IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 October 2023, accepted 31 October 2023, date of publication 8 November 2023, date of current version 15 November 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3331592*

RESEARCH ARTICLE

# Laplacian Eigenvalue Allocation Through Asymmetric Weights in Acyclic Leader-Follower Networks

**GIANFRANCO PARLANGELI**, (Member, IEEE)
Dipartimento di Ingegneria dell'Innovazione, Università del Salento, 73100 Lecce, Italy
e-mail: gianfranco.parlangeli@unisalento.it

This work was supported by the Agreement Between IEEE and Conferenza dei Rettori delle Universitá Italiane (CRUI).

**ABSTRACT** This paper tackles the eigenvalue allocation problem through an appropriate choice of the edge weights for the class of *combinatorially-symmetric* Laplacian matrices of acyclic graphs, namely Laplacian matrices showing symmetric zero/nonzero values in its entries according to a tree graph pattern. The mathematical setting of the problem is remarkably suited for several current multi-agent systems engineering applications, when the communication graph is bidirectional but each agent can set the weight of each incoming neighbor value. The resulting algorithm is inherently iterative and it requires a finite time execution, so that it is well fit for real-world applications as a preliminary routine. For this reason, a special focus is devoted to a distributed implementation of the main algorithm. As a final theoretical result, it is proved that, under the strict interlacing property, the solution is positive, and the algorithm can be iterated. An illustrative example closes the paper, showing how the algorithm works in practice.

**INDEX TERMS** Laplacian eigenvalue allocation, consensus networks, tree graphs, spectral graph theory.

## I. INTRODUCTION

Over the last decades, ever-increasing research activity has been focused on the analysis and design of complex systems made of several interconnected devices, trying to explain the role and effects of local interactions on global features of the system [1].

One major challenge in such a framework is the design of iterative algorithms based on local data that allow to reach a common decision, and the concept of *consensus* is used as a primary tool from various scientific communities [2]. This topic has been investigated with a large effort from researchers of several diverse fields for its impact on several technological areas, ranging from the electric grid and microgrids control [3], [4] distributed signal processing [5], formation control of vehicles and devices [6], robotic networks [7], and several applications of distributed Artificial Intelligence [1], [8].

The dynamical properties of such evolving networks are closely connected and intertwined with the spectral properties of the interconnection graph and in particular with its Laplacian matrix [9], and a renewed thrust of research was devoted to spectral graph theory and the properties of the Laplacian matrix [10].

Graph Laplacians are graph-theoretic matrices whose spectral properties are strictly related to several modern research fields such as graph clustering [11], sparse coding and classification problems [12], consensus, synchronization, and other multi-agent self-organized activities on graphs [13], clustering, and other key tasks in unsupervised learning [14] among others. In recent years, spectral graph theory has been explored also to model and understand signal propagation in the human brain [15], [16].

The spectrum of the Laplacian is strongly related to the fundamental features and limitations of any network dynamics [12], [17], [18]. For example, the smallest positive eigenvalue of a Laplacian is a common measure of how well connected the network is and how fast the system converges, while the largest eigenvalue is fundamental for the stability

The associate editor coordinating the review of this manuscript and approving it for publication was Qiang Li.

of discrete-time consensus and several continuous-time formation control algorithms [19]. Grounded Laplacians, namely Laplacian principal submatrices, have been also widely investigated [20]. The properties of a grounded Laplacian play also a key role in leader-follower networks and opinion dynamics in the presence of stubborn agents [21].

However, real-world applications can be affected by several unavoidable inaccuracies and errors, and some drawbacks may arise. For example, poorly connected networks with a large number of participants may suffer from slow convergence rates. In the presence of slow converge rates together with small disturbances or communication errors, the network evolution often deviates from the nominal evolution toward another fault-driven unpredicted evolution, it may not converge to consensus, and even become unstable [22], [23].

Another real-world urgent issue is the possible presence of a misbehaving agent that behaves to disrupt the network evolution and to ruin the network mission goal [17]. It is often a consequence of the architecture of such networks, which is usually spatially distributed in a large area where other non-interacting agents may be present.

In such applications, the knowledge of the nominal behavior is of great importance to have predictive ability on the network evolution [17], and more specifically the knowledge of the Laplacian eigenvalues is often required for the design of network monitoring filters for detecting topology variations [24], [25]. In this framework, it is worth remarking that pole placement is a well-assessed technique for the design of fault detection filters [26], [27].

The goal of this paper is to thoroughly study the Laplacian eigenvalue allocation problem for a tree graph by an appropriate assignment, possibly asymmetric, of the edge weights.

From a mathematical standpoint, the problem afforded in this paper can be cast into the so-called inverse eigenvalue problems (IEPs) [28]. Indeed, from a mathematical standpoint, the goal of this paper is to understand if it is possible to set the spectrum of some matrices to some prescribed values by the choice of some available parameters. This kind of problem is known as the *inverse eigenvalue problem* [28], and a different yet close problem that falls into this same research area is the pole allocation of a dynamical system by means of state or output feedback [28], which is a long-standing fundamental problem in systems theory [29], [30]. For this reason, we name the problem of this paper as the *eigenvalue allocation problem* by edge weight assignment.

The idea of setting the Laplacian eigenvalues of a graph by an appropriate choice of the edge weights is not new in the control systems community.

Several authors studied the weight design for a graph to impose a favorable spectral structure to the related Laplacian matrix. When dealing with symmetric Laplacians, the exact allocation problem is not solvable and it is usually cast into optimization problems [31], [32]. However, in the last years, there have been attempts to solve the eigenvalue allocation problem by means of asymmetric edge values [19], [33], [34]. In [33], the authors study the problem of the edge weights assignment for directed graphs to impose a prescribed spectrum to their Laplacian matrix, and provide necessary and sufficient conditions for their solvability, together with an explicit solution, in the case of graphs with two and three vertices. In [19], the authors seek conditions for weight allocation in order to bind Laplacian eigenvalues within a prescribed threshold. The mathematical setting adopted in this paper is the same as [34], [35], and [36].

The results of this paper extend those for path graphs [37] to the general class of acyclic graphs. The contribution of this paper is twofold. On one hand, the results achieved in this paper concur to have a thorough insight into the spectral properties of the Laplacian matrix of acyclic graphs and, moreover, they can be directly applied to several multi-agent systems and robotic network applications, as those provided for example in Section II. Though this paper is focused on the real eigenvalue allocation problem, most of the results can be adapted to the general complex case. However, in view of the additional mathematical details and for the sake of clarity, we devote a separate future investigation for the latter framework.

### A. PAPER ORGANIZATION AND NOTATION

This paper is organized as follows. In Section II, a mathematical framework stemming from a number of modern engineering applications is described, and the problem formulation is stated. In Section III some preliminary results on star and path graphs are reported. In Section IV the main theoretical results are derived, which are the basis of the algorithm proposed in this paper. Section VI is devoted to a distributed implementation of the resulting Algorithm, and finally Section VII describes how the algorithm works in practice when it is applied to a 9-nodes tree network.

In the following, we adopt the standard notation of $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{R}_{+}$ for the natural, real, non-negative real numbers, and positive real numbers. The symbol $\mathbf{e}_i^j$ denotes the $i$-th element of the canonical basis of $\mathbb{R}^j$, e.g. $\mathbf{e}_1^5 = \begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}^\top$.

For a square matrix $A \in \mathbb{R}^{n\times n}$, $[A]_{ij}$ is the $(i,j)$-th entry and $(A)_{ij} \in \mathbb{R}^{(n-1)\times(n-1)}$ is the submatrix obtained by removing the $i$-th row and $j$-th column of $A$. The *Laplace expansion rule* for the determinant is:

$$\det A = \sum_{j=1}^{n} (-1)^{i+j} [A]_{ij} \det (A)_{ij} .$$

A matrix $A$ is *non-negative* (positive) if its entries satisfy $[A]_{ij} \geq 0$ ($[A]_{ij} > 0$). A matrix $A \in \mathbb{R}^{n\times n}$ is *combinatorially symmetric* [38] or *structurally symmetric* when $[A]_{ij} \neq 0$ if and only if $[A]_{ji} \neq 0$.

A graph is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = (\{1, 2, \ldots, n\}$ and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$; whenever $(i,j) \in \mathcal{E}$, then $i$ and $j$ are called *neighbors*, and we denote it as $i \sim j$. The set of neighbors of a node $i \in \mathcal{V}$ is denoted as $N_i = \{j \in \mathcal{V} | (i,j) \in \mathcal{E}\}$.

