol{W})|^t}\boldsymbol{u}_l = \lim_{t\to\infty}\sum_{l=L_1+L_2+2}^{n}\alpha_l\left(\frac{\lambda_l(\boldsymbol{W})}{|\lambda_2(\boldsymbol{W})|}\right)^t\boldsymbol{u}_l = \boldsymbol{0}_{n\times 1}.$$

Consequently,

$$\begin{aligned}
\boldsymbol{y}_s &= \lim_{t\to\infty}\left(\frac{\boldsymbol{x}(t)-\boldsymbol{x}(t-2)}{((-1)^s|\lambda_2(\boldsymbol{W})|)^t}+\frac{\boldsymbol{x}(t-1)-\boldsymbol{x}(t-3)}{((-1)^s|\lambda_2(\boldsymbol{W})|)^{t-1}}\right)\\
&= \lim_{t\to\infty}\left(\frac{1-|\lambda_2(\boldsymbol{W})|^{-2}}{(-1)^{st}}\left((-1)^t\boldsymbol{a}_1+\boldsymbol{a}_2+(-1)^s\left((-1)^{t-1}\boldsymbol{a}_1+\boldsymbol{a}_2\right)\right)\right.\\
&\qquad\left.+\frac{\boldsymbol{r}(t)-\boldsymbol{r}(t-2)}{((-1)^s|\lambda_2(\boldsymbol{W})|)^t}+\frac{\boldsymbol{r}(t-1)-\boldsymbol{r}(t-3)}{((-1)^s|\lambda_2(\boldsymbol{W})|)^{t-1}}\right) \qquad (15)\\
&= \lim_{t\to\infty}\left(\frac{1-|\lambda_2(\boldsymbol{W})|^{-2}}{(-1)^{st}}\left((-1)^t\boldsymbol{a}_1+\boldsymbol{a}_2+(-1)^s\left((-1)^{t-1}\boldsymbol{a}_1+\boldsymbol{a}_2\right)\right)\right)\\
&= 2\left(1-|\lambda_2(\boldsymbol{W})|^{-2}\right)\boldsymbol{a}_s, \qquad s\in\{1,2\}
\end{aligned}$$

Combining Equations (14) and (15), we obtain Equation (11).

From Equation (11), (a) implies (b) for all $\boldsymbol{x}(0)\in\mathbb{R}^{n\times 1}$.

As $f(\boldsymbol{w})<1$, from Lemma 1 and Equation (15), we have $\boldsymbol{y}_s\neq\boldsymbol{0}_{n\times 1}$ if and only if $\boldsymbol{a}_s\neq\boldsymbol{0}_{n\times 1}$. Consequently, if (b) holds, the set of $\boldsymbol{x}(0)$ such that $\boldsymbol{a}_s=\boldsymbol{0}_{n\times 1}$ is a vector space whose dimension is less than n; thus it has Lebesgue measure 0. Therefore, (a) and (b) are equivalent for almost every $\boldsymbol{x}(0)\in\mathbb{R}^{n\times 1}$. □

Theorem 2 implies that $\|\boldsymbol{y}_1\|$ and $\|\boldsymbol{y}_2\|$ cannot be zero simultaneously. Therefore, either $\frac{\boldsymbol{y}_1}{\|\boldsymbol{y}_1\|}$ or $\frac{\boldsymbol{y}_2}{\|\boldsymbol{y}_2\|}$ is the unit eigenvector required for computing Equation (10). We notice that the norm of a vector can be computed in a distributed way because it is the square root of n times the average of the squares of its entries. Consequently, we only need to know how to compute Equation (12) in a distributed way, or equivalently, how to compute the cost function f in a distributed way:

Theorem 3. *If $\boldsymbol{w}\in\mathbb{R}^{K\times 1}$ such that $f(\boldsymbol{w})\neq 0$, then*

$$f(\boldsymbol{w})=\lim_{t\to\infty}\frac{\|\boldsymbol{x}(t)-\boldsymbol{x}(t-1)\|}{\|\boldsymbol{x}(t-1)-\boldsymbol{x}(t-2)\|} \qquad (16)$$

for almost every $\boldsymbol{x}(0)\in\mathbb{R}^{n\times 1}$, where $\boldsymbol{x}(t)=(\boldsymbol{W}(\boldsymbol{w}))^t\,\boldsymbol{x}(0)$ for all $t\in\{0,1,2,\ldots\}$.

