

Article

# A Distributed Algorithm for Reaching Average Consensus in Unbalanced Tree Networks

Gianfranco Parlangei 

Department of Engineering for Innovation, University of Salento, Via per Monteroni, 73100 Lecce, Italy; gianfranco.parlangeli@unisalento.it; Tel.: +39-0832-297301

**Abstract:** In this paper, a distributed algorithm for reaching average consensus is proposed for multi-agent systems with tree communication graph, when the edge weight distribution is unbalanced. First, the problem is introduced as a key topic of core algorithms for several modern scenarios. Then, the relative solution is proposed as a finite-time algorithm, which can be included in any application as a preliminary setup routine, and it is well-suited to be integrated with other adaptive setup routines, thus making the proposed solution useful in several practical applications. A special focus is devoted to the integration of the proposed method with a recent Laplacian eigenvalue allocation algorithm, and the implementation of the overall approach in a wireless sensor network framework. Finally, a worked example is provided, showing the significance of this approach for reaching a more precise average consensus in uncertain scenarios.

**Keywords:** laplacian eigenvectors; Perron vector; average consensus problems; multi-agent systems; tree graphs; spectral graph theory



**Citation:** Parlangei, G. A Distributed Algorithm for Reaching Average Consensus in Unbalanced Tree Networks. *Electronics* **2024**, *13*, 4114. <https://doi.org/10.3390/electronics13204114>

Academic Editors: José Sánchez Moreno, María Guinaldo and Raquel Dormido Canto

Received: 13 August 2024  
Revised: 8 October 2024  
Accepted: 14 October 2024  
Published: 18 October 2024



**Copyright:** © 2024 by the author. 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 (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In the last decades, research on the collective behavior of a team of agents through iterative local interactions has been intense, due to the significance of applications in a wide range of fields, from computer science to social networks, from complex technological networks (e.g., electric smart grid), to biological ecosystems [1]. In each framework, a fundamental property to reach agreement among all agents through local interactions is the *consensus* condition, which happens when all agents recursively update a local variable using local information to asymptotically obtain a common value for all participants [2].

One special yet fundamental consensus condition regards the value of such agreements as a function of the initial conditions of all nodes, and, in particular, the average of all initial conditions of the agents is known in the literature as the *average consensus* [3,4]. This special condition is adopted in a large number of applications of distributed estimation, WSN algorithms, robotics, and many more [5].

Average consensus represents the natural equilibrium for a multi-agent system when the communication graph is undirected and unweighted, or more generally, when it is weight-balanced. This condition enables the distributed achievement of various fundamental goals [1]. Even though the average consensus is a well-known condition of a multi-agent system, it is still discussed and debated over the years until our days due to its importance. In [6], the authors study the privacy-preserving problem, namely an algorithm ensuring to reach exact average consensus while keeping the initial state of each agent private. The problem of average consensus is studied in [7] in the presence of communication link failures, and in [8], the same problem is studied in the presence of node faults. In [9,10], the average consensus problem is studied within the wireless sensor network application framework.

Unfortunately, one fundamental limitation when dealing with classical average consensus through the use of undirected communication graphs with the traditional symmetric

edge weight assignment is a limited convergence rate, which significantly decreases when the number of participants grows [5]. This condition leads to a prolonged convergence time, which, in practice, can render consensus unattainable when factoring in the inevitable communication delays, computational errors, or agent faults over extended periods [8,11].

However, the practical significance of this condition for the large number of the applications referenced above pushed the scientific community to a significant effort to derive sophisticated algorithms for general unbalanced graphs, as, for example, through the use of additional state variables [12], dedicated atomic transactions between pairs of neighbors [13], a chain of two integrators that are coupled with a distributed estimator [14] average tracking with incomplete measurement through a distributed averaging filter and a decentralized tracking controller [15], and recently also considering signed networks [16].

In the last few years, advances have been made in adopting the strategy of assigning asymmetric weights to neighbors' edges [17]. Indeed, it was found that a proper choice of asymmetric weights can significantly improve the convergence rate, even over the symmetric optimal design [18], and it makes the convergence rate independent of the size of the graph with asymmetric weights, which is in striking contrast with the fundamental limitations of symmetric weights [1]. On the other hand, the choice of asymmetric weights makes the network converge to a different function of the nodes rather than the average of the initial conditions.

In general, there are several scenarios where edge weight asymmetry should be taken into account, other than design choice. Indeed, empirical research clearly shows that real-world networks usually have a large heterogeneity in the intensity or capacity of connections (and hence weights of the links) [19]. These asymmetries are related to defects and uncertainties in the network, and they result in an inaccurate and biased consensus condition (namely, the asymptotic values of network nodes are only roughly close to each other). The technique proposed in this paper can be useful in these scenarios, as it would produce a much more precise consensus value.

The theoretical backbone of the proposed approach is the distributed estimation of the Perron vector, which allows for a scaling of the initial conditions to reach an average consensus, thus combining the beneficial effects of asymmetric edge weight assignment with the large practical applications of average consensus problems. It is worth remarking that the use of the Perron vector to achieve a more precise average estimation is adopted also in [20] to compensate for inaccurate sensor readings. The results developed in this paper are inspired by [21], as described in detail in Section 3. The approach proposed in this paper can be combined with an asymmetric design of the edge weights as in [22] to achieve the ambitious goal of average consensus with prescribed dynamics, as explained in the Problem Statement description; analogous design techniques have recently been adopted to achieve prescribed-time consensus [23].

This paper is organized as follows. After a brief description of the notation hereafter, in Section 2, some motivating examples of the research pursued in this paper are described, and hence, two facets of the problem are stated. In Section 3, the main theoretical results are provided for two graph topologies, namely path graphs and star graphs, which are prodromal for the inductive general solution, whose theoretical backbone is given in Section 4. These results are then exploited in Section 5, where the iterative algorithm is deduced through a worked example. A final simulation is provided in Section 6, where the application of the results to an uncertain WSN network, computing the average value of an environmental quantity, shows the effectiveness of the proposed results to achieve a more precise consensus value.

### Notation

Here, we briefly describe the notation adopted in this paper. We denote by  $\mathbb{N}$ ,  $\mathbb{R}$ ,  $\mathbb{R}_{\geq 0}$ ,  $\mathbb{R}_+$  the set of natural, real numbers and non-negative real numbers, and positive real numbers. We denote by  $\mathbf{0}_d$ ,  $d \in \mathbb{N}$  the  $d$ -dimensional vectors with components equal to 0 and by  $\mathbf{1}_d$  those vectors whose components are all 1.  $\mathbf{0}_{d_1 \times d_2}$ ,  $d_1, d_2 \in \mathbb{N}$ , is the

$d_1 \times d_2$  matrix made by all zero entries. Vector  $\mathbf{e}_i$  denotes the  $i$ -th canonical vector, e.g.,  $\mathbf{e}_1 = [1 \ 0 \ \dots \ 0]^\top$ . Vectors are denoted in bold letters. For a vector  $\mathbf{v} \in \mathbb{R}^d$ , we denote  $v_i$  the  $i$ th component of  $\mathbf{v}$  so that  $\mathbf{v} = [v_1 \dots v_d]^\top$ . The *spectral radius* of  $A$  is the maximum modulus of its eigenvalues, namely  $\rho_A = \max_{i \in \{1, \dots, n\}} \{|\lambda_i|\}$  where  $\lambda_i \in \Lambda_A$ .

A non-negative (positive) matrix  $A \in \mathbb{R}^{n \times n}$  satisfies  $[A]_{ij} \geq 0$  ( $[A]_{ij} > 0$ ). A permutation matrix  $P$  is a matrix obtained by permuting the rows of the  $n \times n$  identity matrix. A non-negative matrix  $A \in \mathbb{R}^{n \times n}$  is *reducible* if there exist a permutation matrix  $P$  such that the matrix  $PAP^\top$  is in block triangular form, i.e.,

$$PAP^\top = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$

where  $B_{11}, B_{22}$  are square matrices. An *irreducible matrix* is a matrix that is not reducible. A non-negative matrix  $A \in \mathbb{R}^{n \times n}$  is *primitive* if there exists a positive integer  $m$  such that  $A^m > 0$ .

A graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is made of a *vertex set*  $\mathcal{V} = \{1, 2, \dots, n\}$  and an *edge set*  $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ . It is *undirected* if  $(j, i) \in \mathcal{E}$  if and only if  $(i, j) \in \mathcal{E}$ . The *neighbors set* of a node  $i \in \mathcal{V}$  is defined as  $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ . A *path* connecting vertex  $j_1$  with  $j_{k+1}$  is a subset of nodes connected by graph edges  $\{j_\ell \in \mathcal{V} \text{ such that } (j_\ell, j_{\ell+1}) \in \mathcal{E}\}_{\ell \in [1, k]}$ . An undirected graph  $\mathcal{G}$  is *connected* if there is a path connecting  $i$  and  $j$  for every pair of vertices  $i, j \in \mathcal{V}$  such that  $i \neq j$ .

