A Distributed Algorithm for Reaching Average Consensus in Unbalanced Tree Networks

Gianfranco Parlangeli *

Posted Date: 19 August 2024

doi: 10.20944/preprints202408.1310.v1

Keywords: Laplacian eigenvectors; Perron vector; average consensus problems

 doi:10.20944/preprints202408.1310.v1

Article

A Distributed Algorithm for Reaching Average Consensus in Unbalanced Tree Networks

Gianfranco Parlangeli

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

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 get a common value for all participants [2].

One special yet fundamental consensus condition regards the value of such agreement 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 is the natural equilibrium condition for a multi-agent system whenever the communication graph is undirected and unweighted, or more in general, whenever it is weight-balanced, and this special condition allows solving several basic goals in a distributed fashion [1].

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 has the drawback of a large time span for convergence, and in turn, this makes consensus not even possible in practice when the unavoidable presence of communication delays, computation errors or agents' faults over a large time span are accounted for [6].

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 [7], dedicated atomic transactions between pairs of neighbors [8], a chain of two integrators that are coupled with a distributed estimator [9] average tracking with incomplete measurement through a distributed averaging filter and a decentralized tracking controller [10], and recently considering also signed networks [11].

 doi:10.20944/preprints202408.1310.v1

In the last few years, advances have been made in adopting the strategy of assigning asymmetric weights to neighbors' edges [12]. Indeed, it was found that a proper choice of asymmetric weights can significantly improve the convergence rate, even over the symmetric optimal design [13], 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 instead of the average of the initial conditions.

In general, there are several scenarios where edge weight asymmetry should be taken into account, other than a design choice. Indeed, empirical research clearly shows that real-world networks usually have a large heterogeneity in the intensity or capacity of the connections (and hence the weights of the links) [14]. 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 condition.

The approach proposed in this paper can be combined with an asymmetric design of the edge weights as in [15] to achieve the ambitious goal of average consensus with prescribed dynamics. Analogous techniques have been adopted to achieve prescribed-time consensus [16]. The opportunity of setting the Laplacian eigenvalues through an appropriate choice of edge weights has also a beneficial effect on security; indeed most of the algorithms designed to monitor network evolution to exclude the presence of edge faults [17] or malicious nodes [18] require the knowledge of eigenstructure of the network It is worth noting that malicious attacks can destabilize the entire system when there is access of at least one of its eigenvalues [19].

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 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 Perron vector to achieve a more precise average estimation is discussed in [20]. The results developed in this paper are inspired by [21], as described in detail in Section 3.

The 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 along the 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. $0_{d_1 \times d2_2}$, $d_1, d_2 \in \mathbb{N}$, 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. $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & ... & 0 \end{bmatrix}^\top$. Vectors are denoted in bold letters. For a vector $\mathbf{v} \in \mathbb{R}^d$ we denote v_i the ith 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 $\rho_A = \max_{i \in \{1,\dots n\}} \{|\lambda_i| \text{ where } \lambda_i \in \Lambda_A\}$.

A nonnegative (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 nonnegative matrix

 doi:10.20944/preprints202408.1310.v1

$A \in \mathbb{R}^{n\times n}$ 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 nonnegative matrix $A \in \mathbb{R}^{n\times n}$ is *primitive* if there exist a positive integer m such that $A^m > 0$.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is made of a *vertex set* $\mathcal{V} = \{1, 2, ..., n\}$ 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 $\{j_\ell$ 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 $i, j \in V$ 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 *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 $|N_w|$ 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 nonnegative matrices and their related graphs explaining their zero-nonzero pattern, as the following result shows [1]:

Lemma 1.1. *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 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 consensus algorithm [1,2].

This setting is made of a set of N nodes, each holding a variable $x_i(t)$, $i \in \{1, \ldots, N\}$ 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 $\mathbf{x} \in \mathbb{R}^{\mathbf{N}}$ chosen as $[\mathbf{x}]_i(t) = x_i(t)$ and updating as $\dot{\mathbf{x}}(t) = -L_\kappa\mathbf{x}(t)$, which gives rise to the *Laplacian flow* [1]:

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

 doi:10.20944/preprints202408.1310.v1

for a L_κ in the set $\mathcal{L}_w$. Network evolution is dictated by the spectrum of L_κ [22]. It is worth mentioning that the design of asymmetric gains was recently proposed [13,23,24] as a key feature to improve the convergence rate, and it was recently explored also in [25–27].

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 should be chosen sufficiently small to keep the system stable [1], and analogously in this case one gets 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 of (1) and (2), matrix $\Phi(t)$ shows some interesting properties inherited from the properties of L_κ that $L_\kappa \mathbf{1} = \mathbf{0}$, $\mathbf{w}_\mathcal{L}^\top L_\kappa = \mathbf{0}^\top$, namely [1]:

$$\begin{aligned} &\Phi\mathbf{1} = \mathbf{1}, \qquad \mathbf{w}_{\mathcal{L}_k}^\top \Phi = \mathbf{w}_{\mathcal{L}_k}^\top, \\ &\Phi(t) \text{ is positive if } \mathcal{G} \text{ is strongly connected,} \end{aligned} \tag{3}$$