A graph $\mathcal{G}$ is *undirected* when $(j,i) \in \mathcal{E}$ if and only if $(i,j) \in \mathcal{E}$, otherwise it is called directed. A graph 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 $w$, $\mathcal{T}_v^{(w)}$ denotes the tree of $\mathcal{T}^{(w)}$ having $v$ as a vertex. Finally, $\bar{\mathcal{T}}_v^{(w)}$ denotes the subgraph of $\mathcal{T}_v^{(w)}$ after deleting vertex $v$ and all edges incident to $v$. If node $w$ is clear from the context (e.g., when $w$ is the leader node), we use the simplified notation $\mathcal{T}_v$ and $\bar{\mathcal{T}}_v$.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, with associated binary adjacency matrix $A \in \mathbb{R}^{n\times n}$ defined as $[A]_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and $[A]_{ij} = 0$ if $(i,j) \notin \mathcal{E}$. The *weighted adjacency matrix* is defined 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 corresponding *weighted Laplacian matrix* is defined as $[L_w]_{ij} = -[A_w]_{ij}$ if $i \neq j$ and $[L_w]_{ii} = \sum_{j=1, i\neq j}^{n} [A_w]_{ij}$. Finally, the $\mathbb{R}^{(n-1)\times(n-1)}$ matrix obtained by deleting the $\ell$-th row and $\ell$-th column of $L_w$ is called the *grounded Laplacian* (or *Dirichlet Laplacian* matrix) (see f.i. [39], [40]) and it is denoted in the following by $\bar{L}_w^{(\ell)}$; if node $\ell$ is clear from the context (e.g. if $\ell$ is the leader node) we also use the simplified notation $\bar{L}_w$.

The dynamical behavior of a multi-agent networked system is related to the spectral properties of the Laplacian matrix of the communication graph of the network [10]. By construction, $L_w\mathbf{1} = \mathbf{0}$ for any $L_w \in \mathcal{L}_w$ and, by arguments based on the Geršgorin Theorem, its eigenvalues are non-negative, i.e. the Laplacian matrix is a combinatorially symmetric positive semi-definite *zero-row sum* matrix.

In the following, we describe some eigenvalue-related quantities of great importance in modern applications. The *spectrum* of $A \in \mathbb{R}^{n\times n}$ is the set of its eigenvalues and the *spectral radius* of $A$, denoted by $\rho_A$, is the maximum modulus of its eigenvalues. Organize the eigenvalues of $L_w$ as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$; the eigenvalue $\lambda_2$ is known as the *algebraic connectivity* (also known as Fiedler value or Fiedler eigenvalue) of $L_w$, it is nonzero if $\mathcal{G}$ is connected and it dictates the convergence rate of consensus processes [10]. Finally, two quantities of great importance for the synchronization of multi-agent systems are the *spectral gap* $\bar{\delta} = \max\{|1-\lambda_2|, |1-\lambda_n|\}$ [41] in case of discrete-time systems, and the *eigen-ratio* $r = \lambda_n/\lambda_2$ [13] for continuous time dynamic networks [13].

Conversely, for a graph $\mathcal{G}$, we use the symbol $\mathcal{L}_w(\mathcal{G})$ to denote the set of all possible Laplacian matrices whose graph is $\mathcal{G}$.

## II. PROBLEM SETTING

Before stating the problem properly, we briefly describe the abstract mathematical framework of reference, which is adopted in several modern engineering applications in the area of distributed networked systems [10], [42], [43]. After, we briefly provide some examples of fields of application.

In fact, in a few words, the goal of this paper is to derive an algorithm to set the $2\cdot(n-1)$ eigenvalues of some matrices to some prescribed values by the choice of $2\cdot(n-1)$ parameters. This kind of problem is known as the *inverse eigenvalue problem* [28]. From a mathematical perspective, it is necessary to preliminary inquire the *feasibility of the problem*, namely the conditions on $\mathcal{T}$, $\Lambda$ and $\Lambda^{(1)}$ that ensure the existence of a solution. In this respect, it is worth remarking that the general IEP problem regarding the existence of the solution for a tree graph is a long-standing, widely debated and, surprisingly, it is still an open problem [44].

However, the engineering applications motivating this research activity require a working assumption which in turn ensures the existence of a solution for any choice of the desired spectrum. We start by describing the reference framework of this paper.

### A. CONSENSUS-BASED NETWORKS

Consider a set of networked agents, $i \in \{1,\dots,N\}$ each holding a variable $x_i(t)$ which is updated by each node according to $\dot{x}_i(t) = u_i(t)$, where $u_i(t)$ is an input signal that can be set by node $i$.

Each node $i$ is assumed to be able to exchange its own value with a subset of *neighbor nodes* according to a communication graph $G = \{V, E\}$ which describes the communication among nodes, so that each node $i$ can set $u_i(t)$ as

$$u_i(t) = \sum_{j\in\mathcal{N}_i} k_{ij}(x_j(t) - x_i(t)) \tag{1}$$

with gains $k_{ij}$ set by node $i$.

Consider now that one node, say node 1, adds a driving signal $v_1(t)$ to its own $u_1(t)$ as a control input for the team [42]. In this framework, known as *leader follower network* [8], [45], [46], [47], the system dynamics is:

$$G_1(s) = \quad \mathbf{e}_1^T(sI + L_\kappa)^{-1}\mathbf{e}_1 = \frac{\det(sI + \bar{L}_k^{(1)})}{\det(sI + L_\kappa)} = \frac{b(s)}{a(s)}.$$

where $\mathbf{e}_1 = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^\top$ and $L_k \in \mathcal{L}_\kappa$.

It is now worth noting that the poles and zeros of the networked system are, respectively, the eigenvalues of $L_\kappa$ and $\bar{L}_k^{(1)}$, and hence their placement allows to fix the fundamental input-output properties of the whole network from the leader node such as the zero-pole distance, the algebraic connectivity, the spectral gap or the eigen-ratio.

### B. SOME FIELDS OF APPLICATION

The above abstract setting is applied to several modern technological applications, for example:

*Robotic networks* [43], where the multi-agent setting allows robots to coordinate and perform global actions without relying on a central supervisory device. Common global tasks are robot rendezvous (agents meeting at a common location), deployment (agents spreading out in a desired pattern), and formation control (agents maintaining a specific formation). *Artificial intelligence*. Consensus is a

long-debated topic in the field of computer networks [48]. In modern applications, it is one key tool of distributed artificial intelligence, and it is applied in several modern applications, such as blockchain [49]. *Wireless Sensor Networks*. The multi-agent setting is employed to address fundamental issues in wireless sensor networks, such as synchronization [50], or averaging algorithms [51]. *Electrical Power Systems*. With the evolution of the electric grid toward a smart grid, the multi-agent setting is widely adopted to tackle new emerging challenges such as load balancing, fault detection and isolation, demand response, and others [52], [53].

*Remark 1: The above examples deeply inspire the mathematical framework of this paper, which is formalized below. However, the results achieved in this paper are mainly mathematical, namely, how to fix the eigenvalues of an asymmetric Laplacian matrix through the choice of its nonzero entries. These results can be applied also to completely different scenarios, when eigenvalue position is necessary to satisfy some assumption or property required to solve a different problem.*

### C. PROBLEM STATEMENT

We are now ready to state the problem tackled in this paper.

**Problem Statement.** Given a tree graph $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, and two sets $\Lambda = \{\lambda_2, .., \lambda_n\}$ and $\Lambda^{(1)} = \{\mu_1, .., \mu_{n-1}\}$ where $\lambda_i, \mu_i \in \mathbb{R}$ such that

$$0 < \mu_1 < \lambda_2 < \cdots < \mu_{n-1} < \lambda_n \tag{2}$$

find weights $\alpha_{ij}$, such that $\alpha_{ij} = 0$ if $(j, i) \notin \mathcal{E}$ and $\alpha_{ij} > 0$ if $(j, i) \in \mathcal{E}$ so that $\det[sI - L_w] = s(s - \lambda_2) \dots (s - \lambda_n)$ and $\det[sI - \bar{L}_w^{(1)}] = (s - \mu_1) \dots (s - \mu_{n-1})$.

*Remark 2: The problem statement is formulated under the strictly interlacing property to avoid spectrum overlap between $\Lambda$ and $\Lambda^{(1)}$ in order to ensure full controllability and observability by the leader upon the follower team [54] when using the edge weights resulting from the proposed Algorithm. This condition, in turn, ensures the feasibility of the problem for any $\Lambda \subset \mathbb{R}$ and $\Lambda^{(1)} \subset \mathbb{R}$ [28]. Most of the results of this paper can be extended also to the complex plane, however, in view of the additional mathematical details for the sake of clarity, we keep our investigation limited to the real axis.*

## III. PRELIMINARY RESULTS ON STAR AND PATH GRAPHS

In this Section, we describe some preliminary results regarding star and path graphs which are useful in the following of the paper. Indeed the general solution turns out to be iterative, and the final steps of the resulting iterative algorithm require to deal with either path or star shaped subgraphs, which are the outermost portions of any tree graph.