An undirected graph  $\mathcal{T}$  with no cycles is called a *tree* if it is connected, otherwise it is called a *forest*. For a vertex  $w$  of a tree  $\mathcal{T}$ ,  $\mathcal{T}_w$  denotes the forest obtained from  $\mathcal{T}$  by deleting  $w$ , and it is made of  $|\mathcal{N}_w|$  trees. For any  $v$  neighbor of  $w$ ,  $v \in \mathcal{N}_w$ ,  $\mathcal{T}_v$  denotes the subtree of  $\mathcal{T}_w$  having  $v$  as a vertex. The center (or Jordan center) of a graph is the set of all vertices  $u \in \mathcal{V}$  where the greatest distance  $d(u, v)$  to other vertices  $v \in \mathcal{V}$  is minimal. Equivalently, it is the set of vertices with eccentricity equal to the graph's radius. Every tree has a center consisting of one vertex or two adjacent vertices. The center is the middle vertex or middle two vertices in every longest path.

For a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , the *adjacency matrix*  $A \in \mathbb{R}^{n \times n}$  is the matrix  $[A]_{ij} = 1$  if  $(i, j) \in \mathcal{E}$  and  $[A]_{ij} = 0$  if  $(i, j) \notin \mathcal{E}$ , the *weighted adjacency matrix* as  $[A_w]_{ij} = \alpha_{ij}$  with  $\alpha_{ij} > 0$  if  $(i, j) \in \mathcal{E}$  and  $[A_w]_{ij} = 0$  if  $(i, j) \notin \mathcal{E}$ . The *weighted Laplacian matrix* is built upon the rule  $[L_w]_{ij} = -[A_w]_{ij}$  if  $i \neq j$ ,  $[L_w]_{ii} = \sum_{j=1, i \neq j}^n [A_w]_{ij}$ . By construction,  $L_w \mathbf{1} = \mathbf{0}$ , i.e., the Laplacian is a *zero-row-sum* matrix. In the following, given a graph  $\mathcal{G}$ ,  $\mathcal{L}_w(\mathcal{G})$  denotes the set of all  $n \times n$  *zero-row-sum  $\mathcal{G}$ -structured square matrices*. There are connections between the properties of non-negative matrices and their related graphs explaining their zero-nonzero pattern, as the following result shows [1]:

**Lemma 1.** *A matrix  $A \in \mathbb{R}^{n \times n}$  is irreducible if and only if its relative graph  $\mathcal{G}_A$  encoding its zero-nonzero pattern is strongly connected.*

## 2. Motivating Examples

In this section, we introduce the abstract mathematical framework of our problem, namely average consensus for a multi-agent system running a distributed iterative algorithm to accomplish a global task, which is usually in terms of network coordination or synchronization. After that, several applications in modern fields adopting the described framework are reported.

### 2.1. Multi-Agent Systems Running Consensus Algorithms

Here, we introduce the mathematical framework of multi-agent systems running a consensus algorithm [1,2].

This setting is made of a set of  $N$  nodes, each holding a variable  $x_i(t)$ ,  $i \in \{1, \dots, N\}$  updated as  $\dot{x}_i(t) = u_i(t)$  where  $u_i(t)$  is set by each node  $i$  to update its value. Some nodes of the team can communicate between themselves, and two *neighbor nodes*  $i, j$

are able to exchange their values. In this setting, each node  $i$  assigns as input  $u_i(t) = \sum_{j \in \mathcal{N}_i} k_{ij}(x_j(t) - x_i(t))$  where  $k_{ij}$  are set by node  $i$ . The resulting evolution of the team can be effectively computed using the aggregate vector  $\mathbf{x} \in \mathbb{R}^N$  chosen as  $[\mathbf{x}]_i(t) = x_i(t)$ , and updated as  $\dot{\mathbf{x}}(t) = -L_\kappa \mathbf{x}(t)$ , which gives rise to the *Laplacian flow* [1], as follows:

$$\begin{aligned} \mathbf{x}(t) &= \Phi(t)\mathbf{x}(0), \\ \Phi(t) &= e^{-L_\kappa t} = \sum_{i=0}^{\infty} \frac{(-L_\kappa)^i t^i}{i!}, \end{aligned} \tag{1}$$

for a  $L_\kappa$  in the set  $\mathcal{L}_w$ . Network evolution is dictated by the spectrum of  $L_\kappa$  [24]. It is worth mentioning that the design of asymmetric gains was recently proposed [18,25,26] as a key feature to improve the convergence rate.

Other multi-agent processes are better described by discrete updates of a local quantity, namely  $\mathbf{x}(t+1) = (I - \gamma L_\kappa)\mathbf{x}(t)$ , where  $\gamma \in \mathbb{R}_+$  is called *coupling factor*, and it must be sufficiently small to keep the system stable [1], and analogously in this case one obtains the following global evolution:

$$\begin{aligned} \mathbf{x}(t) &= \Phi(t)\mathbf{x}(0), \\ \Phi(t) &= (I - \gamma L_\kappa)^t. \end{aligned} \tag{2}$$

Both in case of (1) and (2), matrix  $\Phi(t)$  shows some interesting properties inherited from the properties of  $L_\kappa$  that  $L_\kappa \mathbf{1} = \mathbf{0}$ ,  $\mathbf{w}_{L_\kappa}^\top L_\kappa = \mathbf{0}^\top$ , namely [1]:

$$\Phi \mathbf{1} = \mathbf{1}, \quad \mathbf{w}_{L_\kappa}^\top \Phi = \mathbf{w}_{L_\kappa}^\top, \tag{3}$$

and matrix  $\Phi(t)$  is positive if  $\mathcal{G}$  is strongly connected. Moreover, even if the eigenvalues of matrix  $\Phi(t)$  are related to those of  $L_\kappa$  according to  $\lambda_\Phi = e^{-\lambda_{L_\kappa} t}$  for systems (1) and  $\lambda_\Phi = (1 - \gamma \lambda_{L_\kappa})^t$  when dealing with (2), eigenvectors of (1) and (2) do coincide.

By direct consequence of (3), system (1) and (2) are called *consensus networks*, namely for every choice of the initial conditions  $x_i(0), i = 1, 2, \dots, N$ , there exists  $\alpha \in \mathbb{R}$  such that

$$\lim_{t \rightarrow +\infty} x_i(t) = \alpha, \quad \forall i \in \{1, \dots, N\}, \tag{4}$$

or, equivalently,  $\lim_{t \rightarrow +\infty} \mathbf{x}(t) = \alpha \mathbf{1}_N, \exists \alpha \in \mathbb{R}$ . The constant  $\alpha$  is called the *consensus value* (or *collective decision*) [2] for system (1), corresponding to the given initial conditions. The consensus value is a key tool in many applications, indeed it encapsulates a global information, equal to

$$\alpha = \frac{\mathbf{w}_{L_\kappa}^\top \mathbf{x}(0)}{\mathbf{w}_{L_\kappa}^\top \mathbf{1}_N}, \tag{5}$$

where  $\mathbf{w}_{L_\kappa}$  is a left eigenvector of  $\Phi(t)$  associated to the unitary eigenvalue and satisfying the normal condition  $\mathbf{w}_{L_\kappa}^\top \mathbf{1} = 1$  [27]. In a wide number of applications, it holds  $\mathbf{w}_{L_\kappa} = \frac{1}{N} \mathbf{1}$  and Equation (5) simplifies to the average of the initial conditions of the whole network, i.e.,

$$\alpha = \sum_{i=1}^N \frac{x_i(0)}{N}, \tag{6}$$

and we refer to this condition as *average consensus*.

Average consensus has become a very popular and significant tool in many applications, as for example, sensor fusion and data aggregation, distributed optimization and machine learning, collective motion, vehicle coordination, and more [1]. However, one main issue of multi-agent systems running standard consensus protocols is a low convergence rate, which is related to the algebraic connectivity of graph  $\mathcal{G}$ , and it is often a key limiting feature of the original consensus protocol [1].

