Distributed Adaptive Estimation of Node-Specific Signals in Wireless Sensor Networks With a Tree Topology

Alexander Bertrand, *Student Member, IEEE*, and Marc Moonen, *Fellow, IEEE*

***Abstract*—We present a distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in a wireless sensor network with a tree topology. The algorithm extends the DANSE algorithm for fully connected sensor networks, as described in previous work. It is argued why a tree topology is the natural choice if the network is not fully connected. If the node-specific desired signals share a common latent signal subspace, it is shown that the distributed algorithm converges to the same linear MMSE solutions as obtained with the centralized version of the algorithm. The computational load is then shared between the different nodes in the network, and nodes exchange only linear combinations of their sensor signal observations and data received from their neighbors. Despite the low connectivity of the network and the multi-hop signal paths, the algorithm is fully scalable in terms of communication bandwidth and computational power. Two different cases are considered concerning the communication protocol between the nodes: point-to-point transmission and local broadcasting. The former assumes that there is a reserved communication link between node-pairs, whereas with the latter, nodes communicate the same data to all of their neighbors simultaneously. The convergence properties of the algorithm are demonstrated by means of numerical examples.**

***Index Terms*—Adaptive estimation, distributed compression, distributed estimation, wireless sensor networks (WSNs).**

I. Introduction

A wireless sensor network (WSN) consists of sensor nodes that cooperate to perform a certain task, such as the estimation of a parameter or signal, where data is shared between nearby nodes through a wireless link. A general objective is to utilize all available data throughout the network, possibly through a fusion center that gathers all sensor signal observations and performs all computations. However, in many cases a distributed approach is preferred, which is scalable with respect to both communication resources and computational power. In this case, data diffuses through the network and each node contributes to the processing (e.g., [1]–[4]).

In this paper, we consider distributed signal estimation, rather than parameter estimation. This means that the number of variables to estimate grows linearly with the number of temporal observations, i.e., for each sample time of the sensors, a new sample of the desired signal(s) needs to be estimated. The estimation then usually relies on linear spatial filtering or beamforming, as often used in signal enhancement [5]–[8]. In parameter estimation problems on the other hand, the number of estimation variables is fixed, i.e., it does not grow with the number of temporal observations [1]–[4], [10], [11]. Usually, intermediate estimates are then exchanged and iteratively improved over time, often without exchanging the actual sensor observations. In the case of distributed beamforming or signal estimation, (fused or compressed) signal observations are exchanged between nodes, and then the aim is to iteratively improve the local fusion rules to better estimate future samples. These type of WSNs are usually smaller in size, and operate with a larger bandwidth and energy consumption compared to traditional large-scale WSNs [7], [8], especially in applications with high sampling rates. They are often also assumed to be more robust, especially in real-time systems, since packet loss can then result in instantaneous signal degradation. Therefore, in this paper, we assume that the communication links are ideal, i.e., they are not subject to noise or random failures.

In [12] and [13], a distributed adaptive node-specific signal estimation (DANSE) algorithm is presented, based on linear MMSE estimation. It operates in a fully connected sensor network where each node has access to multi-channel sensor signal observations. The term "node-specific" refers to the fact that each node estimates a different desired signal. Node-specific estimation is relevant in cases where inherent spatial information in the local observations of the target sources needs to be preserved during the estimation, e.g., to serve as an input for a localization algorithm, or in collaborating hearing aids when the aim is to also preserve the cues for directional hearing [14]. Due to the linearity of the proposed centralized estimator, the algorithm can be made distributed and it significantly compresses the signals that are communicated between nodes, provided that the desired signals of the different nodes share a common low dimensional latent signal subspace (which is assumed to be unknown). Although the nodes broadcast only a few linear combinations of their sensor signal observations, the DANSE algorithm provides linear minimum mean squared error (MMSE) estimates as if all data were available at each node. In [15], the DANSE algorithm is extended to also guarantee convergence when nodes update

Manuscript received July 02, 2010; revised September 21, 2010, December 01, 2010; accepted January 18, 2011. Date of publication January 28, 2011; date of current version April 13, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Aleksandar Dogandzic. A. Bertrand is supported by a Ph.D. grant of the I.W.T. (Flemish Institute for the Promotion of Innovation through Science and Technology). This research work was carried out at the ESAT Laboratory of Katholieke Universiteit Leuven, in the frame of K.U. Leuven Research Council CoE EF/05/006 Optimization in Engineering (OPTEC), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P6/04 (DYSCO, "Dynamical systems, control and optimization," 2007–2011), and Research Project FWO nr. G.0600.08 ("Signal processing and network design for wireless acoustic sensor networks"). The scientific responsibility is assumed by its authors.

The authors are with the Department of Electrical Engineering (ESAT-SCD/SISTA), Katholieke Universiteit Leuven, B-3001 Leuven, Belgium (e-mail: alexander.bertrand@esat.kuleuven.be; marc.moonen@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2011.2108290

simultaneously or asynchronously, which generally results in faster tracking. In [17], a more robust version of DANSE has been formulated, referred to as R-DANSE. Simulations in [7] and [8] illustrate the potential of the algorithm for distributed speech enhancement in acoustic sensor networks.

A limitation of the DANSE algorithm in [12] is that the network is assumed to be fully connected, which is only possible in practice if the nodes have sufficient transmission power, and if the available communication bandwidth is large enough. Furthermore, the amount of data that is received and processed by each node increases with the number of nodes in the network. In this paper, we modify the DANSE algorithm, such that it can operate in a network with a tree topology, avoiding the aforementioned issues. We refer to this algorithm as tree-DANSE or T-DANSE. The choice for a tree topology is justified by the fact that it contains no cycles, and hence does not introduce any feedback paths. We will explain that feedback paths harm the convergence and optimality of the DANSE algorithm. The formulation of T-DANSE was briefly introduced in [16], without a theoretical analysis. In this paper, we provide more details and include a convergence and optimality proof.

In the T-DANSE algorithm, the signal observations of the different nodes are fused in a decentralized way, i.e., each node linearly combines its own sensor signal observations with data obtained from its neighbors before forwarding them to the next node. Remarkably, despite this distributed fusion process, each node is able to optimally estimate its own node-specific signal as if all data of the complete network were available. The local estimation task at each node is equivalent to the centralized estimation problem, but at a much smaller scale, i.e., with a drastically reduced amount of signals.

As opposed to fully connected DANSE, the number of signals that need to be processed by a node in T-DANSE does not depend on the total number of nodes in the network, but only on the number of neighbors of that node. This is important since the number of signals that a single node can receive and process in real-time is usually limited, especially when operating at large sampling rates. An additional advantage of using multi-hop networks, is the fact that nodes can transmit with lower power, and it enables spatial reuse of the spectrum. Furthermore, even when nodes only have access to observations of a single-channel sensor signal, the T-DANSE algorithm yields a benefit in terms of communication bandwidth efficiency, whereas fully connected DANSE is only useful if the number of sensor signals per node is larger than the dimension of the latent signal subspace that contains the desired signals [12].

We will consider two different communication protocols: point-to-point transmission and local broadcasting. The former assumes that there is a reserved communication link between node-pairs, whereas with the latter, a node communicates the same data to multiple neighbors simultaneously. We will show that both cases can be treated equivalently from a theoretical point of view. However, the broadcast protocol is obviously more efficient in terms of communication bandwidth.

The outline of this paper is as follows. In Section II, the general problem statement for distributed node-specific linear MMSE estimation is given. We briefly review the DANSE algorithm [12] for fully connected networks in Section III, which will act as a starting point and which allows us to introduce some necessary notation.[1] In Section IV, we point out that feedback paths in the network topology harm the convergence and optimality of the DANSE algorithm. In Section V, we introduce the T-DANSE algorithm in a network with a tree topology, and prove convergence and optimality. In Section VI, we explain how T-DANSE can be used in a communication protocol that supports local broadcasting. Section VII provides some simulation results. Conclusions are given in Section VIII.

II. Problem Formulation and Notation

In this section, we briefly describe the data model and the problem statement. More details can be found in [12]. This section can be skipped if the reader is familiar with the setup and the notation in [12].

A. Data Model

Consider a network with sensor nodes $\{1,\ldots,J\} = \mathcal{J}$. At this point, we do not yet make any assumptions on the topology of this network. In the sequel, we assume that all mentioned signals are stationary and ergodic.[2] Sensor node k collects observations of a complex[3] valued random M_k-channel sensor signal $\mathbf{y}_k[t]$, where $t \in \mathbb{N}$ is the discrete sample time index. For the sake of an easy exposition, we will omit the time index when referring to a signal, and we will only write the time index when referring to one specific observation, i.e., $\mathbf{y}_k[t]$ is the observation of the signal $\mathbf{y}_k$ at time t. We define $\mathbf{y}$ as the M-channel signal in which all $\mathbf{y}_k$ are stacked, where $M = \sum_{k=1}^{J} M_k$. This scenario is depicted in Fig. 1. The different channels of the signal $\mathbf{y}_k$ may correspond to different sensors at node k (as it is the case in Fig. 1), or different delayed versions of its sensor signals to exploit temporal information.

The objective for each node k is to optimally estimate a node-specific K-channel desired signal $\mathbf{d}_k$ that is assumed to be correlated to $\mathbf{y}$. We consider the general case where $\mathbf{d}_k$ is not an observed signal, i.e., it is assumed to be unknown, as it is the case in signal enhancement. We assume that the node-specific desired signals $\mathbf{d}_k$ share a common K-dimensional latent signal subspace, defined by the channels of an unknown K-channel latent signal $\mathbf{d}$, i.e.,

$$\mathbf{d}_k = \mathbf{A}_k \mathbf{d}, \quad \forall\, k \in \mathcal{J} \tag{1}$$

with $\mathbf{A}_k$ a full rank $K \times K$ matrix with unknown coefficients.[4] It is noted that we assume (without loss of generality) that the

[1]Although this paper does not assume prior knowledge on the fully connected DANSE algorithm, it is recommended to read [12] first, since it addresses a much simpler case, which allows the reader to get familiar with the notation, the problem statement, and the algorithm.

[2]In practice, the stationarity and ergodicity assumption can be relaxed to short-term stationarity and ergodicity, in which case the theory should be applied to finite signal segments that are assumed to be stationary and ergodic.

[3]Throughout this paper, all signals are assumed to be complex valued to permit frequency domain descriptions. In this case, multi-tap estimation is also covered, and the data model (1) then corresponds to a frequency domain description of a convolutive mixture.

[4]It is noted that node-specific estimation also exists in a distributed parameter estimation context, e.g., in random-field estimation [10], [11] However, usually it is assumed that the covariance or other parameters describing the dependencies between the hidden node-specific variables are known. This is not the case in our approach, i.e., we do not know the $\mathbf{A}_k$'s or the cross-correlation between the $\mathbf{d}_k$'s. We only exploit the prior knowledge that the $\mathbf{d}_k$'s share a common laten signal subspace, but we do not know anything about this subspace, except for its dimension.

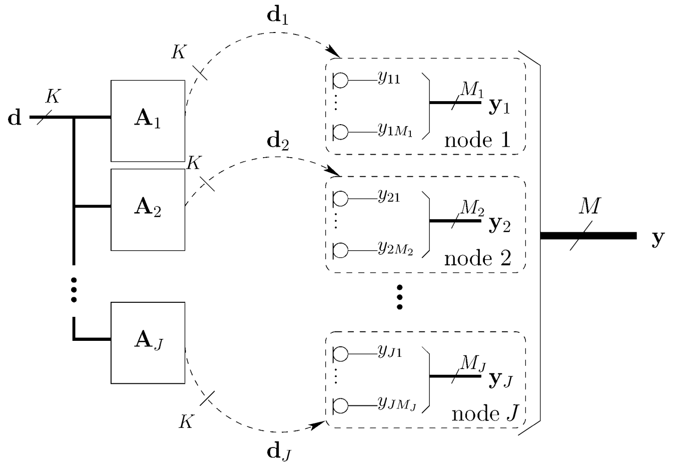


Fig. 1. Description of the scenario. The network (with undefined topology) contains J sensor nodes, $k = 1 \dots J$, where node k collects M_k-channel sensor signal observations and estimates a node-specific desired signal $\mathbf{d}_k$, which is a mixture of the K channels of a common latent signal $\mathbf{d}$.

number of channels of the desired signals $\mathbf{d}_k, \forall\, k \in \mathcal{J}$, is equal to the dimension of the latent signal subspace defined by $\mathbf{d}$. In many practical cases, only a subset of the channels of $\mathbf{d}_k$ may be of actual interest, in which case the other channels should be seen as auxiliary channels to capture the entire K-dimensional signal subspace defined by $\mathbf{d}$ (the reason for this will be explained later).

