A Distributed Adaptive Algorithm for Node-Specific Signal Fusion Problems in Wireless Sensor Networks

Cem Ates Musluoglu and Alexander Bertrand
KU Leuven, Department of Electrical Engineering (ESAT),
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Belgium
{cemates.musluoglu, alexander.bertrand}@esat.kuleuven.be

***Abstract*—The distributed adaptive signal fusion (DASF) framework has been proposed as a generic method to solve spatial filtering and signal fusion problems in a distributed fashion over a wireless sensor network, reducing the communication and energy costs compared to a centralized approach. The DASF framework assumes that there is a common goal across the nodes, i.e., all nodes collaborate to optimize the same network-wide objective function. However, some applications require node-specific objectives, which are linked via a common latent target signal subspace. In this work, we propose the distributed adaptive node-specific signal fusion (DANSF) algorithm which builds upon the DASF framework, and extends it to allow for such node-specific spatial filtering problems.**

***Index Terms*—Distributed Signal Processing, Distributed Spatial Filtering, Feature Fusion.**

I. Introduction

Wireless sensor networks (WSNs) consist of a set of physically distributed wireless sensor nodes that are able to locally process the collected sensor data and share it with other nodes in the network. Typically, the goal is to estimate a signal or parameter satisfying some optimality condition which is dependent on the global data of the network, obtained by combining the data collected at every node. In our work, we are interested in optimal spatial filtering [1], i.e., linearly combining all the signals measured within a WSN to obtain a filtered output signal that is optimal in some sense. Applications of spatial filtering include — but are not restricted to — wireless communication [2], [3], biomedical signal processing [4], [5] and acoustics [6], [7].

Some applications require the nodes to estimate a common spatial filter as in [8]–[10]. This usually translates mathematically as an optimization problem which is common to every node in the network. However, a node-specific spatial filter can be desired, e.g., when each node is interested in different source signals or differently filtered versions of the same source signal(s) [6], [11], [12]. Each node then has a different optimization problem to solve, i.e., the problem is node-specific, yet can be related, e.g., via a common latent signal model, in which case a joint processing is desirable.

Distributed algorithms for some particular node-specific problems have been studied, such as minimum mean-squared error (MMSE) [13]–[15] and linearly constrained minimum variance beamforming (LCMV) [16]–[19], although each problem has been treated separately in the literature. Other distributed algorithms for node-specific problems have been proposed in [20], [21], but can generally not be applied to spatial filtering due to the way the data is partitioned across the network.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 802895). The authors also acknowledge the financial support of the FWO (Research Foundation Flanders) for project G081722N, and the Flemish Government (AI Research Program).

The distributed adaptive signal fusion (DASF) algorithm proposed in [22] is a generic "vanilla" algorithm that allows to solve spatial filtering problems in a distributed and adaptive fashion, i.e., without centralization of the data, while converging to the solution of the central problem under mild constraints [23]. The DASF algorithm captures several existing distributed signal fusion algorithms as special cases. However, it considers a single common optimization problem to be solved across the network, i.e., it does not allow for node-specific optimization problems. In this work, we propose the distributed adaptive node-specific signal fusion (DANSF) algorithm, which builds upon the DASF framework to solve generic node-specific problems in a distributed fashion, where the node-specific objectives are linked via a common latent target signal subspace. The proposed method contains specific algorithms previously described for the MMSE and LCMV problems [13]–[18] as special cases and allows for a generalization over a larger class of problems, such as robust variations of the MMSE/LCMV, regularized total least squares, ℓ_2 regularizations terms, etc. The DANSF algorithm converges to the optimal solution of each node-specific problem under the same assumptions as the original DASF algorithm.

II. Problem Setting

We consider a sensor network with K nodes given in the set $\mathcal{K} = \{1, \dots, K\}$ and connected following a topology given by a graph $\mathcal{G}$, where each link between two nodes k and l implies that nodes k and l can share data with each other. Every node senses an M_k-channel signal $\mathbf{y}_k$ so that the network-wide signal can be defined as

$$\mathbf{y} = [\mathbf{y}_1^T, \dots, \mathbf{y}_K^T]^T, \tag{1}$$

while an observation at time sample t is denoted as $\mathbf{y}(t) \in \mathbb{R}^M$, where $M = \sum_k M_k$. The signal $\mathbf{y}$ should be viewed as a multivariate stochastic variable, assumed to be ergodic and (short-term) stationary. Each node k acts as a data sink and is interested in finding its own optimal (network-wide) spatial filter $X_k \in \mathbb{R}^{M \times Q}$ and the corresponding filter output $X_k^T \mathbf{y}(t)$, which should satisfy a node-specific optimality condition. We envisage a generic problem statement where we assume that the optimal filter X_k is the solution of an optimization problem of the following form (for node k):