Proof. Let $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{w})=\boldsymbol{U}\mathrm{diag}_n\left(1,\lambda_2(\boldsymbol{W}),\ldots,\lambda_n(\boldsymbol{W})\right)\boldsymbol{U}^\top$ be as in the proof of Lemma 1, with $U=(\boldsymbol{u}_1|\boldsymbol{u}_2|\ldots|\boldsymbol{u}_n)$. Then,

$$\begin{aligned}
\boldsymbol{x}(t)=\boldsymbol{W}^t\boldsymbol{x}(0)=\boldsymbol{W}^t\left(\sum_{l=1}^n\alpha_l\boldsymbol{u}_l\right)&=\sum_{l=1}^n\alpha_l\boldsymbol{W}^t\boldsymbol{u}_l\\
=\alpha_1\boldsymbol{W}^t\boldsymbol{u}_1+\sum_{l=2}^n\alpha_l\boldsymbol{W}^t\boldsymbol{u}_l&=\alpha_1\boldsymbol{u}_1+\sum_{l=2}^n\alpha_l\lambda_l(\boldsymbol{W})^t\boldsymbol{u}_l
\end{aligned}$$

for all $t\in\{0,1,2,\ldots\}$, where $\alpha_l=(\boldsymbol{x}(0))^\top\boldsymbol{u}_l$ for all $l\in\{1,2,\ldots,n\}$. Consider $L\in\{0,1,\ldots,n-2\}$ such that $|\lambda_2(\boldsymbol{W})|=|\lambda_3(\boldsymbol{W})|=\ldots=|\lambda_{2+L}(\boldsymbol{W})|$. Observe that $\lambda_2(\boldsymbol{W})\neq 0$, as $|\lambda_2(\boldsymbol{W})|=f(\boldsymbol{w})\neq 0$. Consequently, from the pythagorean theorem,

$$\begin{aligned}\|\boldsymbol{x}(t)-\boldsymbol{x}(t-1)\|^2 &= \left\|\sum_{l=2}^{n}\alpha_l\left(\lambda_l(\boldsymbol{W})^t-\lambda_l(\boldsymbol{W})^{t-1}\right)\boldsymbol{u}_l\right\|^2 = \sum_{l=2}^{n}\alpha_l^2\left(\lambda_l(\boldsymbol{W})^t-\lambda_l(\boldsymbol{W})^{t-1}\right)^2\\ &= \sum_{l=2}^{n}\alpha_l^2\left(\lambda_l(\boldsymbol{W})-1\right)^2\lambda_l(\boldsymbol{W})^{2t-2}\\ &= \lambda_2(\boldsymbol{W})^{2t-2}\left(\sum_{l=2}^{L+2}\alpha_l^2\left(\lambda_l(\boldsymbol{W})-1\right)^2+\sum_{l=L+3}^{n}\alpha_l^2\left(\lambda_l(\boldsymbol{W})-1\right)^2\left(\frac{\lambda_l(\boldsymbol{W})}{\lambda_2(\boldsymbol{W})}\right)^{2t-2}\right)\end{aligned}$$

for all $t \in \{1,2,\ldots\}$. Assume that $\sum_{l=2}^{L+2}\alpha_l^2 \neq 0$, which holds for almost every $\boldsymbol{x}(0) \in \mathbb{R}^{n\times 1}$. As $|\lambda_2(\boldsymbol{W})| > |\lambda_l(\boldsymbol{W})|$ for all $l \in \{L+3,\ldots,n\}$, we conclude that