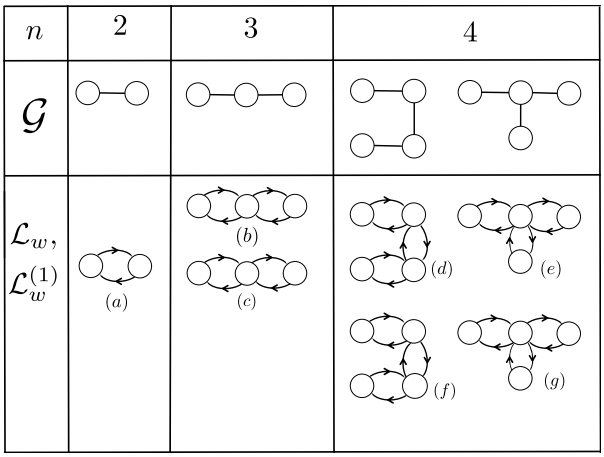


**FIGURE 1.** Sketch of all possible graphs and weighted Laplacians studied in this paper for 2, 3 and 4 nodes.

Consider case *(c)* of Fig.1 first. The related matrices are:

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, A_w = \begin{bmatrix} 0 & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & 0 & 0 \\ \alpha_{31} & 0 & 0 \end{bmatrix}, \tag{3}$$

with associated Laplacian and grounded Laplacian

$$L_w = \begin{bmatrix} \alpha_{12} + \alpha_{13} & -\alpha_{12} & -\alpha_{13} \\ -\alpha_{21} & \alpha_{21} & 0 \\ -\alpha_{31} & 0 & \alpha_{31} \end{bmatrix}, \bar{L}_w^{(1)} = \begin{bmatrix} \alpha_{21} & 0 \\ 0 & \alpha_{31} \end{bmatrix}. \tag{4}$$

Simple computations show that:

$$\begin{aligned} p_w(s) &= \det[sI - L_w] = s(s^2 - a_1 s + a_2), \\ q_w(s) &= \det[sI - \bar{L}_w^{(1)}] = (s - \alpha_{21})(s - \alpha_{31}) \end{aligned} \tag{5}$$

with $a_1 = \alpha_{21}+\alpha_{31}+\alpha_{12}+\alpha_{13}$ and $a_2 = \alpha_{21}\alpha_{31}+\alpha_{12}\alpha_{31}+\alpha_{13}\alpha_{21}$. By recurring to the polynomial identity principle, it is possible to derive that, for any $0 < \mu_1 < \lambda_1 < \mu_2 < \lambda_2$, the choice of

$$\begin{aligned} \alpha_{21} &= \mu_1 \qquad\qquad \alpha_{31} = \mu_2 \\ \alpha_{12} &= \frac{(\lambda_2 - \mu_1)(\lambda_1 - \mu_1)}{(\mu_2 - \mu_1)} \\ \alpha_{13} &= \frac{(\mu_2 - \lambda_1)(\lambda_2 - \mu_2)}{(\mu_2 - \mu_1)} \end{aligned}$$

solve the problem. The above result can be generalized to the case of a $n$-star graph, when the leader is the center. In fact, the following Proposition holds.

*Theorem 3: Let $\mathcal{S}$ be a star graph having n rays, and the central node is selected as leader node. Then, for any given set $\Lambda = \{0, \lambda_2, \lambda_3, .., \lambda_n\}$ and $\Lambda^{(1)} = \{\mu_1, \mu_2, .., \mu_{n-1}\}$ such that $0 < \mu_1 < \lambda_2 < \cdots < \mu_{n-1} < \lambda_n$ there exist positive weights $\alpha_{ij} \in \mathbb{R}_+$ that solve the problem as follows:*

$$L_w = \begin{bmatrix} a_{11} & -\alpha_{12} & \dots & -\alpha_{1n} \\ -\alpha_{21} & \alpha_{21} & 0 & \dots \\ \vdots & 0 & \ddots & \\ -\alpha_{n1} & 0 & \dots & \alpha_{n1} \end{bmatrix} \tag{6}$$

*where coefficients $\alpha_{ij}$ can be computed as*:

$$\begin{cases} \alpha_{j1} &= \mu_j \qquad\qquad for\, j = 2, .., n \\ \alpha_{1j} &= \dfrac{\prod_{k=1}^{n}(\lambda_k - \mu_j)}{\prod_{\substack{k=1 \\ k\neq j}}^{n}(\mu_k - \mu_j)} \end{cases} \tag{7}$$

*and* $a_{11} = \sum_{i=2}^{n} \alpha_{1i}$.

The proof is omitted for the sake of brevity. It can be proved as a special case of general tree graphs, which is thoroughly studied in the next Section.

Further, for the sake of completeness, we provide a statement on the solution of path graphs, which has been recently studied in [37] and, preliminarly in [55]. The interested reader can find any related detail, proof, algorithm in [37].

*Theorem 4: Let $\mathcal{P}$ be a path graph and node 1 one leaf node, and let $L_w$, $\bar{L}_w^{(1)}$ its weighted Laplacian matrix and grounded Laplacian and $\lambda_{i+1}$, $\mu_i$, $i = 1, .., n-1$ satisfying the Problem Statement. Finally, $a(s) = s^n + a_{n-1}s^{n-1} + a_{n-1}s^{n-1} + ..a_1 s$ and $b(s) = s^{n-1} + b_{n-1}s^{n-1} + ..b_1 s + b_0$ are two polynomials having $\lambda_i$ and $\mu_i$ as zeros.*

*It is always possible to compute $\alpha_{ij} \in \mathbb{R}$ such that* $\det[sI - L_w] = p(s)$ *and* $\det[sI - \bar{L}_w^{(1)}] = q(s)$, *by solving iteratively backwards the equations*:

$$\begin{cases} \phi_j(s) = (s - \alpha_{(j-1),j})\phi_{(j-1)}(s) - s\alpha_{j,(j-1)}\psi_{(j-1)}(s) \\ \psi_j(s) = \phi_{(j-1)}(s) - \alpha_{j,(j-1)}\psi_{(j-1)} \end{cases} \tag{8}$$

*with the final conditions $\phi_n(s) = a(s)$, $\psi_n(s) = b(s)$.*

## IV. MAIN RESULTS ON TREE GRAPHS

In this Section, we are ready to tackle the general case of tree graphs. We proceed by induction, starting from the results achieved in the previous Section. There are several differences and additional challenges when dealing with tree graphs compared with star and path graphs. The main difference, from a mathematical perspective, is the cardinality of the neighbor set, starting from the leader node. For path graphs, it is equal to 1, while for tree graphs it is a positive integer. It turns out that, as regards path graphs, the recurrence is realized by each node by computing its own edge weight and sending the remainder to its neighbor (being only one), while in the tree case each node must define a set of polynomials to distribute to its neighbors.

The first result is a technical Lemma which explains how $p(s) = \det[sI - L_w]$ and $q(s) = \det[sI - \bar{L}_w^{(1)}]$ can be written as functions of the weights $\alpha_{1,j}, \alpha_{j,1}$ and the analogous polynomials associated to each subtree $\mathcal{T}_j^{(1)}$, $j \in N_1$. The following analysis is often explored in analogous inverse eigenvalue problems and it is often called neighbors formula (see e.g. [28]). In this paper, Lemma 4 provides a key tool to trigger the iterative procedure that allows each node for computing its own weight.

