Seeing the bigger picture: How nodes can learn their place within a complex ad hoc network topology

Alexander Bertrand$^{*,\diamondsuit}$ and Marc Moonen$^{*,\diamondsuit}$

$*$ KU Leuven, Dept. Electrical Engineering ESAT, SCD-SISTA
$\diamondsuit$ iMinds-Future Health Department
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
E-mail: alexander.bertrand@esat.kuleuven.be
marc.moonen@esat.kuleuven.be
Phone: +32 16 321899, Fax: +32 16 321970

Abstract—**Distributed signal processing algorithms that are operated in complex ad hoc networks are usually 'topology unaware' (TU), as there is generally no a priori knowledge available about the actual network topology. With the aim of developing improved 'topology aware' (TA) algorithms, we explain how nodes can infer information about the topology or topology-related properties of the network based on in-network distributed learning, i.e., without relying on an 'external observer' who has a complete overview over the network. To this end, we review key concepts and basic techniques from the established field of spectral graph theory (SGT), with an emphasis on those concepts that allow for a simple and efficient distributed implementation. In particular, we focus on eigenvector or Katz centrality, algebraic connectivity, and the Fiedler vector. From these concepts, densely-connected node clusters and their sparse cross connections can be identified, as well as the most central and/or important nodes (either on the network- or the cluster-level). We also highlight how this knowledge can be exploited in several distributed signal processing tasks, e.g., for distributed estimation (including consensus-, diffusion-, and gossip-type algorithms), base station or cluster head selection, topology selection, resource allocation, node subset selection, etc.**

I. MOTIVATION

In the past decade, distributed signal processing has become an established research field, mainly due to the increased interest in wireless sensor networks (WSNs) and their applications in, a.o., environmental monitoring, surveillance, multi-robot coordination, etc. This has lead to the development of various distributed algorithms for, e.g., estimation, fusion, compression, detection, data routing, etc. These distributed algorithms are often designed to operate in networks with a complex ad hoc topology, which is a priori unknown and possibly even time-varying [1]–[7]. Due to this ad hoc aspect, the algorithms are usually 'topology-unaware' (TU), i.e., they do not explicitly take any specific topology-related properties of the network into account. The nodes or agents in the network indeed often have no clue about their place, role and/or impact within the overall network. It is a highly non-trivial task for the nodes to infer such information, since each node has a very limited horizon, i.e., it cannot see beyond its neighbors.

On the other hand, if specific (low- or high-level) knowledge on the network topology would be available, this could be exploited to make certain topology-related decisions and/or to improve the performance of distributed algorithms that are operated in the network. Indeed, there is a vast amount of literature demonstrating that the actual network topology significantly impacts the performance of distributed algorithms [1]–[9]. For example, the spectral properties of common topology-related matrices defined in spectral graph theory (SGT) are known to directly influence the robustness and convergence speed of several distributed estimation algorithms, including the popular consensus-, diffusion- and gossip-type algorithms [1]–[3], [6]–[8]. Furthermore, SGT provides several graph analysis techniques that are able to reveal some structure in complex ad hoc network topologies [10]–[22]. However, since the network topology is a property of the entire network as a whole, such a network graph analysis requires an 'external observer', who has a complete overview over the network. Naturally, this requirement is in conflict with the distributed learning paradigm where the nodes themselves can perform the required task by local cooperation. The lack of an 'external observer' then usually hampers the use of distributed 'topology-aware' (TA) algorithms in practice.

However, the common belief that topology-related analysis or design is limited to a centralized and off-line context

The work of A. Bertrand was supported by a Postdoctoral Fellowship of the Research Foundation - Flanders (FWO). This work was carried out at the ESAT Laboratory of KU Leuven, in the frame of KU Leuven Research Council CoE EF/05/006 'Optimization in Engineering' (OPTEC) and PFV/10/002 (OPTEC), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P7/23 (BESTCOM, 'Belgian network on stochastic modeling analysis design and optimization of communication systems' 2012-2017), and Research Project FWO nr. G.0763.12 'Wireless acoustic sensor networks for extended auditory communication'. The scientific responsibility is assumed by its authors.

(either a priori or a posteriori), is not necessarily true [21]–[31]. Indeed, several important matrices defined in SGT, such as the adjacency and Laplacian matrix, are encoded in the network itself, which allows to perform elementary operations with these matrices in a distributed fashion. In this paper, we review key concepts and basic techniques from the field of SGT based on which the 'insiders', i.e., the nodes themselves, can *learn* important high-level information about the network topology or its topology-related properties, without relying on an 'external observer'. One of these concepts is eigenvector centrality, which forms the basis of the celebrated Google Pagerank algorithm [15]. Eigenvector centrality allows to identify central nodes, but also to assign a measure of the network-wide influence of each node. Another very important concept from SGT is the algebraic connectivity and the associated Fiedler vector [11]–[13]. The latter is able to identify densely-connected node clusters that only have a few cross links to other clusters, which also reveals the weak points in the network. We also highlight the potential applicability of these techniques in several distributed signal processing tasks such as distributed estimation (including consensus-, diffusion-, or gossip-type algorithms), base station or cluster head selection, topology selection, resource allocation, node subset selection, etc.

With this tutorial paper, we aim to give a flavor of SGT-based topology inference by introducing easy-to-understand key concepts and basic techniques, and by providing some insight into the application span of these techniques in a distributed signal processing context. Therefore, a variety of (sometimes unrelated) topics and techniques from the fields of SGT and distributed signal processing are touched upon, with an emphasis on conceptual ideas rather than on detailed descriptions or analysis. The included ideas and examples demonstrate how SGT can be a catalyst in the development of TA distributed signal processing algorithms.

The outline of this paper is as follows. In Section II, we describe some basic definitions and notation that will be used throughout this paper. In Section III, we review eigenvector centrality and its generalization to Katz centrality. In Section IV, we review the algebraic connectivity and the Fiedler vector with a focus on their clustering properties. In Section V, we give examples of how these concepts can be exploited in distributed signal processing tasks. Conclusions are drawn in Section VI.

II. Basic definitions, terminology and notation

In accordance with the WSN literature, the vertices of a network graph, i.e., the interacting agents in a network, are referred to as 'nodes', and the edges of the graph are referred to as 'links'. We consider a connected ad hoc network where the set of nodes is denoted by $\mathcal{K}$ (containing $|\mathcal{K}| = K$ elements) and the set of links is denoted by $\mathcal{L}$. We denote $\mathcal{N}_k$ as the set of neighbors of node k, i.e., the nodes that are linked to node k (node k excluded), and $|\mathcal{N}_k|$ is referred to as the *degree* of node k, i.e., the number of neighbors of node k. We define $\mathbf{I}$ as the identity matrix, and $\mathbf{1}$ as a vector with all entries equal to one (dimensions should be clear from the context). We use the notation $\rho(\mathbf{X})$ to denote the spectral radius of the matrix $\mathbf{X}$, i.e., its largest eigenvalue in absolute value.

Two commonly used matrices in SGT are the adjacency matrix and the Laplacian matrix. The entries of the *adjacency matrix* $\mathbf{A} = [a_{kq}]_{K\times K}$ are defined as

$$a_{kq} = a_{qk} = \begin{cases} 1 & \text{if } q \in \mathcal{N}_k \\ 0 & \text{otherwise} . \end{cases} \tag{1}$$

This matrix can be generalized to the case of weighted and/or directed network graphs where a non-negative weight and/or a direction is associated to each link. In a wireless communication network, these weights may correspond to, e.g., the available bandwidth over the links, the received signal strengths, the reliability of the links, and so on. We define the *weighted adjacency matrix* $\mathbf{W} = [w_{kq}]_{K\times K}$, with w_{kq} denoting the weight on the link (k, q) going from node k to q (by definition, $w_{kq} = 0$ if this link does not exist). We assume that all link weights are non-negative, i.e., $w_{kq} \geq 0$, $\forall k, q \in \mathcal{K}$. In the case of directed network graphs, it is possible that $w_{kq} \neq w_{qk}$, i.e., in contrast to $\mathbf{A}$, the weighted adjacency matrix $\mathbf{W}$ may be asymmetric.

The entries of the *Laplacian matrix* $\mathbf{L} = [l_{kq}]_{K\times K}$ of an undirected graph ($w_{kq} = w_{qk}$) are defined as

$$l_{kq} = l_{qk} = \begin{cases} \sum_{j\in\mathcal{K}} w_{kj} & \text{if } k = q \\ -w_{kq} & \text{if } q \in \mathcal{N}_k \\ 0 & \text{otherwise} . \end{cases} \tag{2}$$

For an unweighted network graph, $\mathbf{L}$ has the same definition where the weight for link (k, q) is set to $w_{kq} = 1$ such that $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D} = \text{diag}(|\mathcal{N}_1|, \ldots, |\mathcal{N}_K|)$. We do not define the (asymmetric) Laplacian matrix for the case of directed network graphs, since this exceeds the purpose of this paper, i.e., we only use the Laplacian matrix in a context of clustering, where the direction of the edges is typically ignored.