The fundamental mathematical background for our analysis is the renowned Perron–Frobenius Theorem [1]:

**Theorem 1** (Perron–Frobenius). *Let  $A \in \mathbb{R}^{n \times n}$  be an irreducible non-negative matrix. Then*

- *$A$  has a positive eigenvalue equal to  $\rho(A)$ , and it is a simple eigenvalue of  $A$ .*
- *The left and right eigenvectors relative to the eigenvalue  $\rho(A)$  are positive.*

It is worth remarking that, for any non-negative matrix  $A \in \mathbb{R}^{n \times n}$ ,  $\rho(A)$  is an eigenvalue of  $A$  and the right and left eigenvectors of  $\lambda = \rho(A)$  can be selected as non-negative. If  $A$  is additionally irreducible, then the multiplicity of  $\lambda = \rho(A)$  is one and the relative left and right eigenvector are strictly positive and uniquely determined (up to a scalar factor).

In general, the positive left and right eigenvectors relative to the eigenvalue  $\rho(A)$  are called *left* and *right Perron vectors*. However, considering Equation (3), vector  $\mathbf{1}$  is always a right eigenvector of  $\Phi$  so that in the following, we refer to the *left* eigenvector of  $1 = \rho(A)$  as the *Perron vector* of  $\Phi$  (e.g., vector  $\mathbf{w}_{L_\kappa}$  in Equation (3)).

In the remainder of this paper, for a given Laplacian matrix  $L_w$  related to a connected graph  $\mathcal{G}$ , we denote by  $\mathbf{w}$  its left eigenvector corresponding to  $\lambda = 0$ . If matrix  $L_w$  is the generator of a dynamical flow  $\mathbf{x}(t) = e^{-L_w t} \mathbf{x}(0)$ , then  $\mathbf{w}$  is the Perron vector of the positive matrix  $e^{-L_w}$ , and with a little abuse of notation we refer to  $\mathbf{w}$  as the *Perron vector* of  $L_w$ .

The importance of the Perron vector in such applications is that it explains how the final distribution is [28], so it allows for detecting aggregations, clusters, or, conversely, the weight of each node influence on the asymptotic consensus value.

In the following, we provide some recent technological applications where the above framework has been exploited to design decentralized average consensus algorithms.

## 2.2. Some Examples of Modern Applications

The above setting is the abstract representation of several technological modern applications, where average consensus is the key methodology for distributed algorithm architectures, as described in the following.

In [5], the authors survey the scientific literature on distributed estimation and control applications using linear consensus algorithms. In this paper, there are interesting examples of how some classical estimation and control problems can be rephrased as the average value of some suitable quantities, thus allowing for being efficiently computed in a distributed fashion through average consensus algorithms.

Average consensus is the key strategy for sharing global information through local interactions. In the field of wireless sensor networks, clock synchronization is a fundamental problem and it can be efficiently solved through consensus [29]. In [30], the same synchronization problem is solved through average consensus in IoT applications. Within the above application scenarios, average consensus is also adopted for achieving the estimation of an environmental quantity [13].

In the field of electric grids, smart power grids, and renewable energies, average consensus-based algorithms are widely adopted for the following fundamental issues: generator synchronization [31], economic dispatch [32], and clock synchronization [33].

Finally, several tasks of great value in the area of robotic networks can be accomplished through the exploitation of average consensus. Among the others, it is worth mentioning rendezvous and, conversely, the deployment of a team of robots, which is fundamental for surveillance and, in general, for coverage purposes. Also, attaining and keeping a formation can be executed through properly designed average consensus protocols, without recurring to a supervisory device [34].

## 2.3. Problem Statement

Inspired by the applications discussed in the previous paragraph, we are now ready to state the problem that we afford in the remainder of this paper.

### Problem Statement 1.

Given a weighted Laplacian matrix  $L_\kappa \in \mathcal{L}$  of a tree graph  $\mathcal{T}$ , compute the (left) Perron vector of the positive matrix  $\Phi(t)$ , either in the case of (1) or (2).

However, in view of the motivations discussed above and considering the applications described so far, Problem Statement 1 can be rephrased in a more application-oriented engineering fashion, as follows:

**Problem Statement 2.**

Given a weighted tree graph  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$  and a multi-agent system (either (1) or (2)) running consensus algorithm so that each node asymptotically reaches the value  $\alpha$  as in (4) with  $\alpha$  equal to (5), compute a set of scaling factors  $\zeta_i, i = 1, \dots, N$  such that the state vector  $\chi(t)$  defined as  $(\chi(t))_i = \frac{(\mathbf{x}(t))_i}{\zeta_i}$  converges to  $\chi(t) \xrightarrow{t \rightarrow \infty} \alpha \mathbf{1}$  with

$$\alpha = \sum_{i=1}^N \frac{\mathbf{x}(0)_i}{N},$$

thus solving the average consensus problem for the original problem. The above two problems are indeed two facets of the same argument, the first formulation being closer to a mathematical fashion while the second one is stated in an engineering style. It is worth noting that Statement 2 can also be effectively encapsulated in the Problem Statement of [22], which is related to the Laplacian eigenvalue allocation by a proper asymmetric edge weights assignment, so that the two problems can be integrated into each other as follows:

**Problem Statement 2-bis.** Given a tree graph  $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ , and one node  $\ell$  triggering the distributed algorithm in [35] to allocate the Laplace spectrum and grounded Laplacian to, resp.,  $\lambda_i$  and  $\mu_i \in \mathbb{R}$  (satisfying  $0 < \mu_1 < \lambda_2 < \dots < \mu_{n-1} < \lambda_n$ ), compute a set of scaling factors  $\zeta_i, i = 1, \dots, N$  such that the state vector  $\chi(t)$  defined as  $(\chi(t))_i = \frac{(\mathbf{x}(t))_i}{\zeta_i}$  converges to  $\chi(t) \xrightarrow{t \rightarrow \infty} \alpha \mathbf{1}$  with  $\alpha = \sum_{i=1}^N \frac{\mathbf{x}(0)_i}{N}$ , thus solving the average consensus problem with prescribed dynamics adopting the asymmetric weight distribution.

**Remark 1.** The connection and equivalence of the different Problem Statements stated above is clearly established by considering Equations (5) and (6). Namely, the asymptotic value of  $\chi(t)$  is  $\chi(t) \xrightarrow{t \rightarrow \infty} \alpha \mathbf{1}$  with  $\alpha = \sum_{i=1}^N \frac{\mathbf{w}^\top \chi(0)_i}{\mathbf{w}^\top \mathbf{1}}$ , so that, considering that  $(\chi(0))_i = \frac{(\mathbf{x}(0))_i}{\zeta_i}$  and imposing

$$\frac{w_1}{\mathbf{w}^\top \mathbf{1}} \chi_1(0) + \frac{w_2}{\mathbf{w}^\top \mathbf{1}} \chi_2(0) + \dots = \frac{1}{N} x_1(0) + \frac{1}{N} x_2(0) + \dots, \tag{7}$$

one has that the system evolution seeks average consensus for any set of initial conditions  $x_i(0)$  if the scaling factors  $\zeta_i$  are set equal to

$$\zeta_i = N \frac{w_i}{\mathbf{1}^\top \mathbf{w}}. \tag{8}$$

**Remark 2.** Even if it is not explicitly stated in the above Problem Statement, a further feature of the proposed algorithm is that it can be implemented distributedly, namely, each component of the scaling vector can be computed through the use of local data.

**3. Problem Solution**

We start by seeking the solution of some special topologies, namely path graphs and star graphs, which constitute the topology of the most peripheral branches of any tree, as well as widely adopted graph topologies in general [1]. We further extend the solution to a generic  $n$  tree graph.

In the case of path and star graph, we are able to provide the full characterization of the Perron vector by exploiting some recent mathematical results concerning structured matrix manipulation and inversion.

On the other hand, the general solution for tree graphs is inductive, and it can be implemented through an iterative algorithm.

### 3.1. Solution for Path Graphs

The Laplacian matrix of path graphs is a special case of tridiagonal matrix, so the theoretical background for the construction of the solution is based on [36], adapted to the special structure of our matrices. The following statement is proven in [36], and it is our main mathematical tool to obtain the explicit solution of the path graph.