*Lemma 5: Let $\mathcal{T}$ be a tree graph and $L_w$ its weighted Laplacian with associated polynomials $p(s) = \det[sI - L_w]$ and $q(s) = \det[sI - \bar{L}_w^{(1)}]$, 1 be the leader node and $\ell = |N_1|$. Let $\mathcal{T}_j$, $j = 1, \ldots, \ell$ denote the subtrees of $\mathcal{T}$ after removing node 1, and $p_{\mathcal{T}_i}(s)$ (resp., $q_{\mathcal{T}_i}(s)$) the characteristic polynomials of the weighted Laplacian (weighted grounded Laplacian) of each $\mathcal{T}_i$. The following relations hold*:

$$\begin{cases} p(s) = sq(s) - \sum_{i=1}^{\ell} \alpha_{1i} p_{\mathcal{T}_i}(s) \prod_{\substack{j=1, \\ j\neq i}}^{\ell} \left( p_{\mathcal{T}_j}(s) - \alpha_{j1} q_{\mathcal{T}_j}(s) \right) \\ q(s) = \prod_{i=1}^{\ell} \left( p_{\mathcal{T}_i}(s) - \alpha_{i1} q_{\mathcal{T}_i}(s) \right). \end{cases} \tag{9}$$

*Proof:* As a first step, we write the general weighted Laplacian for a tree, with the labeling as follows. Let $N_1 = \{i_1, .., i_\ell\}$, so that $\mathcal{T}^{(1)}$ is made of $\ell$ subtrees. We label sequentially each subtree, and the first label of each subtree $\mathcal{T}_{i_k}^{(1)}$ is assigned to the neighbor of node 1. It follows that:

$$L_w = \left[\begin{array}{c|ccc|c} a_{11} & -\alpha_{1i_1} & 0 \ \ldots & \ldots & -\alpha_{1i_\ell} \ 0 \ \ldots \\ \hline -\alpha_{i_1 1} & & & & \\ 0 & \bar{L}_{i_1} & & \ldots & \mathbf{0}_{i_1 \times i_\ell} \\ \vdots & & & & \\ \hline \vdots & \vdots & & & \vdots \\ \hline -\alpha_{i_\ell 1} & & & & \\ 0 & \mathbf{0}_{i_\ell \times i_1} & & \ldots & \bar{L}_{i_\ell} \\ \vdots & & & & \end{array}\right] \tag{10}$$

where $\bar{L}_i = L_{\mathcal{T}_i} + \alpha_{i_i 1}\mathbf{e}_1^i \mathbf{e}_1^{i\top}$ with $L_{\mathcal{T}_i}$ being the weighted Laplacian of $\mathcal{T}_i$, and $a_{11} = \sum_{k=1}^{\ell} \alpha_{1i_k}$.

As a first remark, it is worth noting that the grounded Laplacian can be easily written as the product:

$$q_w(s) = \prod_{k=1}^{\ell} \det[sI - \bar{L}_{i_k}] \tag{11}$$

where, analogously, $\bar{L}_{i_k} = L_{\mathcal{T}_{i_k}} + \alpha_{i_{i_k} 1}\mathbf{e}_1^{i_k}\mathbf{e}_1^{i_k\top}$, and that each polynomial $\det[sI - \bar{L}_{i_k}]$ can be directly written as function of $p_{\mathcal{T}_{i_k}}(s)$ and $q_{\mathcal{T}_{i_k}}(s)$:

$$\det[sI - \bar{L}_{i_k}] = p_{\mathcal{T}_{i_k}}(s) - \alpha_{i_k 1} q_{\mathcal{T}_{i_k}}(s) \tag{12}$$

so that the second of (9) follows.

We now compute $p(s) = \det[sI - L_w]$ using the Laplace rule along the first row:

$$\begin{aligned} p(s) = (s - a_{ii}) &\prod_{k=1}^{\ell} \det[sI - \bar{L}_{i_k}] \\ &- \alpha_{1i_1}\alpha_{i_1 1} q_{i_1}(s) \prod_{k=2}^{\ell} \det[sI - \bar{L}_{i_k}] \\ &- \alpha_{1i_2}\alpha_{i_2 1} q_{i_2}(s) \prod_{\substack{k=1, \\ k\neq 2}}^{\ell} \det[sI - \bar{L}_{i_k}] - \ldots \end{aligned} \tag{13}$$

Using (12) it is possible to write

$$\alpha_{1j} p_{\mathcal{T}_j}(s) \prod_{\substack{k=1,\\ k\neq j}}^{\ell} \det[sI - \bar{L}_{i_k}] = \alpha_{1j} \prod_{k=1}^{\ell} \det[sI - \bar{L}_{i_k}] + \alpha_{1j}\alpha_{j1} q_j(s) \prod_{\substack{k=1,\\ k\neq j}}^{\ell} \det[sI - \bar{L}_{i_k}] \tag{14}$$

and, remembering that $a_{11} = \sum \alpha_{1j}$, (13) can be rewritten as

$$p(s) = s \prod_{k=1}^{\ell} \det[sI - \bar{L}_{i_k}] - \sum_{j=1}^{\ell} \left\{ -\alpha_{1j} p_{\mathcal{T}_j}(s) \prod_{\substack{k=1,\\ k\neq j}}^{\ell} \det[sI - \bar{L}_{i_k}] \right\}. \tag{15}$$

Putting together (12) with (15), also the first of (9) follows. □

Equations (9) can be put in a convenient form to set up a two-step algorithm which can be run iteratively with respect to the nodes of the graph. More in detail, the algorithm starts at the leader node with the desired characteristic polynomials $p_w(s)$ and $q_w(s)$ as inputs, and it returns the gain for the edges of the leader node together with a set of $2 \cdot |N_1|$ polynomials related to the $|N_1|$ subtrees of $\mathcal{T}_w$. The resulting equations are derived in the next Theorem.

*Theorem 6: Let $\mathcal{T}$ be a tree graph, considering the notation in Section I-A, $p(s) = \det[sI - L_w]$, $q(s) = \det[sI - \bar{L}_w^{(1)}]$, $a(s) = s(s-\lambda_2)\dots(s-\lambda_n)$, $b(s) = (s-\mu_1)\dots(s-\mu_{n-1})$, $\lambda_i$, $\mu_i$ satisfying the Problem Statement. It is always possible to compute a set of $2\ell$ coefficients $\{\alpha_{1i}, \alpha_{i1}\}_{i\in\{i_1,\dots,i_\ell\}}$ and a set of $2\ell$ polynomials $\{p_i(s), q_i(s)\}_{i\in\{i_1,\dots,i_\ell\}}$, with $p_i(s) = s\hat{p}_i(s)$ and degrees $\partial\hat{p}_i(s) = \partial q_i(s) = n_i - 1$, which satisfy (9) with $p(s) = a(s)$ and $q(s) = b(s)$ (the resulting Algorithm is fully described in the next Subsection (VI)).*

*Proof:* The proof is constructive and the resulting algorithm that solves the stated problem is described in detail after this proof.

Divide $\{\mu_1, .., \mu_{n-1}\}$ into $\ell = |N_1|$ sets, say $\Lambda_{i_1}$, $\Lambda_{i_2}$,..., $\Lambda_{i_\ell}$, and build $q_i(s) = \prod_{\mu_i\in\Lambda_i}(s-\mu_i)$ so that $q(s) = \prod_{i=1}^{\ell} q_i(s)$. Set $\bar{q}_i(s) = \prod_{\substack{j=1,\\ j\neq i}}^{\ell} q_j(s)$, so that $q_k(\mu_i) = 0$ for any $\mu_i \in \Lambda_k$ and $q_k(\mu_i) \neq 0$ for any $\mu_i \in \Lambda_j$ with $j \neq k$, and conversely $\bar{q}_k(\mu_i) \neq 0$ for $\mu_i \in \Lambda_k$ and $\bar{q}_j(\mu_i) = 0$ for $\mu_i \in \Lambda_j$ with $j \neq k$. Finally set

$$\begin{cases} \gamma(s) &= p(s) - sq(s) \\ x_i(s) &= \alpha_{1i} p_{\mathcal{T}_i}(s) \qquad\qquad i = i_1, .., i_\ell \end{cases} \tag{16}$$

where $x_i(s)$ $i = 1, .., \ell$ is a set of $\ell$ unknown polynomials of degree equal to $n_i$ and with a simple zero in 0, namely $x_i(s) = x_{n_i} s^{n_i} + x_{n_i-1} s^{n_i-1} + .. + x_1 s$.

Under the above notation, the first of Eq.(9) takes the form:

$$\gamma(s) = -\sum_{i=1}^{\ell} x_i(s)\bar{q}_i(s) \tag{17}$$

where we recall that $\bar{q}_i(s) = \prod_{\substack{j=1,\\ j\neq i}}^{\ell} q_j(s)$ and it is a generalized polynomial Diophantine equation (an ordinary Diophantine equation for $\ell = 2$) (see, e.g., [56]). The existence and structure of the solution can be studied recurring to the methods of [56] using a vector representation of (17) as $\gamma(s) = \bar{Q}(s)\mathbf{x}(s)$ where $\bar{Q}(s) = [\bar{q}_1(s)\ \bar{q}_2(s)\ \dots\ \bar{q}_\ell(s)]$ and $\mathbf{x}(s) = [x_1(s)\ x_2(s)\ \dots\ x_\ell(s)]^T$. However, in order to trigger an effective induction procedure, it is important to seek solutions with appropriate degree, namely $\partial p_{\mathcal{T}_i}(s) = |\mathcal{T}_i| = n_i$. A convenient way to compute the coefficients of $x_i(s)$ with a prescribed degree $n_i$ is as follows.