The Laplacian matrix $\mathbf{L}$ is always positive semidefinite, and by definition, it has a zero eigenvalue $\lambda_1 = 0$ (eigenvalues are sorted in increasing order of magnitude) with corresponding eigenvector $\frac{1}{\sqrt{K}}\mathbf{1}$. In fact, it can be easily shown that $\mathbf{L}$ has P zero eigenvalues, where P is the number of disconnected subgraphs [13]. Since we only consider connected network graphs ($P = 1$), the eigenvalues of $\mathbf{L}$ satisfy

$$0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_K . \tag{3}$$

Throughout this paper, there is an implicit assumption that the distributed algorithms operate in a synchronous setting. This means that there is a common network-wide iteration index that is incremented deterministically at regular time intervals. In each iteration, the nodes perform a pre-defined task and share the result with their neighbors. This is very different from asynchronous environments, where nodes are allowed to process and transmit data at will (at any time). The latter is less restrictive in terms of network-wide coordination, but the algorithm design is considered to be much more challenging.

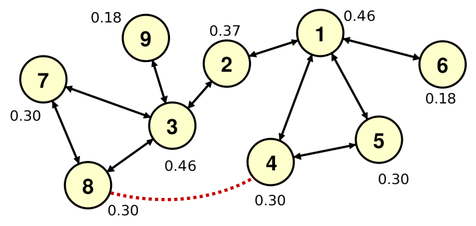


Fig. 1. A small example graph with eigenvector centrality scores (the red dotted line can be ignored, unless stated otherwise).

III. Eigenvector centrality

A. Definition

For the design of TA distributed algorithms, it is often important to infer information about the 'centrality' of each node in the network topology. In graph theory, many different centrality measures are defined, where centrality may have a different meaning in different contexts or measures [10]. For example, in the network depicted in Fig. 1 (ignore the red dotted line for the time being), node 2 could be considered to be the most central node. This intuition relates to the notion of *closeness centrality* [10], i.e., a central node should be close to *any* of the other nodes in the network. In graph theory however, centrality is more often associated with the relative 'influence' of a node with respect to the rest of the network, where nodes with many neighbors are considered to be more influential. In this context, a network may have multiple 'central nodes' at different places in the network, usually at the centers of densily-connected node clusters (see also Section IV). For example, nodes 1 and 3 can be considered to be most influential in the network graph depicted in Fig. 1.

A simple centrality measure in this context is the so-called *degree centrality*, where the centrality of node k is defined by its degree, i.e., $|\mathcal{N}_k|$. However, as it merely measures the local influence of a node, degree centrality does not take the full network topology into account, which is a major disadvantage. The so-called *eigenvector centrality* fixes this problem, relying on the principle that a node has more influence if it is connected to many nodes which in turn are also considered to be influential. If we denote c_k as the centrality of node k, then this principle yields the following implicit definition:

$$\boxed{c_k = \frac{1}{\alpha}\sum_{q\in\mathcal{N}_k} c_q \ , \quad \forall\, k \in \mathcal{K}} \tag{4}$$

where α is an arbitrary normalization factor. We obtain a chicken-and-egg problem where the centrality of node k is proportional to the sum of the centralities of its neighbors. Notice that (4) incorporates knowledge of the full network topology where each node should look beyond its local horizon, since a node's centrality does not only depend on its neighbors, but also implicitely on the neighbors of its neighbors, etc. To solve this chicken-and-egg problem, we rewrite (4) by means of the adjacency matrix $\mathbf{A}$, i.e.,

$$\alpha\, \mathbf{c} = \mathbf{A}\cdot\mathbf{c} \tag{5}$$

where $\mathbf{c}$ is the K-dimensional vector in which all c_k, $\forall\, k \in \mathcal{K}$, are stacked. This shows that $\mathbf{c}$ has to be an eigenvector of $\mathbf{A}$, which also explains the name 'eigenvector centrality'. If we only allow non-negative coefficients in $\mathbf{c}$, then equation (5) has a unique solution, given by the principal eigenvector of $\mathbf{A}$ corresponding to its largest eigenvalue α_K (this can be straightforwardly proven by means of the Perron-Frobenius theorem). It is noted that this defines $\mathbf{c}$ up to a non-zero scaling, and it depends on the application whether and how $\mathbf{c}$ has to be normalized. The definition of $\mathbf{c}$ can be generalized to the case of weighted and/or directed graphs by replacing $\mathbf{A}$ in (5) with $\mathbf{W}$ or $\mathbf{W}^T$, using the weights of the outgoing or incoming links, respectively.

As an example, we have indicated the eigenvector centralities of the different nodes in the example graph depicted in Fig. 1. Nodes 1 and 3 have the highest centrality score, which corresponds to our intuition that these nodes have a large influence. We reiterate that this does not always correspond to the geometrical interpretation of centrality, such as in, e.g., closeness centrality which would rank node 2 higher than nodes 1 and 3. In Section IV-B, we will point out that the Laplacian spectrum often relates better to this geometrical interpretation of centrality than the spectrum of the adjacency matrix.

B. Properties and extensions

Eigenvector centrality is a powerful concept to quantify the topology-induced influence of a node. To give more insight in the actual meaning of eigenvector centrality, we provide another interpretation which also allows for a more tunable generalization. Intuitively, if there are many possible 'walks' through the network that pass through a particular node, this node can be assumed to have an important network-wide contribution (e.g., in terms of information flow, diffusion properties, etc.), and therefore this node should receive a high centrality score. First note that the entry at row k and column q of the matrix $\mathbf{A}^N$ is equal to the number of walks of length N from node k to node q. Therefore, the k-th entry in the vector $\mathbf{A}^N\cdot\mathbf{1}$ is equal to the total number of walks[1] of length N that start at node k. Define the vector

$$\boxed{\mathbf{v}(\phi) = \left(\mathbf{A} + \tfrac{1}{\phi}\mathbf{A}^2 + \tfrac{1}{\phi^2}\mathbf{A}^3 + ...\right)\cdot\mathbf{1}} \tag{6}$$

where $\phi > \alpha_K$ (otherwise the sum diverges). Then the k-th entry of $\mathbf{v}(\phi)$ is equal to the total number of walks that start at node k, where a walk of length N is penalized with a factor $\frac{1}{\phi^{N-1}}$. By using the Taylor series expansion $(\mathbf{I}-\mathbf{X})^{-1} = \sum_{i=0}^{\infty}\mathbf{X}^i$ for $\rho(\mathbf{X}) < 1$ we can rewrite (6) as

$$\mathbf{v}(\phi) = \left(\mathbf{I} - \frac{1}{\phi}\mathbf{A}\right)^{-1}\cdot\mathbf{A}\cdot\mathbf{1} \tag{7}$$

and therefore

$$\mathbf{v}(\phi) = \frac{1}{\phi}\mathbf{A}\cdot\mathbf{v}(\phi) + \mathbf{A}\cdot\mathbf{1}\ . \tag{8}$$

If ϕ approaches α_K from above ($\phi \underset{>}{\rightarrow} \alpha_K$), then $\|\mathbf{v}(\phi)\| \rightarrow \infty$ in (7), in which case the term $\mathbf{A}\cdot\mathbf{1}$ in (8) can be neglected. Therefore $\lim_{\phi \underset{>}{\rightarrow} \alpha_K}\mathbf{v}(\phi) \propto \mathbf{c}$ where '$\propto$' denotes equality up to a non-zero scaling, i.e., eigenvector centrality corresponds

[1]It is noted that similar interpretations exist for weighted and/or directed graphs, e.g., in a context of random walks [15], [21], [22], [32].

to a limit case of (6), where the penalization of longer walks is minimal [33]. To control to which degree neighbors of neighbors of neighbors (etc.) should have an influence on a node's centrality measure, one can also choose a larger penalization factor ϕ in (6). This generalized centrality measure is often referred to as Katz centrality [34], and it forms the basis of, i.a., the successful Pagerank algorithm used by the Google search engine, in the context of a 'random surfer' model [15]. In this case, the penalizing factor ϕ controls how easily the surfer gets bored after clicking too many links. In Section V-E, we will explain how to use similar techniques in TA distributed algorithms to obtain an 'eigenutility' of nodes, based on their topology-induced influence.