$$\lim_{t\to\infty}\frac{\|\boldsymbol{x}(t)-\boldsymbol{x}(t-1)\|}{\|\boldsymbol{x}(t-1)-\boldsymbol{x}(t-2)\|} = |\lambda_2(\boldsymbol{W})|\sqrt{\frac{\sum_{l=2}^{L+2}\alpha_l^2\left(\lambda_l(\boldsymbol{W})-1\right)^2}{\sum_{l=2}^{L+2}\alpha_l^2\left(\lambda_l(\boldsymbol{W})-1\right)^2}} = |\lambda_2(\boldsymbol{W})| = f(\boldsymbol{w}).$$

□

We observe that Equation (16) can be computed in a distributed way because a norm can be computed in a distributed way. Moreover, we observe that the condition $f(\boldsymbol{w}) = 0$ holds if and only if $\boldsymbol{W} = \boldsymbol{P}_n$ (this is possible only if every pair of nodes of the network is connected, i.e., it is a fully-connected network; in this case, $\boldsymbol{W}_{\text{opt}} = \boldsymbol{P}_n$ and consequently the FSDLA problem makes no sense). Therefore, for any non-fully-connected network, $f(\boldsymbol{w}) \neq 0$.

At this point, we have shown that the iterative Equation (9) can be computed in a distributed way. It only remains to be shown that Equation (9) actually converges to $\widehat{\boldsymbol{w}}$:

Theorem 4. *Consider $\boldsymbol{w}_0 \in \mathbb{R}^{K\times 1}$ such that $0 < f(\boldsymbol{w}_0) < 1$. Let $\{\eta_p\}$ be a sequence of real numbers satisfying $\lim_{p\to\infty}\eta_p = 0$ and $\sum_{p=0}^{\infty}\eta_p = \infty$. We also assume that*

$$0 < f(\boldsymbol{w}_p) < 1 \qquad \forall p \in \{1,2,\ldots\} \tag{17}$$

where $\boldsymbol{w}_p$ is defined in Equation (9). Then, $f(\widehat{\boldsymbol{w}}) = \|\boldsymbol{W}_{opt} - \boldsymbol{P}_n\|_2 = \lim_{p\to\infty} f(\boldsymbol{w}_p)$.

Proof. Theorem 1 yields

$$\eta_p\left\|\widetilde{\nabla} f(\boldsymbol{w}_p)\right\| = \eta_p\sqrt{\sum_{k=1}^{K}\left(y_{i_k}-y_{j_k}\right)^4} \leq 4\sqrt{K}\max_{p\in\{0,1,2,\ldots\}}\eta_p.$$

Consequently, as f has a bounded set of minimum points, the result now follows from [13] (Theorem 2.4). □

We observe that the initial point $\boldsymbol{w}_0$ in Theorem 4 can be taken, for instance, as that given by the Metropolis-Hastings algorithm (e.g., [8]). That is, if $\boldsymbol{w}_0$ is that given by the Metropolis-Hastings algorithm, then $[\boldsymbol{w}_0]_{k,1} = \frac{1}{\max\left(d_{i_k}, d_{j_k}\right)}$ for all $k \in \{1,2,\ldots,K\}$, where d_i is the degree of node $i \in \mathcal{V}$ (i.e., the number of nodes to which node i is connected). Therefore, $\boldsymbol{w}_0$ can be computed in a distributed way.

Table 1 relates Algorithm 1 with the theoretical aspects shown in this section.

Table 1. Explanation of Algorithm 1.