Moreover, even if the eigenvalues of matrix $\Phi(t)$ are related to those of L_k according to $\lambda_\Phi = e^{-\lambda_{Lk}t}$ for systems (1) and $\lambda_\Phi = (1 - \gamma\lambda_{Lk})^t$ 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 $x_i(0), i = 1, 2, \ldots, N$, there exists $\alpha \in \mathbb{R}$ such that:

$$\lim_{t \to +\infty} x_i(t) = \alpha, \qquad \forall\, i \in \{1, \ldots, N\}, \tag{4}$$

or, equivalently, $\lim_{t\to+\infty} \mathbf{x}(t) = \alpha \mathbf{1}_N$, $\exists \alpha \in \mathbb{R}$. The constant α 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:

$$\alpha = \frac{\mathbf{w}_\mathcal{L}^\top \mathbf{x}(0)}{\mathbf{w}_\mathcal{L}^\top \mathbf{1}_N}. \tag{5}$$

where $\mathbf{w}_\mathcal{L}$ is a left eigenvector of $\Phi(t)$ associated to the unitary eigenvalue and satisfying the normal condition $\mathbf{w}_\mathcal{L}^\top \mathbf{1} = 1$ [28]. In a wide number of applications, it holds $\mathbf{w}_\mathcal{L} = \frac{1}{N}\mathbf{1}$ 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 became 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 2.1 (Perron–Frobenius). *Let $A \in \mathbb{R}^{n\times n}$ be an irreducible nonnegative 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 nonnegative 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 non-negative. If A is additionally

 doi:10.20944/preprints202408.1310.v1

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 Φ so that in the following, we refer to the *left* eigenvector of $1 = \rho(A)$ as the *Perron vector* of Φ (e.g. vector w_{L_κ} in Equation (3)).

In the following of the paper, for a given Laplacian matrix L_w related to a connected graph $\mathcal{G}$, we denote by $w_{\mathcal{L}}$ its left eigenvector corresponding to $\lambda = 0$. If matrix L_w is the generator of a dynamical flow $x(t) = e^{-L_w}x(0)$, then $w_{\mathcal{L}}$ is the Perron vector of the positive matrix e^{-L_w}, and with a little abuse of notation we refer to $w_{\mathcal{L}}$ 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 [29], so it allows to detect aggregrations, 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 the paper, there are interesting examples on 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 [30]. In [31] the same synchronization problem is solved through average consensus in IoT applications. Within the above application scenarions, average consensus is adopted also for achieving the estimation of an environmental quantity [8].

In the field of Electric grid, Smart Power Grids and Renewable energies, average consensus-based algorithms are widely adopted for the following fundamental issues: (1) generators synchronization[32] (2) economic dispatch [33] (3) clock synchronization [34].

Finally, several tasks of great value in the area of robotic networks can be accomplished through the exploitment of average consensus. Among the others, it is worth mentioning the rendez-vous 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 [35]

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 following 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 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 enginnering 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 α as in (4) with α equal to (5), compute a set of scaling factors ζ_i, $i = 1, \dots, N$ such that the state vector $\chi(t)$ defined as

 doi:10.20944/preprints202408.1310.v1

$(\boldsymbol{\chi}(t))_i = \dfrac{(\mathbf{x}(t))_i}{\zeta_i}$ converges to $\boldsymbol{\chi}(t) \xrightarrow{t\to\infty} \alpha\mathbf{1}$ with $\alpha = \sum_{i=1}^{N} \dfrac{\mathbf{x}(0)}{N}$, thus solving the average consensus problem for the original problem.

The above two problems are indeed two facets of the same argument, the first being closer to a mathematical fashion while the second in an engineering style. It is worth noting that Statement 2 can be also effectively encapsulated into the Problem Statement of [15], which is related to the Laplacian eigenvalue allocation by a proper asymmetric edge weights assignment, so that the two problems can be integrated each other as follows:

Problem Statement 2-bis. Given a tree graph $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, and one node ℓ triggering the distributed algorithm in [36] to allocate the Laplace spectrum and grounded Laplacian to, resp., λ_i and $\mu_i \in \mathbb{R}$ (satisfying $0 < \mu_1 < \lambda_2 < ... < \mu_{n-1} < \lambda_n$), compute a set of scaling factors ζ_i, $i = 1, \ldots, N$ such that the state vector $\boldsymbol{\chi}(t)$ defined as $(\boldsymbol{\chi}(t))_i = \dfrac{(\mathbf{x}(t))_i}{\zeta_i}$ converges to $\boldsymbol{\chi}(t) \xrightarrow{t\to\infty} \alpha\mathbf{1}$ with $\alpha = \sum_{i=1}^{N} \dfrac{\mathbf{x}(0)}{N}$, thus solving the average consensus problem with prescribed dynamics adopting the asymmetric weight distribution.

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

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

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

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

Remark. 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 peripherical branches of any tree, as well as widely adopted graph topologies in general [1]. We further extend the solution to a generic n tree graphs.

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 of the construction of the solution is based on [37] adapted to our special structure of matrices. The following statement is proved in [37], and it is our main mathematical tool to get the explicit solution of Path Graph.

 doi:10.20944/preprints202408.1310.v1

Proposition 3.1 ([37]). *Consider the tridiagonal matrix:*

$$L = \begin{bmatrix} a_1 & b_1 & 0 & \dots \\ c_1 & a_2 & b_2 & \\ & & \ddots & b_{n-1} \\ \dots & 0 & c_{n-1} & a_n \end{bmatrix} \tag{9}$$

and assume that L is nonsigular. Build the two sequences:

$$\theta_i = a_i\theta_{i-1} - b_{i-1}c_{i-1}\theta_{i-2}, \qquad \text{for } i = 2, ..., n, \tag{10}$$

$$\text{with initial conditions} \qquad \theta_0 = 1 \text{ and } \theta_1 = a_1 \tag{11}$$

and

$$\varphi_i = a_i\varphi_{i+1} - b_ic_i\varphi_{i+2}, \qquad \text{for } i = n-1, ..., 1, \tag{12}$$

$$\text{with initial conditions} \qquad \varphi_{n+1} = 1 \text{ and } \varphi_n = a_n, \tag{13}$$