C. In-network computation

To facilitate the design of TA distributed algorithms, we should be able to perform an in-network distributed computation of the centrality vector $\mathbf{c}$. An obvious but important observation is that the (weighted) adjacency matrix is implicitely coded inside the network itself, allowing to perform in-network matrix-vector multiplications with $\mathbf{A}$ or $\mathbf{W}$ in a simple distributed fashion. Indeed, assume we wish to compute the matrix-vector product $\mathbf{y} = \mathbf{A} \cdot \mathbf{x}$, and assume that node k stores the k-th entry of $\mathbf{x}$ (denoted by x_k) and has access to the x_q's of its neighbors ($q \in \mathcal{N}_k$), then the k-th entry of $\mathbf{y}$ can be computed at node k as

$$y_k = \sum_{q=1}^{K} a_{kq} x_q = \sum_{q \in \mathcal{N}_k} x_q \ . \tag{9}$$

Therefore, we can perform an in-network power iteration (PI)

$$\mathbf{x}^{(i+1)} = \mathbf{A} \cdot \mathbf{x}^{(i)} \tag{10}$$

starting from a random vector $\mathbf{x}^{(0)}$. From matrix theory, we know that this PI converges to the principal eigenvector of $\mathbf{A}$, i.e., $\mathbf{x}^{(\infty)} \propto \mathbf{c}$. However, the norm of $\mathbf{x}^{(i)}$ in (10) will diverge or vanish, depending on $\rho(\mathbf{A}) > 1$ or $\rho(\mathbf{A}) < 1$, respectively. Therefore, even though we usually do not require a normalized centrality vector[2], we should still compensate for this growth or shrinking effect. To this end, we apply a compensation factor that continuously approximates the growth/shrinking rate, i.e.,

$$\mathbf{x}^{(i+1)} = \frac{1}{r^{(i)}} \overline{\mathbf{x}}^{(i+1)} = \frac{1}{r^{(i)}} \mathbf{A} \cdot \mathbf{x}^{(i)} \tag{11}$$

such that $r^{(i)} \approx \frac{\|\mathbf{A}\mathbf{x}^{(i)}\|}{\|\mathbf{x}^{(i)}\|} = \frac{\|\overline{\mathbf{x}}^{(i+1)}\|}{\|\mathbf{x}^{(i)}\|}$ (and eventually $\lim_{i\to\infty} r^{(i)} = \lim_{i\to\infty} \frac{\overline{x}_k^{(i+1)}}{x_k^{(i)}} = \rho(\mathbf{A}) = \alpha_K$). The estimation of $r^{(i)}$ can be taken care of by a distributed algorithm that runs in parallel with (10). For example, $r^{(i)}$ is estimated heuristically from the quantities $\frac{\overline{x}_k^{(i+1)}}{x_k^{(i)}}$ by means of a gossip algorithm in [35], and by means of diffusion adaptation in [23]. Note that $r^{(i)}$ does not influence the convergence speed of the PI (11), hence there is no crucial or strict dependency between the PI and the algorithm that estimates $r^{(i)}$ (one can lag behind with respect to the other).

[2]In case a normalized centrality vector is desired, one can use techniques from [1], [6], [7] to compute the norm $\|\mathbf{x}^{(\infty)}\|$ at each node, as done in [35].

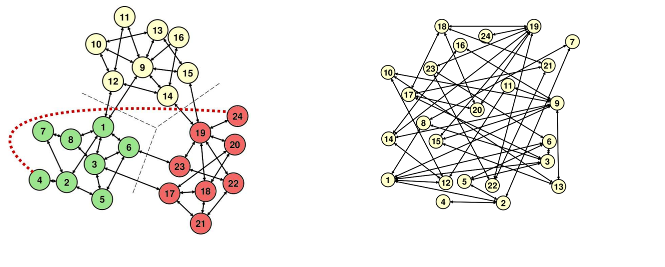

(a) Ordered node placement (b) Random node placement

Fig. 2. Two different visualizations of an ad hoc network consisting of $K = 24$ nodes (the red dotted line can be ignored, unless stated otherwise).

Finally, it is noted that the Katz centrality vector $\mathbf{v}(\phi)$ can be computed with a similar iterative procedure, based on (8), i.e.,

$$\mathbf{x}^{(i+1)} = \frac{1}{\phi} \mathbf{A} \cdot \mathbf{x}^{(i)} + \mathbf{A} \cdot \mathbf{1} \ . \tag{12}$$

and by including a proper compensation for the growth or shrinking rate, by using similar techniques as mentioned above.

IV. Algebraic connectivity and the Fiedler vector

A. Definition

The algebraic connectivity and the Fiedler vector constitute one of the most important concepts in SGT, and they are extracted from the spectrum of the Laplacian matrix $\mathbf{L}$. Recall that the smallest Laplacian eigenvalue is equal to $\lambda_1 = 0$ with corresponding eigenvector $\frac{1}{\sqrt{K}}\mathbf{1}$. The algebraic connectivity is defined as the second-smallest Laplacian eigenvalue λ_2, which is proved to be strictly positive (assuming the network graph is connected) and monotonically increasing when adding links to $\mathcal{L}$. Its corresponding eigenvector, denoted as $\mathbf{f}$, is referred to as the Fiedler (eigen)vector, after Miroslav Fiedler who developed the original theory related to algebraic connectivity [11]. The algebraic connectivity λ_2 and the Fiedler vector $\mathbf{f}$ contain important information about the connectivity and clustering properties of the network graph, as well as the convergence and robustness properties of distributed algorithms such as consensus- [1], [6], diffusion- [2], [3], or gossip-based [7] algorithms (see also Section V-B).

B. Properties

Although the algebraic connectivity and the Fiedler vector have many important properties [13], we will mainly focus on their clustering capabilities, as these yield important information about the network topology, which can be exploited in TA distributed algorithms. To this end, we consider a network in which we would like to identify node clusters, which are internally densely connected, but which have only few cross links with other node clusters. For example, three node clusters can be clearly distinguished in the network depicted in Fig. 2(a), as indicated by their color (in the sequel, the red dotted line can be ignored, unless stated otherwise). The links cut by the black dashed lines form important 'bridges' between the clusters, which usually correspond to bottlenecks in the data diffusion or dissemination in the network. Even though it is intuitively clear how to cluster the network in Fig. 2(a), the node clustering problem becomes a lot harder for the network depicted in Fig. 2(b), even though both networks are actually identical. With this observation, it is not surprising that graph partitioning (node clustering) techniques also play a crucial role in graph visualization software [17].

1) Ratio cut: If we aim to partition a network graph with a set of nodes $\mathcal{K}$ in two non-overlapping node clusters $\mathcal{K}_1$ and $\mathcal{K}_2$, a commonly used approach, leading to the so-called *ratio cut*, is to minimize the edge density [12]

$$\rho(\mathcal{K}_1, \mathcal{K}_2) = \frac{|\mathcal{L}_{(\mathcal{K}_1,\mathcal{K}_2)}|}{|\mathcal{K}_1||\mathcal{K}_2|} \tag{13}$$

where $\mathcal{K}_1 \cap \mathcal{K}_2 = \emptyset$, $\mathcal{K}_1 \cup \mathcal{K}_2 = \mathcal{K}$ and $\mathcal{L}_{(\mathcal{K}_1,\mathcal{K}_2)}$ denotes the set of links that are shared between $\mathcal{K}_1$ and $\mathcal{K}_2$, i.e., the links that are cut. The intuition behind the ratio cut is that the minimization of the numerator of (13) minimizes the number of links between the two node clusters, while the maximization of the denominator yields a driving force towards node clusters of equal size (avoiding trivial solutions). It is noted that the edge density can be generalized for weighted graphs, where $|\mathcal{L}_{(\mathcal{K}_1,\mathcal{K}_2)}|$ is then replaced by the sum of the weights of the links that are cut, i.e.,

$$\rho(\mathcal{K}_1, \mathcal{K}_2) = \frac{\sum_{k\in\mathcal{K}_1, q\in\mathcal{K}_2} w_{kq}}{|\mathcal{K}_1||\mathcal{K}_2|} \ . \tag{14}$$

Finding the ratio cut of a network graph is not straightforward, let alone, to do this in a distributed fashion where each node only sees a small part of the network. Moreover, the problem is shown to be NP-complete [16], and so one has to rely on heuristics to solve it in a reasonable time (this holds in a centralized scenario as well as in a distributed scenario). This is where the algebraic connectivity λ_2 and the Fiedler vector $\mathbf{f}$ come into play.

2) Algebraic connectivity and the ratio cut: One important property of the algebraic connectivity is that it provides a lower bound on the edge density of the ratio cut of the network graph [12], i.e.,

$$\boxed{\min_{\mathcal{K}_1,\mathcal{K}_2} \rho(\mathcal{K}_1, \mathcal{K}_2) \geq \tfrac{\lambda_2}{K} \ .} \tag{15}$$

Basically, this means that a network graph with a large algebraic connectivity requires the removal of relatively many links to partition it into two parts of similar size, i.e., a clear bipartition of the network graph is not inherent. On the other hand, a network graph with a small algebraic connectivity is vulnerable in the sense that two large node clusters are connected by relatively few links. As explained next, the Fiedler vector can then be of great help to identify these links.