**Proposition 1** ([36]). *Consider the following tridiagonal matrix:*

$$L = \begin{bmatrix} a_1 & b_1 & 0 & \dots \\ c_1 & a_2 & b_2 & \\ & & \ddots & b_{n-1} \\ \dots & 0 & c_{n-1} & a_n \end{bmatrix},$$

and assume that  $L$  is nonsingular. Build the two following sequences:

$$\begin{aligned} \theta_i &= a_i\theta_{i-1} - b_{i-1}c_{i-1}\theta_{i-2}, & \text{for } i = 2, \dots, n, \\ &\text{with initial conditions } \theta_0 = 1 \text{ and } \theta_1 = a_1, \end{aligned}$$

and

$$\begin{aligned} \varphi_i &= a_i\varphi_{i+1} - b_i c_i \varphi_{i+2}, & \text{for } i = n - 1, \dots, 1, \\ &\text{with initial conditions } \varphi_{n+1} = 1 \text{ and } \varphi_n = a_n, \end{aligned}$$

then, the inverse of matrix  $L$  can be computed using the following formula:

$$[L^{-1}]_{ij} = \begin{cases} (-1)^{i+j} b_i \dots b_{j-1} \theta_{i-1} \varphi_{j+1} / \theta_n, & \text{if } i < j, \\ \theta_{i-1} \varphi_{j+1} / \theta_n, & \text{if } i = j, \\ (-1)^{i+j} c_j \dots c_{i-1} \theta_{j-1} \varphi_{i+1} / \theta_n, & \text{if } i > j. \end{cases} \tag{9}$$

It is possible to exploit the above result for the computation of the Perron vector in the case of path graphs, as follows.

**Proposition 2.** *Let  $\alpha_{i,j} \in \mathbb{R}_+$ , and consider the Laplacian matrix:*

$$L_w = \begin{bmatrix} \alpha_{1,2} & -\alpha_{1,2} & & \dots \\ -\alpha_{2,1} & a_{2,2} & & \\ & & \ddots & -\alpha_{(n-1),n} \\ \dots & 0 & -\alpha_{n,(n-1)} & \alpha_{n,(n-1)} \end{bmatrix}, \tag{10}$$

with diagonal entries  $a_{i,i} = \alpha_{i,(i-1)} + \alpha_{i,(i+1)}$  for  $i = 2, \dots, n - 1$ . The corresponding (left) Perron vector is equal to

$$\mathbf{w} = \kappa \left[ 1 \quad \frac{\alpha_{12}}{\alpha_{21}} \quad \frac{\alpha_{23}\alpha_{12}}{\alpha_{32}\alpha_{21}} \quad \dots \quad \prod_{j=2}^i \frac{\alpha_{(j-1),j}}{\alpha_{j,(j-1)}} \quad \dots \right], \quad \kappa \in \mathbb{R} - \{0\}. \tag{11}$$

**Proof.** The proof is conducted by direct inspection on the relation  $\mathbf{w}^\top L_w$  with  $L_w$  as in (10) for a nonzero  $\mathbf{w} \in \mathbb{R}^n$ . Consider the partition of  $L_w$  as follows:

$$L_w = \begin{bmatrix} \alpha_{12} & | & L_{1*} \\ \hline L_{*1} & | & L_1 \end{bmatrix},$$

with  $L_{1*} = [-\alpha_{12} \ 0 \ \dots \ 0]$ ,  $L_{*1} = [-\alpha_{21} \ 0 \ \dots \ 0]^\top$  and

$$L_1 = \begin{bmatrix} \alpha_{2,1} + \alpha_{23} & -\alpha_{2,3} & 0 & \dots \\ -\alpha_{3,2} & \alpha_{3,3} & & \\ 0 & & \ddots & -\alpha_{(n-1),n} \\ \dots & 0 & -\alpha_{n,(n-1)} & \alpha_{n,(n-1)} \end{bmatrix},$$

so that, taking  $\mathbf{w} = [w_1 \ \tilde{\mathbf{w}}_1]$  with  $w_1 \in \mathbb{R}$ ,  $\tilde{\mathbf{w}}_1 \in \mathbb{R}^{n-1}$ , one has

$$\mathbf{w}^\top L_w = 0 \Rightarrow \begin{cases} w_1 \alpha_{12} + \tilde{\mathbf{w}}_1^\top L_{*1} = 0; \\ w_1 L_{1*} + \tilde{\mathbf{w}}_1^\top L_1 = 0 \end{cases}; \tag{12}$$

from the second relation it is possible to parametrize  $\mathbf{w}$  as

$$\mathbf{w}^\top = \kappa [1 \ L_{*1} L_1^{-1}], \quad \text{with} \quad \kappa \in \mathbb{R} - \{0\}. \tag{13}$$

Considering that  $L_{1*} = -\alpha_{12} \mathbf{e}_1^\top$ , it is possible to find an explicit expression for (13) by computing the first row of  $L_1^{-1}$  through (9) to obtain

$$L_{*1} L_1^{-1} = (-\alpha_{12}) \left[ -\frac{1}{\alpha_{21}} \quad -\frac{\alpha_{23}}{\alpha_{21} \alpha_{32}} \quad \dots \quad -\frac{\alpha_{23} \dots \alpha_{(n-1),n}}{\alpha_{21} \alpha_{32} \dots \alpha_{n,(n-1)}} \right], \tag{14}$$

so that, putting together (13) with (14), Equation (11) follows.  $\square$

### 3.2. Solution for Star Graphs

The explicit parametric structure of the Perron vector can be easily computed by direct inspection in the case of star graphs. Indeed, it holds

**Proposition 3.** *Let  $\mathcal{G}$  be a star graph having  $n$  rays, and suppose that the central node is selected as the first node. Let  $L_w$  be its corresponding Laplacian matrix for any choice of edge weights, so that*

$$L_w = \begin{bmatrix} a_{11} & -\alpha_{12} & \dots & -\alpha_{1n} \\ -\alpha_{21} & \alpha_{21} & 0 & \dots \\ \vdots & 0 & \ddots & \\ -\alpha_{n1} & 0 & \dots & \alpha_{n1} \end{bmatrix}, \tag{15}$$

where

$$a_{11} = \sum_{j=2}^n \alpha_{1j}.$$

Then, the corresponding Perron vector is equal to

$$\mathbf{w} = \kappa \left[ 1 \quad \frac{\alpha_{12}}{\alpha_{21}} \quad \frac{\alpha_{13}}{\alpha_{31}} \quad \dots \quad \frac{\alpha_{1,(j+1)}}{\alpha_{(j+1),1}} \quad \dots \right], \quad \kappa \in \mathbb{R} - \{0\}. \tag{16}$$

**Proof.** The proof is conducted by direct inspection on the relation  $\mathbf{w}^\top L_w$ , with  $L_w$  as in (15) for a nonzero  $\mathbf{w} \in \mathbb{R}^n$ . Consider the partition of  $L_w$  as

$$L_w = \begin{bmatrix} a_{11} & L_{1*} \\ L_{*1} & L_1 \end{bmatrix},$$

with  $L_{1*} = [-\alpha_{12} \ -\alpha_{13} \ \dots \ -\alpha_{1n}]$ ,  $L_{*1} = [-\alpha_{21} \ -\alpha_{31} \ \dots \ -\alpha_{n1}]^\top$  and

$$L_1 = \begin{bmatrix} \alpha_{2,1} & 0 & 0 & \dots \\ & \alpha_{3,1} & 0 & \\ 0 & & \ddots & 0 \\ \dots & 0 & 0 & \alpha_{n,1} \end{bmatrix} = \text{diag}(\{\alpha_{j,1}\}_{j=2,\dots,n}), \tag{17}$$

so that, taking  $\mathbf{w} = [w_1 \ \tilde{\mathbf{w}}_1]$  with  $w_1 \in \mathbb{R}$ ,  $\tilde{\mathbf{w}}_1 \in \mathbb{R}^{n-1}$ , it is possible to analogously parametrize  $\mathbf{w}$  as

$$\mathbf{w}^\top = \kappa [1 \quad -L_{*1}L_1^{-1}] \quad \text{with} \quad \kappa \in \mathbb{R} - \{0\}. \tag{18}$$

and considering (17) and (18), one has

$$L_{*1}L_1^{-1} = \begin{bmatrix} -\frac{\alpha_{12}}{\alpha_{21}} & -\frac{\alpha_{13}}{\alpha_{31}} & \dots & -\frac{\alpha_{1n}}{\alpha_{n1}} \end{bmatrix}. \tag{19}$$

Putting together Equation (19) with Equation (18), Equation (16) follows.  $\square$

#### 4. A Recursive General Solution Tree Graphs

In this section, we derive a solution for a general tree graph  $\mathcal{T}$ . The structure of the solution in such a case is not in a closed form as for path and star graphs, but it is based on an inductive process over  $\mathbb{N}$ , so that the resulting algorithmic solution is recursive. The proposed approach is inspired by the subdivision of the computation of the Perron vector into smaller problems of [21], and it perfectly fits the eigenvalue allocation algorithm in [22], so it can be integrated into the final algorithmic solution, thus solving the ambitious problem of eigenvalue allocation for average consensus problems.