Lines	Description
2–4	Initialize with Metropolis-Hastings algorithm (Theorem 4)
7–10	Computation of the cost function f according to Theorem 3
11–13	Choose the correct subsequence according to Remark 1
14–15	Compute $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ as in Theorem 2
17–24	Obtain a unit eigenvector $\boldsymbol{y}$ from $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$
27	Compute subgradient as in Theorem 1
28	Update as in Equation (9)

Remark 1. *As f is continuous, from every initial sequence of real numbers $\{\beta_p\}$ with $\lim_{p\to\infty}\beta_p = 0$ and $\sum_{p=0}^{\infty}\beta_p = \infty$ (e.g. $\{\beta_p\} = \{1/\sqrt{p}\}$), a subsequence of stepsizes $\{\eta_p\} = \{\beta_{\sigma(p)}\}$ satisfying Equation (17) can be constructed.*

We finish the section by describing Algorithm 1. For ease of notation, we define

$$\text{ave}_{\boldsymbol{w}}(\boldsymbol{x}, t) := (\boldsymbol{W}(\boldsymbol{w}))^t\,\boldsymbol{x} \qquad t \in \{0,1,2,\ldots\},\ \boldsymbol{x} \in \mathbb{R}^{n\times 1},$$

which is the tth iteration of Equation (1) and can clearly be computed in a distributed way. As for Algorithm 1, we fix t_0 to be the number of iterations of Equation (1) required for a desired precision. We observe that because the worst possible network topology is a path, if we set $t_0 \geq \frac{\log\epsilon}{\log\cos(\pi/n)}$, then $\|\text{ave}_{\boldsymbol{w}}(\boldsymbol{x}, t_0) - x_{\text{ave}}\mathbf{1}_n\|_2 \leq \epsilon\|\boldsymbol{x}\|_2$ (see [15]), and therefore t_0 can also be obtained in a distributed way.

4. Numerical Results

We here present the numerical results obtained using Algorithm 1 for two networks with $n = 16$ nodes. The chosen starting point $\boldsymbol{w}_0$ was that given by the Metropolis-Hastings algorithm [8], and the chosen initial sequence of stepsizes was $\{\beta_p\} = \left\{\frac{1}{\sqrt{p}}\right\}$ for all $p \in \{1,2,\ldots\}$. Moreover, we took $t_0 = 250 \approx \frac{\log 10^{-2}}{\log\cos(\pi/16)}$.

Figure 1 shows the convergence time $\tau\left(\boldsymbol{W}(\boldsymbol{w}_p)\right)$ for the network presented in Figure 2 (solid line). Figure 1 also shows $\tau(\boldsymbol{W}_{\text{opt}}) = 10.03$, which was obtained by using CVX, a package for specifying and solving convex programs in a centralized way [16,17] (dashed line). Finally, Figure 1 also shows the minimum value of $\tau\left(\boldsymbol{W}(\boldsymbol{w}_p)\right)$ obtained up to step p (dotted line). For comparison purposes, we observe that the convergence time yielded by the Metropolis-Hastings algorithm was $\tau\left(\boldsymbol{W}(\boldsymbol{w}_0)\right) = 20.81$, while the minimum convergence time obtained after 150 iterations of our algorithm was 10.31.

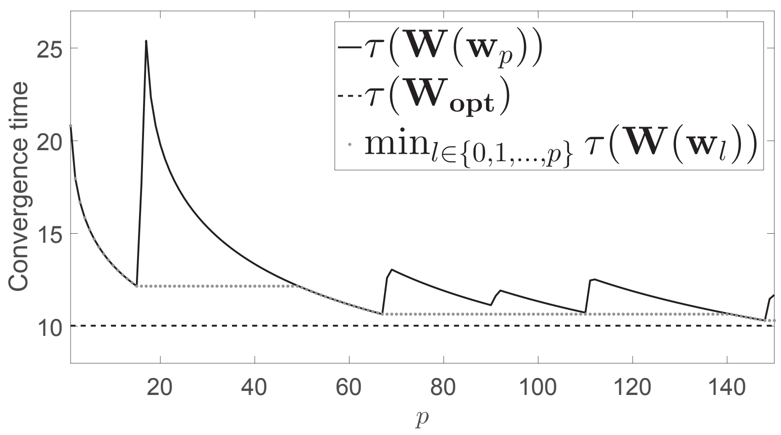

Figure 1. Numerical results for the graph of 16 nodes shown in Figure 2.