3) Fiedler vector-based clustering: The coefficients of the Fiedler vector contain a powerful heuristic to approximate the ratio cut to partition the network graph into two well-separated node clusters [12], [14], [16], [18], which may indeed prove useful in the design of TA distributed algorithms. To see this, observe that the (normalized[3]) Fiedler vector is the solution of

$$\arg\min_{\mathbf{f}} \mathbf{f}^T\mathbf{L}\mathbf{f} \tag{16}$$

$$\text{s.t. } \|\mathbf{f}\| = 1 \tag{17}$$

$$\mathbf{f}^T\mathbf{1} = 0 \tag{18}$$

and it can be easily verified that

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{(k,q)\in\mathcal{L}} w_{kq}(f_k - f_q)^2 \ . \tag{19}$$

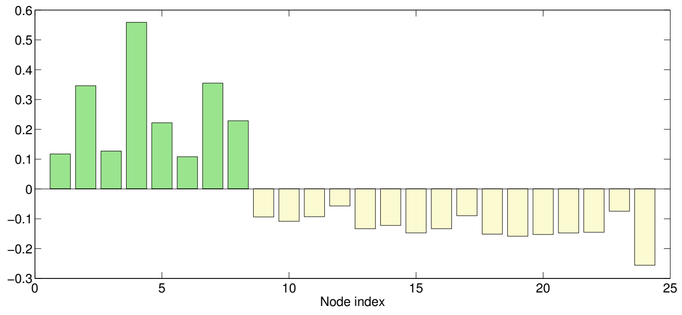
where f_k denotes the k-th entry of $\mathbf{f}$. Minimizing (19) yields a driving force such that, if node k and q are connected (with a large weight w_{kq}), the values f_k and f_q are close to each other. This means that the nodes in densely connected node clusters will have similar entries in $\mathbf{f}$. Since $\|\mathbf{f}\| = 1$ and $\mathbf{f}^T\mathbf{1} = 0$, there is also a driving force such that the entries in $\mathbf{f}$ are centered around zero and not all equal to each other, such that trivial clustering solutions are avoided. Therefore, sorting the nodes with respect to their corresponding entries in $\mathbf{f}$ provides a good heuristic to partition the network graph into two node clusters based on the ratio cut. In most cases, the distinction between positive entries and negative entries in $\mathbf{f}$ is used to form the two node clusters, although other approaches also exist (see, e.g., [14], [36]). If a network graph has to be clustered into more than two node clusters, this technique can be applied recursively [16], [18], [37], which is often referred to as Recursive Spectral Bisection (RSB). In this case, the value of λ_2 (on the cluster level) can be used as an indicator to decide whether or not a specific cluster can be easily partitioned into two smaller clusters, which is motivated by the lower bound in (15). A possible RSB-based algorithm to cluster a network graph in Q node clusters is presented in Table I. It is noted that (15) can also be used to define a stop criterion if Q is not known a priori. The RSB-based algorithm described in Table I is a top-down clustering algorithm, starting from the complete network graph and subsequently dividing it into smaller clusters. It is also possible to start from a given node and to identify the local 'natural cluster' that contains this node by using related Fiedler-based techniques [20]. These techniques can then be used in a bottom-up approach where a couple of smaller clusters are identified first, which can then be merged together with heuristic procedures [19].

As a brief illustration, consider the example network graph depicted in Fig. 2(a) with the entries of the corresponding Fiedler vector given in Fig. 3(a). Notice that the nodes in $\mathcal{K}_1 = \{1, \ldots, 8\}$ correspond to positive entries, whereas the other nodes correspond to negative entries. This indicates that $\mathcal{K}_1$ can be considered as a densely-connected node cluster, having only a small number of links to nodes in $\mathcal{K}\backslash\mathcal{K}_1$. A more subtle observation is that, within $\mathcal{K}_1$, nodes 1, 3 and 6 have entries that are significantly closer to zero than the entries of the others. This is because these nodes contain the bridge links

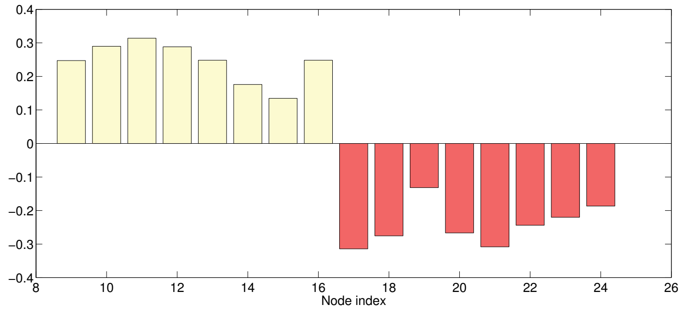
[3]It is noted that the unity norm constraint in (17) is an arbitrary choice, merely to avoid a trivial solution. The actual value of the norm of $\mathbf{f}$ is usually not important in practice.

TABLE I
DESCRIPTION OF A RSB NODE CLUSTERING ALGORITHM TO IDENTIFY Q CLUSTERS

1) Let $\mathcal{C}$ denote the set of clusters and initialize $\mathcal{C} \leftarrow \{\mathcal{G}_1\}$ where $\mathcal{G}_1 = \mathcal{K}$.
2) $\forall\, \mathcal{G}_i \in \mathcal{C}$, $i = 1 \ldots |\mathcal{C}|$, compute the cluster-level Fiedler vector $\mathbf{f}(\mathcal{G}_i)$ and algebraic connectivity $\lambda_2(\mathcal{G}_i)$, where bridge links between clusters are ignored.
3) Find $\mathcal{G}^* \leftarrow \arg\min_{\mathcal{G}\in\mathcal{C}} \frac{\lambda_2(\mathcal{G})}{|\mathcal{G}|}$.
4) Partition $\mathcal{G}^*$ into two clusters $\mathcal{G}^*_-$ and $\mathcal{G}^*_+$, where $\mathcal{G}^*_-$ contains the nodes which have negative entries in $\mathbf{f}(\mathcal{G}^*)$, and where $\mathcal{G}^*_+$ contains the nodes which have positive entries in $\mathbf{f}(\mathcal{G}^*)$.
5) Set $\mathcal{C} \leftarrow \{\mathcal{G}^*_-, \mathcal{G}^*_+\} \cup \mathcal{C}\backslash\{\mathcal{G}^*\}$.
6) If $|\mathcal{C}| < Q$, return to step 2.



(a) Entries of the Fiedler vector of the full network



(b) Entries of the Fiedler vector of the isolated cluster $\{9, 10, \ldots, 24\}$

Fig. 3. Fiedler vectors computed in the network depicted in Fig. 2(a).

between $\mathcal{K}_1$ and $\mathcal{K}\backslash\mathcal{K}_1$. Similarly, within $\mathcal{K}\backslash\mathcal{K}_1$, nodes 9, 12, 17 and 23 have the entries that are closest to zero. Finally, the larger node cluster $\mathcal{K}\backslash\mathcal{K}_1$ can further be partitioned into two node clusters by computing the Fiedler vector of the subgraph corresponding to this node cluster (ignoring the bridge links with $\mathcal{K}_1$). The entries of this vector are given in Fig. 3(b). Again, it is observed that the positive entries correspond to one node cluster, and the negative entries to the second node cluster. This finally results in the three node clusters indicated by the different colors in Fig. 2(a).

4) Fiedler vector-based centrality: In addition to their usage in node clustering, the entries in the Fiedler vector may also be used as a measure of centrality of the nodes, this time in a more geometrical context of centrality (similar to closeness centrality). As illustrated in the previous example, the entries that are close to zero correspond to nodes that connect the two node clusters, since assigning small values to such nodes indeed yields a smaller value in (19). These nodes have short paths to the nodes in *both* clusters, and therefore they are usually close to most other nodes in the network. We will not further elaborate on this Fiedler vector-based centrality, as it is beyond the scope of this paper. However, it is noted that it could be incorporated in many of the techniques explained in Section V (including base station and topology selection) instead of eigenvector centrality. This would favor solutions with small network delay, rather than the typical high-throughput solutions in the case of eigenvector centrality.

C. In-network computation

Similar to the adjacency matrix, it is observed that the Laplacian matrix is implicitely coded inside the network itself, again allowing to perform an in-network PI with $\mathbf{L}$. However, as we are now interested in the second-smallest eigenvalue of $\mathbf{L}$, some modifications are required to compute $\mathbf{f}$ and λ_2 [23].