To make this more concrete, we give an example in the context of noise reduction for speech enhancement that fits the aforementioned data model. Assume a scenario with K speech sources, stacked in the signal $\mathbf{d}$. The observed signals at the M_k sensors (i.e., microphones) of node k are then described by the linear sensor data model

$$\mathbf{y}_k = \mathbf{U}_k \mathbf{d} + \mathbf{n}_k \tag{2}$$

with $\mathbf{U}_k$ an (unknown) $M_k \times K$ steering matrix to the M_k microphones of node k, and $\mathbf{n}_k$ a noise component containing point-source interferers, diffuse noise and sensor noise. Note that (2) is a frequency domain representation, transforming the convolutive acoustic mixture of the speech signals in an instantaneous mixture. The goal of each node is to estimate the mixture of the signals in $\mathbf{d}$ observed at one of their microphones,[5] referred to as the reference microphone. If $K > 1$, additional auxiliary reference microphones need to be selected to obtain a K-dimensional node-specific desired signal $\mathbf{d}_k$. If the first K microphones are selected in each node, then $\mathbf{A}_k$ in (1) corresponds to the first K rows of $\mathbf{U}_k$ in (2). For more information regarding this acoustic application, we refer to [7].

We emphasize that expression (2) is given here as an example, and we do not restrict ourselves to this sensor data model in the design of the algorithms in the sequel.

B. Centralized Linear MMSE Estimation

We first consider the centralized estimation problem, i.e., we assume that all nodes have access to observations of all M channels of $\mathbf{y}$. Node k uses a linear estimator $\mathbf{W}_k$ to estimate $\mathbf{d}_k$ as

$$\bar{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y} \tag{3}$$

where $\mathbf{W}_k$ is a complex $M \times K$ matrix, and where superscript H denotes the conjugate transpose operator. This is similar to beamforming frameworks [5], where multiple signals are linearly combined to generate an output signal with suppressed interferers and background noise. We consider linear MMSE estimation (similar to multi-channel Wiener filtering [6]) based on a node-specific estimator $\hat{\mathbf{W}}_k$, i.e.,

$$\hat{\mathbf{W}}_k = \underset{\mathbf{W}_k}{\arg\min}\; E\left\{\left\|\mathbf{d}_k - \mathbf{W}_k^H \mathbf{y}\right\|^2\right\} \tag{4}$$

where $E\{\cdot\}$ denotes the expected value operator. Assuming that the correlation matrix $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$ has full rank, the solution of (4) is

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1} \mathbf{R}_{yd_k} \tag{5}$$

with $\mathbf{R}_{yd_k} = E\{\mathbf{y}\mathbf{d}_k^H\}$. Based on the assumption that the signals are ergodic, $\mathbf{R}_{yy}$ and $\mathbf{R}_{yd_k}$ can be estimated by time averaging. The $\mathbf{R}_{yy}$ is estimated from the sensor signal observations. Since $\mathbf{d}_k$ is assumed to be unknown, $\mathbf{R}_{yd_k}$ has to be estimated indirectly. A possible way to estimate $\mathbf{R}_{yd_k}$, is to periodically transmit training sequences, or by exploiting the ON–OFF behavior of the desired signal, e.g., in speech enhancement applications [7], [17]. More information on the estimation of $\mathbf{R}_{yd_k}$ can be found in [12]. In the sequel we will assume that $\mathbf{R}_{yd_k}$, or its distributed variants, can be estimated adaptively during operation of the algorithm.

In the distributed case, each node k only has access to observations of $\mathbf{y}_k$ which is a subset of the channels of the full signal $\mathbf{y}$. Therefore, to find the optimal MMSE solution (5) in each node, the observations of $\mathbf{y}_k$ in principle have to be communicated to all nodes in the network, which requires a large communication bandwidth, especially if the network is not fully connected, i.e., if multi-hop transmission is required. As shown in the sequel, due to the linearity of the centralized estimator (5) and the low dimension of the latent signal subspace, we are still able to obtain the linear MMSE solution (5) at each node, even when the data communicated by the nodes is significantly compressed. We assume ideal communication links, i.e., they are not subject to noise or random failures. The issue of link failures in real-time signal estimation in wireless networks is briefly addressed in [18].

III. The DANSE Algorithm in a Fully Connected Network

In this section, we briefly review the DANSE_K algorithm[6] in a fully connected sensor network, as introduced in [12] and [13]. This is important to introduce some notation, and to grasp the underlying idea on how we can distribute each estimator over different nodes. In Section V, we will extend this framework for signal estimation in networks with a tree topology.

We define a partitioning of the matrix $\mathbf{W}_k$ as $\mathbf{W}_k = [\mathbf{W}_{k1}^T \dots \mathbf{W}_{kJ}^T]^T$ where $\mathbf{W}_{kq} \in \mathbb{C}^{M_k \times K}$ is the part of $\mathbf{W}_k$

[5]This is the best one can do in a blind approach, i.e., when there is neither knowledge about $\mathbf{d}$ nor the mixing system. The desired signal is then the observed mixture of target sources at the local sensors. This technique is often used in noise reduction applications for speech enhancement [6]–[8].

[6]The subscript K refers to the number of channels in the broadcast signals of each node. To obtain the optimal estimators, this number should be equal to the dimension of the latent signal subspace defined by $\mathbf{d}$, as proven in [12].

that is applied to the sensor signal observations of $\mathbf{y}_q$. The equivalent of (4) is then

$$\hat{\mathbf{W}}_k = \begin{bmatrix} \hat{\mathbf{W}}_{k1} \\ \hat{\mathbf{W}}_{k2} \\ \vdots \\ \hat{\mathbf{W}}_{kJ} \end{bmatrix} = \underset{\{\mathbf{W}_{k1},\ldots,\mathbf{W}_{kJ}\}}{\arg\min} \; E\left\{ \left\| \mathbf{d}_k - \sum_{q=1}^{J} \mathbf{W}_{kq}^H \mathbf{y}_q \right\|^2 \right\}. \quad (6)$$

In the DANSE_K algorithm, $\mathbf{y}_k$ is linearly compressed to a K-channel signal $\mathbf{z}_k$ (the compression rule will be defined later), which is then broadcast to the remaining $J-1$ nodes. We define the $(J-1)K$-channel signal $\mathbf{z}_{-k} = [\mathbf{z}_1^T \ldots \mathbf{z}_{k-1}^T \mathbf{z}_{k+1}^T \ldots \mathbf{z}_J^T]^T$. Node k then collects observations of its own sensor signals in $\mathbf{y}_k$ and the signals in $\mathbf{z}_{-k}$ obtained from the other nodes in the network. Similar to (4), node k can then compute the linear MMSE estimator with respect to these input signals, i.e.,

$$\begin{bmatrix} \mathbf{W}_{kk} \\ \mathbf{G}_{k,-k} \end{bmatrix} = \underset{\mathbf{W}_{kk},\mathbf{G}_{k,-k}}{\arg\min} \; E\left\{ \left\| \mathbf{d}_k - \begin{bmatrix} \mathbf{W}_{kk}^H \; \mathbf{G}_{k,-k}^H \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k} \end{bmatrix} \right\|^2 \right\} \quad (7)$$

where $\mathbf{W}_{kk}$ is the part of the estimator that is applied to node k's own sensor signals in $\mathbf{y}_k$ and where $\mathbf{G}_{k,-k} = [\mathbf{G}_{k1}^T \ldots \mathbf{G}_{k,k-1}^T \mathbf{G}_{k,k+1}^T \ldots \mathbf{G}_{kJ}^T]^T$ with $\mathbf{G}_{kq} \in \mathbb{C}^{K\times K}$ denoting the part of the estimator that is applied to $\mathbf{z}_q$. The linear compression rule that generates the broadcast signal $\mathbf{z}_k$ is then given by

$$\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k. \quad (8)$$

A schematic illustration of this scheme is shown in Fig. 2, for a network with $J = 3$ nodes. It is noted that $\mathbf{W}_{kk}$ both acts as a compressor and as a part of the estimator $\mathbf{W}_k$. Based on Fig. 2, it can be seen that the parametrization of the $\mathbf{W}_k$ effectively applied at node k, to generate $\bar{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y}$, is then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix} \quad (9)$$

where we assume that $\mathbf{G}_{kk} = \mathbf{I}_K$ with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix. Expression (9) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously, where node k can only control the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_{k,-k}$. The following theorem explains how this parametrization is still able to provide an optimal signal estimate in each node. A similar result will be obtained in Section V for the case of tree topology networks.

Theorem III. 1: If (1) holds, then the optimal estimators $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, given in (5) are in the solution space defined by parametrization (9).

Proof: Since $\mathbf{d}_k = \mathbf{A}_k \mathbf{d}$, and because of (5), we know that the following holds:

$$\forall\, k, q \in \mathcal{J} : \hat{\mathbf{W}}_k = \hat{\mathbf{W}}_q \mathbf{A}_{kq} \quad (10)$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k^H$. The theorem is proven by comparing (10) with (9), and by setting $\mathbf{G}_{kq} = \mathbf{A}_{kq}$, $\forall\, q \in \mathcal{J}$. ■

The DANSE_K algorithm iteratively updates the parameters in (9), by letting each node k compute (7), $\forall\, k \in \mathcal{J}$, in a sequential round robin fashion.[7] It is noted that each node then essentially performs a similar task as in a centralized computation, but on a smaller scale, i.e., with less signals. In [12], it is proven that this procedure converges to the centralized linear MMSE estimators at all nodes, i.e., $\lim_{i\to\infty} \mathbf{W}_k^i = \hat{\mathbf{W}}_k, \forall\, k \in \mathcal{J}$. It is noted that we are not directly interested in the $\mathbf{W}_k$'s, but rather in the estimated samples of the $\mathbf{d}_k$'s. The estimate of a sample $\mathbf{d}_k[t]$ of the desired signal $\mathbf{d}_k$ in node k at any point in the iterative process is computed as

$$\bar{\mathbf{d}}_k[t] = \mathbf{W}_{kk}^H \mathbf{y}_k[t] + \sum_{q\neq k} \mathbf{G}_{kq}^H \mathbf{z}_q[t]. \quad (11)$$

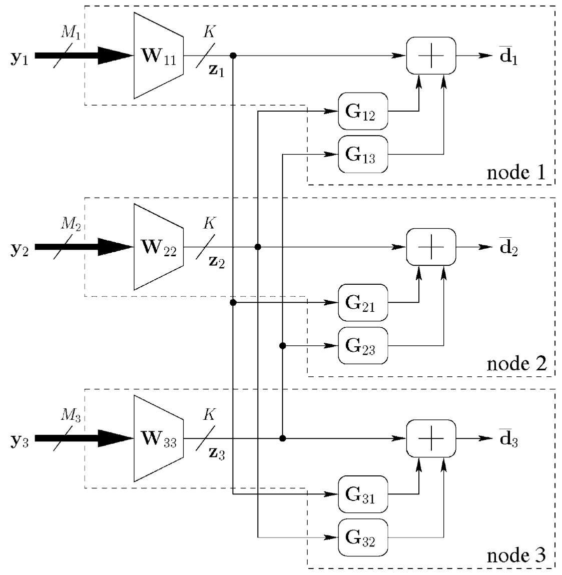


Fig. 2. The DANSE_K scheme with three nodes ($J = 3$). Each node k estimates a signal $\mathbf{d}_k$ using its own M_k-channel sensor signal observations, and two compressed K-channel signal observations broadcast by the other two nodes.

Remark I: It is assumed that the cross-correlations $E\{\mathbf{y}_k \mathbf{d}_k^H\}$ and $E\{\mathbf{z}_{-k}\mathbf{d}_k^H\}$ can be (re-)estimated during operation of the algorithm. As explained earlier, this is only possible in certain applications, e.g., when the target source has an ON–OFF behavior or when training sequences can be used. We will not elaborate on this issue here, and we refer to [12] instead for further details.

Remark II: The iterative nature of the DANSE_K algorithm may suggest that the same sensor signal observations are compressed and broadcast multiple times, i.e., once after every iteration. However, in practical applications, iterations are spread over time, which means that successive updates of the estimators use different sensor signal observations. By exploiting the (short-term) stationarity assumption, updated estimators are only used for new (future) observations. This means that the iterations are not performed on the same block of data, but only on the local fusion rules at the nodes. For a detailed non-batch description of the algorithm, we refer to [12].

IV. DANSE IN SIMPLY CONNECTED NETWORKS WITH CYCLES

If the network is not fully connected, information must be passed from one side of the network to the other over multiple

[7]Results where nodes update simultaneously are also available [15], but these are not addressed here.

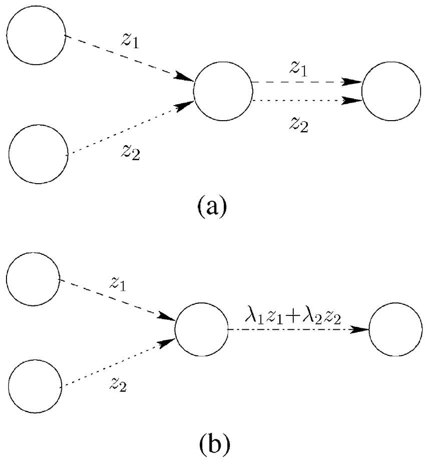


Fig. 3. Two different types of data fusion in a network that is not fully connected. (a) The relay case. (b) Linear combination of inputs.