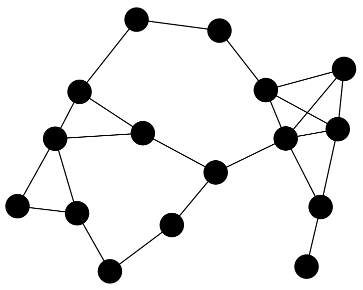

Figure 2. Graph with $n = 16$ nodes considered in Figure 1.

Figure 3 is of the same type as Figure 1, but in this case, the considered network was a 4×4 grid. In this case, if the problem is optimally solved in a centralized way it yields $\tau(\boldsymbol{W}_{\mathrm{opt}}) = 2.89$. The convergence time yielded by the Metropolis-Hastings algorithm was $\tau\left(\boldsymbol{W}(\boldsymbol{w}_0)\right) = 4.91$, while the minimum convergence time obtained after 150 iterations of our algorithm was 2.99.

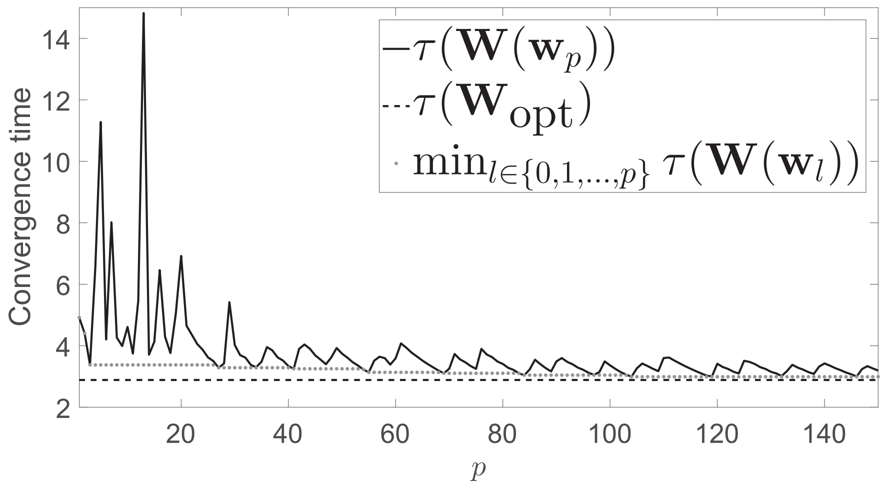

Figure 3. Numerical results for the grid of 16 nodes (4 rows and 4 columns).

We finish the section with a note on the number of exchanged messages (number of transmissions). For every iteration p of Algorithm 1, the number of exchanged messages per node was at most $5t_0$, divided as follows: t_0 message exchanges were required for lines 8 and 9, another $2t_0$ message exchanges were needed in line 10 (lines 14 and 15 did not require new message exchanges), and line 16 required another t_0 message exchanges. Finally, depending on the if-clause, another t_0 message exchanges were required in line 21. Therefore, the overall number of required transmissions per node was between $4p_{\max}t_0$ and $5p_{\max}t_0$.

5. Conclusions

In this paper we have provided an algorithm for the in-network computation of the optimal weighting matrix for distributed consensus. The algorithm can be viewed as an iterative repetition of, at most, five distributed consensus operations. Our algorithm is especially useful for networks that do not have a central entity and that change with time. In fact, if a network never changes with time (and its topology is known a priori), it seems easier to solve the FSDLA problem offline (in a centralized rather than a distributed way) using [6], and then pre-configuring the nodes with the obtained weights. However, if the network topology changes randomly with time (e.g., if sensors are added or removed) and there is no central entity, our algorithm would so far be the only way of obtaining the optimal solution to the FSDLA problem.