Consider the matrix

$$\mathbf{G} = \mathbf{I} - \sigma\mathbf{L} \tag{20}$$

with $0 \leq \sigma \leq \frac{1}{\lambda_K}$ such that $\mathbf{G}$ is positive (semi-)definite. To guarantee the latter, σ can be safely set to $\sigma = \left(\overline{\lambda}\right)^{-1}$, based on the upper bound [23]

$$\lambda_K \leq \overline{\lambda} = \max_{k\in\mathcal{K}} \left(S_k + \frac{1}{S_k} \sum_{q\in\mathcal{N}_k} S_q \right) \tag{21}$$

where $S_k = \sum_{q\in\mathcal{N}_k} w_{kq}$. It is noted that $\mathbf{G}$ is a doubly stochastic matrix since the entries in any row and column of $\mathbf{L}$ sum to zero. We also define the matrix

$$\mathbf{V} = \mathbf{G}^N \mathbf{L} \tag{22}$$

which has the same eigenvectors as $\mathbf{L}$ and $\mathbf{G}$, and of which the eigenvalues are given by

$$\nu_j = \lambda_j (1 - \sigma\lambda_j)^N \ , \quad j = 1, \ldots, K \ . \tag{23}$$

The aim is now to perform an in-network PI with the matrix $\mathbf{V}$, similar to (11). To make $\mathbf{f}$ the dominant eigenvector of $\mathbf{V}$, the eigenvalues in (23) should satisfy $\nu_2 > \nu_j$, $\forall\, j \neq 2$. Since $\nu_1 = 0$, $\forall\, N \in \mathbb{N}$, it is easy to see that this will indeed be the case for sufficiently large N (see [23] for sufficient conditions on N), and therefore we can indeed use the PI

$$\mathbf{x}^{(i+1)} = \mathbf{V} \cdot \mathbf{x}^{(i)} = \mathbf{G}^N \cdot \mathbf{L} \cdot \mathbf{x}^{(i)} \tag{24}$$

to compute $\mathbf{f}$. This PI can be easily computed in a distributed

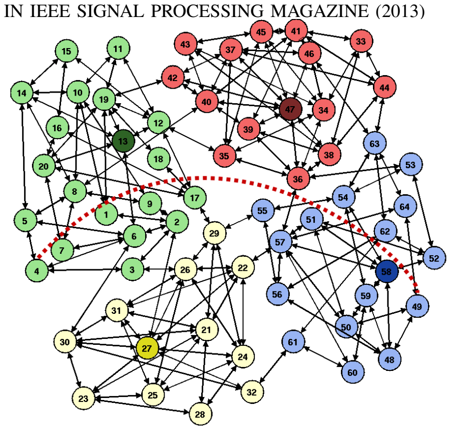


Fig. 4. A large example graph with four densely-connected node clusters and corresponding cluster heads as selected by the algorithm in Table II (the red dotted line can be ignored, unless stated otherwise).

fashion, by alternating between a single in-network multiplication with $\mathbf{L}$, and N subsequent in-network multiplications with $\mathbf{G}$ (and by including a proper compensation for the growth or shrinking rate, as in Section III-C). After convergence, the k-th entry f_k is known to node k, as well as λ_2, since the latter can be computed from the limiting growth/shrinking rate.

It is noted that there exist alternative algorithms to compute $\mathbf{f}$ and/or λ_2. E.g., in [24], a continuous-time algorithm is proposed based on a combination of a PI and consensus averaging (CA) techniques. In [25], an improved convergence is targeted based on a combination of an inverse PI and CA techniques. If only an estimate of λ_2 is of interest, we refer to [22], [26]. The algorithms in [27], [28] do not focus on the Fiedler vector in particular, but are able to compute a set of dominant eigenvectors of $\mathbf{L}$ or $\mathbf{G}$ in a distributed fashion.

V. Application in Distributed Signal Tasks

In this section, we present a number of distributed signal processing tasks that can benefit from a TA approach based on the information that is included in the centrality vector $\mathbf{c}$, the Fiedler vector $\mathbf{f}$ and the algebraic connectivity λ_2. In particular, we consider base station or cluster head selection, distributed estimation (including consensus- and diffusion-type algorithms), topology selection, and eigenutility. We also briefly address resource allocation and node subset selection.

It is noted that the entries in $\mathbf{c}$ and $\mathbf{f}$ are distributed over the different nodes, which may require some additional measures to make global decisions based on the entries in these vectors. In many cases (including the examples in this section), the decision making process merely involves a maximum and/or minimum search over the entries in $\mathbf{c}$ and $\mathbf{f}$, which can be easily performed by means of gossip or max-/min-consensus techniques [38]. If node clustering is involved, one can rely on the signs of the entries in $\mathbf{f}$, hence nodes automatically know which cluster they belong to. We will not elaborate further on more sofisticated decision making processes, as this is an application-specific issue which mostly relies on heuristics or divide and conquer strategies (see also, e.g., [19]).

A. Base station or cluster head selection ($\mathbf{c}$ / $\mathbf{f}$)

Base station or cluster head selection amounts to choosing one or more nodes in a network to serve as a data collector/distributor for a large part of the network. This is a highly non-trivial task, especially so in a distributed setting [37], [39]. Ideally, these base stations should be easily reachable (central) nodes and well-separated from each other. For example, in Fig. 2(a), an 'external observer' will usually select nodes 3, 9 and 18 as base stations. In a distributed setting, the centrality vector $\mathbf{c}$ will provide a good heuristic to make this choice, since a node with a large centrality has many paths or walks going through it, and therefore it is able to quickly collect/distribute a large amount of information from/to many different nodes. However, if the network can be easily partitioned in multiple node clusters (if $\frac{\lambda_2}{K}$ is small), it is of utmost importance that the base stations are properly distributed over the different clusters. If not, the relatively few bridge links between the clusters may get congested, since these will be used to transfer the data of an entire cluster to reach a base station in another cluster.

The algorithm presented in Table II uses both $\mathbf{c}$ and $\mathbf{f}$ to select Q base stations in a network (or Q cluster heads in Q different clusters). When this algorithm is applied to the network graph in Fig. 2(a), it indeed selects nodes 3, 9 and 18 as the base stations. When applied to the large-scale network graph depicted in Fig. 4 (again ignoring the red dotted line), it selects nodes 13, 27, 47, 58 as base stations after succesfully identifying 4 densely-connected node clusters (indicated by 4 colors).

B. λ_2-dependency of distributed estimation algorithms

It has been shown that the algebraic connectivity directly influences the performance of several distributed estimation algorithms. To provide some intuition why this is the case, and considering the important implications for distributed signal processing in general, we will first elaborate in more detail on this λ_2-dependency, in particular for the well-known consensus averaging (CA) algorithm [6], as well as for diffusion-type algorithms [2], [3]. Although beyond the scope of this paper, it is noted that similar results on λ_2-dependency have been presented for other algorithms such as gossip-type algorithms [7] and general consensus algorithms [1]. In the next section, we will demonstrate how the Fiedler vector can be used as a heuristic to optimize λ_2 in an ad hoc network.

1) Consensus averaging (CA): Assume that each node $k \in \mathcal{K}$ collects an observation $x_k^{(0)}$, then the goal of the CA algorithm is to compute the average $\overline{x} = \frac{1}{K}\mathbf{1}^T\mathbf{x}^{(0)}$ (where $\mathbf{x}^{(0)}$ has $x_k^{(0)}$ as its k-th entry). A common approach is to use Degroot's algorithm [40], i.e., $x_k^{(0)}$ is iteratively updated at node k as $x_k^{(i+1)} = g_{kk}x_k^{(i)} + \sum_{q\in\mathcal{N}_k} g_{kq}x_q^{(i)}$, which can be written as the in-network PI

$$\mathbf{x}^{(i+1)} = \mathbf{G}\cdot\mathbf{x}^{(i)} \tag{25}$$

where $\mathbf{G} = [g_{kq}]_{K\times K}$. If $\mathbf{G}$ satisfies some necessary and sufficient [6], each node will eventually obtain $x_k^{(\infty)} = \overline{x}$, $\forall\, k \in \mathcal{K}$, i.e., consensus is reached. A popular choice for the

TABLE II
DESCRIPTION OF A BASE STATION OR CLUSTER HEAD SELECTION ALGORITHM

1) Partition the network in Q clusters $\mathcal{C} = \{\mathcal{G}_1, \ldots, \mathcal{G}_Q\}$, e.g., with the algorithm in Table I.
2) In each cluster $\mathcal{G}_i \in \mathcal{C}$, $i = 1 \ldots Q$, compute the cluster-level eigenvector centrality $\mathbf{c}(\mathcal{G}_i)$, where bridge links between clusters are ignored.
3) In each cluster $\mathcal{G}_i \in \mathcal{C}$, $i = 1 \ldots Q$, find the maximum entry in $\mathbf{c}(\mathcal{G}_i)$. Each node corresponding to such a maximum is selected as a base station or cluster head.