Compute Eq. (17) at $s = \mu_i$ for each $\mu_i \in \Lambda^{(1)}$; by construction of $\bar{q}_i(s)$, it results:

$$\begin{aligned} p(\mu_i) &= -x_1(\mu_i)\bar{q}_1(\mu_i) && \forall \mu_i \in \bar{\Lambda}_1 \\ p(\mu_i) &= -x_2(\mu_i)\bar{q}_2(\mu_i) && \forall \mu_i \in \bar{\Lambda}_2 \\ &\vdots \end{aligned} \tag{18}$$

and in general

$$p(\mu_i) = -x_k(\mu_i)\bar{q}_k(\mu_i) \quad \forall \mu_i \in \bar{\Lambda}_k \quad k = 1, .., \ell. \tag{19}$$

The above equations have the advantage of decoupling the computation for each subtree $\mathcal{T}_k$. Considering the unknowns $x_{n_k}, x_{n_k-1}, ..$ for a given $k$, equations (19) can be put as follows:

$$\underbrace{\begin{bmatrix} \mu_1^{n_k-1} & \mu_1^{n_k-2} & \dots & 1 \\ \mu_2^{n_k-1} & \mu_2^{n_k-2} & \dots & 1 \\ \vdots & 0 & \ddots & \vdots \\ \mu_{n_k-1}^{n_k-1} & \mu_{n_k-1}^{n_k-2} & \dots & 1 \end{bmatrix}}_{V_k} \begin{bmatrix} x_{n_k} \\ x_{n_k-1} \\ \vdots \\ x_1 \end{bmatrix} = \begin{bmatrix} -\dfrac{p(\mu_1)}{\bar{q}_k(\mu_1)\mu_1} \\ -\dfrac{p(\mu_2)}{\bar{q}_k(\mu_2)\mu_2} \\ \vdots \\ -\dfrac{p(\mu_{n_k-1})}{\bar{q}_k(\mu_{n_k-1})\mu_{n_k-1}} \end{bmatrix} \tag{20}$$

where $V_k$ is a Vandermonde matrix built on the zeros of $q_k(s)$. Its determinant is equal to

$$\det V_k = \prod_{1\leq i<j\leq n_k} (\mu_j - \mu_i),$$

so that it is nonsingular under the strict interlacing property, and it allows to compute the polynomial $x_k(s) = x_{n_k} s^{n_k} + x_{n_k-1} s^{n_k-1} + .. + x_1 s$ for a $\mathcal{T}_k$ with $n_k = |\mathcal{T}_k|$.

Once that $x_k(s)$ is determined, the complete solution can be easily computed. As a first remark, it is worth noting that $x_{n_k} \neq 0$. Indeed, $x_{n_k} = \alpha_{1k}$ and if $\alpha_{1k} = 0$, then the standing Assumption of strict interlacing inequality would be violated (see, e.g., Eq. (10)).

Considering the second of (16), it is easy to compute $\alpha_{1k}$ and $p_{\mathcal{T}_k}(s)$ as:

$$\begin{cases} \alpha_{1k} & = x_{n_k} \\ p_{\mathcal{T}_k}(s) & = s^{n_k} + \dfrac{x_{n_k-1}}{x_{n_k}} s^{n_k-1} + .. + \dfrac{x_{n_k-1}}{x_{n_k}} s \end{cases} \tag{21}$$

and finally $\alpha_{k1}$ and $q_{\mathcal{T}_k}(s)$ can be easily retrieved using the above solution (21) together with relation $q_k(s) = \prod_{\mu_i \in \Lambda_i}(s-\mu_i) = p_{\mathcal{T}_k}(s) - \alpha_{k1} q_{\mathcal{T}_k}(s)$ so that

$$\begin{cases} \alpha_{k1} = \dfrac{x_{n_k-1}}{x_{n_k}} + \displaystyle\sum_{\mu_i \in \Lambda_i} \mu_i \\ q_{\mathcal{T}_k}(s) = \dfrac{1}{\alpha_{k1}} \left[ \left( s^{n_k} + \dfrac{x_{n_k-1}}{x_{n_k}} s^{n_k-1} + .. + \dfrac{x_{n_k-1}}{x_{n_k}} s \right) \right. \\ \qquad \left. - \displaystyle\prod_{\mu_i \in \Lambda_i} (s-\mu_i) \right]. \end{cases} \tag{22}$$

As a final remark, it is easily seen that $q_{\mathcal{T}_k}(s)$ is a monic polynomial of degree $n_k - 1$. □

## V. DISCUSSION ON THE POSITIVITY OF THE SOLUTION AND ON THE ITERATIVENESS OF ALGORITHM 1

In this Section, we address two main technical issues that are necessary to ensure that the resulting Algorithm provides an effective solution for any possible graph and leader node, and every choice of reference polynomials $a(s)$, $b(s)$, and it can be iterated until all edge weights are assigned.

In this respect, it is worth noting that matrices $L_w$ are not symmetric, so they do not necessarily have a real spectrum, nor they are granted to satisfy the interlacing property in advance with any principal submatrix. However, to iterate the result of Theorem 6, these two properties of $p(s)$ and $q(s)$ should hold since they are required as assumptions of the next step of iteration.

Hence, one main point is to study the properties of polynomials $p_{\mathcal{T}_k}(s)$, $q_{\mathcal{T}_k}(s)$ as solutions of the Algorithm resulting from Theorem 6, and to understand if their zeros are real or complex, simple or multiple, and finally, if they satisfy a strict interlace condition between themselves.

A different yet close investigation is about the sign of the edge weights achieved through the Algorithm. Indeed, considering the motivating applicative scenarios, we are primarily interested in positive values for the resulting edge weights.

Fortunately, it comes out that it is possible to prove that the resulting polynomials of Theorem 6 have the required features, as we describe in the next Theorem, and it ensures that the algorithm of Theorem 6 can be iterated without any restriction.

*Proposition 7: Consider the setting and the notation of Theorem 6, and set* $p(s) = \det[sI - L_w]$ *and* $q(s) = \det[sI - \bar{L}_w^{(1)}]$. *The resulting solutions have the following features*:

- *The* $2\ell$ *coefficients* $\{\alpha_{1i}, \alpha_{i1}\}_{i \in \{i_1,..,i_\ell\}}$ *are positive.*
- *The set of* $2\ell$ *polynomials* $\{p_{\mathcal{T}_i}(s), q_{\mathcal{T}_i}(s)\}_{i \in \{i_1,..,i_\ell\}}$ *have real distinct roots which satisfy the interlacing property between themselves and, moreover,* $p_{\mathcal{T}_i}(s)$ *interlaces with* $q_i(s) = \prod_{\mu_i \in \Lambda_i}(s-\mu_i)$.

*Proof:* In this proof, we use the notation $\mathcal{I}_i = \{k \in \mathcal{V} \,|\, \mu_k \in \Lambda_i\}$ and $\bar{\mathcal{I}}_i = \mathcal{V} \backslash \mathcal{I}_i$, namely the sets of indices of all $\mu_i$ belonging to a $\Lambda_i$, and to a $\bar{\Lambda}_i$. Since we set $a(s) = s(s-\lambda_2)\dots(s-\lambda_n)$, $b(s) = (s-\mu_1)\dots(s-\mu_{n-1})$, recurring to the simple partial fraction expansion it is possible to write

$$\frac{(s-\lambda_2)\dots(s-\lambda_n)}{(s-\mu_1)\dots(s-\mu_{n-1})} = 1 + \frac{R_1}{s-\mu_1} + \dots + \frac{R_{n-1}}{s-\mu_{n-1}} \tag{23}$$

where $R_i = \dfrac{\prod_{j=2}^{n}(\mu_i - \lambda_j)}{\prod_{\substack{i=1 \\ j\neq i}}^{n-1}(\mu_i - \mu_j)}$, and it easily seen that the interlacing property among $\lambda_i$ and $\mu_i$ ensures that $R_i < 0$ for any $i$. Based on (23), it is possible to write:

$$a(s) - sb(s) = \sum_{i=1}^{n-1} \left( R_i s \prod_{\substack{j=1 \\ i\neq j}}^{n-1} (s-\mu_j) \right). \tag{24}$$

It is now useful to organize the right-hand side according to the following logic. Group together the terms of the sum according to the sets $\Lambda_i$, so that Eq. (24) can be rewritten as follows:

$$\begin{aligned} a(s) - sb(s) = & \left( \sum_{i\in\mathcal{I}_1} R_i s \prod_{\substack{j\in\mathcal{I}_1 \\ i\neq j}} (s-\mu_j) \right) \prod_{k\in\bar{\mathcal{I}}_1} (s-\mu_k) \\ & + \left( \sum_{i\in\mathcal{I}_2} R_i s \prod_{\substack{j\in\mathcal{I}_2 \\ i\neq j}} (s-\mu_j) \right) \prod_{k\in\bar{\mathcal{I}}_2} (s-\mu_k) + \dots . \end{aligned} \tag{25}$$

Considering that each $\prod_{k\in\bar{\mathcal{I}}_i}(s-\mu_k)$ is equal to $\bar{q}_j(s)$, and comparing Eq.(25) to Eq.(16), it is easy to see that the structure of $x_i(s)$ is:

$$x_i(s) = \sum_{k\in\mathcal{I}_i} R_i s \prod_{\substack{j\in\mathcal{I}_i \\ j\neq k}} (s-\mu_j) \tag{26}$$

and, in turn, Eq.(26) together with (16), reveals that $\alpha_{1k} = -\sum_{k\in\mathcal{I}_i} R_k > 0$.