Acknowledgments: This work was supported in part by the Spanish Ministry of Economy and Competitiveness through the RACHEL project (TEC2013-47141-C4-2-R), the CARMEN project (TEC2016-75067-C4-3-R) and the COMONSENS network (TEC2015-69648-REDC).

Author Contributions: Xabier Insausti conceived the research question and performed the simulations. Xabier Insausti, Jesús Gutiérrez-Gutiérrez, and Marta Zárraga-Rodríguez proved the main results (Theorems 1 to 4). Xabier Insausti, Jesús Gutiérrez-Gutiérrez, Marta Zárraga-Rodríguez, and Pedro M. Crespo wrote the paper. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bertrand, A.; Moonen, M. Distributed computation of the Fiedler vector with application to topology inference in ad hoc networks. *Signal Process.* **2013**, *93*, 1106–1117.
2. Weng, Y.; Xiao, W.; Xie, L. Diffusion-Based EM Algorithm for Distributed Estimation of Gaussian Mixtures in Wireless Sensor Networks. *Sensors* **2011**, *11*, 6297–6316.
3. Mohammadi, A.; Asif, A. Consensus-based distributed dynamic sensor selection in decentralised sensor networks using the posterior Cramér-Rao lower bound. *Signal Process.* **2015**, *108*, 558–575.
4. Zeng, Y.; Hendriks, R.C. Distributed estimation of the inverse of the correlation matrix for privacy preserving beamforming. *Signal Process.* **2015**, *107*, 109–122.
5. Olshevsky, A.; Tsitsiklis, J. Convergence speed in distributed consensus and averaging. *SIAM Rev.* **2011**, *53*, 747–772.
6. Xiao, L.; Boyd, S. Fast linear iterations for distributed averaging. *Syst. Control Lett.* **2004**, *53*, 65–78.
7. Boyd, S.; Ghosh, A.; Prabhakar, B.; Shah, D. Randomized gossip algorithms. *IEEE Trans. Inf. Theory* **2006**, *52*, 2508–2530.
8. Xiao, L.; Boyd, S.; Kimb, S.J. Distributed average consensus with least-mean-square deviation. *J. Parallel Distrib. Comput.* **2007**, *67*, 33–46.
9. Bertrand, A.; Moonen, M. Topology-aware distributed adaptation of Laplacian weights for in-network averaging. In Proceedings of the 21st European Signal Processing Conference, Marrakech, Morocco, 9–13 September 2013; pp. 1–5.
10. Insausti, X.; Camaró, F.; Crespo, P.M.; Beferull-Lozano, B.; Gutiérrez-Gutiérrez, J. Distributed Pseudo-Gossip Algorithm and Finite-Length Computational Codes for Efficient In-Network Subspace Projection. *IEEE J. Sel. Top. Signal Process.* **2013**, *7*, 163–174.
11. Bernstein, D.S. *Matrix Mathematics*; Princeton University Press: Princeton, NJ, USA, 2009.
12. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
13. Shor, N.Z. *Minimization Methods for Non-Differentiable Functions*; Springer: Berlin/Heidelberg, Germany, 1985.
14. Zhang, S.; Tepedelenlioğlu, C.; Banavar, M.K.; Spanias, A. Distributed Node Counting in Wireless Sensor Networks in the Presence of Communication Noise. *IEEE Sens. J.* **2017**, *17*, 1175–1186.
15. Boyd, S.; Diaconis, P.; Sun, J.; Xiao, L. Fastest mixing Markov chain on a path. *Am. Math. Mon.* **2006**, *113*, 70–74.
16. Grant, M.; Boyd, S. CVX: Matlab Software for Disciplined Convex Programming, Version 2.1. Available online: http://cvxr.com/cvx (accessed on 1 March 2017).
17. Grant, M.; Boyd, S. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*; Lecture Notes in Control and Information Sciences; Blondel, V., Boyd, S., Kimura, H., Eds.; Springer: London, UK, 2008; pp. 95–110.

© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).