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

$$[L^{-1}]_{ij} = \begin{cases} (-1)^{i+j}b_i \dots b_{j-1}\theta_{i-1}\varphi_{j+1}/\theta_n, & \text{if } i < j, \\ \theta_{i-1}\varphi_{j+1}/\theta_n, & \text{if } i = j, \\ (-1)^{i+j}c_j \dots c_{i-1}\theta_{j-1}\varphi_{i+1}/\theta_n, & \text{if } i > j \end{cases}, \tag{14}$$

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

Proposition 3.2. *Let $\alpha_{i,j} \in \mathbb{R}_+$, and consider the Laplacian matrix:*

$$L_w = \begin{bmatrix} \alpha_{(1,2} & -\alpha_{1,2} & & \dots \\ -\alpha_{2,1} & a_{2,2} & & \\ & & \ddots & -\alpha_{(n-1),n} \\ \dots & 0 & -\alpha_{n,(n-1)} & \alpha_{n,(n-1)} \end{bmatrix} \tag{15}$$

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

$$\mathbf{w}_{\mathcal{L}} = \kappa \begin{bmatrix} 1 & \frac{\alpha_{12}}{\alpha_{21}} & \frac{\alpha_{23}\alpha_{12}}{\alpha_{32}\alpha_{21}} & \dots & \prod_{j=2}^{i} \frac{\alpha_{(j-1),j}}{\alpha_{j,(j-1)}} & \dots \end{bmatrix} \qquad \kappa \in \mathbb{R} - \{0\}. \tag{16}$$

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 follows:

$$\mathcal{L}_w = \left[\begin{array}{c|c} \alpha_{12} & L_{1*} \\ \hline L_{*1} & L_1 \end{array}\right] \tag{17}$$

with $L_{1*} = \begin{bmatrix} -\alpha_{12} & 0 & \dots & 0 \end{bmatrix}$, $L_{*1} = \begin{bmatrix} -\alpha_{21} & 0 & \dots & 0 \end{bmatrix}^\top$ and

$$L_1 = \begin{bmatrix} \alpha_{2,1} + \alpha_{23} & -\alpha_{2,3} & 0 & \dots \\ -\alpha_{3,2} & a_{3,3} & & \\ 0 & & \ddots & -\alpha_{(n-1),n} \\ \dots & 0 & -\alpha_{n,(n-1)} & \alpha_{n,(n-1)} \end{bmatrix}, \tag{18}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 August 2024 doi:10.20944/preprints202408.1310.v1

so that, taking $\mathbf{w} = \begin{bmatrix} w_1 & \tilde{\mathbf{w}}_1 \end{bmatrix}$ with $w_1 \in \mathbb{R}$, $\tilde{\mathbf{w}}_1 \in \mathbb{R}^{n-1}$, one has:

$$\mathbf{w}^\top L_w = 0 \Rightarrow \begin{cases} w_1 \alpha_{12} + \tilde{\mathbf{w}}_1{}^\top L_{*1} = 0 \\ w_1 L_{1*} + \tilde{\mathbf{w}}_1{}^\top L_1 = 0 \end{cases}; \tag{19}$$

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

$$\mathbf{w}^\top = \kappa \begin{bmatrix} 1 & L_{*1} L_1^{-1} \end{bmatrix} \qquad \text{with} \qquad \kappa \in \mathbb{R} - \{0\}. \tag{20}$$

Considering that $L_{1*} = -\alpha_{12} \mathbf{e}_1^\top$, it is possible to find an explicit expression for (20) by computing the first row of L_1^{-1} through (14) to get:

$$L_{*1} L_1^{-1} = (-\alpha_{12}) \begin{bmatrix} -\dfrac{1}{\alpha_{21}} & -\dfrac{\alpha_{23}}{\alpha_{21}\alpha_{32}} & \dots & -\dfrac{\alpha_{23} \dots \alpha_{(n-1),n}}{\alpha_{21}\alpha_{32} \dots \alpha_{n,(n-1)}} \end{bmatrix} \tag{21}$$

so that, putting together (20) with (21), Equation (16) follows. □

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,

Proposition 3.3. *Let $\mathcal{G}$ be a star graph having n rays, and suppose that the central node is selected as first node. Let L_w be its corresponding Laplacian matrix for any choice of edge weights, so that:*

$$L_w = \begin{bmatrix} a_{11} & -\alpha_{12} & \dots & -\alpha_{1n} \\ -\alpha_{21} & \alpha_{21} & 0 & \dots \\ \vdots & 0 & \ddots & \\ -\alpha_{n1} & 0 & \dots & \alpha_{n1} \end{bmatrix} \tag{22}$$

where

$$a_{11} = \sum_{j=2}^{n} \alpha_{1j}. \tag{23}$$

Then, the corresponding Perron vector is equal to:

$$\mathbf{w}_{\mathcal{L}} = \kappa \begin{bmatrix} 1 & \dfrac{\alpha_{12}}{\alpha_{21}} & \dfrac{\alpha_{13}}{\alpha_{31}} & \dots & \dfrac{\alpha_{j,(j+1)}}{\alpha_{(j+1),j}} & \dots \end{bmatrix} \qquad \kappa \in \mathbb{R} - \{0\}. \tag{24}$$