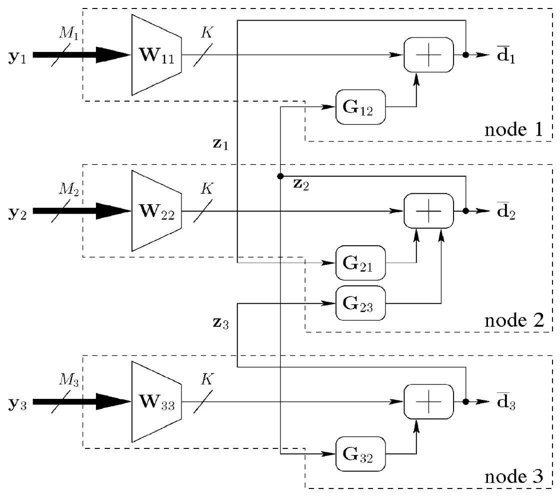


Fig. 4. Visualization of parametrization (13) in a three-node sensor network with a line topology.

edges of the network graph. One can make the network virtually fully connected by letting nodes act as relays and so eventually provide every node with all observations of $\mathbf{z}_k$, as shown in Fig. 3(a). However, this is not scalable in terms of communication bandwidth and computational power, and the routing of the data streams can become very complex for large networks. A more desirable approach is to let each node transmit linear combinations of all its inputs, i.e., its own sensor signal observations as well as the data provided by other nodes, as shown in Fig. 3(b). We will first describe a straightforward fusion rule, and we will point out that this approach is problematic if the network contains cycles, since this introduces feedback components. We will then explain how this feedback can be avoided, which will lead to the tree-DANSE algorithm in Section V.

A. A Straightforward Fusion Rule

To pass information from node to node without increasing the communication bandwidth, one can apply the same DANSE_K algorithm as in the previous section, but now let each node k transmit the observations of the K-channel signal

$$\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k + \sum_{q \in \mathcal{N}_k} \mathbf{G}_{kq}^H \mathbf{z}_q \tag{12}$$

to its neighbors, with $\mathcal{N}_k$ denoting the set of nodes that are connected to node k, node k excluded. The $\mathbf{z}_k$ signal is then a fused signal containing all the input signals of node k. Notice that this $\mathbf{z}_k$ corresponds to the node-specific estimated signal of node k, i.e., $\mathbf{z}_k = \bar{\mathbf{d}}_k$. This is illustrated in Fig. 4 for a three-node network with a line topology. From this figure, it can be seen that the implicit definition of the $\mathbf{W}_k$, that is applied to $\mathbf{y}$ to generate $\bar{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y}$, is then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{O} \\ \mathbf{W}_{kk} \\ \mathbf{O} \end{bmatrix} + \sum_{q \in \mathcal{N}_k} \mathbf{W}_q \mathbf{G}_{kq} \tag{13}$$

with $\mathbf{O}$ denoting an all-zero matrix of appropriate dimension. Parametrization (13) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously. We assume that the matrices $\mathbf{G}_{kq}$ are all-zero matrices, or do not exist, if there is no connection between node k and node $q \notin \mathcal{N}_k$.

In [16], it is pointed out that the parametrization (13) has an impact on the dynamics of the algorithm, but also on the solution space. Unfortunately, if (1) holds, i.e., if the desired signals share a K-dimensional latent signal subspace,[8] the algorithm cannot obtain the optimal estimators (5), which was proven in [16] for a two-node network. In the following theorem, we proof the general statement.

Theorem IV.1: Consider a network with any topology. If (1) holds, then the optimal estimators $\hat{\mathbf{W}}_k$ given by (5) are not in the solution space defined by parametrization (13).

Proof: We prove the theorem by contradiction, so we assume that the optimal centralized solution $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, is in the solution space defined by (13). By substituting (10) in parametrization (13) for $k = 1$ (w.l.o.g.), we obtain

$$\begin{bmatrix} \mathbf{O}_{M_1 \times K} \\ \hat{\mathbf{W}}_{12} \\ \vdots \\ \hat{\mathbf{W}}_{1J} \end{bmatrix} = \sum_{q \in \mathcal{N}_k} \begin{bmatrix} \hat{\mathbf{W}}_{11} \\ \hat{\mathbf{W}}_{12} \\ \vdots \\ \hat{\mathbf{W}}_{1J} \end{bmatrix} \mathbf{A}_{q1} \mathbf{G}_{1q} \tag{14}$$

where $\mathbf{O}_{M_1 \times K}$ denotes an all-zero $M_1 \times K$ matrix. From the first M_1 rows in (14), we obtain

$$\mathbf{O}_{M_1 \times K} = \hat{\mathbf{W}}_{11} \sum_{q \in \mathcal{N}_k} \mathbf{A}_{q1} \mathbf{G}_{1q}. \tag{15}$$

From the last $M - M_1$ rows in (14), we obtain[9]

$$\mathbf{I}_K = \sum_{q \in \mathcal{N}_k} \mathbf{A}_{q1} \mathbf{G}_{1q}. \tag{16}$$

Combining (15) and (16) yields $\hat{\mathbf{W}}_{11} = \mathbf{O}_{M_1 \times K}$, meaning that the sensor signals of node 1 are not used in the centralized solution (5). Instead of choosing $k = 1$, we can use a similar reasoning for all $k \in \mathcal{J}$ to eventually find that $\hat{\mathbf{W}}_k = \mathbf{O}_{M \times K}$, $\forall\, k \in \mathcal{J}$, which contradicts (5). ■

[8]Remarkably, if (1) does not hold, the solution space defined by parametrization (13) contains the same estimators as when using parametrization (9) [16]. However, the optimal solution (5) cannot be achieved in this case since it cannot be parametrized by (9).

[9]We implicitly assume that the submatrix $[\hat{\mathbf{W}}_{12}^T \ldots \hat{\mathbf{W}}_{1J}^T]^T$ of the optimal centralized estimator given by (5), has full rank. Although this is not fully guaranteed by the imposed assumptions, this is satisfied in most practical cases since $M - M_1 \gg K$, and because $\mathbf{R}_{yy}^{-1}$ and the stacked $M_k \times K$ submatrices of $\mathbf{R}_{yd_k}$ in (5) have full rank (due to spatial diversity of the sensors).

The fundamental problem with parametrization (13) is the feedback in the signal paths. Indeed, the observations of $\mathbf{z}_q$ that node k receives from neighboring nodes $q \in \mathcal{N}_k$ also contain a component with the sensor signal observations $\mathbf{y}_k$ of node k itself. This results in a solution space for the DANSE_K algorithm that does not contain the optimal estimators (5), as pointed out by Theorem IV.1.

We will distinguish between two forms of feedback: direct and indirect feedback. Direct feedback is caused by the feedback path from node k to a neighboring node q and back to node k, i.e., a cycle of length two. In Section IV-B, we show that this type of feedback can be easily controlled. Indirect feedback is more difficult to deal with. It occurs when data transmitted by node k travels through a path in the network, containing more than two different nodes, and eventually arrives back at node k. In Section IV-C, we will explain that indirect feedback can be avoided by removing direct feedback and by pruning the network to a tree topology.

B. Direct Feedback Cancellation

To avoid direct feedback, each node can send different data to each of its neighbors instead of locally broadcasting (12). Let $\mathbf{z}_{kq}$ denote the signal of which observations are transmitted from node k to node q, then direct feedback is avoided by choosing

$$\begin{aligned}\forall\, k \in \mathcal{J}, \forall\, q \in \mathcal{N}_k :&\\ \mathbf{z}_{kq} &= \mathbf{W}_{kk}^H \mathbf{y}_k + \sum_{l \in \mathcal{N}_k \backslash \{q\}} \mathbf{G}_{kl}^H \mathbf{z}_{lk} \quad (17)\\ &= \mathbf{z}_k - \mathbf{G}_{kq}^H \mathbf{z}_{qk}. \quad (18)\end{aligned}$$

We will refer to this strategy as "transmitter feedback cancellation" (TFC), since the direct feedback at node q is cancelled by the transmitting node k. We will refer to the signals defined in (17) as TFC-signals. It is noted that expression (17) provides an implicit definition of the TFC-signals, and that it is difficult to obtain a general closed form expression due to the remaining indirect feedback.

The TFC strategy matches perfectly with a point-to-point communication protocol, in which each individual node pair has a reserved communication link. In this case, direct feedback can be avoided without an increase in communication bandwidth. However, when the communication protocol supports local broadcasting, a more efficient strategy is possible, based on expression (18), which we will describe in Section VI-B. This strategy will be referred to as "receiver feedback cancellation" (RFC). However, from a theoretical point of view, the TFC and RFC strategies are equivalent. For the sake of an easy exposition, we use the former in the theoretical analysis.

C. Removal of Indirect Feedback

As mentioned in Section IV-B, direct feedback can be easily removed. Unfortunately, indirect feedback is more difficult to avoid. However, if direct feedback is removed, the data diffuses through the network in one direction, i.e., data sent by node k over a specific edge of the network graph cannot return to node k through the same edge in opposite direction, and so it can only return through a different edge that is part of a cycle. Hence, if the network has a tree topology, i.e., the network graph contains no cycles, indirect feedback is automatically removed if direct feedback is removed. In the sequel, we assume that the network graph has been pruned to a spanning tree of the initial graph. Optimal spanning trees can be defined and computed in several ways. An overview of different spanning tree problems can be found in [19].

It is noted that the combination of TFC with a tree topology has some similarities with the message passing for belief propagation (BP) in trees (see, e.g., [20]). Furthermore, in Section VI-A, we will decompose the signal flow in an inwards fusion and an outwards diffusion flow, which is also similar in BP. Despite these strong similarities in the data flow, the estimation frameworks of both algorithms are very different and incomparable, even on a higher level of abstraction.

V. DANSE IN A NETWORK WITH A TREE TOPOLOGY (T-DANSE)

In this section, we will extend the DANSE_K algorithm, to operate in networks with a tree topology. We will refer to this as tree-DANSE_K or T–DANSE_K. In the sequel, we will often refer to Fig. 5, showing an example network with a tree topology.

A. T–DANSE_k *Algorithm*

A node k transmits observations of the K-channel TFC-signal $\mathbf{z}_{kq}$, defined by (17), to a node $q \in \mathcal{N}_k$. Fig. 6 illustrates this for a network graph with a line topology, which is a subgraph of the network graph in Fig. 5.

It is noted that a tree topology defines a unique path between any pair of nodes, if an edge can only be used once. Let $P_{p_1 \to p_t} = (p_1, p_2, \dots, p_{t-1}, p_t)$ denote the ordered set of nodes defining the unique path from node p_1 to node p_t, and let $P_{p_1 \leftarrow p_t}$ denote the inverse path, i.e., $P_{p_1 \leftarrow p_t} = P_{p_t \to p_1}$. Define

$$\mathbf{G}_{p_1 \leftarrow p_t} = \mathbf{G}_{p_{t-1}p_t} \mathbf{G}_{p_{t-2}p_{t-1}} \dots \mathbf{G}_{p_2p_3} \mathbf{G}_{p_1p_2} \quad (19)$$

with p_j denoting the jth node in the path $P_{p_1 \to p_t}$. We define $\mathbf{G}_{k \leftarrow k} = \mathbf{G}_{kk} = \mathbf{I}_k$. The order of the $\mathbf{G}$'s in (19) must be the same as the order of the nodes in the inverse path $P_{p_1 \leftarrow p_t}$.

Example: The matrix $\mathbf{G}_{1 \leftarrow 8}$ for the network graph depicted in Fig. 5 is $\mathbf{G}_{1 \leftarrow 8} = \mathbf{G}_{48}\mathbf{G}_{34}\mathbf{G}_{13}$. This structure is clearly visible in the network graph of Fig. 6, defined by the path $P_{1 \leftarrow 8}$. Notice that $\mathbf{G}_{8 \leftarrow 1} = \mathbf{G}_{1 \to 8} = \mathbf{G}_{31}\mathbf{G}_{43}\mathbf{G}_{84}$.

The parametrization of the $\mathbf{W}_k$ effectively applied at node k, to generate $\overline{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y}$, is then

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k \leftarrow 1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{k \leftarrow J} \end{bmatrix}. \quad (20)$$

Parametrization (20) defines a solution space for all $\mathbf{W}_k$, $k \in \mathcal{J}$, simultaneously, that depends on the network topology. Notice that its structure is very similar to (9), as used in fully connected DANSE.

Theorem V.1: If (1) holds, then the optimal estimators $\hat{\mathbf{W}}_k$ given in (5) are in the solution space defined by parametrization (20).