Remark: *the available data throughput of each link can also be incorporated by using a weighted adjacency and/or Laplacian matrix when computing* $\mathbf{f}$ *and* $\mathbf{c}$.

weight matrix $\mathbf{G}$, corresponding to the so-called Laplacian rule [6], is

$$\mathbf{G} = \mathbf{I} - \sigma \mathbf{L} \tag{26}$$

which is also the matrix defined earlier in (20). Note that this choice partially affects generality, i.e., it allows node k to give a different weight g_{kq} to each neighbor $q \in \mathcal{N}_k$, but the weights are confined to be symmetric, i.e., $g_{kq} = g_{qk}$ since $l_{kq} = l_{qk}$. Considering the spectrum of $\mathbf{L}$, the two dominant (in absolute value) eigenvalues of $\mathbf{G}$ are equal to $\gamma_K = 1 - \sigma 0 = 1$ and $\gamma_{K-1} = (1 - \sigma\lambda_2)$ if

$$0 < \sigma \leq \sigma^* = \frac{2}{\lambda_2 + \lambda_K} \ . \tag{27}$$

Therefore, if σ is not too large, the principal eigenvector of $\mathbf{G}$ is equal to the λ_1-eigenvector of $\mathbf{L}$, i.e., $\frac{1}{\sqrt{K}}\mathbf{1}$. Since the absolute value of all other eigenvalues is strictly smaller than 1, we find that $\lim_{N\to\infty} \mathbf{G}^N = \frac{1}{K}\mathbf{1}\mathbf{1}^T$, and therefore (25) indeed converges to $\mathbf{x}^{(\infty)} = \frac{1}{K}\mathbf{1}\mathbf{1}^T\mathbf{x}^{(0)} = \overline{x}\mathbf{1}$.

Since the convergence rate of a PI directly depends on the ratio β between the two dominant eigenvalues, the CA algorithm (25) converges at the rate

$$\boxed{\beta = \frac{1}{1-\sigma\lambda_2} \leq \frac{1}{1-\sigma^*\lambda_2} = \frac{\lambda_K+\lambda_2}{\lambda_K-\lambda_2} \ .} \tag{28}$$

Therefore, increasing the algebraic connectivity λ_2 (see Section V-C) will always improve the CA convergence rate β for any σ that satisfies (27). For the optimal choice $\sigma = \sigma^*$, the convergence rate increases when $\lambda_2 \to \lambda_K$. To determine σ^*, λ_K can be computed in a distributed fashion from the limiting growth or shrinking rate of the in-network PI $\mathbf{x}^{(i+1)} = \mathbf{L} \cdot \mathbf{x}^{(i)}$. If suboptimal convergence is sufficient, then the nodes can safely use the value $\sigma = \frac{2}{\lambda_2 + \overline{\lambda}}$ where $\overline{\lambda}$ is defined in (21).

2) Diffusion adaptation: A similar conclusion can be made for diffusion-based adaptive algorithms [2], [3], which combine adaptive filtering techniques with a weighted averaging step similar to (25). In this case, each node $k \in \mathcal{K}$ collects observations $\{\mathbf{u}_{k,i}\}_{i\in\mathbb{N}}$ of a stochastic vector $\mathbf{u}_k$ and observations $\{d_{k,i}\}_{i\in\mathbb{N}}$ of a response signal $d_k = \mathbf{u}_k^T\mathbf{w}^o + n_k$, where n_k denotes an additive noise term and $\mathbf{w}^o$ denotes an unknown parameter vector (common to all nodes). The goal is to estimate and track $\mathbf{w}^o$ at each node $k \in \mathcal{K}$ by means of local cooperation. As an example, consider the diffusion least mean squares (LMS) algorithm [3], where each node alternates between a stochastic gradient descent step (with stepsize μ)

$$\boldsymbol{\psi}_k^{(i)} = \mathbf{w}_k^{(i)} + \mu\mathbf{u}_{k,i}(d_{k,i} - \mathbf{u}_{k,i}^T\mathbf{w}_k^{(i)}) \tag{29}$$

and a diffusion step involving a weighted averaging with weight matrix $\mathbf{G} = [g_{kq}]_{K\times K}$

$$\mathbf{w}_k^{(i+1)} = g_{kk}\boldsymbol{\psi}_k^{(i)} + \sum_{q\in\mathcal{N}_k} g_{kq}\boldsymbol{\psi}_q^{(i)} \ . \tag{30}$$

In the sequel, we give an intuitive argument to demonstrate how the diffusion step improves the performance of the LMS algorithm, and how this improvement is influenced by the algebraic connectivity. For the sake of an easy exposition, we consider the scalar case where $\mathbf{u}$, $\mathbf{w}$ and $\boldsymbol{\psi}$ become u, w and ψ, respectively. We define the instantaneous error vectors $\widetilde{\mathbf{w}}^{(i)}$ and $\widetilde{\boldsymbol{\psi}}^{(i)}$, where the k-th entry is given by $\widetilde{w}_k^{(i)} = w_k^o - w_k^{(i)}$ and $\widetilde{\psi}_k^{(i)} = w_k^o - \psi_k^{(i)}$, respectively. The norm $\|\widetilde{\mathbf{w}}^{(i)}\|$ can then be viewed as an instantaneous network-wide performance measure at iteration i of the diffusion LMS algorithm. By defining $\boldsymbol{\Lambda}_i = \text{diag}\{u_{1,i}, \ldots, u_{K,i}\}$ and $\mathbf{n}_i = [n_{1,i} \ldots n_{K,i}]^T$, (29)-(30) can straightforwardly be rewritten as

$$\widetilde{\boldsymbol{\psi}}^{(i+1)} = \left(\mathbf{I} - \mu\boldsymbol{\Lambda}_i^2\right) \cdot \widetilde{\mathbf{w}}^{(i)} - \mu\boldsymbol{\Lambda}_i \cdot \mathbf{n}_i \tag{31}$$

$$\widetilde{\mathbf{w}}^{(i+1)} = \mathbf{G} \cdot \widetilde{\boldsymbol{\psi}}^{(i+1)} \ . \tag{32}$$

Let us now compare $\widetilde{\boldsymbol{\psi}}^{(i)}$ with $\widetilde{\mathbf{w}}^{(i)}$, i.e., the network-wide instantaneous error before and after the diffusion step. Note that $\widetilde{\boldsymbol{\psi}}^{(i)}$ can be decomposed as $\widetilde{\boldsymbol{\psi}}^{(i)} = \overline{\psi}^{(i)}\mathbf{1} + \boldsymbol{\eta}^{(i)}$ where $\overline{\psi}^{(i)} = \frac{1}{K}\mathbf{1}^T\widetilde{\boldsymbol{\psi}}^{(i)}$ such that $\boldsymbol{\eta}^{(i)}$ can be viewed as a 'disagreement' component. In steady state, this disagreement component is mainly dominated by the noise $\mathbf{n}_i$ and the variance of the local stochastic gradients, hence $\boldsymbol{\eta}^{(i)}$ should preferably be eliminated. If we again choose the Laplacian rule $\mathbf{G} = \mathbf{I} - \sigma\mathbf{L}$ in the diffusion step (32), it can be easily found that (note that $\boldsymbol{\eta}^{(i)}$ and $\mathbf{1}$ are orthogonal)

$$\|\widetilde{\mathbf{w}}^{(i)}\|^2 \leq \|\overline{\psi}^{(i)}\mathbf{1}\|^2 + (1-\sigma\lambda_2)^2\|\boldsymbol{\eta}^{(i)}\|^2 \leq \|\widetilde{\boldsymbol{\psi}}^{(i)}\|^2 \tag{33}$$

i.e., the network-wide error $\|\widetilde{\mathbf{w}}^{(i)}\|$ (after the diffusion step) is strictly smaller than $\|\widetilde{\boldsymbol{\psi}}^{(i)}\|$ (before the diffusion step), unless $\widetilde{\boldsymbol{\psi}}^{(i)}$ is equal in each node ($\|\boldsymbol{\eta}^{(i)}\| = 0$), in which case the diffusion step has no impact. Note that (31)-(32) is a recursive expression, hence the beneficial effect of $\mathbf{G}$ builds up over the iterations (we refer to [3] for a rigorous performance analysis which incorporates this recursive effect). The suppresion of $\boldsymbol{\eta}^{(i)}$ in every iteration of the diffusion LMS algorithm directly depends on the magnitude of $(1 - \sigma\lambda_2)$, hence increasing the algebraic connectivity (see Section V-C) generally has a beneficial effect on the convergence speed and steady-state performance.

C. Increasing the algebraic connectivity ($\mathbf{f}$)

In the previous section, we have explained that a large algebraic connectivity λ_2 is beneficial for the performance of distributed estimation algorithms. Furthermore, a network graph with a small λ_2 is vulnerable to link failures or congestion, due to the presence of crucial bridge links between the densely connected node clusters. Therefore, a TA distributed algorithm should be able to modify the set of active links in the network graph, to maintain a large algebraic connectivity.