Proof. The proof is conducted by direct inspection on the relation $\mathbf{w}^\top L_w$, with L_w as in (22) for a nonzero $\mathbf{w} \in \mathbb{R}^n$. Consider the partition of L_w as::

$$\mathcal{L}_w = \left[\begin{array}{c|c} a_{11} & L_{1*} \\ \hline L_{*1} & L_1 \end{array}\right] \tag{25}$$

with $L_{1*} = \begin{bmatrix} -\alpha_{12} & -\alpha_{13} & \dots & -\alpha_{1n} \end{bmatrix}$, $L_{*1} = \begin{bmatrix} -\alpha_{21} & -\alpha_{31} & \dots & -\alpha_{n1} \end{bmatrix}^\top$ and

$$L_1 = \begin{bmatrix} \alpha_{2,1} & 0 & 0 & \dots \\ & \alpha_{3,1} & 0 & \\ & & \ddots & 0 \\ \dots & 0 & 0 & \alpha_{n,1} \end{bmatrix} = \operatorname{diag}(\{\alpha_{j,1}\}_{|j=2,..,n}), \tag{26}$$

 doi:10.20944/preprints202408.1310.v1

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

$$\mathbf{w}^\top = \kappa \begin{bmatrix} 1 & -L_{*1}L_1^{-1} \end{bmatrix} \qquad \text{with} \qquad \kappa \in \mathbb{R} - \{0\}. \tag{27}$$

and considering (26) and (27), one has:

$$L_{*1}L_1^{-1} = \begin{bmatrix} -\frac{\alpha_{12}}{\alpha_{21}} & -\frac{\alpha_{13}}{\alpha_{31}} & \dots & -\frac{\alpha_{1n}}{\alpha_{n1}} \end{bmatrix}. \tag{28}$$

Putting together Equation (28) with Equation (27), Equation (24) follows. □

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 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 [15], so it can be integrated into the final algorithmic solutiono, thus solving the ambitious problem of eigenvalue allocation for average consensus problems.

Proposition 4.1. *Consider a tree graph $\mathcal{T}$, L_w its corresponding Laplacian, and let $\ell = |N_1|$. $\mathcal{T}_j$, $j = 1, \dots, \ell$ represent the subtrees of $\mathcal{T}$ after remotion of node 1, each made of $\ell_j = |\mathcal{T}_j|$ nodes, $j = 1, \dots, \ell$ 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 $\mathbf{w} = \begin{bmatrix} w_1 & \tilde{\mathbf{w}}_1 & \dots & \tilde{\mathbf{w}}_\ell \end{bmatrix}$ 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 $\tilde{w}_j$ satisfies:*

$$(\tilde{\mathbf{w}}_i)_1 = w_1 \frac{\alpha_{1i}}{\alpha_{i1}}. \tag{29}$$

Proof. Let $N_1 = \{i_1, .., i_\ell\}$, so that a convenient labeling allows to write the Laplacian matrix without loss of generality as:

$$L_w = \left[\begin{array}{c|ccc|c} a_{11} & -\alpha_{1i_1} & 0 & \dots & -\alpha_{1i_\ell} \quad 0 \quad \dots \\ \hline -\alpha_{i_1 1} \\ 0 \\ \vdots & \multicolumn{3}{c|}{\bar{L}_{i_1}} & \multicolumn{2}{c}{\mathbf{0}_{i_1 \times i_\ell}} \\ \hline \vdots & \multicolumn{3}{c|}{\vdots} & \multicolumn{2}{c}{\vdots} \\ \hline -\alpha_{i_\ell 1} \\ 0 \\ \vdots & \multicolumn{3}{c|}{\mathbf{0}_{i_\ell \times i_1}} & \multicolumn{2}{c}{\bar{L}_{i_\ell}} \end{array}\right] \tag{30}$$

where $\bar{L}_i = L_{\mathcal{T}_i} + \alpha_{i_i 1}\mathbf{e}_1\mathbf{e}_1^\top$ 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 (30), so that $\mathbf{w} = \begin{bmatrix} w_1 & \tilde{\mathbf{w}}_1 & \dots & \tilde{\mathbf{w}}_\ell \end{bmatrix}$ with

 doi:10.20944/preprints202408.1310.v1

dimension of each $\tilde{\mathbf{w}}_i$ according to those of the related subtree $\mathcal{T}$ and detailed in the above statement. By direct inspection of the relation $\mathbf{w}^\top L_w = \mathbf{0}_n^\top$ one gets:

$$\begin{aligned} &\sum_{k=1}^{\ell} \alpha_{1i_k} w_1 - \sum_{k=1}^{\ell} (\tilde{\mathbf{w}}_k)_1 = 0 \\ &w_1 \begin{bmatrix} \alpha_{1i} & 0 & \dots & 0 \end{bmatrix} + \tilde{\mathbf{w}}_1^\top \bar{L}_1 = \mathbf{0}_{\ell_1}^\top \\ &w_1 \begin{bmatrix} \alpha_{2i} & 0 & \dots & 0 \end{bmatrix} + \tilde{\mathbf{w}}_2^\top \bar{L}_2 = \mathbf{0}_{\ell_2}^\top \\ &\vdots \end{aligned} \tag{31}$$

and considering that $\tilde{\mathbf{w}}_1^\top \bar{L}_i = \tilde{\mathbf{w}}_1^\top \left(L_{\mathcal{T}_i} + \alpha_{i_1 1} \mathbf{e}_1 \mathbf{e}_1^\top \right) = \tilde{\mathbf{w}}_1^\top L_{\mathcal{T}_i} + \alpha_{i_1 1} (\tilde{\mathbf{w}}_i)_1 \mathbf{e}_1^\top$, Equation (33) can be rewritten as:

$$\begin{aligned} &\sum_{k=1}^{\ell} \alpha_{1i_k} w_1 - \sum_{k=1}^{\ell} (\tilde{\mathbf{w}}_k)_1 = 0, \\ &\begin{bmatrix} w_1 \alpha_{1i} + \alpha_{i_1 1} (\tilde{\mathbf{w}}_1)_1 & 0 & \dots & 0 \end{bmatrix} + \tilde{\mathbf{w}}_1^\top L_{\mathcal{T}_1} = \mathbf{0}_{\ell_1}^\top, \\ &\begin{bmatrix} w_1 \alpha_{2i} + \alpha_{i_2 1} (\tilde{\mathbf{w}}_2)_1 & 0 & \dots & 0 \end{bmatrix} + \tilde{\mathbf{w}}_2^\top L_{\mathcal{T}_2} = \mathbf{0}_{\ell_2}^\top, \\ &\vdots \end{aligned} \tag{32}$$

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

$$\begin{aligned} &\sum_{k=1}^{\ell} \alpha_{1i_k} w_1 - \sum_{k=1}^{\ell} (\tilde{\mathbf{w}}_k)_1 = 0, \\ &w_1 \alpha_{1i} + \alpha_{i_1 1} (\tilde{\mathbf{w}}_1)_1 = 0 \\ &w_2 \alpha_{2i} + \alpha_{i_1 1} (\tilde{\mathbf{w}}_2)_1 = 0 \\ &\vdots \end{aligned} \tag{33}$$

and hence the statement follows. □

The previous result provides the main tools for setting 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 Problem 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 to establish an interative 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 [36]. 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 be the reference node for the algorithm execution. Considering the notation adopted in this paper, which is reported in Fig. 1 (b), we take a vector $\mathbf{w} \in \mathbb{R}^9$ partitioned as:

$$\mathbf{w}^\top = \begin{bmatrix} w_1 & \tilde{\mathbf{w}}_2 & \tilde{\mathbf{w}}_4^\top & \tilde{\mathbf{w}}_7^\top \end{bmatrix} \tag{34}$$

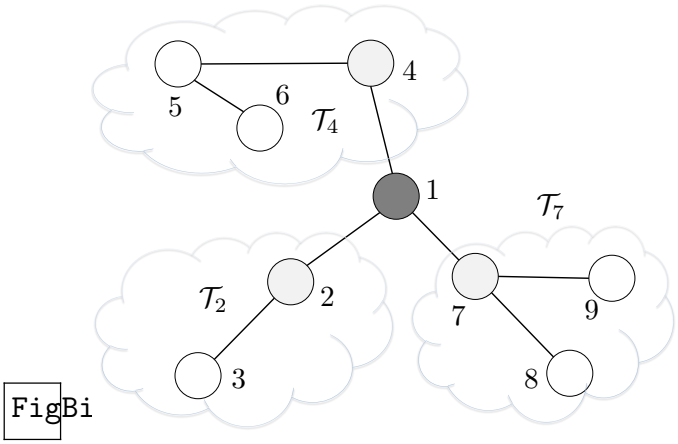
where

$$\tilde{\mathbf{w}}_2^\top L_{\mathcal{T}_2} = 0^\top \qquad \tilde{\mathbf{w}}_4^\top L_{\mathcal{T}_4} = 0^\top \qquad \tilde{\mathbf{w}}_7^\top L_{\mathcal{T}_7} = 0^\top \tag{35}$$

 doi:10.20944/preprints202408.1310.v1

and the first component of each subvector must satisfy:

$$(\tilde{\mathbf{w}}_2)_1 = w_1 \frac{\alpha_{12}}{\alpha_{21}} \qquad (\tilde{\mathbf{w}}_4)_1 = w_1 \frac{\alpha_{14}}{\alpha_{41}} \qquad (\tilde{\mathbf{w}}_7)_1 = w_1 \frac{\alpha_{17}}{\alpha_{71}} \tag{36}$$



FigBi

Figure 1. Worked Example from [36] : (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 equiv. a star with one ray, $\mathcal{T}_4$ is a path made of 3 nodes and finally $\mathcal{T}_7$ is a three-node two-ray star, thus:

$$\tilde{\mathbf{w}}_2 = \begin{bmatrix} \kappa_1 & \kappa_1 \frac{\alpha_{23}}{\alpha_{32}} \end{bmatrix} \quad \tilde{\mathbf{w}}_4 = \begin{bmatrix} \kappa_2 & \kappa_2 \frac{\alpha_{45}}{\alpha_{54}} & \kappa_2 \frac{\alpha_{45}\alpha_{56}}{\alpha_{54}\alpha_{65}} \end{bmatrix} \quad \tilde{\mathbf{w}}_7 = \begin{bmatrix} \kappa_3 & \kappa_3 \frac{\alpha_{78}}{\alpha_{87}} & \kappa_3 \frac{\alpha_{79}}{\alpha_{97}} \end{bmatrix}. \tag{37}$$

Combining (37) with (36), one has that the structure of $\mathbf{w}$ is as follows:

$$\mathbf{w}^\top = \begin{bmatrix} w_1 & w_1 \frac{\alpha_{12}}{\alpha_{21}} & w_1 \frac{\alpha_{12}}{\alpha_{23}} \frac{\alpha_{21}}{\alpha_{32}} & w_1 \frac{\alpha_{14}}{\alpha_{41}} & w_1 \frac{\alpha_{14}}{\alpha_{41}} \frac{\alpha_{45}}{\alpha_{54}} & w_1 \frac{\alpha_{14}}{\alpha_{41}} \frac{\alpha_{45}\alpha_{56}}{\alpha_{54}\alpha_{65}} & w_1 \frac{\alpha_{17}}{\alpha_{71}} & w_1 \frac{\alpha_{17}}{\alpha_{71}} \frac{\alpha_{78}}{\alpha_{87}} & w_1 \frac{\alpha_{17}}{\alpha_{71}} \frac{\alpha_{79}}{\alpha_{97}} \end{bmatrix} \tag{38}$$