**Proposition 4.** Consider a tree graph  $\mathcal{T}$ ,  $L_w$  its corresponding weighted Laplacian, let node 1 be a reference node with neighbor set  $N_1 = \{i_1, \dots, i_\ell\}$ , so that  $\ell = |N_1|$ . Let  $\mathcal{T}_j, j = 1, \dots, \ell$  represent the subtrees of  $\mathcal{T}$  after remotion of node 1, each made of  $\ell_j = |\mathcal{T}_j|$  nodes, and finally  $L_w^j \in \mathbb{R}^{\ell_j \times \ell_j}$  the submatrix of  $L_w$  being the Laplacian of  $\mathcal{T}_j$ .

If we denote by  $\mathbf{w} = [w_1 \ \tilde{\mathbf{w}}_1 \ \dots \ \tilde{\mathbf{w}}_\ell]$  a Perron vector for  $\mathcal{T}$ , where  $w_1 \in \mathbb{R}$  and each  $\tilde{\mathbf{w}}_j \in \mathbb{R}^{\ell_j}$ , then it holds that

- Each  $\tilde{\mathbf{w}}_j$  is a Perron vector for each  $L_w^j$ ;
- The first component of each  $\tilde{\mathbf{w}}_j$  satisfies

$$(\tilde{\mathbf{w}}_j)_1 = w_1 \frac{\alpha_{1i_j}}{\alpha_{i_j1}}, \quad j = 1, \dots, \ell. \tag{20}$$

**Proof.** Let  $N_1 = \{i_1, \dots, i_\ell\}$ , so that convenient labeling allows for writing the Laplacian matrix without loss of generality as

$$L_w = \begin{bmatrix} a_{11} & -\alpha_{1i_1} & 0 & \dots & \dots & -\alpha_{1i_\ell} & 0 & \dots \\ -\alpha_{i_11} & & & & & & & \\ 0 & \bar{L}_{i_1} & & \dots & & \mathbf{0}_{i_1 \times i_\ell} & & \\ \vdots & & & & & & & \\ & & \vdots & & & & \vdots & \\ -\alpha_{i_\ell1} & & & & & & & \\ 0 & \mathbf{0}_{i_\ell \times i_1} & & \dots & & \bar{L}_{i_\ell} & & \\ \vdots & & & & & & & \end{bmatrix}, \tag{21}$$

where  $\bar{L}_i = L_{\mathcal{T}_i} + \alpha_{i1}\mathbf{e}_1\mathbf{e}_1^\top$ , with  $L_{\mathcal{T}_i}$  being the weighted Laplacian of  $\mathcal{T}_i$ , and  $a_{11} = \sum_{k=1}^{\ell} \alpha_{1i_k}$ . Now, take a vector  $\mathbf{w}$  partitioned conformably to the Laplacian (21), so that  $\mathbf{w} = [w_1 \ \tilde{\mathbf{w}}_1 \ \dots \ \tilde{\mathbf{w}}_\ell]$  with dimension of each  $\tilde{\mathbf{w}}_i$  according to those of the related

subtree  $\mathcal{T}_i$ , as detailed in the statement. By direct inspection of the relation  $\mathbf{w}^\top L_w = \mathbf{0}^\top$  one obtains

$$\begin{aligned} \sum_{k=1}^{\ell} \alpha_{1i_k} w_1 - \sum_{k=1}^{\ell} (\tilde{\mathbf{w}}_k)_1 &= 0, \\ w_1 [\alpha_{1i} \quad 0 \quad \dots \quad 0] + \tilde{\mathbf{w}}_1^\top \bar{L}_1 &= \mathbf{0}_{\ell_1}^\top, \\ w_1 [\alpha_{2i} \quad 0 \quad \dots \quad 0] + \tilde{\mathbf{w}}_2^\top \bar{L}_2 &= \mathbf{0}_{\ell_2}^\top, \\ &\vdots \end{aligned} \tag{22}$$

and considering that  $\tilde{\mathbf{w}}_1^\top \bar{L}_i = \tilde{\mathbf{w}}_1^\top (L_{\mathcal{T}_i} + \alpha_{i1} \mathbf{e}_1 \mathbf{e}_1^\top) = \tilde{\mathbf{w}}_1^\top L_{\mathcal{T}_i} + \alpha_{i1} (\tilde{\mathbf{w}}_1)_1 \mathbf{e}_1^\top$ , Equation (22) can be rewritten as

$$\begin{aligned} \sum_{k=1}^{\ell} \alpha_{1i_k} w_1 - \sum_{k=1}^{\ell} (\tilde{\mathbf{w}}_k)_1 &= 0, \\ [w_1 \alpha_{1i} + \alpha_{i1} (\tilde{\mathbf{w}}_1)_1 \quad 0 \quad \dots \quad 0] + \tilde{\mathbf{w}}_1^\top L_{\mathcal{T}_1} &= \mathbf{0}_{\ell_1}^\top, \\ [w_1 \alpha_{2i} + \alpha_{i2} (\tilde{\mathbf{w}}_2)_1 \quad 0 \quad \dots \quad 0] + \tilde{\mathbf{w}}_2^\top L_{\mathcal{T}_2} &= \mathbf{0}_{\ell_2}^\top, \\ &\vdots \end{aligned}$$

so that, assuming that each  $\tilde{\mathbf{w}}_i$  satisfies  $\tilde{\mathbf{w}}_i^\top L_{\mathcal{T}_i} = \mathbf{0}^\top$ , one has

$$\begin{aligned} \sum_{k=1}^{\ell} \alpha_{1i_k} w_1 - \sum_{k=1}^{\ell} (\tilde{\mathbf{w}}_k)_1 &= 0, \\ w_1 \alpha_{1i} + \alpha_{i1} (\tilde{\mathbf{w}}_1)_1 &= 0, \\ w_2 \alpha_{2i} + \alpha_{i2} (\tilde{\mathbf{w}}_2)_1 &= 0, \\ &\vdots \end{aligned}$$

and hence the statement follows.  $\square$

The previous result provides the main tools for setting up an algorithm, which solves Problem 1 iteratively, though some remarks are needed to fully describe the proposed approach. In the next section, we exploit the results gained in this section, and we deduce an algorithm for the solution of Problems 1 and 2, thus achieving our main goal for this paper.

### 5. A Distributed Algorithm for the General Solution

The results of Section 4 are useful for establishing an iterative procedure for the distributed computation of the Perron vector. In the following, we solve Problem 1 for a worked example, which is taken from the simulation results of [35]. Then, we provide the general algorithm, based on the experience gained through the example.

#### 5.1. An Illustrative Example

Consider the weighted tree graph in Figure 1 and Node 1 as the reference node for the algorithm execution. Considering the notation adopted in this paper, which is reported in Figure 1b, we take a vector  $\mathbf{w} \in \mathbb{R}^9$  partitioned as

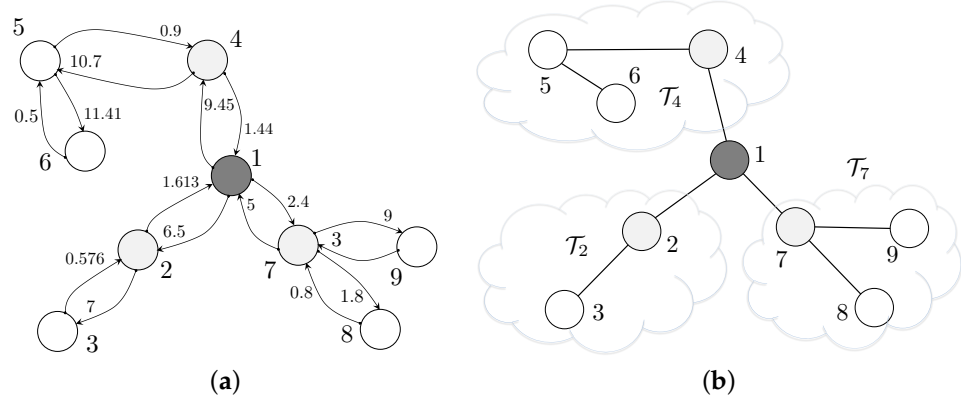
$$\mathbf{w}^\top = [w_1 \quad \tilde{\mathbf{w}}_2 \quad \tilde{\mathbf{w}}_4 \quad \tilde{\mathbf{w}}_7],$$

where

$$\tilde{\mathbf{w}}_2^\top L_{\mathcal{T}_2} = \mathbf{0}^\top, \quad \tilde{\mathbf{w}}_4^\top L_{\mathcal{T}_4} = \mathbf{0}^\top, \quad \tilde{\mathbf{w}}_7^\top L_{\mathcal{T}_7} = \mathbf{0}^\top,$$

and the first component of each subvector must satisfy

$$(\tilde{\mathbf{w}}_2)_1 = w_1 \frac{\alpha_{12}}{\alpha_{21}}, \quad (\tilde{\mathbf{w}}_4)_1 = w_1 \frac{\alpha_{14}}{\alpha_{41}}, \quad (\tilde{\mathbf{w}}_7)_1 = w_1 \frac{\alpha_{17}}{\alpha_{71}}. \quad (23)$$



**Figure 1.** Worked example from [22]: (a) unbalanced tree graph, (b) subdivision of the graph according to the notation.