$$\begin{aligned} \mathbb{P}_k : \underset{X_k \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \varphi_k\big(X_k^T \mathbf{y}(t), X_k^T B\big) \\ \text{subject to} \quad & \eta_{k,j}\big(X_k^T \mathbf{y}(t), X_k^T B\big) \leq 0 \;\; \forall j \in \mathcal{J}_I, \\ & \eta_{k,j}\big(X_k^T \mathbf{y}(t), X_k^T B\big) = 0 \;\; \forall j \in \mathcal{J}_E, \end{aligned} \tag{2}$$

where the sets $\mathcal{J}_\mathcal{I}$ and $\mathcal{J}_\mathcal{E}$ represent the index sets for inequality and equality constraints respectively. Some examples of problems of the form (2) are shown in Table I. The functions

TABLE I: Examples of problems with node-specific objectives as in (2). MMSE is the minimum mean squared error problem and LCMV the linearly constrained minimum variance beamforming problem. $\mathbb{E}$ denotes the expectation operator.

Problem	Cost function φ_k	Constraints
MMSE	$\min \mathbb{E}[\|\|\mathbf{d}_k(t) - X_k^T\mathbf{y}(t)\|\|^2]$	—
LCMV	$\min \mathbb{E}[\|\|X_k^T\mathbf{y}(t)\|\|^2]$	$X_k^T B = H_k$

φ_k and $\eta_{k,j}$, $j \in \mathcal{J}_\mathcal{I} \cup \mathcal{J}_\mathcal{E}$ are real and scalar-valued functions, while the subscript k specifies that a function or variable is specific for node k. Moreover, as the filter output $(X_k^T\mathbf{y}(t))$ is a stochastic variable, the functions in (2) should contain an operator to extract a real-valued deterministic quantity from this term, such as an expectation operator. Note that a solution $X_k^*(t)$ of (2) depends on the time sample t, as the statistics of the signal $\mathbf{y}$ are allowed to change in time. The proposed DANSF algorithm will act as a block-adaptive filter that estimates and tracks the changes in the data statistics. However, from short-term stationarity, we assume that a solution $X_k^*(t)$ of (2) changes slowly in time compared to the convergence rate of the DANSF algorithm. Therefore we omit the time-dependence of solutions of (2) for mathematical tractability and assume stationarity within convergence time of the algorithm.

B is a deterministic $M \times L$ matrix independent of the time index t, and is commonly encountered to enforce a structure on the variable X_k (as in, e.g., LCMV in Table I). Although treated similarly, the distinction between stochastic signals $\mathbf{y}$ and deterministic matrices B are made to emphasize the adaptive and stochastic properties of the algorithm we will present. Similarly to the partitioning of $\mathbf{y}$ in (1), these deterministic matrices are assumed to be obtained by stacking $M_k \times L$ matrices B_k, where B_k is supposed to be available at node k, i.e., $B = [B_1^T, \ldots, B_K^T]^T$. Moreover, for a fixed node k, we also allow Problem (2) to have multiple variables, signals and deterministic matrices (in addition to X_k, $\mathbf{y}$ and B, respectively), which are however not represented in (2) for conciseness. We assume that every parameter in (2), except $\mathbf{y}$ and B, is available at node k. For example, the signal $\mathbf{d}_k$ and parameter H_k in the MMSE and LCMV examples of Table I should be available at node k.

For each node k, let us denote by $\mathcal{X}_k^*$ the solution set of $\mathbb{P}_k$ and $X_k^* \in \mathcal{X}_k^*$ a specific solution. We then make the following assumption on the set of problems $\mathbb{P}_k$ which links the solutions across the nodes.

Assumption 1. *There exists a set of invertible $Q \times Q$ matrices $\{D_{k,l}\}_{(k,l)\in\mathcal{K}^2}$ such that for any pair (k,l) of nodes, the solutions $X_k^* \in \mathcal{X}_k^*$ and $X_l^* \in \mathcal{X}_l^*$ satisfy $X_k^* = X_l^* \cdot D_{k,l}$.*

These properties were also exploited in [13]–[17], [19] for the design of distributed fusion algorithms for MMSE and LCMV problems. Note that Assumption 1 implies that the solutions at the different nodes are instantaneous mixtures of each other. However, this also allows to model convolutive mixtures if the problem is formulated in the short-time Fourier transform domain, as for example in [15], [17]. Taking the example of the MMSE problem in Table I, it can be shown that Assumption 1 is satisfied if $\mathbf{d}_k(t) = D_{k,l}^T\mathbf{d}_l(t)$. This is true if the desired signals at the different nodes are all different mixtures from the same set of latent sources. This is common in, e.g., hearing aids where the acoustic mixing process needs to be preserved for spatial hearing [15], [17]. For the LCMV example of Table I, Assumption 1 is satisfied if $H_k = D_{k,l}^T H_l$, which is a common case in wireless acoustic sensor networks [17]–[19]. In this paper, we propose a unifying algorithmic framework, which has [13]–[18] as special cases, while also admitting new problems (see, e.g., Section IV), assuming they can be written in the form (2).