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:

$$1 = w_1 \left[1 + \frac{\alpha_{12}}{\alpha_{21}} \underbrace{\left(1 + \frac{\alpha_{23}}{\alpha_{32}} \right)}_{c_{\mathcal{T}_2}} + \frac{\alpha_{14}}{\alpha_{41}} \underbrace{\left(1 + \frac{\alpha_{45}}{\alpha_{54}} + \frac{\alpha_{45}\alpha_{56}}{\alpha_{54}\alpha_{65}} \right)}_{c_{\mathcal{T}_4}} + \frac{\alpha_{17}}{\alpha_{71}} \underbrace{\left(1 + \frac{\alpha_{78}}{\alpha_{87}} + \frac{\alpha_{79}}{\alpha_{97}} \right)}_{c_{\mathcal{T}_7}} \right] \tag{39}$$

so that (38) together with

$$w_1 = \left(1 + \frac{\alpha_{12}}{\alpha_{21}} c_{\mathcal{T}_2} + \frac{\alpha_{14}}{\alpha_{41}} c_{\mathcal{T}_4} + \frac{\alpha_{17}}{\alpha_{71}} c_{\mathcal{T}_7} \right)^{-1} \tag{40}$$

solves Problem 1. 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

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

and each $c_{\mathcal{T}_i}$ can be analogously be computed referring to the subgraphs of each $\mathcal{T}_i$, thus opening the path for a distributed implementation for its computation. Coefficients $c_{\mathcal{T}7}$ are reminiscent of the

 doi:10.20944/preprints202408.1310.v1

normalizing terms in [21] called *coupling factors*, and in the following if 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 Equation (39), (40), (41), it is straight to see that a first stage of the computation of the coupling factors should flow from the peripherical 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.

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 $v \in \mathcal{V}$ should 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

$$c_v = 1 + \sum_{w \in \mathcal{N}_v} \frac{\alpha_{vw}}{\alpha_{wv}} c_w, \tag{42}$$

  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, that we call it 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, as follows:

- makes the computation as in Equation (41), namely:

$$w_c = \left(1 + \sum_{v \in \mathcal{N}_c} \frac{\alpha_{cv}}{\alpha_{vc}} c_{\mathcal{T}_v})^{-1}\right), \tag{43}$$

  thus computing the component of the Perron vector corresponding to its own location w_c.
- sends the result back to its neighbors v (but ℓ), $v \in \mathcal{N}_c - \{\ell\}$.
- According to (8), node c sets its own scaling factor ζ_c equal to $\zeta_c = N w_c$.

Then, this latter procedure propagates from node c back to the leaf nodes. At this stage, when any node, say node j, receives w_ℓ from one of its neighbors ℓ, then:

- 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 (38),
- sends the result back to its neighbors v, $v \in \mathcal{N}_j$,
- According to (8), node j sets its scaling factor ζ_j equal to $\zeta_j = N w_j$.

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

The above procedure allows 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 foillowing. However, it is not an issue for the algorithm execution, but only for an estimation in advance of the time needed for execution.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 August 2024 doi:10.20944/preprints202408.1310.v1

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 have a finite execution with a correct solution, though it cannot be computed in advancethe execution time nor the location of the last node(s) running the algorithm.

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 [15], where weights are set in a distributed fashion, starting from the leader node to leaf nodes. However, several different frameworks are possible, also adopting a virtual leader. In this latter setting, where a leader triggers the algorithm, 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 ε denotes an uncertainty term and it is a variation of the edge value with respect to the nominal one equal to 1.

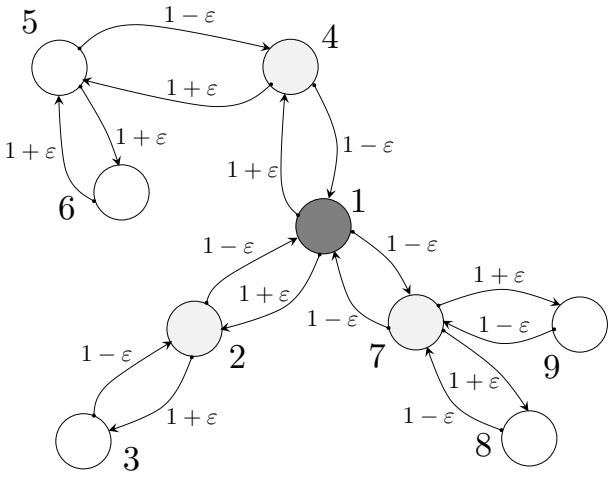


Figure 2. Simulation results: perturbed consensus network topology adopted in the simulations.

Our reference scenario for this simulation is the WSN Implementation of the Average Consensus Algorithm as described in [38] Figure 3 , where the architecture of a wireless sensor node is sketched, allowing the scaling of the initial measurements in the preliminary initialization step (Stage I and II).