As for the localization of the roots, note that the value of $x_i(\mu_k)$ changes sign for successive $\mu_k$ so that, if $\nu_i$ denotes a zero of $x_i(s)$, it is easy to infer that

$$\mu_1 < \nu_2 < \mu_2 < \dots < \nu_{n-1} < \mu_{n-1}. \tag{27}$$

Consider now Eq.(22). As for $\alpha_{k1}$, upon defining $\nu_1 = 0$, it is worth noting that the first of Eq.(22) can also be written as $\alpha_{k1} = \sum_{i=1}^{n_k-1}(\mu_i - \nu_i)$ and it is positive in view of Eq.(27).

As a final result, denote the zeros of $q_{\mathcal{T}_i}(s)$ as $\varsigma_i$. Considering the second of (22), namely

$$q_{\mathcal{T}_i}(s) = \frac{p_{\mathcal{T}_i}(s) - \prod_{j\in\mathcal{I}_i}(s-\mu_j)}{\alpha_{i1}},$$

together with the condition previously achieved (27), it is possible to prove analogously that $0 < \varsigma_1 < \nu_2 < \cdots < \varsigma_{n-2} < \nu_{n-1}$. □

## VI. A DISTRIBUTED IMPLEMENTATION OF THE ALGORITHM

Theorem 6 basically allows to splitting of the desired polynomials into a set of reference polynomials of lower degree, together with a set of weights to be assigned to the edges of the leader node. The idea is that such a scheme should be iterated by the leader's neighbors, and so on, to let each node of the network retrieve its own edge weights. In this respect, it is possible to put the solution in a form so that each node can retrieve its own gains by simply means of local data, and transmit the resulting polynomials to its neighbors, thus providing a distributed structure to the resulting Algorithm. In more detail, it is possible to execute the whole Algorithm for the Laplacian eigenvalue assignment as discussed in Theorem 6 as follows. The leader node sets the reference polynomials, it runs Eq. (20), (21) to retrieve the weights of its incoming edges, and finally it transmits the reference polynomials of each subtree to the neighbor nodes. Then, each node applies Eq. (20), (21), (22) to retrieve the weights of its incoming edges, and finally it transmits the reference polynomials to its neighbors, and so on.

The leader node triggers the procedure by running the following algorithm:

**Algorithm 1** Leader Node

**Input:** $\Lambda = \{\lambda_2, \lambda_3, \ldots, \lambda_n\}$, $\Lambda^{(1)} = \{\mu_1, \mu_2, .., \mu_{n-1}\}$
**Output:** $\alpha_{1k}$ and $p_{\mathcal{T}_k}(s)$, $k = 1, .., \ell$
1 Split $\Lambda^{(1)}$ into $\ell$ disjoint sets $\Lambda_{i_1}$,.., $\Lambda_{i_\ell}$ such that $|\Lambda_{i_k}| = |\mathcal{T}_{i_k}| = n_{i_k}$ ;
2 Build polynomials $a(s)$, $q_i(s)$ and compute $\gamma(s)$ as in (16);
3 **for** $k = 1, \ldots, \ell$ **do**
4 Solve (20);
5 Use (21) to retrieve $\alpha_{1k}$ and $p_{\mathcal{T}_k}(s)$.;
6 **end**

and it transmits $p_{\mathcal{T}_k}(s)$, $q_k(s)$ to each neighbor $k \in N_1$. Upon receipt of such data, and based on it, each node $k$ runs the algorithm below:

In general, the lines of Algorithm 1 and Algorithm 2 do not require a special computational effort by the elaborating node $k$. As regards line 4 of Algorithm 1, it is worth noting that the Vandermonde matrix has been widely studied, and its inverse is well known in closed form (see, e.g., [57]).

**Algorithm 2** Node $k$

**Input:** $p_{\mathcal{T}_k}(s)$, $q_k(s)$
**Output:** $\alpha_{kj}$ and $p_{\mathcal{T}_j}(s)$, $j \sim k$
1 Compute (22) to retrieve $\alpha_{k\ell}$ and $q_{\mathcal{T}_k}(s)$ ;
2 Compute the real simple roots of $q_{\mathcal{T}_k}(s)$;
3 Split the roots of $q_{\mathcal{T}_k}(s)$ into $\ell_k$ disjoint sets $\Lambda_{i_1}$,.., $\Lambda_{i_{\ell_k}}$ such that $|\Lambda_{i_k}| = |\mathcal{T}_{i_k}| = n_{i_k}$;
4 **for** $k = 1, \ldots, \ell$ **do**
5 Solve (20);
6 Use (21) to retrieve $\alpha_{kj}$ and $p_{\mathcal{T}_j}(s)$ for each $j \sim k$;
7 **end**

Lastly, line 2 of Algorithm 2 requires to compute the roots of $q_{\mathcal{T}_k}(s)$, and it requires an interative numerical computation. However, this step is made easy by the previous result of Proposition 7, namely the feature that $q_{\mathcal{T}_k}(s)$ has all real simple roots which are roughly localized thanks to the bounds of the interlacing property. In our simulations, we used the technique [58], where it is possible to find all the numerical details of this routine. In particular, it is possible to find all the details in terms of precision of the solution with respect to the cost (i.e. number of bit operations). Starting from these relations, together with the details of a specific framework (namely, the computational capacity of each node), it is always possible to find an upper bound of the time execution of Algorithm 2. All the other lines require the execution of simple elaborations. The above discussion clarifies that Algorithm 1 and Algorithm 2 are feasible, and an upper bound of the time execution is computable, so the algorithm is well fit for real-time applications. Indeed, the whole procedure requires a finite amount of time equal to the eccentricity of the leader node, and this makes it possible to run this algorithm as a preliminary routine and to set the weights corresponding to a prescribed spectrum before a mission execution.

## VII. A CONSTRUCTIVE EXAMPLE

In this Section, we describe the distributed algorithm in practice through an illustrative example.

Consider a multi-agent system as depicted in Fig. 2, the goal for the leader it to let the agents assign their edge weights to get the Laplacian spectrum equal to:

$$\Lambda = \{0, 2, 4, 6, 8, 10, 12, 14, 16\},$$
$$\Lambda^{(1)} = \{1, 3, 5, 7, 9, 11, 13, 15\};$$

the leader node is the dark node with label 1. Node 1 triggers the algorithm, it computes:

$$a(s) = s\prod_{i=1}^{8}(s-2i), \qquad b(s) = \prod_{i=0}^{7}(s-2i-1),$$
$$\Lambda_1 = \{3, 11\}, \quad \Lambda_2 = \{1, 9, 13\}, \quad \Lambda_3 = \{5, 7, 15\},$$

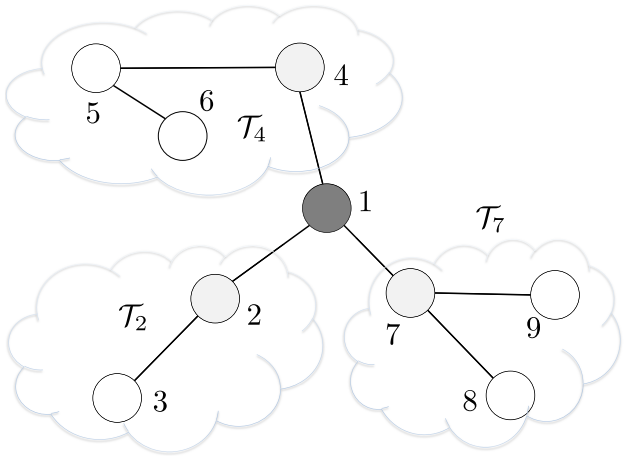


**FIGURE 2.** Simulation results: Subdivision of the graph according to the notation of Section II.

so that:

$$q_1(s) = (s-3)(s-11), \quad q_2(s) = (s-1)(s-9)(s-13),$$
$$q_3(s) = (s-5)(s-7)(s-15),$$
$$\bar{q}_1(s) = q_2(s)q_3(s), \bar{q}_2(s) = q_1(s)q_3(s), \bar{q}_3(s) = q_1(s)q_2(s).$$