III. DASF for Node-Specific Problems

In this section, we derive the DANSF algorithm which extends the DASF framework [22], [23] to also admit node-specific problems of the form (2). We refer to [22] for a thorough presentation of the DASF algorithm, from which we here only extract the essential ingredients to allow us to define the proposed DANSF algorithm.

At each iteration i, an updating node $q \in \mathcal{K}$ is selected and the network represented by the graph $\mathcal{G}$ is temporarily pruned to a tree $\mathcal{T}^i(\mathcal{G}, q)$, such that there is a unique path between any pair of nodes in the network. The pruning function is a free design choice, however it should not remove any links between node q and its neighbors $n \in \mathcal{N}_q$ [22], where $\mathcal{N}_q$ denotes the set of neighboring nodes of node q. In the remaining parts of the algorithm derivation, the neighbors of any node are defined to be the ones after pruning the network, i.e., based on the edges of $\mathcal{T}^i(\mathcal{G}, q)$. Note that the updating node changes at each iteration i of the algorithm, which implies that a different tree is used in each iteration.

Let us partition each X_k, i.e., the network-wide spatial filter that generates the desired node-specific output signal for node k, as

$$X_k = [X_{k1}^T, \ldots, X_{kK}^T]^T, \tag{3}$$

such that each X_{kl} is $M_l \times Q$. For each $k \in \mathcal{K}$, we define X_{kk}, the $k-$th block of X_k, to be the compressor at node k. At iteration i, every node $k \neq q$ uses its current estimate of X_{kk} to compress the local M_k-dimensional sensor signal $\mathbf{y}_k$ into a $Q-$dimensional one, with $Q < M_k$. A similar compression applied to node k's matrix B_k leads to

$$\widehat{\mathbf{y}}_k^i \triangleq X_{kk}^{iT}\mathbf{y}_k, \; \widehat{B}_k^i \triangleq X_{kk}^{iT}B_k, \tag{4}$$

where X_{kk}^0 is initialized randomly for each k. The nodes will then fuse and forward their compressed data (4) towards node q as explained next. We note that a batch of N time samples of $\widehat{\mathbf{y}}_k^i$ should be transmitted by node k, where N should be chosen such that there are enough samples to estimate the relevant statistics of $\widehat{\mathbf{y}}_k^i$ that are used in the objective and constraints of (2) (in most practical examples, this consists of all the second-order statistics). At each iteration, a different block of N samples is used, so that in the case of changes in statistics of the signal $\mathbf{y}$, the proposed method can adaptively track these changes. Each node k first waits until receiving data from all of its neighbors except one, which we denote as n. Node k then fuses the data received from its neighbors $l \in \mathcal{N}_k \backslash \{n\}$ with its data (4) and the result is then transmitted to node n, which receives N samples of

$$\widehat{\mathbf{y}}_{k\to n}^i \triangleq X_{kk}^{iT}\mathbf{y}_k + \sum_{l\in\mathcal{N}_k\backslash\{n\}} \widehat{\mathbf{y}}_{l\to k}^i, \tag{5}$$

where $\widehat{\mathbf{y}}_{l\to k}^i$ is the data received by node k from its neighbor l. Note that the second term of (5) is recursive and vanishes for nodes with a single neighbor, i.e., leaf nodes, implying that the recursion in (5) is bootstrapped at the leaf nodes of the tree $\mathcal{T}^i(\mathcal{G}, q)$. The fused data eventually reaches node q, which receives N samples of

$$\widehat{\mathbf{y}}_{n\to q}^i = X_{nn}^{iT}\mathbf{y}_n + \sum_{k\in\mathcal{N}_n\backslash\{q\}} \widehat{\mathbf{y}}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i, \tag{6}$$

from all its neighbors $n \in \mathcal{N}_q$. In (6), $\mathcal{B}_{nq}$ is defined to be the subgraph of $\mathcal{T}^i(\mathcal{G}, q)$ which contains node n, obtained after