We assume to have a set of 9 sensor nodes connected as depicted in Figure 2, each holding a measured quantity, the initial values are set equal to

$$x0 = \begin{bmatrix} 33 & -12 & -12 & 4 & -17 & 6 & 48 & 2 & -10 \end{bmatrix}$$

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 $\varepsilon = 0.05$. The evolution of the system is shown in Figure 3, solid lines.

 doi:10.20944/preprints202408.1310.v1

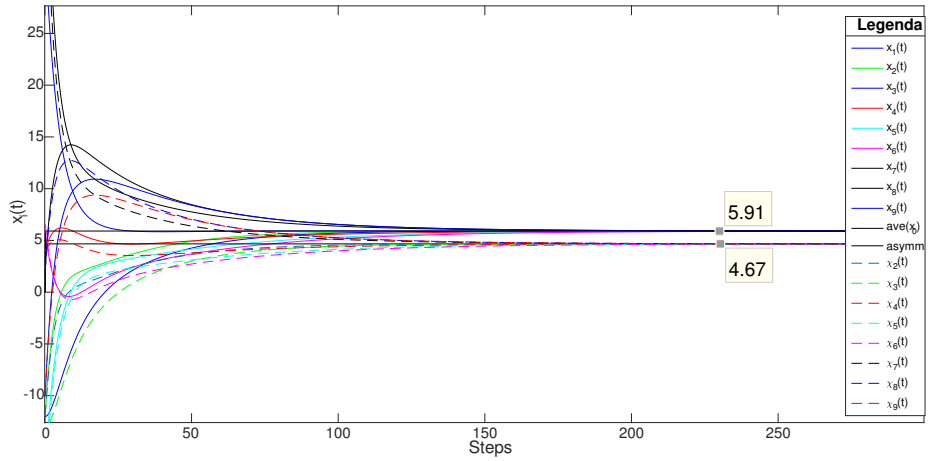


Figure 3. Simulation results: perturbed consensus network. Case $\varepsilon = 0.05$. Solid lines plots show uncompensated trajectories, dashed lines show the evolution with scaled initial conditions.

A first notable point is the magnitude of the drift of the consensus value from the initial average value. Indeed, in such a case, the consensus value turns to 5.9, which is 26, 4% away from the nominal average value, even if the variation was only of magnitude 5%.

The above phenomenon is even more evident for values of $\varepsilon = 0.2$. Indeed, in such case the asymptotic value is equal to 10.02, so that it is more than 100% far 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 of the Perron vector. Even if each component has limited variation, the overall weighted sum results so far from the average. The Perron vector is equal to

$$\mathbf{w} = \begin{bmatrix} 1.5 & 1 & 0.7 & 1 & 0.7 & 0.7 & 1.5 & 1 & 1 \end{bmatrix}'$$

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

$$\chi(0) = \begin{bmatrix} 2 & -12 & -18 & 4 & -25.5 & 9 & 32 & 2 & -10 \end{bmatrix}'$$

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

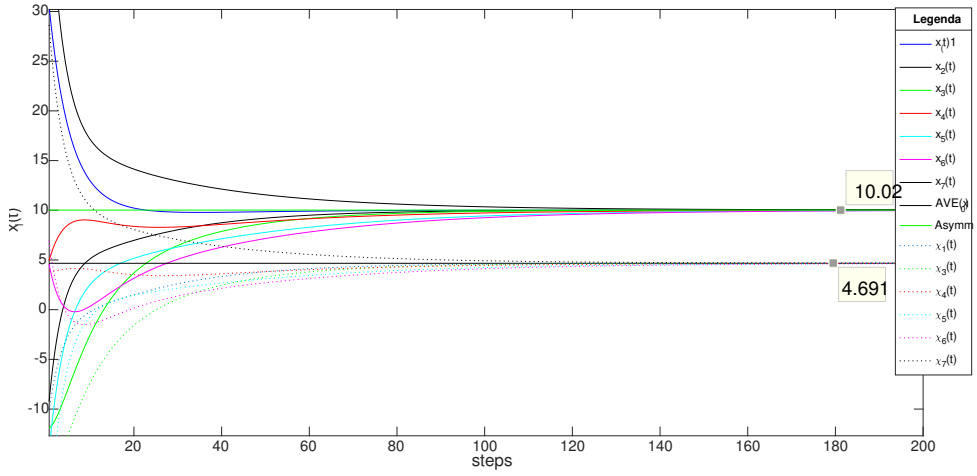


Figure 4. Simulation results: perturbed consensus network. Case $\varepsilon = 0.2$. Solid lines plots show uncompensated trajectories, dashed lines show the evolution with scaled initial conditions.

 doi:10.20944/preprints202408.1310.v1

7. Conclusions

In this paper, a distributed algorithm is derived for the computation of suitable scaling factors which 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. Further research is oriented in different directions, e.g. the extension of the proposed approach to more general and time-varying graphs, considering more general dynamics of a single agent allowing for vehicle and mobile robot network applications, and testing the algorithm in experimental stages of WSN applications.