Then, node 1 runs Algorithm 1 with the following output:

| | |
|---|---|
| $\alpha_{12}$ | 1.613 |
| $p_{\mathcal{T}_2}$ | $s^2 - 7.5s$ |
| $\alpha_{14}$ | 1.44 |
| $p_{\mathcal{T}_4}$ | $s^3 - 23.55s^2 + 133s$ |
| $\alpha_{17}$ | 4.95 |
| $p_{\mathcal{T}_7}$ | $s^3 - 14.6s^2 + 28.8s$ |

and finally it transmits $p_{\mathcal{T}_2}(s)$, $q_2(s)$ to its neighbor 2, and analogously $p_{\mathcal{T}_4}(s)$, $q_4(s)$ to node 4 and $p_{\mathcal{T}_7}(s)$, $q_7(s)$ to node 7.

Then, the procedure prosecutes with the results of Section III and, more in details, node 4 and node 2 are the leaves of path-shaped subtrees, so they first apply Eq. (22), and then Theorem 6, while node 7 is the center of a star-shaped subgraph, and it applies Eq. (22) and then Theorem 4.

Fig. 3 shows the exchanged data and the edge weights computed by each node, and finally Fig. 4 shows the edge values corresponding to the prescribed spectrum $\Lambda$, $\Lambda^{(1)}$.

The resulting Laplacian matrix (28), as shown at the bottom of the page.

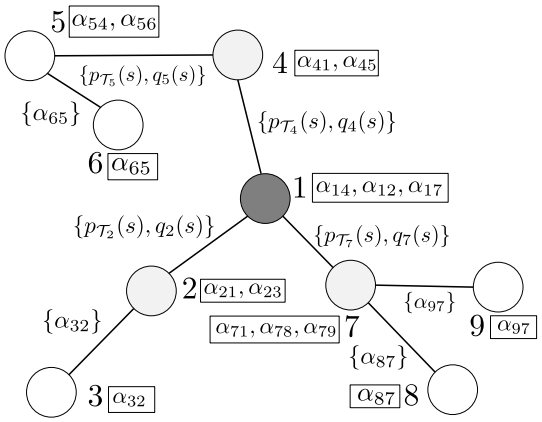


**FIGURE 3.** Simulation results: exchanged data.

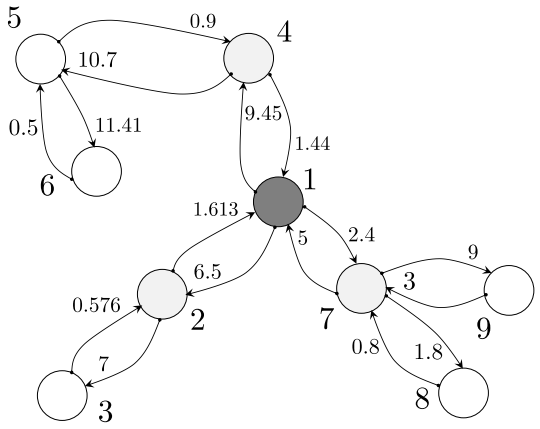


**FIGURE 4.** Simulation results: computed edge weights for the eigenvalue allocation $\Lambda = \{0, 2, 4, 6, 8, 10, 12, 14, 16\}$, $\Lambda^{(1)} = \{1, 3, 5, 7, 9, 11, 13, 15\}$.

*Remark 8: A strict comparison of the above example with other algorithms is not possible, because the inverse eigenvalue problem for such systems was developed only for networks made of $n = 2, 3, 4, 5$ nodes [33], being the investigation regarding asymmetric nodes, as well as optimal symmetric, focused on improving the convergence rate. Considering such a goal, it was already discovered for path graphs [34] that asymmetric weight design outperforms even optimal symmetric, and the convergence rate can be*

$$L_w = \begin{bmatrix} 8 & -1.61 & 0 & -1.44 & 0 & 0 & -4.95 & 0 & 0 \\ -6.48 & 7.06 & -0.576 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -7 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -9.45 & 0 & 0 & 10.34 & -0.9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -10.7 & 11.25 & -0.55 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -11.41 & 11.41 & 0 & 0 & 0 \\ -2.4 & 0 & 0 & 0 & 0 & 0 & 6.2 & -0.8 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.8 & 1.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -9 & 0 & 9 \end{bmatrix} \tag{28}$$

*uniformly bounded away from zero. The results of this paper confirm the beneficial effects of asymmetric weights for the broader class of tree graphs.*

## VIII. CONCLUSION

In this paper, we derived an algorithm for the Laplacian eigenvalue assignment for tree graphs by an appropriate choice of the gains by each agent. This problem is motivated by several recent applications of distributed algorithms for multi-agent systems and robotic networks. The resulting algorithm is inherently iterative and it requires a finite time equal to the eccentricity of the leader position in the communication graph, so that it is well fit in real-world applications as a preliminary routine to be run before any mission or task. Several results are promising, however they are limited to the case of acyclic graphs, and agents modeled as first-order dynamics. Several directions of research stem from the results of this paper, as for example the analogous investigations for the general complex eigenvalue problem and/or investigating more general graphs with cycles, or extending the results of this paper to more complex kinematics including nonholonomic constraints, as those explored in [59].

## ACKNOWLEDGMENT

The author would like to thank Dr Rossella Attanasi for the fruitful discussions on the topic of this paper.

## REFERENCES

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[2] Z. Li, Z. Duan, G. Chen, and L. Huang, "Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 213–224, Jan. 2010.

[3] S. Ullah, L. Khan, I. Sami, and N. Ullah, "Consensus-based delay-tolerant distributed secondary control strategy for droop controlled AC microgrids," *IEEE Access*, vol. 9, pp. 6033–6049, 2021.

[4] M. M. Rana and A. Shahirinia, "Distributed dynamic state estimation considering packet losses in interconnected smart grid subsystems: Linear matrix inequality approach," *IEEE Access*, vol. 8, pp. 2687–2693, 2020.

[5] S. Sardellitti, M. Giona, and S. Barbarossa, "Fast distributed average consensus algorithms based on advection-diffusion processes," *IEEE Trans. Signal Process.*, vol. 58, no. 2, pp. 826–842, Feb. 2010.

[6] Y. Zheng, S. Zhao, Y. Liu, Y. Li, Q. Tan, and N. Xin, "Weighted algebraic connectivity maximization for optical satellite networks," *IEEE Access*, vol. 5, pp. 6885–6893, 2017.

[7] K. Li, R. Gong, S. Wu, C. Hu, and Y. Wang, "Decentralized robust connectivity control in flocking of multi-robot systems," *IEEE Access*, vol. 8, pp. 105250–105262, 2020.

[8] A. Amirkhani and A. H. Barshooi, "Consensus in multi-agent systems: A review," *Artif. Intell. Rev.*, vol. 55, no. 5, pp. 3897–3935, Jun. 2022.

[9] J. A. Almendral and A. Díaz-Guilera, "Dynamical and spectral properties of complex networks," *New J. Phys.*, vol. 9, no. 6, p. 187, Jun. 2007.

[10] F. Bullo, *Lectures on Network Systems*, 1st ed. Seattle, WA, USA: Kindle Direct Publishing, 2022. [Online]. Available: http://motion.me.ucsb.edu/book-lns

[11] M. C. V. Nascimento and A. C. P. L. F. de Carvalho, "Spectral methods for graph clustering—A survey," *Eur. J. Oper. Res.*, vol. 211, no. 2, pp. 221–231, Jun. 2011.

[12] S. Gao, I. W. Tsang, and L.-T. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 92–104, Jan. 2013.

[13] L. Kempton, G. Herrmann, and M. D. Bernardo, "Self-organization of weighted networks for optimal synchronizability," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 4, pp. 1541–1550, Dec. 2018.

[14] P. Niyogi, S. Smale, and S. Weinberger, "A topological view of unsupervised learning from noisy data," *SIAM J. Comput.*, vol. 40, no. 3, pp. 646–663, Jan. 2011.