cutting the link between nodes n and q. A similar recursion applies for the compressed matrices $\widehat{B}_k^i$, and we define $\widehat{B}_{n\to q}^i$ as the matrix analogous to (6) received by node q. Defining $\mathcal{N}_q \triangleq \{n_1, \ldots, n_{|\mathcal{N}_q|}\}$, the data collected at node q can be structured as

$$\begin{aligned} \widetilde{\mathbf{y}}_q^i &\triangleq [\mathbf{y}_q^T, \widehat{\mathbf{y}}_{n_1\to q}^{iT}, \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T \in \mathbb{R}^{\widetilde{M}_q}, \\ \widetilde{B}_q^i &\triangleq [B_q^T, \widehat{B}_{n_1\to q}^{iT}, \ldots, \widehat{B}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T \in \mathbb{R}^{\widetilde{M}_q \times L} \end{aligned} \tag{7}$$

where $\widetilde{M}_q = |\mathcal{N}_q| \cdot Q + M_q$. Using the local data in (7), node q creates a compressed version of its original problem $\mathbb{P}_q$:

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi_q\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big) \\ \text{subject to} \quad & \eta_{q,j}\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big) \le 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_{q,j}\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T \widetilde{B}_q^i\big) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{8}$$

Note that (8) has the same objective and constraint functions as (2) (for $k = q$), hence a solver for (2) can also be used locally at node q to solve the compressed problem (8). This is an interesting feature of the DASF framework, which is inherited in DANSF as well.

Node q then solves its local problem (8) to obtain $\widetilde{X}_q^{i+1}$. In the cases where (8) has multiple solutions, we choose $\widetilde{X}_q^{i+1}$ by minimizing $||\widetilde{X}_q - \widetilde{X}_q^i||_F$ over all solutions of (8), with

$$\widetilde{X}_q^i = [X_{qq}^{iT}, I_Q, \ldots, I_Q]^T \tag{9}$$

and where X_{qq}^i corresponds to the current estimate of the compressor X_{qq} of node q. The optimal solution $\widetilde{X}_q^{i+1}$ is then partitioned as

$$\widetilde{X}_q^{i+1} = [X_{qq}^{(i+1)T}, G_{qn_1}^{(i+1)T}, \ldots, G_{qn_{|\mathcal{N}_q|}}^{(i+1)T}]^T, \tag{10}$$

with each $G-$matrix being $Q \times Q$. The new estimate of the variable $X_q = [X_{q1}^T, \ldots, X_{qK}^T]^T$ at iteration i is then

$$X_{qk}^{i+1} = \begin{cases} X_{qq}^{i+1} & \text{if } k = q \\ X_{kk}^i G_{qn}^{i+1} & \text{if } k \in \mathcal{B}_{nq},\ n \in \mathcal{N}_q. \end{cases} \tag{11}$$

For nodes $k \neq q$, a new estimate of their variable X_k can be estimated in the following way. Node q first transmits

$$\mathbf{z}_q^{i+1}(t) \triangleq \widetilde{X}_q^{(i+1)T} \widetilde{\mathbf{y}}_q^i(t), \quad Z_q^{i+1} \triangleq \widetilde{X}_q^{(i+1)T} \widetilde{B}_q^i \tag{12}$$

to each node k, which can either be broadcast by node q or transmitted following the pruned network topology (see [13] for a discussion on efficient ways to achieve this). Note that again N samples of $\mathbf{z}_q^{i+1}$ should be sent by node q to the other nodes. Node k then solves

$$\begin{aligned} \underset{F_{kq} \in \mathbb{R}^{Q \times Q}}{\text{minimize}} \quad & \varphi_k\big(F_{kq}^T \mathbf{z}_q^{i+1}(t), F_{kq}^T Z_q^{i+1}\big) \\ \text{subject to} \quad & \eta_{k,j}\big(F_{kq}^T \mathbf{z}_q^{i+1}(t), F_{kq}^T Z_q^{i+1}\big) \le 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_{k,j}\big(F_{kq}^T \mathbf{z}_q^{i+1}(t), F_{kq}^T Z_q^{i+1}\big) = 0 \ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{13}$$

such that a new estimate of its variable X_k at iteration i is

$$X_k^{i+1} = X_q^{i+1} F_{kq}^{i+1}, \tag{14}$$

where F_{kq}^{i+1} is a solution of (13) at node k. Note that the compressor at node k, i.e., X_{kk} is part of X_k, i.e., the compression matrix of node k is also updated by (14).