To obtain the largest possible λ_2, the network graph should have a fully connected topology, which is then very energy-inefficient. In most cases however, λ_2 can be significantly increased by adding only a few links between carefully chosen node pairs [9], [41]. This can be achieved based on an iterative greedy approach where, in each iteration, the link is added that yields the largest increase in λ_2. As it turns out, the Fiedler vector $\mathbf{f}$ provides an important heuristic to select this link. Indeed, it can be shown that [41]

$$\boxed{\frac{\partial \lambda_2}{\partial w_{kq}} = \mathbf{f}^T \frac{\partial \mathbf{L}}{\partial w_{kq}} \mathbf{f} = (f_k - f_q)^2} \tag{34}$$

i.e., a small perturbation Δ on the weight $w_{kq} = w_{qk}$ of the link (k, q) will yield a change of $\Delta(f_k - f_q)^2$ in λ_2. Therefore, to achieve a significant increase in λ_2, it suffices to identify $\underset{(k,q)\notin\mathcal{L}}{\arg\max}\,(f_k - f_q)^2$ (i.e., identify the minimum and maximum entry in $\mathbf{f}$), and to create a link between the resulting nodes k and q. In the network graphs depicted in Fig. 1, Fig. 2(a) and Fig 4, this optimal link is indicated with a red dotted line, demonstrating that this procedure will try to connect nodes on the outer edges of 2 different clusters. It is noted that a similar procedure can also be used to remove links that do not have much impact on λ_2, to reduce the overall energy consumption [9].

Observe that this procedure can also be used to manipulate the Laplacian combiner weights g_{kq} in (26) to improve the performance of distributed estimation algorithms (see Section V-B). Furthermore, if σ is set to $\sigma = \sigma^*$ in (26), (28) shows that a joint minimization of λ_K and maximization of λ_2 should improve performance. λ_K can be minimized by computing a similar heuristic based on the principal eigenvector of $\mathbf{L}$ corresponding to λ_K (which can also be easily computed in a distributed fashion). Based on the above heuristics, a greedy method is proposed in [9], which incorporates the simultaneous effects on λ_K, λ_2 and the resulting increase/decrease in energy consumption when adding and/or removing a specific link. Similar Fiedler-based techniques have also been used to control the motion of moving agents with limited communication range, to increase their interconnectivity [42].

D. Topology selection and network pruning ($\mathbf{c}$ / $\mathbf{f}$)

For some distributed algorithms, the network graph is required to belong to a pre-defined topology class, e.g., a tree topology [43], a ring topology [44], etc. Setting up such a specific topology by pruning links from an initial ad hoc network graph is often non-trivial, especially in a distributed context. Again, the vectors $\mathbf{c}$ and $\mathbf{f}$ can be of great help in this process. For example, if a highly branched spanning tree is required, the node corresponding to the maximum entry in $\mathbf{c}$ should be chosen as the root node. The tree can then grow further by using a similar heuristic on the branches, i.e., a so-called brush fire search. A similar procedure can be used to approximate an optimal spanning tree in a weighted network graph [45]. If it is important that each node has a short path to the root node, a Fiedler vector-based centrality measure can also be used instead (see Section IV-B4).

A more difficult example is the construction of a ring topology or a so-called Hamiltonian cycle [44], i.e., a cyclic path that visits each node exactly once, which is an NP-complete problem. However, if an approximate Hamiltonian cycle is allowed where a few nodes can be visited more than once, the Fiedler vector can be used to identify node clusters and bridge links. Many of the bridge links will be part of the (pseudo-)Hamiltonian cycle and should therefore be treated as anchor points in the definition of the cycle. It is noted that the corresponding node clustering also allows for a divide-and-conquer approach, where each cluster sets up a (pseudo-)Hamiltonian path from one bridge link to another bridge link.

E. Eigenutility of nodes ($\mathbf{c}$)

In most of the currently available distributed algorithms, the nodes themselves do not know how they influence the overall network performance. At best, they only know how they influence the signal processing task of their direct neighbors. Therefore, an interesting objective would be to assign some ‘utility’ measure to each node based on the node’s influence in a *network-wide* context. Depending on the application or context, ‘utility’ can have different meanings, e.g., the usefulness of the data that a node provides [46], the quality of its local estimate [3], its reliability [47], its willingness to cooperate, its data throughput, etc.

A first approach to defining such a utility measure is to let each node $k \in \mathcal{K}$ assign a non-negative utility score w_{kq} to each of its neighbors $q \in \mathcal{N}_k$, where $\sum_{q\in\mathcal{N}_k} w_{kq} = 1$. The utility of node k could then be defined as $u_k = \sum_{q\in\mathcal{N}_k} w_{qk}$. However, this is again a local measure, which does not relate to the impact of node k on the entire network. For instance, if node k is only found useful by nodes that themselves are not found useful by their own neighbors, node k should get a small u_k as it is indeed not useful in a network-wide context. Similarly, a node should get a higher utility if it is found useful by other high-utility nodes. This chicken-and-egg problem straightforwardly brings us back to the notion of eigenvector centrality in (5), this time with a weighted adjacency matrix equal to the (possibly assymetric) scoring matrix $\mathbf{W} = [w_{kq}]_{K\times K}$, i.e.,

$$\mathbf{u} = \mathbf{W}^T \cdot \mathbf{u}\ . \tag{35}$$

The unique positive solution for $\mathbf{u}$ is equal to the principal eigenvector of $\mathbf{W}^T$. It is noted that the scaling factor α in (5) is omitted here since $\rho(\mathbf{W}) = 1$ due to the normalization $\sum_{q\in\mathcal{N}_k} w_{kq} = 1$. The iteration (35) is strongly related to the invariant distribution of random walks over a Markov chain where the transition probabilities are given by the entries in $\mathbf{W}$ [32, Chapter 2].

The entries in $\mathbf{u}$ are referred to as the nodes' eigenutility. They are the sensor network equivalent of sociometric eigenvector status indices [34], [48]. The eigenutility can be very useful for resource allocation, node subset selection, weight selection in weighted averaging, etc. In some cases, it may be desired to dampen the effect of far-away nodes on a node's utility. For example, in large-scale networks for online tracking of rapidly time-varying parameters, there may be insufficient time to diffuse information over the entire network, as old information may rapidly become obsolete. In this case, far-away nodes cannot affect a node's performance, hence they should be neglected in the determination of a node's utility. In such cases, Katz centrality may prove a valid alternative. For example, Katz centrality is used in the EigenTrust algorithm that assigns a measure of reliability to each node [47], and in Google's Pagerank algorithm that assigns scores to internet web pages [15]. In [31], a unifying framework is proposed that captures several of these measures in a single cost function, and a single distributed algorithm to compute all of these (including Katz centrality, (personal) Pagerank, peer-to-peer rating, information centrality, and even consensus averaging).

F. Other applications of $\mathbf{c}$ *and* $\mathbf{f}$

There are many other distributed signal processing problems that can benefit from the information that is included in the centrality vector $\mathbf{c}$ and/or the Fiedler vector $\mathbf{f}$. Although beyond the scope of this paper, we briefly mention the following applications:

- *Resource allocation* ($\mathbf{c}$ / $\mathbf{f}$): Resource allocation is a common problem appearing in almost all distributed algorithms, and it may refer to assigning to each node/link, e.g., a quantization level, a bit-rate, a channel code length, a transmission power, and so on. In this context, node clustering can be used to identify the 'weakest links', i.e., the bridge links between densely-connected node clusters[4]. These bridge links should ideally be protected with strong channel codes and fine quantization to facilitate robust data dissemination between clusters. Central nodes or nodes with a large eigenutility should also receive more resources, because these are assumed to have a large overall influence.
- *Divide-and-conquer techniques* ($\mathbf{f}$): Node clustering can be helpful to divide difficult problems involving large-scale networks into small-scale sub problems on the cluster level, which can be solved more easily (e.g., for stability or robustness control [30], or to find a Hamiltonian cycle).
- *Node subset selection* ($\mathbf{c}$ / $\mathbf{f}$): Node subset selection refers to selecting a set of nodes that contribute most to the overall network performance, while putting less useful nodes to sleep to save energy. This problem can use input from the eigenutility procedure, but also node clustering may be helpful, e.g., to protect important bridge links from being removed.

[4]In [29], another (related) heuristic method is proposed to identify critical links without using explicit node clustering procedures.

VI. Conclusions

In this paper, we have explained how nodes in a network graph can infer information about the network topology or its topology-related properties, based on in-network distributed learning, i.e., without relying on an 'external observer' who has a complete overview over the network. We have reviewed some key concepts from the field of SGT, with a focus on those that allow for a simple distributed implementation, i.e., eigenvector or Katz centrality, algebraic connectivity and the Fiedler vector. We have explained how the nodes themselves can quantify their individual network-wide influence, as well as identify densely-connected node clusters and the sparse bridge links between them. The addressed concepts, as well as more advanced concepts from the field of SGT, are believed to be crucial catalysts in the design of topology-aware distributed algorithms. We have provided examples of how these techniques can be exploited in several non-trivial distributed signal processing tasks.