Proof: The proof is based on expression (10), which follows from (1) and (5), and which is repeated here for convenience:

$$\forall\, k, q \in \mathcal{J} : \hat{\mathbf{W}}_k = \hat{\mathbf{W}}_q \mathbf{A}_{kq} \quad (21)$$

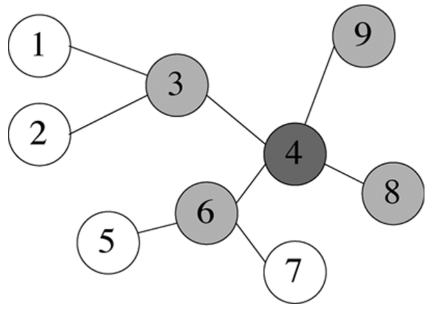


Fig. 5. Example of a network graph with tree topology with nine sensor nodes.

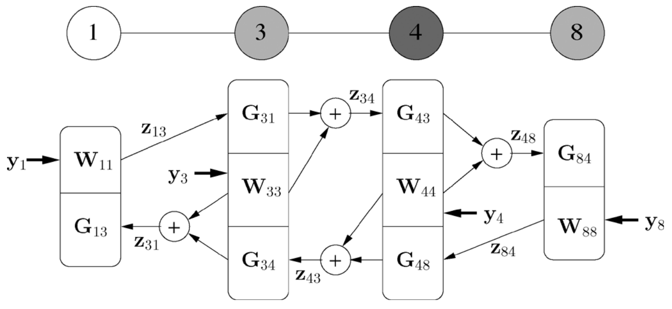


Fig. 6. The T–DANSE_K scheme in a network graph with a line topology.

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k^H$. By setting all $\mathbf{G}_{kq}$ matrices equal to $\mathbf{G}_{kq} = \mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k^H$, we automatically have that $\mathbf{G}_{k\leftarrow l} = \mathbf{A}_{kl}$ for any k and l, since $\mathbf{A}_{nl}\mathbf{A}_{kn} = \mathbf{A}_{kl}$, for any k,l and n. By using a similar reasoning as in the proof of Theorem III.1, we can show that all $\hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, are in the solution space defined by parametrization (20). ■

Let the matrix $\mathbf{G}_{k,-q}$ denote the stacked version of all $\mathbf{G}_{kn}$ matrices for which $n \in \mathcal{N}_k\backslash\{q\}$. Vector $\mathbf{z}_{\rightarrow k}$ denotes the vector in which all K-channel signals $\mathbf{z}_{qk}$ are stacked, for all $q \in \mathcal{N}_k$, i.e., it contains all signals that node k receives from its neighbors. Let P denote an ordered set of nodes that contains all nodes in the network, possibly with repetition of nodes. Let p_j denote the jth element in this set and let $|P|$ denote the number of elements in P. In general, we will use X^i to denote X at iteration i, where X can be a signal or a parameter.

The T–DANSE_K algorithm then consists of the following steps:

The T–DANSE$_K$ Algorithm

1) • Initialize $\mathbf{W}_{qq}^0$ and $\mathbf{G}_{q,-q}^0$, $\forall\, q \in \mathcal{J}$, as random matrices.
• $i \leftarrow 0$.
• $k \leftarrow p_1$.

2) • Node k updates its local parameters $\mathbf{W}_{kk}^i$ and $\mathbf{G}_{k,-k}^i$ by minimizing its MSE criterion, based on observations of its own inputs sensor signal $\mathbf{y}_k$ and of the compressed signals $\mathbf{z}_{qk}^i$, that it receives from nodes $q \in \mathcal{N}_k$:

$$\begin{bmatrix}\mathbf{W}_{kk}^{i+1}\\ \mathbf{G}_{k,-k}^{i+1}\end{bmatrix} = \underset{\mathbf{W}_{kk},\mathbf{G}_{k,-k}}{\arg\min}\; E\left\{\left\|\mathbf{d}_k - \left[\mathbf{W}_{kk}^H\ \mathbf{G}_{k,-k}^H\right]\begin{bmatrix}\mathbf{y}_k\\ \mathbf{z}_{\rightarrow k}^i\end{bmatrix}\right\|^2\right\}. \quad (22)$$

• The other nodes do not change their variables:

$$\forall\, q \in \mathcal{J}\backslash\{k\}:\ \mathbf{W}_{qq}^{i+1} = \mathbf{W}_{qq}^i,\ \mathbf{G}_{q,-q}^{i+1} = \mathbf{G}_{q,-q}^i. \quad (23)$$

3) • $i \leftarrow i+1$.

4) • $k \leftarrow p_t$ with $t = (i \bmod |P|) + 1$.

5) • Return to step 2

The estimate of a sample $\mathbf{d}_k[t]$ of the desired signal $\mathbf{d}_k$ at node k at any point in the iterative process is computed as

$$\bar{\mathbf{d}}_k[t] = \mathbf{W}_{kk}^{i\,H}\mathbf{y}_k[t] + \sum_{q\in\mathcal{N}_k}\mathbf{G}_{kq}^{i\,H}\mathbf{z}_{qk}^i[t]. \quad (24)$$

Remark I: We emphasize again that the iterative nature of the algorithm does not mean that the same sensor signal observations are retransmitted after every iteration. In practical applications, iterations are spread over time, i.e., if $\mathbf{W}_{kk}^i$ is updated to $\mathbf{W}_{kk}^{i+1}$ at time t_0, this is only used to compress new observations $\mathbf{y}_k[t]$ and estimate the samples $\mathbf{d}_k[t]$ for which $t > t_0$, while previous observations for $t \leq t_0$ are neither recompressed, retransmitted nor re-estimated. Effectively, each sensor signal observation is compressed and transmitted only once. This is motivated by the stationarity assumption of the involved signals.

Remark II: In the T–DANSE_K algorithm, each node solves an estimation problem that is equivalent to the centralized problem (4), but on a much smaller scale, i.e., with less signals. This means that the computations are distributed over the nodes in the network.

Remark III: It is noted that, even when nodes only have access to observations of a single-channel sensor signal, i.e., $M_k = 1, \forall\, k \in \mathcal{J}$, the T–DANSE_K algorithm yields a benefit in terms of communication bandwidth efficiency, compared to the relay case depicted in Fig. 3(a). In the fully connected case, DANSE_K only yields an improvement in bandwidth efficiency if $M_k > K$ [12]. Furthermore, the T–DANSE_K algorithm is fully scalable, i.e., the amount of data transmitted between each node pair does not depend on the number of nodes J, and the computational effort at each node only depends on the number of neighboring nodes (but not on J). This is important since the number of signals that a single node can receive and process in real-time is usually limited, especially when operating at large sampling rates. Based on the complexity analysis in [12], we find that the computations at node k (in a recursive implementation) have a complexity of

$$O((M_k + K|\mathcal{N}_k|)^2). \quad (25)$$

This is to be compared with the complexity $O((M_k + K(J-1))^2)$ of fully connected DANSE, which depends on the total number of nodes J.

B. Convergence and Optimality

The following theorem provides a sufficient condition on the updating order P for T–DANSE_K to guarantee convergence to the optimal estimators.

Theorem V.2: Consider a network with a tree topology. Let P denote an ordered set of nodes that defines a path through the network that starts in k and ends in any $q \in \mathcal{N}_k$, such that $\forall\, p \in \mathcal{J} : p \in P$. If (1) holds, then the T–DANSE$_K$ algorithm as described in Section V-A converges for any initialization of its parameters to the linear MMSE solution (5) for all k.

Proof: The proof of this theorem is elaborate, and can be found in Appendix A. Some additional concepts and lemmas are used in the proof, which can be found in Appendix A and B. ■

Theorem V.2 states that the updating order of the nodes must correspond to a path through the network. This means that if node k updates in iteration i, then the node that updates in iteration $i+1$ must be in $\mathcal{N}_k$. For example, for the network in Fig. 5, a possible choice for P is $P = (1,3,2,3,4,9,4,8,4,6,5,6,7,6,4,3)$.

Remark I: Extensive simulations show that the condition in Theorem V.2 on the updating order P is sufficient, but not necessary. In fact, the algorithm is always observed to converge, regardless of the updating order of the nodes (see also Section VII). This is stated here as an observation since a proof is not yet available. However, choosing an updating order satisfying the condition in Theorem V.2 usually results in a faster convergence for most of the nodes. This can be explained by the fact that this condition generally implies that nodes with many neighbors are updated more often. Since these nodes act as bottlenecks in the data diffusion, it is important that these are indeed updated frequently, whereas updates of nodes at the boundary of the network have less impact on the data flow in the rest of the network.

Remark II: The proof of convergence in Appendix A shows that the algorithm is at least as fast as a centralized alternating optimization (AO) algorithm or block-coordinate descent method (a.k.a. nonlinear Gauss-Seidel iteration [21]), where alternating blocks of variables are optimized while the other variables are fixed. The T–DANSE$_K$ algorithm is usually even faster, since none of the variables are actually fixed, but rather constrained to a subspace. Since an AO method converges Q-linearly for strictly convex objective functions [22], we can conclude that T–DANSE$_K$ converges at least Q-linearly. The convergence proof in Appendix A also shows that the MSE monotonically decreases at node k (when evaluated after each update of node k). When the network grows, convergence takes longer due to the sequential nature of the method, i.e., it takes more iterations to have a full round of updates. When nodes would update simultaneously, the convergence time usually scales much better with the network size, but convergence cannot be guaranteed anymore. Relaxation methods, similar to the ones applied in [15] may again yield convergence in this case, but this is beyond the scope of this paper.

Remark III: For the sake of an easy exposition, we have ignored clock- and transmission delays in the signal paths. However, if proper compensating delays are added at the right places in the signal path, the theoretical analysis above is not affected by this practical aspect. Section VI also describes a data-driven computation of the sample estimates, which has the advantage that nodes do not need any information on the transmission delays.

Remark IV: In the special case where all the nodes estimate the same signal, i.e., $\mathbf{d}_k = \mathbf{d}, \forall\, k \in \mathcal{J}$, the convergence properties remain the same as in the scenario with node-specific estimation problems, i.e., the fact that different signals are estimated at each node does not affect the convergence speed of the algorithm. This straightforwardly follows from the proof of Theorem V.2.

VI. T-DANSE WITH LOCAL BROADCASTING

In this section, we describe how T-DANSE can operate in a WSN that allows local broadcasting to neighboring nodes, instead of using reserved communication links between node pairs. By describing the system as a data-driven feed-forward data flow, we will be able to transform the TFC procedure into another feedback cancellation procedure that exploits the higher communication bandwidth efficiency of a local broadcast communication protocol. However, these practical aspects have no effect on the iterations of the T-DANSE algorithm, and therefore the theoretical analysis of the previous section remains valid.

For reasons that will become clear, we will first group the different nodes of the network in subsets, which we will refer to as 'shells' of the network. Let h_k denote the maximum number of hops between node k and any other node in the network, and let $h_{\min} = \min_{k\in\mathcal{J}} h_k$ and $h_{\max} = \max_{k\in\mathcal{J}} h_k$. We define $N = h_{\max} - h_{\min}$. Let S_N denote the subset of nodes that form the outer shell of the network, i.e., $S_N = \{k : h_k = h_{\min} + N\}$. Similarly, we define the shells

$$S_n = \{k : h_k = h_{\min} + n\}, \quad n = 0 \ldots N. \qquad (26)$$

The inner shell S_0 contains maximally 2 nodes, which we refer to as the root nodes. It is noted that the nodes in the outer shell S_N are leaf nodes, i.e., nodes with a single neighbor, but S_N does not necessarily contain all the leaf nodes of the network.

Example: The shells of the network depicted in Fig. 5 are $S_2 = \{1,2,5,7\}$ (white), $S_1 = \{3,6,8,9\}$ (light grey) and $S_0 = \{4\}$ (dark grey).

In the sequel, we assume that each node knows the shell index to which it belongs, together with the shell indices of its neighboring nodes. This requires some upper layer protocol or coordination.

A. Data-Driven Computation of TFC-Signals

The T-DANSE algorithm uses the TFC-signals $\mathbf{z}_{kq}$ as defined in (17), which is an implicit definition. This signal definition is illustrated in Fig. 7 for node 3 of the graph depicted in Fig. 5. Because the network is assumed to have a tree topology, (17) can be easily solved for the $\mathbf{z}_{kq}$'s by backwards substitution, where the leaf nodes act as starting points. Indeed, if k is a leaf node, then $\mathbf{z}_{kq} = \mathbf{W}_{kk}^H \mathbf{y}_k$, which does not contain contributions from any other node. This backwards substitution defines causality constraints, and can be described by means of a data-driven signal flow graph, where elementary building blocks (as the one depicted in Fig. 7) are interconnected. Each internal operator is triggered when it has received a sample or a data packet on each of its input lines, generating a new packet of data on its output line. This description can also be used in a practical implementation, and has the advantage that nodes do not need any information on the transmission delays (which is particularly interesting in communication networks with variable transmission delays).