Remark 1. *Similar to the proof in [22], it can be shown that for each node k, X_k^i defined as in (11) and (14) always satisfies the constraints of the corresponding problem (2) at node k for every $i > 0$. Furthermore, $\widetilde{X}_q^{i+1} \in \widetilde{\mathcal{S}}_q^i \iff X_q^{i+1} \in \mathcal{S}$, where $\widetilde{\mathcal{S}}_q^i$ and $\mathcal{S}$ denote the constraint sets of (8) and (2), respectively (proof omitted, see [22]).*

Since only the compressor matrices X_{kk} (for all k) play a role within the algorithm (as these define the transmitted signals), the update of the blocks X_{kl} with $k \neq l$ can be omitted, unless the nodes are explicitly interested in knowing the coefficients of the full matrix X_k. However, in most applications, the filter output signal $\mathbf{z}_k(t) = X_k^T \mathbf{y}(t)$ is sought after, rather than the filter X_k itself, which can be computed at each node k as

$$\mathbf{z}_k^{i+1}(t) \triangleq \begin{cases} \widetilde{X}_q^{(i+1)T} \widetilde{\mathbf{y}}_q^i(t) & \text{if } k = q \\ F_{kq}^{(i+1)T} \mathbf{z}_q^{i+1}(t) & \text{if } k \neq q, \end{cases} \tag{15}$$

without keeping track of other subblocks X_{kq} for $k \neq q$. This is because the filtering of subblocks X_{kq} is done at node q instead of node k, using the compressor X_{qq}, of which the output is transformed with F_{kq} at node k.

At each iteration, this process is repeated by selecting a different updating node. Algorithm 1 summarizes the steps of the DANSF algorithm described above. We note that [14]–[17] are special cases of this algorithm. The method is able to adapt to and track changes in the signal statistics of $\mathbf{y}$, as is the case for the original DASF algorithm. This is because a new block of N samples, e.g., $\{\mathbf{y}(t)\}_{t=iN}^{(i+1)N-1}$, is measured and used at each iteration to solve (8), i.e., different iterations of the DANSF algorithm are spread over different sample blocks across the time dimension, similar to an adaptive filter, making each X_k^i an estimate of $X_k^*(t)$, where $t = iN$. Note that the communication cost per node and per iteration is at most $2NQ$ samples, independent of the topology (note that $Q < M_k$). In a setting where all the raw data is relayed to a fusion center via multi-hop transmissions, this cost is much higher (i.e., PNM_k, where P is the number of hops between node k and the fusion center). Additionally, as the problems (8) are compressed versions of the original problem (2) at each node k, the computational cost required to solve the former is smaller.

Theorem 1 below gives a convergence guarantee in cost for the DANSF algorithm. The full convergence for each node k to a solution $X_k^* \in \mathcal{X}_k^*$ of $\mathbb{P}_k$ is described afterwards in Theorem 2. However, the proof of the latter is omitted due to space constraints, but follows similar steps as in [23].

Theorem 1. *Let us denote by $(\varphi_k^i)_i$ the sequence of function values $\varphi_k\big(X_k^{iT}\mathbf{y}(t), X_k^{iT}B\big)$, $\forall k \in \mathcal{K}$, obtained from Algorithm 1. Then, the sequence $(\varphi_k^i)_i$ is non-increasing, and converges for each k, i.e., the cost function at each node is monotonically decreasing.*

Proof. For conciseness, we omit the matrix B in this proof (it can be treated similarly to $\mathbf{y}$). For the updating node q, at iteration i, we have that $\varphi_q\big(\widetilde{X}_q^{(i+1)T}\widetilde{\mathbf{y}}_q^i(t)\big) \le \varphi_q\big(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\big)$ for any $\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i$, since $\widetilde{X}_q^{i+1}$ solves the local problem (8). Moreover, since $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$ (see Remark 1), we have $\varphi_q\big(\widetilde{X}_q^{(i+1)T}\widetilde{\mathbf{y}}_q^i(t)\big) \le \varphi_q\big(\widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i(t)\big)$. This shows a monotonic decrease of the cost at the current updating node q. For the case $k \neq q$, let us (hypothetically) assume that the updating node q would have used the cost function of node k instead,

Algorithm 1: DANSF algorithm

X_{kk}^0 initialized randomly for each k, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) The network $\mathcal{G}$ is pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$.
  2) Each node k collects N samples of $\mathbf{y}_k$, compress these to N samples of $\widehat{\mathbf{y}}_k^i$ and also compute $\widehat{B}_k^i$ as in (4).
  3) The nodes sum-and-forward their compressed data towards node q via the recursive rule (5) (and a similar rule for the $\widehat{B}_k^i$'s). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ given in (6) and similarly, $\widehat{B}_{n\to q}^i$, from all its neighbors $n \in \mathcal{N}_q$.
  at *Node* q **do**
    4a) Compute the solution of (8) to obtain $\widetilde{X}_q^{i+1}$. If the solution is not unique, select $\widetilde{X}_q^{i+1}$ which minimizes $||\widetilde{X}_q^{i+1} - \widetilde{X}_q^i||_F$ with $\widetilde{X}_q^i$ defined in (9).
    4b) Partition $\widetilde{X}_q^{i+1}$ as in (10).
    4c) Update the estimate of X_q as in (11).
    4d) Disseminate Z_q^{i+1} and N samples of $\mathbf{z}_q^{i+1}$ as in (12) within the tree to each data sink node.
  end
  5) Nodes $k \neq q$ update X_k according to (14) by solving (13) and can estimate its filtered output as in (15).
  $i \leftarrow i + 1$