VII. Acknowledgements

The authors would like to thank Dr. B. Cornelis and the anonymous reviewers for proofreading this manuscript and for their valuable comments and suggestions.

References

[1] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[2] S.-Y. Tu and A. H. Sayed, "Mobile adaptive networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 649–664, Aug. 2011.

[3] F. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035 –1048, March 2010.

[4] S. Kar and J. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315 –3326, July 2008.

[5] G. Mateos and G. B. Giannakis, "Distributed recursive least-squares: Stability and performance analysis," *IEEE Trans. Signal Processing*, vol. 60, no. 7, pp. 3740–3754, July 2012.

[6] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[7] J. Lavaei and R. Murray, "On quantized consensus by means of gossip algorithm - part II: Convergence time," in *American Control Conference*, June 2009, pp. 2958–2965.

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

[9] C. Asensio-Marco and B. Beferull-Lozano, "A greedy perturbation approach to accelerating consensus algorithms and reducing its power consumption," in *IEEE Statistical Signal Processing Workshop (SSP)*, June 2011, pp. 365–368.

[10] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[11] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[12] B. Mohar, "Laplace eigenvalues of graphs - a survey," *Discrete Mathematics*, vol. 109, no. 13, pp. 171–183, 1992.

[13] ——, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, Y. Alavi, G. Chartrand, O. Oellermann, and A. Schwenk, Eds. Wiley, 1991, pp. 871–898.

[14] T. F. Chan, T. C. Ciarlet, and W. K. Szeto, "On the optimality of the median cut spectral bisection graph partitioning method," *SIAM Journal on Scientific Computing*, vol. 18, pp. 943–948, 1997.

[15] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 107–117, 1998.

[16] L. Hagen and A. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074 –1085, Sep. 1992.

[17] W. Richards and A. Seary, "Spectral methods for analyzing and visualizing networks: an introduction," *Dynamic Social Network Modeling and Analysis*, pp. 209–228, 2003.

[18] S. Barnard and H. Simon, "Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems," *Concurrency and computation: Practice and Experience*, vol. 6, pp. 101–117, 1994.

[19] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, pp. 27–64, 2007.

[20] P. Orponen and S. Schaeffer, "Local clustering of large graphs by approximate Fiedler vectors," in *Experimental and Efficient Algorithms*, ser. Lecture Notes in Computer Science, S. Nikoletseas, Ed. Springer Berlin Heidelberg, 2005, vol. 3503, pp. 524–533.

[21] A. D. Sarma, "Algorithms for large graphs," Ph.D. dissertation, Georgia Institute of Technology, 2010.

[22] D. Nanongkai, "Graph and geometric algorithms on distributed networks and databases," Ph.D. dissertation, Georgia Institute of Technology, 2011.

[23] A. Bertrand and M. Moonen, "Distributed computation of the fiedler vector with application to topology inference in ad hoc networks," *Signal Processing*, vol. 93, no. 5, pp. 1106–1117, May 2013.

[24] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, Feb. 2010.

[25] R. K. Williams and G. S. Sukhatme, "Distributed connectivity control in mobile networks under local topology constraints," *Internal report of the Departments of Electrical Engineering and Computer Science at the University of Southern California, Los Angeles*, 2012.

[26] R. Aragues, G. Shi, D. V. Dimarogonas, C. Sagues, and K. H. Johansson, "Distributed algebraic connectivity estimation for adaptive event-triggered consensus," in *Proc. American Control Conference*, Fairmont Queen Elizabeth, Montreal, Canada, June 2012, pp. 32–37.

[27] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 70–83, 2008.

[28] T. Sahai, A. Speranzon, and A. Banaszuk, "Hearing the clusters of a graph: A distributed algorithm," *Automatica*, vol. 48, no. 1, pp. 15–24, 2012.

[29] C. Gkantsidis, G. Goel, M. Mihail, and A. Saberi, "Towards topology aware networks," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, May 2007, pp. 2591–2595.

[30] A. Banaszuk, V. A. Fonoberov, T. A. Frewen, M. Kobilarov, G. Mathew, I. Mezic, A. Pinto, T. Sahai, H. Sane, A. Speranzon, and A. Surana, "Scalable approach to uncertainty quantification and robust design of interconnected dynamical systems," *Annual Reviews in Control*, vol. 35, no. 1, pp. 77–98, 2011.

[31] D. Bickson and D. Malkhi, "A unifying framework of rating users and data items in peer-to-peer and social networks," *Peer-to-Peer Networking and Applications*, vol. 1, no. 2, pp. 93–103, 2008.

[32] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[33] G. L. Thompson, "Lectures on Game Theory, Markov Chains and Related Topics," *Monograph SCR-11, Sandia Corporation, Albuquerque, New Mexico*, 1958.

[34] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[35] M. Jelasity, G. Canright, and K. Engo-Monsen, "Asynchronous distributed power iteration with gossip-based normalization," in *Lecture Notes in Computer Science*, vol. 4641. Springer, 2007, pp. 514–525.

[36] D. A. Spielman and S.-H. Teng, "Spectral partitioning works: Planar graphs and finite element meshes," *Linear Algebra and its Applications*, vol. 421, no. 23, pp. 284–305, 2007.

[37] B. Elbhiri, S. El Fkihi, R. Saadane, and D. Aboutajdine, "Clustering in wireless sensor networks based on near optimal bi-partitions," in *Proc. Conf. on Next Generation Internet (NGI)*, June 2010, pp. 1–6.

[38] A. Tahbaz-Salehi and A. Jadbabaie, "A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times," in *Proc. IEEE Conference on Decision and Control*, Dec. 2006, pp. 4664–4669.

[39] B. Deosarkar, N. Yadav, and R. Yadav, "Clusterhead selection in clustering algorithms for wireless sensor networks: A survey," in *Proc. Int. Conf. on Computing, Communication and Networking (ICCCN).*, Dec. 2008, pp. 1–8.

[40] M. H. Degroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[41] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *IEEE Conference on Decision and Control*, Dec. 2006, pp. 6605–6611.

[42] M. C. DeGennaro and A. Jadbabaie, "Decentralized control of connectivity for multiagent systems," in *Proc. IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006, pp. 3628–3633.

[43] A. Bertrand and M. Moonen, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2196–2210, 2011.

[44] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, no. 8, pp. 4064–4077, Aug. 2007.

[45] A. Robles-Kelly and E. R. Hancock, "Spanning tree recovery via random walks in a riemannian manifold." in *Proc. Iberoamerican Congress on Pattern Recognition*, ser. Lecture Notes in Computer Science, vol. 3287. Springer, 2004, pp. 303–311.

[46] A. Bertrand, J. Szurley, P. Ruckebusch, I. Moerman, and M. Moonen, "Efficient calculation of sensor utility and sensor removal in wireless sensor networks for adaptive signal estimation and beamforming," *IEEE Trans. Signal Processing*, vol. 60, no. 11, pp. 5857–5869, Nov. 2012.

[47] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. international conference on World Wide Web*. New York, USA: ACM, 2003, pp. 640–651.

[48] J. R. Seeley, "The net of reciprocal influence: A problem in treating sociometric data," *Canadian Journal of Psychology*, vol. 3, pp. 234–240, 1949.

Alexander Bertrand received the electrical engineering degree (2007) and the Ph.D. degree in engineering sciences (2011), both from KU Leuven, Belgium. He is currently a Postdoctoral Fellow of the Research Foundation - Flanders (FWO), affiliated with the E.E. Department of KU Leuven. In 2010, he was a visiting researcher at the University of California, Los Angeles (UCLA). He has served as a Technical Program Committee (TPC) Member for the European Signal Processing Conference (EUSIPCO) 2012 & 2013, and is currently co-editor of the newsletter of the European Association for Signal Processing (EURASIP). He received the 2012 IBM (Belgium) Award.
[*alexander.bertrand@esat.kuleuven.be*]

Marc Moonen (IEEE Fellow 2007) is a Full Professor at the E.E. Department of KU Leuven, Belgium. He has served as Editor-in-Chief for the EURASIP Journal on Applied Signal Processing (2003-2005), and has been a member of the editorial board of IEEE Transactions on Circuits and Systems, IEEE Signal Processing Magazine, EURASIP Journal on Wireless Communications and Networking, and Signal Processing. He is currently a member of the editorial board of EURASIP Journal on Advances in Signal Processing and Feature Articles Area Editor in IEEE Signal Processing Magazine. He was President of EURASIP (2007-2007, 2011-2012) and is currently Past-President of EURASIP.
[*marc.moonen@esat.kuleuven.be*]