Consider now that each subtree has the topology of either a path or a star graph, so that we can recur to the results of Sections 3.1 and 3.2. Specifically,  $\mathcal{T}_2$  is either a path of two nodes (or, equivalently, a star with one ray),  $\mathcal{T}_4$  is a path made of three nodes, and finally,  $\mathcal{T}_7$  is a three-node two-ray star; thus:

$$\tilde{\mathbf{w}}_2 = \begin{bmatrix} \kappa_1 & \kappa_1 \frac{\alpha_{23}}{\alpha_{32}} \end{bmatrix}, \quad \tilde{\mathbf{w}}_4 = \begin{bmatrix} \kappa_2 & \kappa_2 \frac{\alpha_{45}}{\alpha_{54}} & \kappa_2 \frac{\alpha_{45}\alpha_{56}}{\alpha_{54}\alpha_{65}} \end{bmatrix}, \quad \tilde{\mathbf{w}}_7 = \begin{bmatrix} \kappa_3 & \kappa_3 \frac{\alpha_{78}}{\alpha_{87}} & \kappa_3 \frac{\alpha_{79}}{\alpha_{97}} \end{bmatrix}. \quad (24)$$

Combining (24) with (23), one has that the structure of  $\mathbf{w}$  is as follows:

$$\mathbf{w}^\top = \begin{bmatrix} w_1 & w_1 \frac{\alpha_{12}}{\alpha_{21}} & w_1 \frac{\alpha_{12}}{\alpha_{23}} \frac{\alpha_{21}}{\alpha_{32}} & w_1 \frac{\alpha_{14}}{\alpha_{41}} & w_1 \frac{\alpha_{14}}{\alpha_{41}} \frac{\alpha_{45}}{\alpha_{54}} & w_1 \frac{\alpha_{14}}{\alpha_{41}} \frac{\alpha_{45}\alpha_{56}}{\alpha_{54}\alpha_{65}} & w_1 \frac{\alpha_{17}}{\alpha_{71}} & w_1 \frac{\alpha_{17}}{\alpha_{71}} \frac{\alpha_{78}}{\alpha_{87}} & w_1 \frac{\alpha_{17}}{\alpha_{71}} \frac{\alpha_{79}}{\alpha_{97}} \end{bmatrix}, \quad (25)$$

for a nonzero  $w_1 \in \mathbb{R}$ . We now seek the value of  $w_1$  to match the normal condition  $\mathbf{w}^\top \mathbf{1}_N = 1$ , which leads to:

$$1 = w_1 \left[ 1 + \frac{\alpha_{12}}{\alpha_{21}} \underbrace{\left( 1 + \frac{\alpha_{23}}{\alpha_{32}} \right)}_{c_{\mathcal{T}_2}} + \frac{\alpha_{14}}{\alpha_{41}} \underbrace{\left( 1 + \frac{\alpha_{45}}{\alpha_{54}} + \frac{\alpha_{45}\alpha_{56}}{\alpha_{54}\alpha_{65}} \right)}_{c_{\mathcal{T}_4}} + \frac{\alpha_{17}}{\alpha_{71}} \underbrace{\left( 1 + \frac{\alpha_{78}}{\alpha_{87}} + \frac{\alpha_{79}}{\alpha_{97}} \right)}_{c_{\mathcal{T}_7}} \right], \quad (26)$$

so that (25) solves Problem 1 with the choice

$$w_1 = \left( 1 + \frac{\alpha_{12}}{\alpha_{21}} c_{\mathcal{T}_2} + \frac{\alpha_{14}}{\alpha_{41}} c_{\mathcal{T}_4} + \frac{\alpha_{17}}{\alpha_{71}} c_{\mathcal{T}_7} \right)^{-1}. \quad (27)$$

It is worth noting here, in view of Problem 2 and a distributed implementation of the proposed approach, that coefficients  $c_{\mathcal{T}_i}$  are analogous coefficients related to each subtree, and indeed we can write

$$w_1 = c_{\mathcal{T}}^{-1} \text{ where } c_{\mathcal{T}} = 1 + \frac{\alpha_{12}}{\alpha_{21}} c_{\mathcal{T}_2} + \frac{\alpha_{14}}{\alpha_{41}} c_{\mathcal{T}_4} + \frac{\alpha_{17}}{\alpha_{71}} c_{\mathcal{T}_7}, \quad (28)$$

and each  $c_{\mathcal{T}_i}$  can be analogously computed referring to the subgraphs of each  $\mathcal{T}_i$ , thus opening the path for a distributed implementation for its computation. Coefficients  $c_{\mathcal{T}_i}$  are reminiscent of the normalizing terms in [21] called *coupling factors*, and in the following of this paper we use the same name.

### 5.2. Algorithm Description

Starting from the experience gained from the example, we now generalize the iterative algorithm to retrieve the value of the scaling factors that allow for reaching average consensus, as stated in Problem Statement 2.

Considering the solution computed in the example, and in particular Equations (26)–(28), it is straight to see that a first stage of the computation of the coupling factors should flow from the peripheral nodes to the inner ones. Indeed, each coupling factor  $c_{\mathcal{T}_i}$  can be determined only following this order, since  $c_{\mathcal{T}_i} = 1$  for leaf nodes while it is unknown in advance for the other nodes. In the following, for the sake of clearness, we describe each stage separately.

#### 5.2.1. First Stage

The first stage of the algorithm is as follows. Each leaf node starts the computation with  $c_\ell = 1$ , and sends it to its (only) neighbor  $v$ . At each time instant, each node  $v \in \mathcal{V}$  has to run the following algorithm:

- If  $v$  has received less than  $|\mathcal{N}_v| - 1$  coefficients  $c_w$  from its neighbors, node  $v$  must stay idle.
- If  $v$  has received  $|\mathcal{N}_v| - 1$  coefficients  $c_w$  from its neighbors, node  $v$  must compute

$$c_v = 1 + \sum_{w \in \mathcal{N}_v} \frac{\alpha_{vw}}{\alpha_{wv}} c_w,$$

and send it to the remaining neighbor.

The above computation is active until there is at least one node executing the second line, and it is easy to see that there exists an instant when a node, which we call  $c$  in the following, receives all the coefficients  $c_v$  from each one of its neighbors. When this condition happens, then node  $c$  triggers the second stage of the algorithm.

#### 5.2.2. Second Stage

As soon as one node  $c$  receives coefficients  $c_v$  from all its neighbors, then it triggers the second stage of the algorithm as follows.

- Node  $c$  makes the computation as in Equation (28), namely

$$w_c = \left( 1 + \sum_{v \in \mathcal{N}_c} \frac{\alpha_{cv}}{\alpha_{vc}} c_{\mathcal{T}_v} \right)^{-1},$$

thus computing the component of the Perron vector corresponding to its own location  $w_c$ ;

- Node  $c$  sends the result back to its neighbors  $v, v \in \mathcal{N}_c$ ;
- According to (8), node  $c$  sets its own scaling factor  $\zeta_c$  equal to  $\zeta_c = Nw_c$ .

Then, this latter procedure propagates from node  $c$  back to the leaf nodes. At this stage, when any node, say node  $j$ , receives  $w_\ell$  from one of its neighbors  $\ell$ , then  $j$  prosecutes the second stage of the algorithm as follows.

- Node  $j$  computes  $w_j = \frac{\alpha_{\ell j}}{\alpha_{j\ell}} w_\ell$ , namely the component of the Perron vector corresponding to its own location, according to the structure of Equation (25).
- It sends the result back to its neighbors  $v$  but  $\ell, v \in \mathcal{N}_j - \{\ell\}$ .
- According to (8), node  $j$  sets its scaling factor  $\zeta_j$  equal to  $\zeta_j = Nw_j$ .

The above procedure prosecutes until there are still neighbor nodes, thus ending when it reaches the leaf nodes.

The above procedure allows us to compute the scaling factors for average consensus in a distributed fashion. A final issue of the above procedure is the time instant when the leaf node should start running stage one. There are several possible strategies according to the specific setting, described in the following. However, this aspect is not relevant

for the algorithm execution, but only for estimation in advance of the time needed for algorithm execution.