i.e., it solves

$$\begin{aligned} \underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{minimize}} \quad & \varphi_k\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t)\big) \\ \text{subject to} \quad & \eta_{k,j}\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t)\big) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_{k,j}\big(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t)\big) = 0 \ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{16}$$

instead of (8). As mentioned earlier, (8) and (16) are compressed versions of the problem (2) for node q and node k respectively, i.e., the data $\mathbf{y}$ of (2) is replaced by $\widetilde{\mathbf{y}}_q^i$ in (8) and (16). Therefore, the local problems (8)-(16) satisfy Assumption 1, i.e., there exists a matrix $\widetilde{D}_{q,k}$ such that $\widetilde{X}_q^{i+1} = \widetilde{X}_k^{i+1} \cdot \widetilde{D}_{q,k}$. This implies that node q also optimizes φ_k up to a transformation with a $Q \times Q$ matrix. The latter transformation is compensated for by finding a proper transformation F_{kq} at node k by solving (13). Since this argument holds for any iteration i, and from the relationship $X_k^{i+1} = X_q^{i+1} F_{kq}^{i+1}$ in (14), we have $\varphi_k(X_k^{(i+1)T}\mathbf{y}(t)) \leq \varphi_k(X_k^{iT}\mathbf{y}(t))$ even though node q optimizes φ_q instead of φ_k at iteration i. The sequence $(\varphi_k^i)_i$ is therefore non-increasing for each k. Since these sequences are respectively lower bounded by the minimal value of φ_k achieved for X_k^* over the constraint set of $\mathbb{P}_k$ in (2), they are converging sequences. □

Theorem 2 (Proof Omitted)**.** *Suppose that, for each node k, Problem (2) satisfies Assumption 1 and the conditions for convergence of the original DASF algorithm (see [22], [23]). Then the sequences $(X_k^i)_i$, for each $k \in \mathcal{K}$, obtained from the DANSF algorithm also converge respectively to an optimal point $X_k^* \in \mathcal{X}_k^*$ of Problem (2).*

Note that this also implies that each node has access to its optimal node-specific filter output $\mathbf{z}_k^*(t) = X_k^{*T}\mathbf{y}(t)$ for all samples collected after convergence of the algorithm. Sensor observations used *during* convergence of the algorithm are fused suboptimally (similar to how an adaptive filter initially produces suboptimal filter outputs). The algorithm can be used in an adaptive or tracking context if the dynamics of the statistics change slowly, i.e., slower than the convergence time of the DANSF algorithm.

IV. Simulations

To demonstrate the generic nature of the DANSF algorithm, we consider a new "toy" problem that has not been investigated in the literature in the context of node-specific spatial filtering. For results on practical scenarios, we refer readers to [13]–[18], which can be shown to be special cases of DANSF. The problem can be formulated as:

$$\begin{aligned} \underset{X_k \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \text{trace}(X_k^T B_k) \\ \text{subject to} \quad & \text{trace}(X_k^T R_{\mathbf{yy}} X_k) \leq 1, \end{aligned} \tag{17}$$

where $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$. Taking $B_k \neq 0$, the unique solution of the problem is given by $X_k^* = -\beta_k \cdot R_{\mathbf{yy}}^{-1} B_k$, where $\beta_k = \sqrt{\text{trace}(B_k^T R_{\mathbf{yy}}^{-1} B_k)^{-1}}$. For each k, we take $B_k = B \cdot D_k$, where each element of B and D_k's are drawn from a Gaussian distribution with zero-mean and variance 1, i.e., $\mathcal{N}(0,1)$. Note that Problem (17) satisfies Assumption 1 as $X_k^* = X_l^* \cdot D_{k,l}$, where $D_{k,l} = D_l^{-1} D_k \cdot \beta_k/\beta_l$. We will first look at the convergence properties of the DANSF algorithm under stationarity conditions and for different topologies. In a second experimental setting, we will show that the DANSF algorithm is able to track changes in the statistical properties of the signals, demonstrating its adaptive properties in a non-stationary setting. In every experiment described below, $\mathcal{T}^i(\cdot, q)$ is taken to be the shortest path pruning function.