[15] P. Srivastava, E. Nozari, J. Z. Kim, H. Ju, D. Zhou, C. Becker, F. Pasqualetti, G. J. Pappas, and D. S. Bassett, "Models of communication and control for brain networks: Distinctions, convergence, and future outlook," *Netw. Neurosci.*, vol. 4, no. 4, pp. 1122–1159, Jan. 2020.

[16] S. Mostafa, L. Tang, and F.-X. Wu, "Diagnosis of autism spectrum disorder based on eigenvalues of brain networks," *IEEE Access*, vol. 7, pp. 128474–128486, 2019.

[17] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf.*, Dec. 2011, pp. 2195–2201.

[18] G. Parlangeli and M. E. Valcher, "Accelerating consensus in high-order leader-follower networks," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 381–386, Jul. 2018.

[19] S. Y. Shafi, M. Arcak, and L. E. Ghaoui, "Graph weight allocation to meet Laplacian spectral constraints," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1872–1877, Jul. 2012.

[20] J. J. Molitierno, "The spectral radius of submatrices of Laplacian matrices for trees and its comparison to the Fiedler vector," *Linear Algebra Appl.*, vol. 406, pp. 253–271, Sep. 2005.

[21] W. Xia and M. Cao, "Analysis and applications of spectral properties of grounded Laplacian matrices for directed networks," *Automatica*, vol. 80, pp. 10–16, Jun. 2017.

[22] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proc. Amer. Control Conf. (ACC)*. IEEE, Jun. 2005, pp. 1859–1864.

[23] G. Parlangeli, "A supervisory algorithm against intermittent and temporary faults in consensus-based networks," *IEEE Access*, vol. 8, pp. 98775–98786, 2020.

[24] Y. Hao, Q. Wang, Z. Duan, and G. Chen, "Discernibility of topological variations for networked LTI systems," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 377–384, Jan. 2023.

[25] M. E. Valcher and G. Parlangeli, "On the effects of communication failures in a multi-agent consensus network," in *Proc. 23rd Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2019, pp. 709–720.

[26] M. A. Eissa, A. Sali, F. A. Ahmad, and R. R. Darwish, "Observer-based fault detection approach using fuzzy adaptive poles placement system with real-time implementation," *IEEE Access*, vol. 9, pp. 83272–83284, 2021.

[27] K. Kavinelavu and S. Kalaivani, "Pole placement-based sensor fault detection and isolation of a single phase PWM rectifier for electric railway traction," in *Proc. Int. Conf. Comput. Power, Energy, Inf. Commun. (ICCPEIC)*, Apr. 2014, pp. 194–199.

[28] M. T. Chu and G. H. Golub, *Inverse Eigenvalue Problems: Theory, Algorithms, and Applications*, vol. 13. London, U.K.: Oxford Univ. Press, 2005.

[29] R. Schmid, L. Ntogramatzidis, T. Nguyen, and A. Pandey, "A unified method for optimal arbitrary pole placement," *Automatica*, vol. 50, no. 8, pp. 2150–2154, Aug. 2014.

[30] V. Katewa and F. Pasqualetti, "Minimum-gain pole placement with sparse static feedback," *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3445–3459, Aug. 2021.

[31] S. Boyd, "Convex optimization of graph Laplacian eigenvalues," in *Proc. Int. Congr. Mathematicians*, vol. 3, nos. 1–3. Princeton, NJ, USA: Citeseer, 2006, pp. 1311–1319.

[32] S. Sardellitti, S. Barbarossa, and A. Swami, "Optimal topology control and power allocation for minimum energy consumption in consensus networks," *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 383–399, Jan. 2012.

[33] J. Hermann and U. Konigorski, "Eigenvalue assignment for the Laplacian matrix of directed graphs," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 4036–4042.

[34] H. Hao and P. Barooah, "Improving convergence rate of distributed consensus through asymmetric weights," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 787–792.

[35] A.-K. Schug, A. Eichler, and H. Werner, "A decentralized asymmetric weighting approach for improved convergence of multi-agent systems with undirected interaction," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 8317–8322, 2014.

[36] S. Dhuli and Y. N. Singh, "Analysis of average consensus algorithm for asymmetric regular networks," 2018, *arXiv:1806.03932*.
[37] G. Parlangeli, "A distributed algorithm for the assignment of the Laplacian spectrum for path graphs," *Mathematics*, vol. 11, no. 10, p. 2359, May 2023.
[38] J. S. Maybee, "Combinatorially symmetric matrices," *Linear Algebra Appl.*, vol. 8, no. 6, pp. 529–537, Dec. 1974.
[39] P. Barooah and J. P. Hespanha, "Graph effective resistance and distributed control: Spectral properties and applications," in *Proc. 45th IEEE Conf. Decis. Control*, Dec. 2006, pp. 3479–3485.
[40] M. Pirani and S. Sundaram, "On the smallest eigenvalue of grounded Laplacian matrices," *IEEE Trans. Autom. Control*, vol. 61, no. 2, pp. 509–514, Feb. 2016.
[41] Z. Liu and L. Guo, "Synchronization of multi-agent systems without connectivity assumptions," *Automatica*, vol. 45, no. 12, pp. 2744–2753, Dec. 2009.
[42] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2010.
[43] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks* (Applied Mathematics Series). Princeton, NJ, USA: Princeton Univ. Press, 2009.
[44] S. M. Fallat, H. T. Hall, R. H. Levene, S. A. Meyer, S. Nasserasr, P. Oblak, and H. Šmigoc, "Spectral arbitrariness for trees fails spectacularly," 2023, *arXiv:2301.11073*.
[45] J. A. Torres and S. Roy, "Graph-theoretic analysis of network input–output processes: Zero structure and its implications on remote feedback control," *Automatica*, vol. 61, pp. 73–79, Nov. 2015.
[46] S. Roy, J. A. Torres, and M. Xue, "Sensor and actuator placement for zero-shaping in dynamical networks," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 1745–1750.
[47] M. Xue and S. Roy, "Input-output properties of linearly-coupled dynamical systems: Interplay between local dynamics and network interactions," in *Proc. IEEE 56th Annu. Conf. Decis. Control (CDC)*, Dec. 2017, pp. 487–492.
[48] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, vol. 19. Hoboken, NJ, USA: Wiley, 2004.
[49] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 2567–2572.
[50] A. R. Swain and R. C. Hansdah, "A model for the classification and survey of clock synchronization protocols in WSNs," *Ad Hoc Netw.*, vol. 27, pp. 219–241, Apr. 2015.
[51] L. M. Oliveira and J. J. Rodrigues, "Wireless sensor networks: A survey on environmental monitoring," *J. Commun.*, vol. 6, no. 2, pp. 143–151, Apr. 2011.
[52] H. Farhangi, "The path of the smart grid," *IEEE Power Energy Mag.*, vol. 8, no. 1, pp. 18–28, Jan. 2010.
[53] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov. 2017.
[54] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, "Controllability of multi-agent systems from a graph-theoretic perspective," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 162–186, Jan. 2009.
[55] G. Parlangeli, "Laplacian eigenvalue allocation by asymmetric weight assignment for path graphs," in *Proc. 26th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2022, pp. 503–508.
[56] V. Kučera, "Diophantine equations in control—A survey," *Automatica*, vol. 29, no. 6, pp. 1361–1375, Nov. 1993.
[57] D. E. Knuth, *The Art of Computer Programming*, vol. 1. Reading, MA, USA: Addison-Wesley, 1973.
[58] A. Kobel, F. Rouillier, and M. Sagraloff, "Computing real roots of real polynomials... and now for real!" in *Proc. ACM Int. Symp. Symbolic Algebr. Comput.*, Jul. 2016, pp. 303–310.
[59] G. Parlangeli and G. Indiveri, "Single range observability for cooperative underactuated underwater vehicles," *Annu. Rev. Control*, vol. 40, pp. 129–141, Jan. 2015.

**GIANFRANCO PARLANGELI** (Member, IEEE) received the M.Sc. degree (Hons.) in electrical engineering from the University of Pisa, Pisa, Italy, in 1999, and the Ph.D. degree in information engineering from the University of Lecce, Lecce, Italy, in 2005. He is currently an Associate Professor with the Department of Innovation Engineering, University of Salento, Lecce. He has published over 60 papers in the field and has contributed to several national and international projects in the areas of robotics and industrial automation. His research interests include analysis and design of multi-agent systems and robotic networks, fault tolerant control, variable structure control systems, and robotics. He is a member of the Scientific Committee of the Interuniversity Center of Integrated Systems for the Marine Environment (ISME).

Open Access funding provided by 'Università del Salento' within the CRUI CARE Agreement