Furthermore, the chain of computations in this data-driven procedure will show that the signal flow naturally decomposes in an inwards flow followed by an outwards flow. To this end, an important observation is that any non-root node k has only one neighbor q that is in a deeper shell, i.e., $\forall\, n \in \{1, \ldots, N\}, \forall\, k \in$

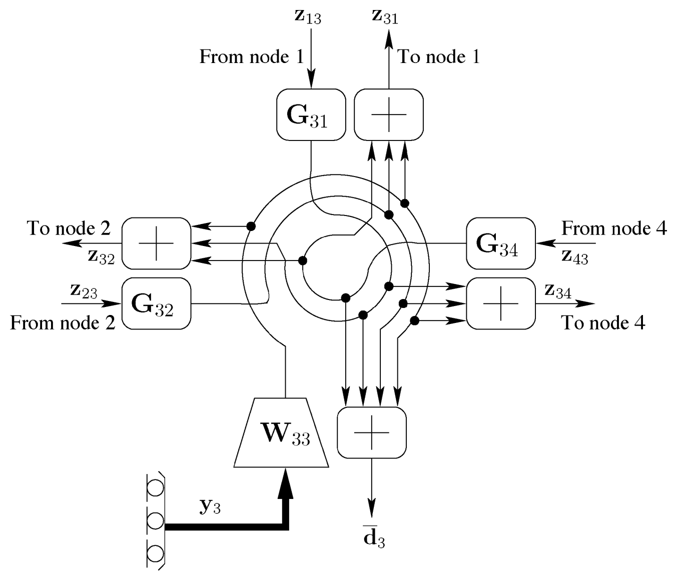


Fig. 7. Illustration of the internal data flow in node 3 of the network depicted in Fig. 5.

S_n : $|\mathcal{N}_k \cap S_{n-1}| = 1$. The data flow is then decomposed as follows:

1) **Fusion flow** $(S_N \to S_0)$: The fusion flow is initiated by the leaf nodes, who fire immediately after the collection of a new sensor observation. A non-leaf node fires when it has received data from all of its neighbors in the outer shell. At this event, the node fuses and forwards this data to the single neighbor in the deeper shell that has not fired yet. In this way, the data travels inwards from S_N to S_0, where it is fused on the way with other local sensor data, and eventually arrives at the root node(s). If there are 2 root nodes, they have to exchange observations of the TFC signals, such that each of them has access to (fused) data that contains all sensor observations.
2) **Diffusion flow** $(S_0 \to S_N)$: After arrival of the fusion flow in S_0, the root nodes initiate the diffusion flow by firing to all their neighbors in the outer shell, providing each with its node-specific TFC-signal. The neighbors at their turn do the same until the data is spread out over the entire network.

Remark I: In practice, the fusion flow and diffusion flow can run simultaneously, be it with a relative time lag. In other words, a leaf node k can fire each time it collects a new sensor observation $\mathbf{y}_k[t]$, even though the previous estimation sample $\bar{\mathbf{d}}_k[t-1]$ cannot be computed yet due to the fact that the required data is still on its way through the network.

Remark II: It is noted that in the fusion flow, each node transmits only one TFC-signal, whereas in the diffusion flow, a node k has to transmit $\mathcal{N}_k - 1$ TFC-signals (and a single root node k will transmit $\mathcal{N}_k$ TFC-signals). This will be exploited in the next subsection to reduce the communication bandwidth.

B. Receiver Feedback Cancellation

In Section IV-B, it is explained how direct feedback can be avoided by transmitter feedback cancellation. Although this is an efficient strategy in point-to-point communication protocols, TFC is very inefficient if the communication protocol allows local broadcasting. A better strategy would then be to let a node broadcast the same signal to all of its neighbors, and let the receiving nodes themselves remove their node-specific feedback component. We will refer to this strategy as 'receiver feedback cancellation' (RFC). The natural choice for the broadcast signal would be to use the $\mathbf{z}_k$ as defined in (12). A node q that receives the signal $\mathbf{z}_k$ from node k can then remove its own feedback component,[10] using

$$\mathbf{z}_{kq} = \mathbf{z}_k - \mathbf{G}_{kq}^H \mathbf{z}_q. \tag{27}$$

However, (12) and (27) are implicit definitions, and it is not possible to compute the broadcast signals (12), $\forall\, k \in \mathcal{J}$, due to causality issues. Indeed, the computation of the sample $\mathbf{z}_k[t]$ based on (12) requires the samples $\mathbf{z}_q[t]$, $\forall\, q \in \mathcal{N}_k$, but $\mathbf{z}_q[t]$ cannot be computed by node q without the sample $\mathbf{z}_k[t]$, which results in a deadlock.

To resolve this deadlock, we have to combine TFC with RFC, since the former is computable. To this end, we redefine the signals $\mathbf{z}_k$, $\forall\, k \in \mathcal{J}$, with an explicit definition based on the TFC-signals:

$$\begin{array}{ll} \forall\, k \in \mathcal{J}\backslash\mathcal{L}: & \mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k + \sum_{l \in \mathcal{N}_k} \mathbf{G}_{kl}^H \mathbf{z}_{lk} \\ \forall\, k \in \mathcal{L}: & \mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k \end{array} \tag{28}$$

where $\mathcal{L}$ denotes the set of leaf nodes. We denote the $\mathbf{z}_k$ signals (on the left-hand side) as RFC-signals. By comparing (28) with (17), we find that

$$\begin{array}{ll} \forall\, k \in \mathcal{J}\backslash\mathcal{L}, \forall\, q \in \mathcal{N}_k: & \mathbf{z}_{kq} = \mathbf{z}_k - \mathbf{G}_{kq}^H \mathbf{z}_{qk} \\ \forall\, k \in \mathcal{L}, \forall\, q \in \mathcal{N}_k: & \mathbf{z}_{kq} = \mathbf{z}_k. \end{array} \tag{29}$$

If a receiving node $q \in \mathcal{N}_k$ would have access to the signals $\mathbf{z}_k$ and $\mathbf{z}_{qk}$, and the parameters $\mathbf{G}_{kq}$, then node q itself can compute the TFC-signal $\mathbf{z}_{kq}$ by using expression (29). Therefore, some of the TFC-signals will have to be broadcast together with the RFC-signals. The natural question is then how to organize this. The answer straightforwardly follows from the decomposition of the signal flow in a fusion and a diffusion flow, as explained in the previous subsection.

In the fusion flow, each node provides only one neighbor with signal observations, i.e., its single neighbor in the deeper shell. In this case, RFC cannot provide any benefit, and therefore TFC-signals are transmitted. In the diffusion flow on the other hand, node k will provide $\mathcal{N}_k - 1$ nodes with signal observations. In this case, however, a root node k can compute observations of the RFC-signal $\mathbf{z}_k$ according to (28), since it has access to observations of all the signals on the right-hand side (which are provided by the fusion flow). Node k then broadcasts observations of $\mathbf{z}_k$, and its neighbors in the outer shell can extract observations of their node-specific TFC-signal from the observations of $\mathbf{z}_k$, by using (29). A receiving node q, can then compute the observations of $\mathbf{z}_q$, similarly to (28), and broadcast this to its $\mathcal{N}_q - 1$ neighbors in the next shell. This is illustrated in Fig. 8 for a subgraph of the graph in Fig. 5.

When this RFC strategy is used, each node (except leaf nodes and single root nodes) transmits observations of 2 signals; a TFC-signal in the fusion flow and an RFC-signal in the diffusion flow. Hence, the amount of data that a node transmits is independent of the number of neighbors of that node.

[10]Here it is assumed that $\mathbf{G}_{kq}$ is known at node q, which requires some minor additional information exchange between nodes, assuming that the sampling rate of the sensors is significantly larger than the update frequency of the estimation parameter at the different nodes.

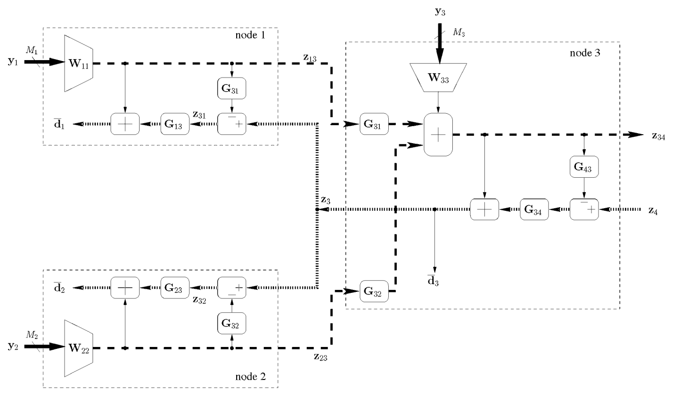


Fig. 8. Illustration of the fusion flow (dashed line) and the diffusion flow (dotted line) between nodes 1, 2, and 3 for broadcast communication in the network depicted in Fig. 5.

VII. Simulations

In this section, we provide batch mode simulation results for the T–DANSE$_3$ algorithm in networks with a tree topology. The first experiment is performed based on the network depicted in Fig. 5. The network contains 9 nodes $(J = 9)$, each having 10 sensors $(M = 90)$. The dimension of the latent signal subspace defined by $\mathbf{d}$ is $K = 3$. All three signals in $\mathbf{d}$ are uniformly distributed random processes on the interval $[-0.5, 0.5]$ from which $N = 10000$ samples are generated. All sensor measurements correspond to a random linear combination of the three generated signals to which zero-mean white noise is added with half the power of the signals in $\mathbf{d}$. The $\mathbf{W}_{kk}$ variables are initialized randomly, whereas the $\mathbf{G}_{kq}$ variables are initialized as all-zero matrices. All MMSE cost functions are replaced by their related least-squares (LS) cost functions, i.e., (for node k)

$$J_k(\mathbf{W}_k) = \sum_{t=0}^{N} \left\| \mathbf{d}_k[t] - \mathbf{W}_k^H \mathbf{y}[t] \right\|^2 . \tag{30}$$

The results are given in Fig. 9(a), showing the LS cost of node 1 (above) and the summed LS cost of all the nodes (below) versus the iteration index i. Notice that one iteration corresponds to the time needed for a node to estimate the statistics of its inputs and to calculate the new parameter setting. Three different cases are simulated. In the first case, the network is assumed to be fully connected, and the DANSE$_3$ algorithm of [12] is applied where the updating order is round-robin. In the second and third case, the network has the tree topology shown in Fig. 5 and the T–DANSE$_3$ algorithm is applied. In case 2, the updating order is $P_1 = (1, 3, 2, 3, 4, 9, 4, 8, 4, 6, 5, 6, 7, 6, 4, 3)$, which satisfies the condition of Theorem V.2, whereas in case 3 the updating order is $P_2 = (1, 2, \ldots, 9)$, i.e., round-robin, and so the condition of Theorem V.2 is not satisfied.

Remarkably, the updating order P_1 yields a faster convergence than P_2 at node 1, despite the fact that the update rate of node 1 is higher in the latter. As mentioned already in Section V-B, this holds for most of the nodes. If we only retain the iteration indexes in $(1, 17, 33, \ldots,)$, i.e., the iteration steps in which node 1 updates its parameters, then the cost function J_1 decreases monotonically for updating order P_1, as indicated by (42) in the proof of Theorem V.2. This does not hold in the round-robin case.

In the second experiment, T–DANSE$_3$ is applied in a nine-node network with a line topology, i.e., each node has exactly two neighbors, except for the 2 leaf nodes. The results are shown in Fig. 9(b). Here, we compare the updating order P_2 (round robin) with $P_3 = (1, 2, \ldots, 9, 8, 7, \ldots, 2)$, where the latter satisfies the conditions of Theorem V.2. The difference in convergence speed between updating order P_2 and P_3 is even more significant in this case, where the latter converges much faster than the round-robin updating order P_2.

In both plots, it is observed that the LS cost function J_1 may increase significantly, even after it nearly reached the optimal level. This is due to the fact that the other nodes did not yet achieve optimal estimation performance, yielding significant changes in their estimation parameters. Since the estimators of all the nodes are intertwined due to (20), these changes have a significant effect on the local cost function J_1 at node 1, resulting in an increase. However, after a couple of iterations, an equilibrium state is reached where each node has optimal performance. The reason why the cost J_1 remains almost constant in the initial iterations, is due to the fact that node 1 chooses small entries for $\mathbf{G}_{13}$ in the first iteration, and therefore node 1 is basically cut off from the network until its next update.

VIII. Conclusion

In this paper, we have extended the DANSE$_K$ algorithm, introduced in [12] and [13] for a fully connected sensor network, to the T–DANSE$_K$ algorithm which operates in a network with a tree topology. It is argued that feedback is to be avoided, when a straightforward modification of DANSE$_K$ is applied in a network that is not fully connected, since it harms the convergence and optimality properties of the algorithm. Direct feedback can be avoided easily, whereas indirect feedback is more difficult to remove in a network topology that has cycles. A tree topology is then a natural choice, since it has no cycles, for which the T–DANSE$_K$ algorithm can subsequently be derived. A condition is given on the updating order of the nodes to guarantee convergence to the optimal estimators. Simulations have shown that the condition on the updating order of the nodes is sufficient but not necessary, although convergence is faster if the condition is satisfied.