References

1. Bullo, F. *Lectures on Network Systems*, 1 ed.; 2022.
2. Olfati-Saber, R.; Fax, A.J.; Murray, R. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE* **2007**, *95*, 215–233.
3. Rezaee, H.; Abdollahi, F. Average consensus over high-order multiagent systems. *IEEE Transactions on Automatic Control* **2015**, *60*, 3047–3052.
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 Systems Magazine* **2019**, *39*, 40–72.
5. Garin, F.; Schenato, L. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked control systems*; Springer, 2010; pp. 75–107.
6. Patterson, S.; Bamieh, B.; El Abbadi, A. Convergence rates of distributed average consensus with stochastic link failures. *IEEE Transactions on Automatic Control* **2010**, *55*, 880–892.
7. 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. IEEE, 2011, pp. 1956–1961.
8. Guyeux, C.; Haddad, M.; Hakem, M.; Lagacherie, M. Efficient distributed average consensus in wireless sensor networks. *Computer Communications* **2020**, *150*, 115–121.
9. Sun, S.; Chen, F.; Ren, W. Distributed average tracking in weight-unbalanced directed networks. *IEEE Transactions on Automatic Control* **2020**, *66*, 4436–4443.
10. Sen, A.; Sahoo, S.R.; Kothari, M. Distributed average tracking with incomplete measurement under a weight-unbalanced digraph. *IEEE Transactions on Automatic Control* **2022**, *67*, 6025–6037.
11. Du, M.; Meng, D.; Wu, Z.G. Distributed averaging problems over directed signed networks. *IEEE Transactions on Control of Network Systems* **2021**, *8*, 1442–1453.
12. Shafi, S.Y.; Arcak, M.; El Ghaoui, L. Graph weight allocation to meet Laplacian spectral constraints. *IEEE Transactions on Automatic Control* **2011**, *57*, 1872–1877.
13. Hao, H.; Barooah, P. Improving convergence rate of distributed consensus through asymmetric weights. In Proceedings of the 2012 American Control Conference (ACC). IEEE, 2012, pp. 787–792.
14. Gao, L.; Zhao, J.; Di, Z.; Wang, D. Asymmetry between odd and even node weight in complex networks. *Physica A: Statistical Mechanics and its Applications* **2007**, *376*, 687–691.
15. Parlangeli, G. Laplacian Eigenvalue Allocation Through Asymmetric Weights in Acyclic Leader-Follower Networks. *IEEE Access* **2023**, *11*, 126409–126419.
16. Parlangeli, G. Prescribed-time average consensus through data-driven leader motion. *IEEE Access* **2024**.
17. Valcher, M.E.; Parlangeli, G. On the effects of communication failures in a multi-agent consensus network. In Proceedings of the 2019 23rd International Conference on System Theory, Control and Computing (ICSTCC). IEEE, 2019, pp. 709–720.
18. Parlangeli, G. A Supervisory Algorithm Against Intermittent and Temporary Faults in Consensus-Based Networks. *IEEE Access* **2020**, *8*, 98775–98786.
19. Dibaji, S.M.; Pirani, M.; Flamholz, D.B.; Annaswamy, A.M.; Johansson, K.H.; Chakrabortty, A. A systems and control perspective of CPS security. *Annual reviews in control* **2019**, *47*, 394–411.
20. Kenyeres, M.; Kenyeres, J. Average Consensus with Perron Matrix for Alleviating Inaccurate Sensor Readings Caused by Gaussian Noise in Wireless Sensor Networks. In Proceedings of the Software Engineering and Algorithms: Proceedings of 10th Computer Science On-line Conference 2021, Vol. 1. Springer, 2021, pp. 391–405.

 doi:10.20944/preprints202408.1310.v1

21. Meyer, C.D. Uncoupling the Perron eigenvector problem. *Linear Algebra and its applications* **1989**, *114*, 69–94.
22. Liu, Z.; Guo, L. Synchronization of multi-agent systems without connectivity assumptions. *Automatica* **2009**, *45*, 2744–2753.
23. Sardellitti, S.; Giona, M.; Barbarossa, S. Fast distributed average consensus algorithms based on advection-diffusion processes. *IEEE Transactions on Signal Processing* **2009**, *58*, 826–842.
24. 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. IEEE, 2011, pp. 6660–6665.
25. Song, Z.; Taylor, D. Asymmetric Coupling Optimizes Interconnected Consensus Systems. *arXiv preprint arXiv:2106.13127* **2021**.
26. Parlangeli, G.; Valcher, M.E. Leader-controlled protocols to accelerate convergence in consensus networks. *IEEE Transactions on Automatic Control* **2018**, *63*, 3191–3205.
27. Parlangeli, G.; Valcher, M.E. Accelerating consensus in high-order leader-follower networks. *IEEE Control Systems Letters* **2018**, *2*, 381–386.
28. Mirzaev, I.; Gunawardena, J. Laplacian dynamics on general graphs. *Bulletin of mathematical biology* **2013**, *75*, 2118–2149.
29. Dietzenbacher, E. Aggregation in multisector models: using the Perron vector. *Economic Systems Research* **1992**, *4*, 3–24.
30. Schenato, L.; Fiorentin, F. Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica* **2011**, *47*, 1878–1886.
31. 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 of Things Journal* **2022**, *10*, 8215–8227.
32. Wu, J.; Li, X. Collective synchronization of Kuramoto-oscillator networks. *IEEE Circuits and Systems Magazine* **2020**, *20*, 46–67.
33. Duan, Y.; He, X.; Zhao, Y. Distributed algorithm based on consensus control strategy for dynamic economic dispatch problem. *International Journal of Electrical Power & Energy Systems* **2021**, *129*, 106833.
34. Macii, D.; Rinaldi, S. Time synchronization for smart grids applications: Requirements and uncertainty issues. *IEEE Instrumentation & Measurement Magazine* **2022**, *25*, 11–18.
35. Bullo, F.; Cortés, J.; Martínez, S. *Distributed Control of Robotic Networks*; Princeton University Press, 2009. Applied Mathematics Series.
36. Parlangeli, G. A distributed algorithm for the assignment of the Laplacian spectrum for path graphs. *Mathematics* **2023**, *11*, 2359.
37. Usmani, R.A. Inversion of a tridiagonal Jacobi matrix. *Linear Algebra and its Applications* **1994**, *212*, 413–414.
38. 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. VDE, 2011, pp. 1–8.