A. Stationary Setting

In this experiment, we consider a stationary signal $\mathbf{y}$ which follows the mixture model $\mathbf{y}(t) = A \cdot \mathbf{d}(t) + \mathbf{n}(t)$, where each element of $\mathbf{d} \in \mathbb{R}^Q$ and $\mathbf{n} \in \mathbb{R}^M$ independently follows $\mathcal{N}(0, 0.5)$ and $\mathcal{N}(0, 0.1)$ respectively, at each time sample. Every entry of $A \in \mathbb{R}^{M \times Q}$ is drawn from $\mathcal{N}(0, 0.2)$.

At each iteration i of the DANSF algorithm, each node k transmits $N = 10^4$ samples of $\widehat{\mathbf{y}}_k^i$ to the updating node q. The updating node q then solves its local problem (8) given by

$$\underset{\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q \times Q}}{\text{min.}} \ \text{trace}(\widetilde{X}_q^T \widetilde{B}_q^i), \quad \text{s. t. } \text{trace}(\widetilde{X}_q^T R_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q}^i \widetilde{X}_q) \leq 1, \tag{18}$$

where $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$ are defined as in (7) and $R_{\widetilde{\mathbf{y}}_q \widetilde{\mathbf{y}}_q}^i = \mathbb{E}[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$.

The performance of the DANSF algorithm is assessed by computing the relative mean squared error (MSE) $\epsilon_k(X_k^i) = ||X_k^i - X_k^*||_F^2 \cdot ||X_k^*||_F^{-2}$, where X_k^* is the solution of (17) for node k. In all our experiments, we take $Q = 3$, $K = 10$ and $M_k = 7$ for every $k \in \mathcal{K}$. Figure 1 shows the MSE ϵ_k for every node k and for different network topologies namely fully-connected networks, networks with line topologies, i.e., each node has two neighbors except two which have a single neighbor, and networks with randomly generated topologies. We observe that the DANSF algorithm converges to X_k^* for every node k of the network, as stated in Theorem 2, although at different convergence rates for different topologies. Fully-connected networks converge the fastest, while the slowest

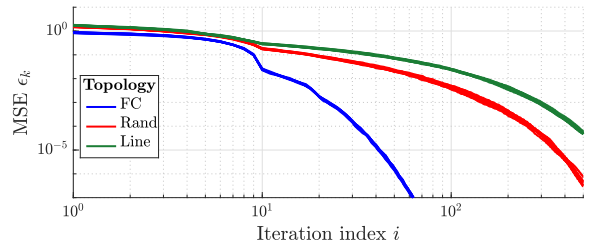

Fig. 1: MSE ϵ_k for all nodes k of the DANSF algorithm in a stationary setting for various network topologies, namely fully-connected networks (*FC*), randomly generated networks using the Erdős-Rényi model (*Rand*) and graphs in a line topology (*Line*). Each point has been obtained by taking the median of 100 Monte-Carlo runs.

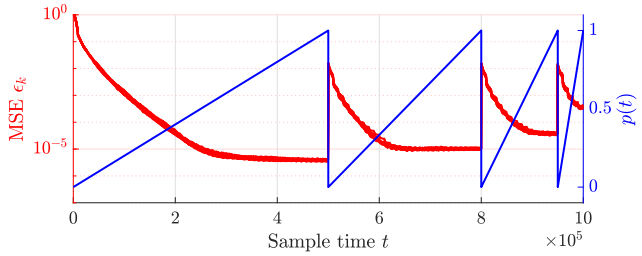

Fig. 2: MSE ϵ_k for all nodes k in an adaptive setting in a randomly generated networks using the Erdős-Rényi model. Each point has been obtained by taking the median of 100 Monte-Carlo runs.

convergence rate is obtained for networks with a line topology. A similar result was observed for the DASF algorithm, where networks with more connected topologies lead to faster convergence rates [22]. Additionally, we see from Figure 1 that each node's estimate of its variable X_k converges to the respective optimal value X_k^* without large deviations in convergence rate between different nodes.