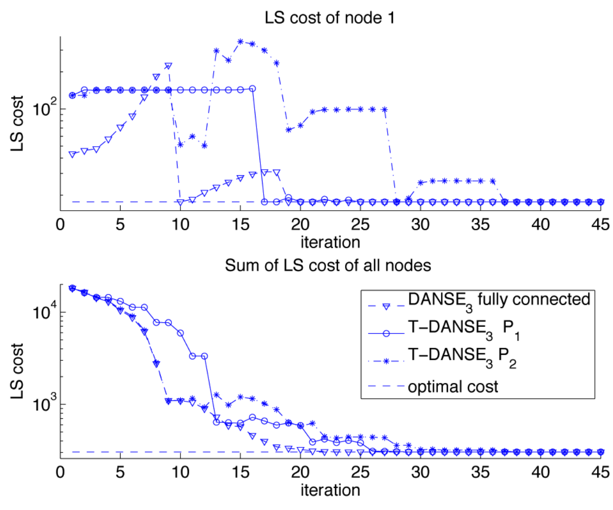
Two different communication protocols have been considered, i.e., point-to-point transmission between node pairs and local broadcasts. For both protocols it is possible to remove direct feedback, and so from a theoretical point of view, there is no true difference. However, the local broadcast communication protocol is more efficient and scalable in terms of connectivity, i.e., the amount of data that is transmitted is independent of the number of neighbors at each node.

Appendix

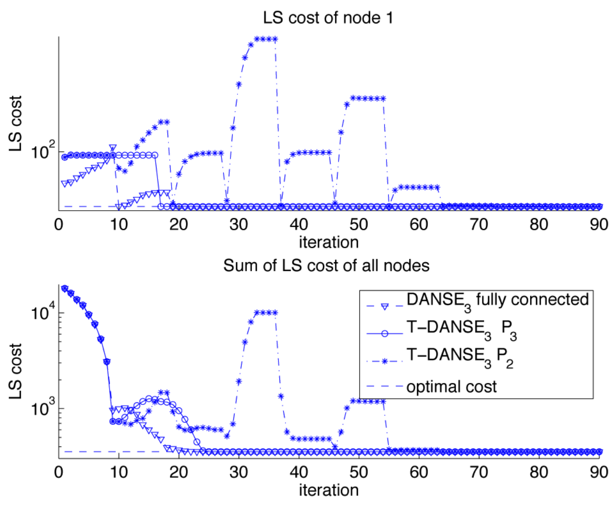
A. *Proof of Theorem V.2 (Convergence of* T–DANSE$_K$*)*

Before proving Theorem V.2, we first introduce some new concepts and lemmas. We will consider a partitioning $\mathcal{P}$, which is an ordered set of non-overlapping subsets of a set of nodes $\mathcal{J}$. The first subset of a partitioning $\mathcal{P}$ is referred to as the free subset, whereas the other subsets are referred to as the constrained subsets.

Example: For a five-node network a possible partitioning is $\mathcal{P} = (\{2\}; \{1, 4\}, \{3, 5\})$. $\{2\}$ is the free subset, and $\{1, 4\}$ and $\{3, 5\}$ are constrained subsets.



(a) Tree of Fig. 5



(b) Line topology

Fig. 9. The LS cost of node 1 and summed LS cost versus the number of iterations (a) in the network depicted in Fig. 5 and (b) in a nine-node network with line topology.

For a certain subset of nodes S, let $\mathbf{W}_{k|S}$ denote the stacked version of all the $\mathbf{W}_{kq}$'s (see Section III) for which $q \in S$. Assume that the estimator $\mathbf{W}_k$ is initialized with a certain matrix $\mathbf{W}_k^0$. We consider an updating scheme that updates the values in $\mathbf{W}_k$ by a sequence of n alternating optimizations[11] (AO), defined by a sequence of partitionings $(\mathcal{P}^i)_{i=0\ldots n-1}$. In each step i of the AO sequence, MMSE optimization (6) is performed, but with constraints added to the variables in the constrained subsets of $\mathcal{P}^i$. In the ith optimization step, the columns of $\mathbf{W}_{k|C}$ are constrained to the subspace defined by the columns of the current values of $\mathbf{W}_{k|C}^i$, where C is a constrained subset of $\mathcal{P}^i$. For $\mathcal{P}^i = (F^i; C_1^i, \ldots, C_{t_i})$, the corresponding AO update step is then

$$\begin{aligned}\left\{\mathbf{W}_{k1}^{i+1}, \ldots, \mathbf{W}_{kJ}^{i+1}, \mathbf{C}_1, \ldots, \mathbf{C}_{t_i}\right\} &= \underset{\{\mathbf{W}_{k1},\ldots,\mathbf{W}_{kJ},\mathbf{C}_1,\ldots,\mathbf{C}_{t_i}\}}{\arg\min} E\left\{\left\|\mathbf{d}_k - \sum_{l=1}^{J} \mathbf{W}_{kl}^H \mathbf{y}_l\right\|^2\right\} \\ \text{s.t.} &\begin{cases}\mathbf{W}_{k|C_1^i} = \mathbf{W}_{k|C_1^i}^i \mathbf{C}_1 \\ \vdots \\ \mathbf{W}_{k|C_{t_i}^i} = \mathbf{W}_{k|C_{t_i}^i}^i \mathbf{C}_{t_i}\end{cases}\end{aligned} \tag{31}$$

where $\mathbf{C}_1, \ldots, \mathbf{C}_{t_i}$ are $K \times K$ matrices.

Example: the optimization in the AO-step defined by $\mathcal{P}^i = (\{2\}; \{1,4\}, \{3,5\})$ is

$$\begin{aligned}\left\{\mathbf{W}_{k1}^{i+1}, \ldots, \mathbf{W}_{k5}^{i+1}, \mathbf{C}_1, \mathbf{C}_2\right\} &= \underset{\{\mathbf{W}_{k1},\ldots,\mathbf{W}_{k5}\}}{\arg\min} E\left\{\left\|\mathbf{d}_k - \sum_{l=1}^{5} \mathbf{W}_{kl}^H \mathbf{y}_l\right\|^2\right\} \\ \text{s.t.} \begin{bmatrix}\mathbf{W}_{k1} \\ \mathbf{W}_{k4}\end{bmatrix} &= \begin{bmatrix}\mathbf{W}_{k1}^i \\ \mathbf{W}_{k4}^i\end{bmatrix} \mathbf{C}_1, \begin{bmatrix}\mathbf{W}_{k3} \\ \mathbf{W}_{k5}\end{bmatrix} = \begin{bmatrix}\mathbf{W}_{k3}^i \\ \mathbf{W}_{k5}^i\end{bmatrix} \mathbf{C}_2.\end{aligned}$$

[11]The AO used in this paper is different from the AO algorithms (a.k.a. nonlinear Gauss-Seidel algorithms) described in [21], [22] in which the optimization is done over a subset of the variables, while other variables are fixed to their current value. In this paper, the optimization in an AO step is performed over all variables, but constraints are added to certain variables in an alternating fashion.

At first, we will consider the hypothetical case where all sensor signal observations are available to all nodes, and a node k can compute any AO-step on its full unparametrized $\mathbf{W}_k$. Later, we will consider the AO procedure that is related to the actual network, by linking the sequence of partitionings $(\mathcal{P}^i)_{i=0\ldots n-1}$ to the network topology. To analyze this AO process, we will first analyze the convergence properties in a two-node network where the AO steps of both nodes are linked, based on the sensors to which each node has access. Based on this results, we will prove the convergence of the T–DANSE$_K$ algorithm by making a hierarchical decomposition of the entire network into simpler two-node networks.

In the sequel, we will refer to the MSE cost function of node k, as given in (31), with $J_k(\mathbf{W}_k) = J_k([\mathbf{W}_{k1}^T \ldots \mathbf{W}_{kJ}^T]^T)$. Notice that for any AO-step performed on $J_k(\mathbf{W}_k)$:

$$J_k\left(\mathbf{W}_k^{i+1}\right) \leq J_k\left(\mathbf{W}_k^i\right) \tag{32}$$

and because the optimization problem (31) is strictly convex[12]:

$$\mathbf{W}_k^{i+1} \neq \mathbf{W}_k^i \Leftrightarrow J_k\left(\mathbf{W}_k^{i+1}\right) < J_k\left(\mathbf{W}_k^i\right). \tag{33}$$

Lemma A.1: Consider an arbitrary AO sequence defined by a sequence of partitionings $(\mathcal{P}^i)_{i=0\ldots n-1}$ with $\mathcal{P}^i = \{F^i; C_1^i, \ldots, C_{t_i}^i\}$. This AO sequence is simultaneously applied to the cost function $J_k(\mathbf{W}_k)$ of node k to update $\mathbf{W}_k^i$ and to the cost function $J_q(\mathbf{W}_q)$ of node q to update $\mathbf{W}_q^i$. Assume that the two initial centralized estimators $\mathbf{W}_k^0$ and $\mathbf{W}_q^0$ are related through $\mathbf{W}_{q|C_j^0}^0 = \mathbf{W}_{k|C_j^0}^0 \mathbf{G}_j$, $\forall\, j \in \{1, \ldots, t_0\}$, where $\mathbf{G}_j$ is a non-singular $K \times K$ matrix. If (1) is satisfied, then the following holds for any $i \in \{1, \ldots, n\}$:

$$\mathbf{W}_k^i = \mathbf{W}_q^i \mathbf{A}_{kq} \tag{34}$$

with $\mathbf{A}_{kq} = \mathbf{A}_q^{-H} \mathbf{A}_k^H$.

Proof: See Appendix B. ■

[12]Strict convexity is satisfied if the sensor measurements $\mathbf{y}_k$ are not perfectly correlated, which is always satisfied in practice due to sensor noise.

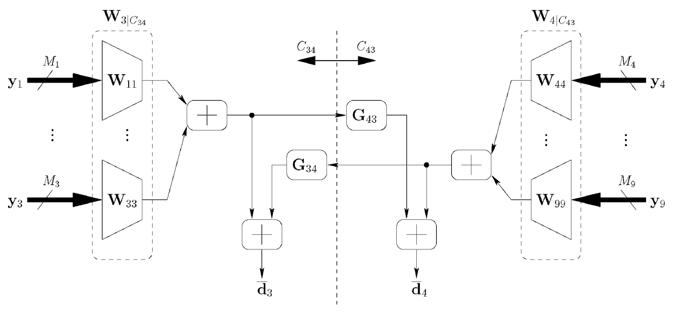


Fig. 10. The hypothetical two-node network for the edge (3, 4) in the network depicted in Fig. 5.

Lemma A.1 shows that all the (centralized) AO-steps at the different nodes (minimizing different cost functions) result in $\mathbf{W}_k$'s that have the same K-dimensional column subspace, if they are initialized properly.

In a network with a tree topology, there always exists a unique graph cut that cuts the edge (k, q) between two neighboring nodes k and q, and no other edges. This cut partitions the graph in two complementary sets of nodes: C_{kq} denotes the one that contains node k, and C_{qk} denotes the one that contains node q.

Consider a hypothetical two-node network with nodes k and q, where node k has access to the sensor signal observations $\mathbf{y}_{C_{kq}}$ and node q has access to the sensor signal observations $\mathbf{y}_{C_{qk}}$, where $\mathbf{y}_{C_{kq}}$ and $\mathbf{y}_{C_{qk}}$ denote the stacked version of all $\mathbf{y}_j$ for which $j \in C_{kq}$ and $j \in C_{qk}$, respectively. We now parametrize a subset of $\mathbf{W}_k$ and $\mathbf{W}_q$ as follows:

$$\mathbf{W}_{k|C_{qk}} = \mathbf{W}_{q|C_{qk}}\mathbf{G}_{kq} \tag{35}$$

$$\mathbf{W}_{q|C_{kq}} = \mathbf{W}_{k|C_{kq}}\mathbf{G}_{qk}. \tag{36}$$

It is assumed that node k has access to the variables $\mathbf{W}_{k|C_{kq}}$ and $\mathbf{G}_{kq}$, and that node q has access to the variables $\mathbf{W}_{q|C_{qk}}$ and $\mathbf{G}_{qk}$, as it is the case in the DANSE_K algorithm with parametrization (9).

Example: Consider the network in Fig. 5. Let $k = 3$ and $q = 4$, then $C_{34} = \{1, 2, 3\}$ and $C_{43} = \{4, 5, 6, 7, 8, 9\}$, and the two-node network corresponding to (35)–(36) is shown in Fig. 10.