If all the leaf nodes start synchronously at time  $\bar{t}$ , then the algorithm lasts until  $\bar{t} + 2R$ , where  $R$  is the radius of the graph. In this case, the ending node is the center of the tree. However, if the leaves are not synchronized and each node has its own starting time, then the algorithm still has a finite execution, and it ends with a correct solution, though neither the execution time nor the location of the last node(s) running the algorithm can be computed in advance.

Finally, there are scenarios where one leader node is present and it can trigger the algorithm. In this latter case, our reference scenario is the Laplacian allocation algorithm in [22], where weights are set in a distributed fashion from one leader node to the leaves; however, several different frameworks are possible, also adopting a virtual leader. In this latter setting, where a leader triggers the algorithm in general, the total algorithm time execution is equal to  $e + 2R$ , where  $e$  represents the eccentricity of the leader within the network graph.

### 6. Simulation Results

In this section, we show the results of the proposed approach in the consensus network described by the graph in Figure 2, where a parameter  $\varepsilon$  denotes an uncertainty term, and it is a variation in the edge value with respect to the nominal one equal to 1.

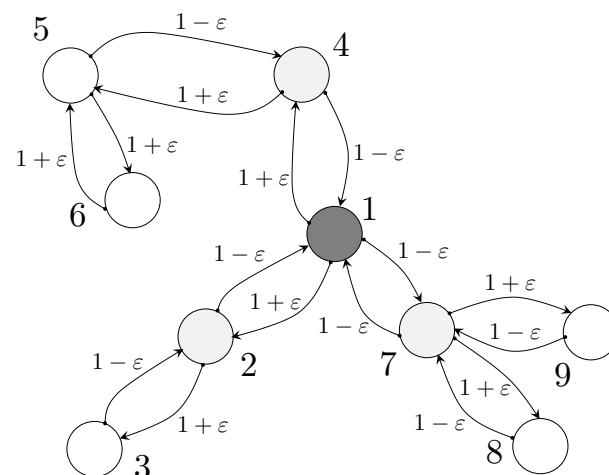


Figure 2. Simulation results: perturbed consensus network topology adopted in the simulations.

Our reference scenario for this simulation is the WSN Implementation of the average consensus algorithm as described in ([37], Figure 3), where the architecture of a wireless sensor node is sketched, allowing the scaling of the initial measurements in the preliminary initialization step (Stages I and II).

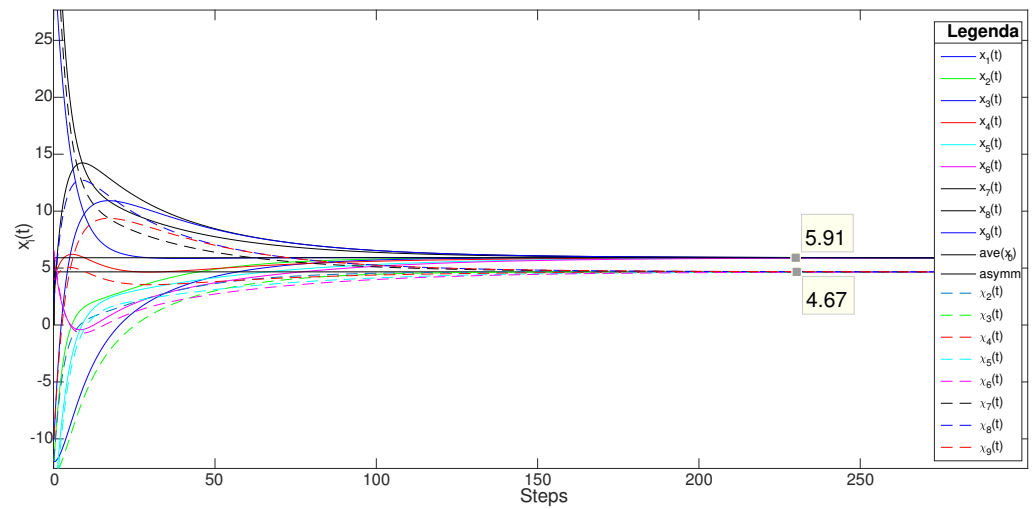
We assume to have a set of nine sensor nodes connected as depicted in Figure 2, each holding a measured quantity. The initial values are set as equal to

$$x(0) = [33 \quad -12 \quad -12 \quad 4 \quad -17 \quad 6 \quad 48 \quad 2 \quad -10]^T,$$

with average value equal to  $\sum_{i=1}^9 x_i(0)/9 = 4.667$ .

The first simulation shows the results of the evolution of the system when the uncertainty parameter is  $\varepsilon = 0.05$ . The evolution of the system is shown in Figure 3 as solid lines.

A first notable point is the magnitude of the drift of the consensus value from the initial average value. Indeed, the consensus value changes to 5.91, which is 26.4% away from the nominal average value, equal to 4.667 when no perturbation is present, even if the variation in the edge values was only of magnitude 5%.



**Figure 3.** Simulation results: perturbed consensus network. Case  $\epsilon = 0.05$ . Solid lines show uncompensated trajectories, dashed lines show the evolution with scaled initial conditions.

The above phenomenon is even more evident for values of  $\epsilon = 0.2$ . Simulation results of this case are reported in Figure 4. Indeed, in such a case the asymptotic value is equal to 10.02, so that it is more than 100% away from the nominal one, and it is useless even as a rough estimation of the average value.

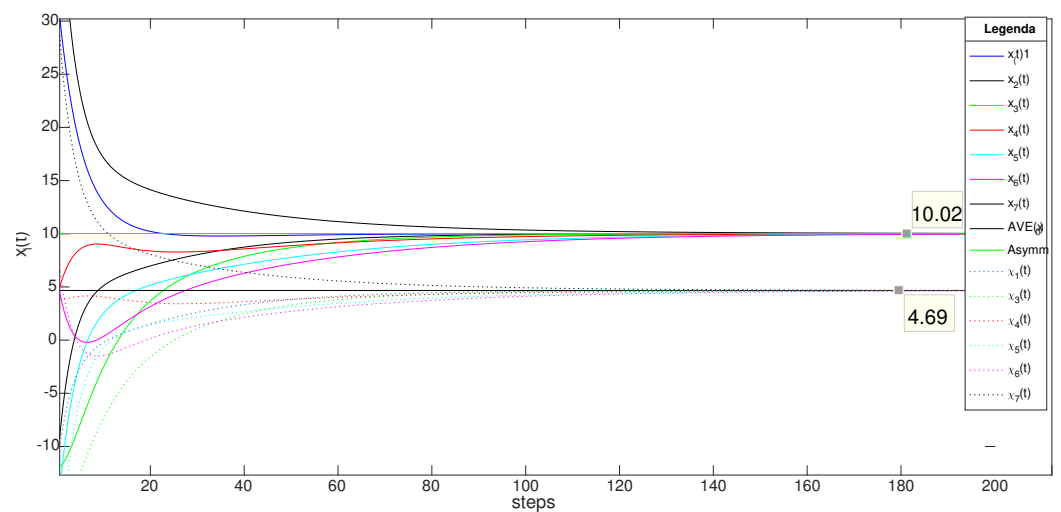
The reason for this phenomenon is related to the variation in the Perron vector. Even if each component has limited variation, the overall weighted sum results are far from the average. The Perron vector is equal to

$$\mathbf{w} = [1.5 \ 1 \ 0.7 \ 1 \ 0.7 \ 0.7 \ 1.5 \ 1 \ 1]^T,$$

so that a suitable rescaling of the initial values according to Equation (8) provides

$$\chi(0) = [2 \ -12 \ -18 \ 4 \ -25.5 \ 9 \ 32 \ 2 \ -10]^T,$$

which restores the original consensus value to the average of  $\mathbf{x}(0)$  (dashed lines).



**Figure 4.** Simulation results: perturbed consensus network. Case  $\epsilon = 0.2$ . Solid lines show uncompensated trajectories, dashed lines show the evolution with scaled initial conditions.

## 7. Conclusions

In this paper, a distributed algorithm is derived for the computation of suitable scaling factors that allow reaching average consensus in unbalanced tree networks. The general algorithm is iterative and based on local data, and it is appropriate to be run as a preliminary routine. Simulation results show that the proposed algorithm can be effectively integrated with other set-up routines, and it gathers beneficial effects on the precision of the computed average value.