B. Adaptive Setting

In this second experimental setting, we demonstrate that the DANSF algorithm is able to track changes in the signal statistics, therefore making the method adaptive. We consider the same setting as previously, except $N = 10^3$ and the mixture matrix A changes at each time sample t. In particular, we have $A(t) = A_0 \cdot (1 - p(t)) + (A_0 + \Delta) \cdot p(t)$, where the elements of A_0 and Δ are independently drawn from $\mathcal{N}(0, 0.2)$ and $\mathcal{N}(0, 0.01)$ respectively, and p is given in Figure 2. This implies that the stationarity condition does not hold and X^* is now time-dependent. Figure 1 shows the MSE value for each node k in networks with randomly generated topologies. We see that an abrupt change in the signal statistics imply a sudden increase in the MSE value, while the algorithm gradually adapts to slow rates of change, as can be seen by the decreasing MSE. We also observe that the algorithm reaches MSE floors instead of converging towards 0 as in the stationary case. This is due to the fact that the optimal solutions X_k^*, for each node k, change at each iteration, where the error floor is higher for faster rates of change in the signal statistics.

V. Conclusion

We have proposed the DANSF algorithm to solve node-specific signal fusion problems in a distributed fashion over a network. The DANSF algorithm builds upon the principles of the DASF framework and extends it to problems with different optimization problems at each node, yet with coupled solution sets, which leads to analogous convergence results between both algorithms. We provided a proof for the convergence in cost, which showed that we obtain a monotonic decrease of the cost at each node. Simulations of the DANSF algorithm applied on a new problem validated our convergence claims.

References

[1] S. Haykin and K. R. Liu, *Handbook on array processing and sensor networks*. John Wiley & Sons, 2010.

[2] E. Björnson and L. Sanguinetti, "Scalable cell-free massive MIMO systems," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 4247–4261, 2020.

[3] L. Sanguinetti, E. Björnson, and J. Hoydis, "Toward massive MIMO 2.0: Understanding spatial correlation, interference suppression, and pilot contamination," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 232–257, 2019.

[4] A. Bertrand, "Distributed signal processing for wireless EEG sensor networks," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 6, pp. 923–935, 2015.

[5] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Muller, "Optimizing spatial filters for robust EEG single-trial analysis," *IEEE Signal processing magazine*, vol. 25, no. 1, pp. 41–56, 2007.

[6] N. Furnon, R. Serizel, I. Illina, and S. Essid, "Distributed speech separation in spatially unconstrained microphone arrays," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4490–4494.

[7] J. Zhang, R. Heusdens, and R. C. Hendriks, "Rate-distributed spatial filtering based noise reduction in wireless acoustic sensor networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2015–2026, 2018.

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

[9] ——, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[10] S. Markovich-Golan, S. Gannot, and I. Cohen, "Distributed multiple constraints generalized sidelobe canceler for fully connected wireless acoustic sensor networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 343–356, 2012.

[11] S. Markovich, S. Gannot, and I. Cohen, "Multichannel eigenspace beamforming in a reverberant noisy environment with multiple interfering speech signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1071–1086, 2009.

[12] S. Doclo, T. J. Klasen, T. Van den Bogaert, J. Wouters, and M. Moonen, "Theoretical analysis of binaural cue preservation using multi-channel wiener filtering and interaural transfer functions," in *Proc. Int. Workshop Acoust. Echo Noise Control (IWAENC)*, 2006, pp. 1–4.

[13] J. Szurley, A. Bertrand, and M. Moonen, "Topology-independent distributed adaptive node-specific signal estimation in wireless sensor networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 130–144, 2016.

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

[15] S. Doclo, M. Moonen, T. Van den Bogaert, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 38–51, 2009.

[16] A. Bertrand and M. Moonen, "Distributed node-specific LCMV beamforming in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 233–246, 2011.

[17] S. Markovich-Golan, S. Gannot, and I. Cohen, "A reduced bandwidth binaural MVDR beamformer," in *Proc. of the International Workshop on Acoustic Echo and Noise Control (IWAENC), Tel-Aviv, Israel*, 2010.

[18] S. Markovich-Golan, A. Bertrand, M. Moonen, and S. Gannot, "Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks," *Signal Processing*, vol. 107, pp. 4–20, 2015.

[19] X. Guo, M. Yuan, Y. Ke, C. Zheng, and X. Li, "Distributed node-specific block-diagonal LCMV beamforming in wireless acoustic sensor networks," *Signal Processing*, vol. 185, p. 108085, 2021.

[20] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2733–2748, 2015.

[21] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3448–3460, 2015.

[22] C. A. Musluoglu and A. Bertrand, "A unified algorithmic framework for distributed adaptive signal and feature fusion problems—part I: Algorithm derivation," *IEEE Transactions on Signal Processing*, vol. 71, pp. 1863–1878, 2023.

[23] C. A. Musluoglu, C. Hovine, and A. Bertrand, "A unified algorithmic framework for distributed adaptive signal and feature fusion problems — part II: Convergence properties," *IEEE Transactions on Signal Processing*, vol. 71, pp. 1879–1894, 2023.