The parametrization of the hypothetical two-node network as described above corresponds to partitionings $\mathcal{P}$ that satisfy a specific form. Indeed, node k can never freely manipulate the variables in $\mathbf{W}_{k|C_{qk}}$, and therefore any AO-step performed by node k must have partitionings of the form

$$\mathcal{P}_{k\leftarrow q} = \{(F; C_1, \ldots, C_t) : \exists\, n \in \{1, \ldots, t\} : C_{qk} \subseteq C_n\}. \tag{37}$$

Expression (37) implies that, during any optimization step at node k, the search space of the variables in $\mathbf{W}_{k|C_{qk}}$ is constrained to the current column space of $\mathbf{W}^i_{k|C_{qk}}$, which is due to (35) and the fact that node k can only change $\mathbf{W}_{k|C_{qk}}$ through $\mathbf{G}_{kq}$. Similarly, an AO-step performed by node q must have partitionings of the form $\mathcal{P}_{q\leftarrow k}$, which is also given by (37) (by switching k and q). It is noted that, even though it is assumed that node k can directly manipulate all entries in $\mathbf{W}_{k|C_{kq}}$, we do not assume that $\mathcal{P}_{k\leftarrow q} = (C_{kq}; C_{qk})$. The set C_{kq} can be divided in a free subset together with one or more constrained subsets. The latter can even be merged with C_{qk}.

Example: Consider the network in Fig. 5. Let $k = 4$ and $q = 3$, then a possible partitioning $\mathcal{P}_{4\leftarrow 3}$ is $\mathcal{P}_{4\leftarrow 3} = (\{5, 6, 7\}; \{1, 2, 3, 4\}, \{8, 9\})$. Notice that $C_{34} = \{1, 2, 3\}$ is a subset of one of the constrained subsets of $\mathcal{P}_{4\leftarrow 3}$, as defined by (37).

We will consider AO sequences that are computed by the two nodes in this hypothetical network as follows. If the partitioning is of the form $\mathcal{P}_{k\leftarrow q}$, then the AO step (31) is performed by node k where the optimization is with respect to cost function $J_k(\mathbf{W}_k)$. This node k has direct access to the vector $\mathbf{W}_{k|C_{kq}}$, although it may also be constrained by other constrained sets in $\mathcal{P}_{k\leftarrow q}$. The variables in $\mathbf{W}_{k|C_{qk}}$ are parameterized as in (35) and manipulated through the $\mathbf{G}_{kq}$ matrix. Similarly, if the partitioning is of the form $\mathcal{P}_{q\leftarrow k}$ node q will optimize its parameters $\mathbf{W}_{q|C_{qk}}$ and $\mathbf{G}_{qk}$ with respect to cost function $J_q(\mathbf{W}_q)$. It is noted that, due to (35)–(36), an update of $\mathbf{W}_{k|C_{kq}}$ also changes $\mathbf{W}_{q|C_{kq}}$ and an update of $\mathbf{W}_{q|C_{qk}}$ changes $\mathbf{W}_{k|C_{qk}}$.

Lemma A.2: Consider a network with tree topology, and consider the hypothetical two-node network defined by the edge (k, q) as explained above, with the corresponding linked parametrization of $\mathbf{W}_k$ and $\mathbf{W}_q$ as defined by (35)–(36). Consider an AO sequence of n steps defined by a sequence of partitionings $\left(\mathcal{P}^0_{k\leftarrow q}, (\mathcal{P}^i_{q\leftarrow k})_{i=1\ldots(n-2)}, \mathcal{P}^{n-1}_{k\leftarrow q}\right)$, where the first and last AO step are performed by node k and the others by node q. Assume that in the first and last AO step, the set C_{qk} is a constrained subset as such, without additional nodes. If (1) is satisfied, then the resulting $\mathbf{W}^n_k$ will be the same as if node k had access to all sensor signal observations and performed all optimizations in the AO sequence by itself with respect to its own cost function J_k.

Proof: This lemma is a straightforward consequence of Lemma A.1, which shows that any $\mathbf{W}^i_q$ that results from an AO sequence with respect to $J_q(\mathbf{W}_q)$, is the same as $\mathbf{W}^i_k$ that results from the same AO sequence with respect to $J_k(\mathbf{W}_k)$, except for a transformation by the matrix $\mathbf{A}_{kq}$. The latter can be compensated by the $\mathbf{G}$'s in parametrization (35)–(36). Since node k performs the last AO step, and since C_{qk} is a constrained subset, $\mathbf{G}^n_{kq}$ indeed compensates for this transformation with respect to $\mathbf{W}^{n-1}_{q|C_{qk}}$. Since C_{qk} is a constrained subset in the initial AO step, the assumption in Lemma A.1 on the initialization of the parameters is automatically satisfied by the parametrization (35)–(36). ■

We now define a one-to-one correspondence between a node k and a partitioning $\mathcal{P}$ as follows:

$$k \leftrightarrow \mathcal{P} = (\{k\}; C_{\mathcal{N}_k}) \tag{38}$$

with $C_{\mathcal{N}_k}$ denoting the set containing all the sets C_{jk} for which $j \in \mathcal{N}_k$.

Example: In the case of the network depicted in Fig. 5, we have that $4 \leftrightarrow (\{4\}; \{1, 2, 3\}, \{5, 6, 7\}, \{8\}, \{9\})$.

The correspondence (38) defines another correspondence between a path P over $n-1$ edges through the network, and an n-step AO procedure defined by the sequence of partitionings

$$P \leftrightarrow (\mathcal{P}^i)_{i=0\ldots n-1}. \tag{39}$$

In the sequel, we assume that $\mathbf{W}_k, \forall\, k \in \mathcal{J}$, is parametrized according to (20), and that the AO step with partitioning of the form (38) is applied to the cost function $J_k(\mathbf{W}_k)$. Notice that an update of node k by the T–DANSE$_K$ algorithm is equivalent to such an AO step. Indeed, the update of $\mathbf{W}_{kk}$ and $\mathbf{G}_{k-k}$ is defined by a constrained optimization over the variables in $\mathbf{W}_k$, where $\mathbf{W}_{kk}$ are unconstrained variables (free subset) and where the matrix $\mathbf{G}_{k-k}$ is used to manipulate the constrained variables in the constrained subsets of (38). The T–DANSE$_K$ algorithm thus performs the AO sequence based on the sequence of partitioning defined by (39), in which the actual cost function and optimization variables change along with the corresponding node that is actually updating. It is noted that, even though an AO step at node k is defined on the variable $\mathbf{W}_k$, the resulting update of $\mathbf{W}_{kk}$ and $\mathbf{G}_{k-k}$ has an indirect influence on the other variables $\mathbf{W}_q$ with $q \neq k$, since they are linked through parametrization (20).

Lemma A.3: Consider a network with a tree topology and a path $P_{k\to k}$ through this network with length $n-1$, that never passes through node k, except at the start and at the end. Consider the T–DANSE$_K$ updating sequence equivalent to the n-step AO sequence defined by the partitionings $(\mathcal{P}^i)_{i=0\ldots n-1} \leftrightarrow P_{k\to k}$. If (1) is satisfied, then the resulting $\mathbf{W}_k^n$, parameterized by (20), will be the same as if node k had access to all sensor signal observations and performed all optimizations in the AO sequence by itself with respect to its own cost function J_k.

Proof: This lemma can be proven by recursively applying Lemma A.2 on the different edges that are visited in the path $P_{k\to k}$. Let q be the second node that is visited in the path $P_{k\to k}$, then $P_{k\to k} = (k, P_{q\to q}, k)$. Notice that the new path $P_{q\to q}$ again starts and ends with the same edge. This is a consequence of the fact that the network has a tree topology. Since the path $P_{k\to k}$ eventually returns to node k through the edge (k,q), Lemma A.2 can be applied at this edge, in which $k \leftrightarrow \mathcal{P}^0_{k\leftarrow q} = \mathcal{P}^{n-1}_{k\leftarrow q}$ and $P_{q\to q} \leftrightarrow (\mathcal{P}^i_{q\leftarrow k})_{i=1\ldots(n-2)}$. Indeed, $\mathcal{P}^0_{k\leftarrow q}$ and $\mathcal{P}^{n-1}_{k\leftarrow q}$ both contain C_{qk} as a constrained subset, and the partitionings in $(\mathcal{P}^i_{q\leftarrow k})_{i=1\ldots(n-2)}$ all contain C_{kq} in one of their constrained subset. This can then be viewed as a hypothetical two-node network in which node q performs the AO sequence defined by $(\mathcal{P}^i_{q\leftarrow k})_{i=2\ldots(n-1)}$, assuming that node q has access to all sensor signal observations in $\{\mathbf{y}_n : n \in C_{qk}\}$. Lemma A.2 then states that the resulting $\mathbf{W}_k^n$ is the same as if node k performed the whole AO sequence defined by $(\mathcal{P}^i)_{i=0\ldots n-1}$ by itself.

Obviously, node q does not have direct access to all sensor signal observations in $\{\mathbf{y}_n : n \in C_{qk}\}$ and cannot perform the AO sequence $(\mathcal{P}^i_{q\leftarrow k})_{i=1\ldots(n-2)}$. However, since the new path $P_{q\to q}$ also starts and ends with the same edge, we can again apply Lemma A.2 on the edge between node q and the node that is visited third in the path $P_{k\to k}$. This line of thought can be continued further until the problem has been recursively decomposed into a hierarchy of two-node networks. Applying Lemma A.2 backwards on all these recursive problems proves the theorem. ■

Lemma A.4: Under the same assumptions of Lemma A.3, the following holds:

$$J_k\left(\mathbf{W}_k^n\right) = J_k\left(\mathbf{W}_k^1\right)$$
$$\Rightarrow \left(\mathbf{W}^n_{k|C_{kq}} = \mathbf{W}^1_{k|C_{kq}},\ \mathbf{G}^n_{kq} = \mathbf{A}_{kq},\ \mathbf{G}^n_{qk} = \mathbf{A}_{qk}\right) \quad (40)$$

where q is the neighboring node of k that is visited first and last but one in the path $P_{k\to k}$.

Proof: Assume hypothetically that node k performs all the updates of the AO sequence defined by the path $P_{k\to k}$, on its own cost function $J_k(\mathbf{W}_k)$. Since $J_k(\mathbf{W}_k^n) = J_k(\mathbf{W}_k^1)$, and because of (32), we find that $J_k(\mathbf{W}_k^i) = J_k(\mathbf{W}_k^1)$, $\forall\, i \in \{1,\ldots,n\}$. The latter, combined with (33), yields

$$\mathbf{W}_k^i = \mathbf{W}_k^1, \quad \forall\, i \in \{1,\ldots,n\}. \quad (41)$$

Keep in mind that (41) only holds if node k itself would perform all the AO steps, which is not the case. However, since $P_{k\to k}$ starts and ends in node k, the first and last AO step are performed by node k itself, and Lemma A.3 then explains that $\mathbf{W}_k^1$ and $\mathbf{W}_k^n$ are equal to the result we would obtain if node k performed all updates by itself. Therefore, (41) does indeed hold for $i = n$, which proves the first part of the right-hand side of (40).

If node q would perform all the updates of the AO sequence defined by the path $P_{k\to k}$, on its own cost function $J_q(\mathbf{W}_q)$, then (41) and Lemma A.1 would imply that $\mathbf{W}_q^i = \mathbf{W}_k^1\mathbf{A}_{qk}$, $\forall\, i \in \{1,\ldots,n\}$. Since node q is the second to last node that is updated, the latter does indeed hold for $i = n-1$ (implied by Lemma A.3). Therefore, $\mathbf{G}_{qk}^{n-1} = \mathbf{A}_{qk}$, and since $\mathbf{G}_{qk}$ is not updated in the last step of the AO sequence, $\mathbf{G}_{qk}^n = \mathbf{G}_{qk}^{n-1} = \mathbf{A}_{qk}$. The fact that $\mathbf{G}_{kq}^n = \mathbf{A}_{kq}$ can be proven with a similar argument. ■

We can now prove the main theorem:

Proof of Theorem V.2: Consider the node $k = p_1$, i.e., the first node that is updated by the T–DANSE$_K$ algorithm. Consider the infinite path $\bar{P} = (P, P, \ldots)$, i.e., the periodic extension of path P, that defines the updating order of the T–DANSE$_K$ algorithm. Path $\bar{P}$ can be split into a sequence of subpaths $(P^j_{k\to k})_{j\in\mathbb{N}}$ that all satisfy the conditions in Lemma A.3. This lemma shows that, every time node k is updated, the result is as if node k performed all optimizations in the AO sequence with respect to its own cost function J_k, even though the AO updates are performed by different nodes with respect to different cost functions. With (32), we therefore find that

$$J_k\left(\mathbf{W}_k^{s_k^{j+1}}\right) \leq J_k\left(\mathbf{W}_k^{s_k^j}\right) \quad (42)$$

where the subsequence $(s_k^j)_{j\in\mathbb{N}} \subset (i)_{i\in\mathbb{N}}$ corresponds to the iteration indices at which node k updates the entries in $\mathbf{W}_{kk}^i$ and $\mathbf{G}_{k-k}^i$. By using a similar reasoning, it can be shown that expression (42) holds for any node k. This shows that all sequences $(J_k(\mathbf{W}_k^{s_k^j}))_{j\in\mathbb{N}}, \forall\, k \in \mathcal{J}$, are decreasing sequences, and since they have a lower bound, they converge. It can be shown that convergence of the sequence $(J_k(\mathbf{W}_k^{s_k^j}))_{j\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$, implies convergence of the sequences $(\mathbf{W}_{kk}^i)_{i\in\mathbb{N}}$ and $(\mathbf{G}^i_{k,-k})_{i\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$, by applying Lemma A.4 to each subpath of the form $\left(P^j_{k\to k}\right)_{j\in\mathbb{N}}$, $\forall\, k \in \mathcal{J}$. This proves convergence of the T–DANSE$_K$ algorithm.