### Future Work

There are several directions for further research stemming from the results of this paper; in the following, we describe some related activities that are now under investigation or consideration. From a theoretical/methodological standpoint, we are interested in the extension of the proposed approach to more general and time-varying graphs, graphs with cycles and directed graphs, and the use of a more general dynamics for a single agent, thus allowing for vehicle and mobile robot network applications. On the other hand, there are several perspective activities that use the results of this paper in applications, as for example testing the algorithm in experimental stages of WSN applications.

**Funding:** This research received no external funding.

**Data Availability Statement:** The datasets generated for this study are available on request to the corresponding author.

**Conflicts of Interest:** The author declares no conflicts of interest.

## References

1. Bullo, F. *Lectures on Network Systems*, 1st ed.; Kindle Direct Publishing: Seattle, WA, USA, 2022.
2. Olfati-Saber, R.; Fax, A.J.; Murray, R. Consensus and Cooperation in Networked Multi-Agent Systems. *Proc. IEEE* **2007**, *95*, 215–233. [\[CrossRef\]](#)
3. Rezaee, H.; Abdollahi, F. Average consensus over high-order multiagent systems. *IEEE Trans. Autom. Control* **2015**, *60*, 3047–3052. [\[CrossRef\]](#)
4. Kia, S.S.; Van Scoy, B.; Cortes, J.; Freeman, R.A.; Lynch, K.M.; Martinez, S. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. *IEEE Control Syst. Mag.* **2019**, *39*, 40–72. [\[CrossRef\]](#)
5. Garin, F.; Schenato, L. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked Control Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 75–107.
6. Chen, W.; Wang, Z.; Liu, Q.; Yue, D.; Liu, G.P. A new privacy-preserving average consensus algorithm with two-phase structure: Applications to load sharing of microgrids. *Automatica* **2024**, *167*, 111715. [\[CrossRef\]](#)
7. Xing, L.; Xu, Q.; Wen, C.; Tian, Y.C.; Mishra, Y.; Ledwich, G.; Song, Y. Robust event-triggered dynamic average consensus against communication link failures with application to battery control. *IEEE Trans. Control Netw. Syst.* **2020**, *7*, 1559–1570. [\[CrossRef\]](#)
8. Parlangeli, G. A Supervisory Algorithm Against Intermittent and Temporary Faults in Consensus-Based Networks. *IEEE Access* **2020**, *8*, 98775–98786. [\[CrossRef\]](#)
9. Kenyeres, M.; Kenyeres, J. Distributed mechanism for detecting average consensus with maximum-degree weights in bipartite regular graphs. *Mathematics* **2021**, *9*, 3020. [\[CrossRef\]](#)
10. Kenyeres, M.; Kenyeres, J. Average consensus over mobile wireless sensor networks: Weight matrix guaranteeing convergence without reconfiguration of edge weights. *Sensors* **2020**, *20*, 3677. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Patterson, S.; Bamieh, B.; El Abbadi, A. Convergence rates of distributed average consensus with stochastic link failures. *IEEE Trans. Autom. Control* **2010**, *55*, 880–892. [\[CrossRef\]](#)
12. Cai, K.; Ishii, H. Average consensus on general digraphs. In Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, USA, 12–15 December 2011; pp. 1956–1961.
13. Guyeux, C.; Haddad, M.; Hakem, M.; Lagacherie, M. Efficient distributed average consensus in wireless sensor networks. *Comput. Commun.* **2020**, *150*, 115–121. [\[CrossRef\]](#)
14. Sun, S.; Chen, F.; Ren, W. Distributed average tracking in weight-unbalanced directed networks. *IEEE Trans. Autom. Control* **2020**, *66*, 4436–4443. [\[CrossRef\]](#)
15. Sen, A.; Sahoo, S.R.; Kothari, M. Distributed average tracking with incomplete measurement under a weight-unbalanced digraph. *IEEE Trans. Autom. Control* **2022**, *67*, 6025–6037. [\[CrossRef\]](#)
16. Du, M.; Meng, D.; Wu, Z.G. Distributed averaging problems over directed signed networks. *IEEE Trans. Control Netw. Syst.* **2021**, *8*, 1442–1453. [\[CrossRef\]](#)

17. Shafi, S.Y.; Arcak, M.; El Ghaoui, L. Graph weight allocation to meet Laplacian spectral constraints. *IEEE Trans. Autom. Control* **2011**, *57*, 1872–1877. [[CrossRef](#)]
18. Hao, H.; Barooah, P. Improving convergence rate of distributed consensus through asymmetric weights. In Proceedings of the 2012 American Control Conference (ACC), Montreal, QC, Canada, 27–29 June 2012; pp. 787–792.
19. Gao, L.; Zhao, J.; Di, Z.; Wang, D. Asymmetry between odd and even node weight in complex networks. *Phys. A Stat. Mech. Its Appl.* **2007**, *376*, 687–691. [[CrossRef](#)]
20. Kenyeres, M.; Kenyeres, J. Average Consensus with Perron Matrix for Alleviating Inaccurate Sensor Readings Caused by Gaussian Noise in Wireless Sensor Networks. In *Software Engineering and Algorithms: Proceedings of 10th Computer Science On-line Conference 2021*; Springer: Cham, Switzerland, 2021; Volume 1, pp. 391–405.
21. Meyer, C.D. Uncoupling the Perron eigenvector problem. *Linear Algebra Its Appl.* **1989**, *114*, 69–94. [[CrossRef](#)]
22. Parlange, G. Laplacian Eigenvalue Allocation Through Asymmetric Weights in Acyclic Leader-Follower Networks. *IEEE Access* **2023**, *11*, 126409–126419. [[CrossRef](#)]
23. Parlange, G. Prescribed-time average consensus through data-driven leader motion. *IEEE Access* **2024**, *12*, 71970–71981. [[CrossRef](#)]
24. Liu, Z.; Guo, L. Synchronization of multi-agent systems without connectivity assumptions. *Automatica* **2009**, *45*, 2744–2753. [[CrossRef](#)]
25. Sardellitti, S.; Giona, M.; Barbarossa, S. Fast distributed average consensus algorithms based on advection-diffusion processes. *IEEE Trans. Signal Process.* **2009**, *58*, 826–842. [[CrossRef](#)]
26. Chen, Y.; Tron, R.; Terzis, A.; Vidal, R. Corrective consensus with asymmetric wireless links. In Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, USA, 12–15 December 2011; pp. 6660–6665.
27. Mirzaev, I.; Gunawardena, J. Laplacian dynamics on general graphs. *Bull. Math. Biol.* **2013**, *75*, 2118–2149. [[CrossRef](#)] [[PubMed](#)]
28. Dietzenbacher, E. Aggregation in multisector models: Using the Perron vector. *Econ. Syst. Res.* **1992**, *4*, 3–24. [[CrossRef](#)]
29. Schenato, L.; Fiorentin, F. Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica* **2011**, *47*, 1878–1886. [[CrossRef](#)]
30. Shi, F.; Yang, S.X.; Mukherjee, M.; Jiang, H.; da Costa, D.B.; Wong, W.K. Parameter-sharing-based average-consensus time synchronization in IoT networks. *IEEE Internet Things J.* **2022**, *10*, 8215–8227. [[CrossRef](#)]
31. Wu, J.; Li, X. Collective synchronization of Kuramoto-oscillator networks. *IEEE Circuits Syst. Mag.* **2020**, *20*, 46–67. [[CrossRef](#)]
32. Duan, Y.; He, X.; Zhao, Y. Distributed algorithm based on consensus control strategy for dynamic economic dispatch problem. *Int. J. Electr. Power Energy Syst.* **2021**, *129*, 106833. [[CrossRef](#)]
33. Macii, D.; Rinaldi, S. Time synchronization for smart grids applications: Requirements and uncertainty issues. *IEEE Instrum. Meas. Mag.* **2022**, *25*, 11–18. [[CrossRef](#)]
34. Bullo, F.; Cortés, J.; Martínez, S. *Distributed Control of Robotic Networks*; Applied Mathematics Series; Princeton University Press: Princeton, NJ, USA, 2009.
35. Parlange, G. A distributed algorithm for the assignment of the Laplacian spectrum for path graphs. *Mathematics* **2023**, *11*, 2359. [[CrossRef](#)]
36. Usmani, R.A. Inversion of a tridiagonal Jacobi matrix. *Linear Algebra Its Appl.* **1994**, *212*, 413–414. [[CrossRef](#)]
37. Kenyeres, J.; Kenyeres, M.; Rupp, M.; Farkas, P. WSN implementation of the average consensus algorithm. In Proceedings of the 17th European Wireless 2011-Sustainable Wireless Technologies, Vienna, Austria, 27–29 April 2011; pp. 139–146.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.