Notice that, from Lemma A.4, it follows that

$$\forall\, k \in \mathcal{J},\ \forall\, q \in \mathcal{N}_k :\ \mathbf{G}_{kq}^\infty = \mathbf{A}_{kq} \quad (43)$$

and therefore, $\mathbf{G}^\infty_{k\leftarrow l} = \mathbf{A}_{kl}$ for any k and l, since $\mathbf{A}_{nl}\mathbf{A}_{kn} = \mathbf{A}_{kl}$, for any k,l and n. Parametrization (20) then shows that

$$\forall\, k, q \in \mathcal{J} :\ \mathbf{W}_k^\infty = \mathbf{W}_q^\infty \mathbf{A}_{kq} \quad (44)$$

which satisfies the mutual property of the MMSE solutions given by (10). With this, we can show that $\mathbf{W}_k^\infty = \hat{\mathbf{W}}_k$, $\forall\, k \in \mathcal{J}$, by using the same arguments as in the optimality proof for the DANSE_K algorithm in a fully connected network, as given in [12]. It is therefore omitted here. ■

B. Proof of Lemma A.1

Proof: We will prove the following induction hypothesis:

$$\Big(\forall\, j \in \{1,\ldots,t_i\},\ \exists\, \mathbf{G}_j \in (\mathbb{C}^{K\times K})^{-1}: \\ \mathbf{W}^i_{k|C^i_j} = \mathbf{W}^i_{q|C^i_j}\mathbf{G}_j\Big) \Rightarrow \mathbf{W}^{i+1}_k = \mathbf{W}^{i+1}_q \mathbf{A}_{kq} \quad (45)$$

with $(\mathbb{C}^{K\times K})^{-1}$ denoting the set of non-singular complex valued $K \times K$ matrices. The left-hand side of (45) states that any $\mathbf{W}^i_{k|C^i_j}$ has the same column space as $\mathbf{W}^i_{q|C^i_j}$. This implies that the search space for $\mathbf{W}_k$ and $\mathbf{W}_q$ is the same in the ith optimization step.

For notational convenience, we omit the superscript i in C^i_j and the subscript i in t_i. By substituting the constraints of (31) in the cost function, we obtain the unconstrained optimization problem:

$$\min_{\mathbf{V}_k} E\left\{\left\|\mathbf{d}_k - \mathbf{V}^H_k \tilde{\mathbf{y}}^i_k\right\|^2\right\} \quad (46)$$

with

$$\mathbf{V}^H_k = \left[\mathbf{W}^H_{k|F}\,\Big|\,\mathbf{C}^H_{1,k}\,|\ldots|\,\mathbf{C}^H_{t,k}\right]$$
$$\tilde{\mathbf{y}}^i_k = \begin{bmatrix} \mathbf{y}_F \\ \hline \mathbf{W}^{i\,H}_{k|C_1}\mathbf{y}_{C_1} \\ \hline \vdots \\ \hline \mathbf{W}^{i\,H}_{k|C_t}\mathbf{y}_{C_t} \end{bmatrix} \quad (47)$$

where $\mathbf{y}_F$ and $\mathbf{y}_{C_j}$ denote the stacked vector of all $\mathbf{y}_l$ for which $l \in F$ and $l \in C_j$, respectively. The solution of(46) is

$$\mathbf{V}_k = \left(\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}\right)^{-1}\mathbf{R}^i_{\tilde{y}_k d_k} \quad (48)$$

with $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k} = E\{\tilde{\mathbf{y}}^i_k\tilde{\mathbf{y}}^{i\,H}_k\}$ and $\mathbf{R}^i_{\tilde{y}_k d_k} = E\{\tilde{\mathbf{y}}^i_k\mathbf{d}^H_k\}$. Consider the optimization problem (46) for another node q, i.e., $\min_{\mathbf{V}_q} E\{\|\mathbf{d}_q - \mathbf{V}^H_q\tilde{\mathbf{y}}^i_q\|^2\}$. The solution of this optimization problem is

$$\mathbf{V}_q = \left(\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}\right)^{-1}\mathbf{R}^i_{\tilde{y}_q d_q}. \quad (49)$$

If the left-hand side of (45) is satisfied, then we can write

$$\tilde{\mathbf{y}}^i_k = \mathbf{D}^H\tilde{\mathbf{y}}^i_q \quad (50)$$

with $\mathbf{D}$ a block diagonal matrix defined by $\mathbf{D} = \mathrm{diag}(\mathbf{I}, \mathbf{G}_1, \ldots, \mathbf{G}_t)$, where $\mathbf{I}$ denotes an identity matrix of appropriate dimensions. From (1) and (50) we find the following relations between correlation matrices:

$$\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k} = \mathbf{D}^H\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}\mathbf{D} \quad \mathbf{R}^i_{\tilde{y}_k d_k} = \mathbf{D}^H\mathbf{R}^i_{\tilde{y}_q d_q}\mathbf{A}_{kq} \quad (51)$$

with $\mathbf{A}_{kq} = \mathbf{A}^{-H}_q\mathbf{A}^H_k$. Comparison of (48) and (49), together with (51), yields

$$\mathbf{V}_k = \mathbf{D}^{-1}\mathbf{V}_q\mathbf{A}_{kq}. \quad (52)$$

By comparing the left-hand side and the right-hand side of (52), we find that

$$\mathbf{W}^{i+1}_{k|F} = \mathbf{W}^{i+1}_{q|F}\mathbf{A}_{kq} \quad (53)$$
$$\begin{aligned}\forall\, j \in \{1,\ldots,t\}: \mathbf{W}^{i+1}_{k|C_j} &= \mathbf{W}^i_{k|C_j}\mathbf{C}_{j,k} \\ &= \mathbf{W}^i_{k|C_j}\mathbf{G}^{-1}_j\mathbf{C}_{j,q}\mathbf{A}_{kq} \\ &= \mathbf{W}^i_{q|C_j}\mathbf{C}_{j,q}\mathbf{A}_{kq} \\ &= \mathbf{W}^{i+1}_{q|C_j}\mathbf{A}_{kq} \quad (54)\end{aligned}$$

which proves the induction hypothesis (45). Notice that

$$\mathbf{W}^{i+1}_k = \mathbf{W}^{i+1}_q\mathbf{A}_{kq} \Rightarrow \forall\, j \in \{1,\ldots,t_{i+1}\}: \\ \mathbf{W}^{i+1}_{k|C^{i+1}_j} = \mathbf{W}^{i+1}_{q|C^{i+1}_j}\mathbf{A}_{kq} \quad (55)$$

i.e., if the right-hand side of (45) holds for $i = s$, then the left-hand side of (45) holds for $i = s + 1$. With (45) and (55), and since the left-hand side of (45) is satisfied for $i = 0$, the lemma is proven by induction. ■

References

[1] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, Jul. 2008.

[2] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[3] S. Kar and J. Moura, "Distributed linear parameter estimation in sensor networks: Convergence properties," in *Proc. 42nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2008, pp. 1347–1351.

[4] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 2320–2330, May 2011.

[5] B. Van Veen and K. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Mag.*, vol. 5, pp. 4–24, 1988, Apr.

[6] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, pp. 38–51, Jan. 2009.

[7] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP J. Adv. Signal Process.*, 2009, Article ID 530435; doi:10.1155/2009/530435.

[8] A. Bertrand, J. Callebaut, and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks," presented at the Int. Workshop Acoust. Echo, Noise Control (IWAENC), Tel Aviv, Israel, Aug. 2010.

[9] I. Schizas, G. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4284–4299, Aug. 2007.

[10] A. Dogandzic and B. Zhang, "Distributed estimation and detection for sensor networks using hidden markov random field models," *IEEE Trans. Signal Process.*, vol. 54, no. 8, pp. 3200–3215, Aug. 2006.

[11] J. Fang and H. Li, "Distributed estimation of Gauss–Markov random fields with one-bit quantized data," *IEEE Signal Process. Lett.*, vol. 17, pp. 449–452, May 2010.

[12] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks—Part I: Sequential node updating," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5277–5291, 2010, accepted for publication.

[13] A. Bertrand and M. Moonen, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," presented at the IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP), Taipei, Taiwan, R.O.C., Apr. 2009.

[14] S. Doclo, T. Klasen, T. Vanden Bogaert, J. Wouters, and M. Moonen, "Theoretical analysis of binaural cue preservation using multi-channel Wiener filtering and interaural transfer functions," presented at the Int. Workshop Acoust. Echo Noise Control (IWAENC), Paris, France, Sep. 2006.

[15] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks—Part II: Simultaneous & asynchronous node updating," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5292–5306, 2010.

[16] A. Bertrand and M. Moonen, "Distributed adaptive node-specific MMSE signal estimation in sensor networks with a tree topology," presented at the Eur. Signal Process. Conf. (EUSIPCO), Glasgow, Scotland, U.K., Aug. 2009.

[17] A. Spriet, M. Moonen, and J. Wouters, "The impact of speech detection errors on the noise reduction performance of multi-channel Wiener filtering and generalized sidelobe cancellation," *Signal Process.*, vol. 85, pp. 1073–1088, Jun. 2005.

[18] A. Bertrand and M. Moonen, "Efficient sensor subset selection and link failure response for linear MMSE signal estimation in wireless sensor networks," in *Proc. Eur. Signal Processing Conf. (EUSIPCO)*, Aalborg, Denmark, Aug. 2010, pp. 1092–1096.

[19] H. Chen, A. Campbell, B. Thomas, and A. Tamir, "Minimax flow tree problems," *Networks*, vol. 54, pp. 117–129, Mar. 2009.

[20] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artif. Intell.*, vol. 29, no. 3, pp. 241–288, 1986.

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

[22] J. C. Bezdek and R. J. Hathaway, "Some notes on alternating optimization," in *Advances in Soft Computing*. Berlin, Germany: Springer, 2002, pp. 187–195.

Alexander Bertrand (S'08) was born in Roeselare, Belgium, in 1984. He received the M.Sc. degree in electrical engineering from Katholieke Universiteit Leuven, Belgium, in 2007.

Since 2007, he has been working towards the Ph.D. degree under the supervision of Prof. M. Moonen, at the Electrical Engineering Department (ESAT), Katholieke Universiteit Leuven. In 2010, he was a visiting researcher at the Adaptive Systems Laboratory, University of California, Los Angeles (UCLA), under the supervision of Prof. A. H. Sayed. His research interests are in multi-channel signal processing, ad hoc sensor arrays, wireless sensor networks, distributed signal enhancement, speech enhancement, and distributed estimation.

Mr. Bertrand received a Ph.D. scholarship of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) from 2008–2011, and an FWO (Research Foundation–Flanders) travel grant for a Visiting Research Collaboration at UCLA in 2010.

Marc Moonen (M'94–SM'06–F'07) received the Electrical Engineering degree and the Ph.D. degree in applied sciences from Katholieke Universiteit Leuven, Belgium, in 1986 and 1990, respectively.

Since 2004, he has been a Full Professor at the Electrical Engineering Department of Katholieke Universiteit Leuven, where he heads a research team working in the area of numerical algorithms and signal processing for digital communications, wireless communications, DSL, and audio signal processing.

Dr. Moonen received the 1994 KU Leuven Research Council Award, the 1997 Alcatel Bell (Belgium) Award (with P. Vandaele), the 2004 Alcatel Bell (Belgium) Award (with R. Cendrillon), and was a 1997 "Laureate of the Belgium Royal Academy of Science." He received a journal best paper award from the IEEE TRANSACTIONS ON SIGNAL PROCESSING (with G. Leus) and from *Elsevier Signal Processing* (with S. Doclo). He was Chairman of the IEEE Benelux Signal Processing Chapter from 1998 to 2002, and is currently President of EURASIP (European Association for Signal Processing). He served as Editor-in Chief for the *EURASIP Journal on Applied Signal Processing* from 2003 to 2005 and has been a member of the Editorial Board of *Integration*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II from 2002 to 2003, and IEEE SIGNAL PROCESSING MAGAZINE from 2003 to 2005, and *Integration, the VLSI Journal*. He is currently a member of the Editorial Board of the *EURASIP Journal on Advances in Signal Processing*, the *EURASIP Journal on Wireless Communications and Networking*, and *Signal Processing*.