A Unified Algorithmic Framework for Distributed Adaptive Signal and Feature Fusion Problems
— Part I: Algorithm Derivation

Cem Ates Musluoglu, and Alexander Bertrand, *Senior Member, IEEE*

***Abstract*—In this paper, we describe a general algorithmic framework for solving linear signal or feature fusion optimization problems in a distributed setting, for example in a wireless sensor network (WSN). These problems require linearly combining the observed signals (or features thereof) collected at the various sensor nodes to satisfy a pre-defined optimization criterion. The framework covers several classical spatial filtering problems, including minimum variance beamformers, multi-channel Wiener filters, principal component analysis, canonical correlation analysis, (generalized) eigenvalue problems, etc. The proposed distributed adaptive signal fusion (DASF) algorithm is an iterative method that solves these types of problems by allowing each node to share a linearly compressed version of the local sensor signal observations with its neighbors to reduce the energy and bandwidth requirements of the network. We first discuss the case of fully-connected networks and then extend the analysis to more general network topologies. The general DASF algorithm is shown to have several existing distributed algorithms from the literature as a special case, while at the same time allowing to solve new distributed problems as well with guaranteed convergence and optimality. This paper focuses on the algorithm derivation of the DASF framework along with simulations demonstrating its performance. A technical analysis along with convergence conditions and proofs are provided in a companion paper.**

***Index Terms*—Distributed optimization, distributed signal processing, spatial filtering, signal fusion, feature fusion, wireless sensor networks.**

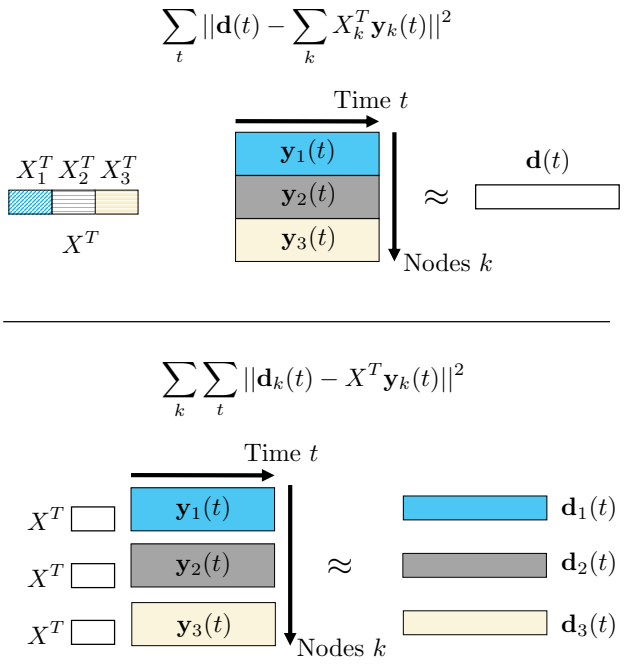

Fig. 1. Comparison of the 3−node problem setting of the DSFO (top) and the traditional consensus-type setting (bottom) with corresponding example objective functions for the case of least squares estimation. In the consensus setting, note that the objective is per-node separable and has a shared optimization variable X which is assumed to be the same across all nodes (which is why there is no node subscript k).

I. Introduction

WIRELESS Sensor Networks (WSNs) consist of a wireless network of sensor nodes, which collect, process, and share data in order to solve a specific signal processing task in a collaborative fashion. Such WSNs allow to easily acquire data at multiple locations simultaneously, which is useful in several application domains including health monitoring [2], [3], acoustics [4], [5], structural monitoring [6], environmental studies [7], [8], and many others [9], [10].

In many cases, the aim is to find or estimate a common signal, filter or a set of parameters that satisfy a pre-defined optimality criterion involving the observation data from all the nodes [11], [12]. Various "work horse" strategies and frameworks have been described previously to solve such problems in a distributed fashion. Well-known examples are consensus [13], [14], incremental strategies [15], [16], diffusion [17]–[19], gossip [20], [21], or the alternating direction method of multipliers [22], [23].

In order to achieve a distributed implementation, most of these methods rely on the separability of the global cost function f as a sum of local functions f_k: $f(X) = \sum_k f_k(X)$, where f_k depends only on the local data of node k, and where X is a *shared* optimization variable across all nodes. In this case, we say that f is per-node separable. However, there exist various cases where this property is not satisfied, e.g., when optimizing a spatial filter X that linearly combines the signals from different nodes. Classical examples are adaptive beamformers [24], [25], multi-channel Wiener filtering [26], [27], principal component analysis, filters based on (generalized) eigenvectors of spatial covariance matrices [28], [29], etc. In these problems, the aim is to find a network-wide linear spatial filter $X \in \mathbb{R}^{M\times Q}$, $Q < M$ to apply to 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).

C.A. Musluoglu and A. Bertrand are with KU Leuven, Department of Electrical Engineering (ESAT), Stadius Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium and with Leuven.AI - KU Leuven institute for AI. e-mail: cemates.musluoglu, alexander.bertrand @esat.kuleuven.be

A companion paper submitted together with this paper is provided in [1].

wide $M-$channel time signal $\mathbf{y}(t) \in \mathbb{R}^M$ containing all sensor channels[1] from all nodes in the network, where t denotes the time or sample index. The filter X is typically designed to exploit the spatial correlation across the different nodes to optimize some network-wide objective function in the form $f(X^T\mathbf{y}(t))$, with T the transpose operator. In this case, the function f itself is not per-node separable, but the argument is, i.e., $f(X^T\mathbf{y}(t)) = f(\sum_k X_k^T\mathbf{y}_k(t))$. If $\mathbf{y}$ is interpreted as a feature vector, this type of separation of the observed data is also known as feature partitioning or distributed features [30]–[35]. In this paper, we refer to such cases as a distributed signal fusion optimization (DSFO) problem to emphasize that the framework also applies to traditional array processing problems such as beamforming or spatial filtering. A visual example that illustrates the conceptual difference between both aforementioned types of data separation, i.e., separability of the cost function versus separability of the argument, is given in Figure 1 for the case of least squares estimation.

A commonly encountered strategy to solve DSFO problems is to compute all inner products involving the data vector $\mathbf{y}$ via a standard consensus-type subroutine that performs in-network averaging or summation [36], [37], or by artificially rewriting the problem as a consensus problem (e.g., by treating the filter output $\mathbf{d}$ itself as a shared (consensus) optimization variable in the example of Figure 1). However, this strategy typically results in a distributed algorithm with nested iterative loops for each sample time t, each in itself requiring a substantial number of communication rounds. Such an iterative distributed subroutine typically has to be executed from scratch for each new sample observation at the sensors. This leads to a large communication burden, which also scales poorly with network size, i.e., a larger network increases the *per node* transmission cost in these subroutines. In many cases, the use of such consensus-type subroutines even leads to a setting where each node shares more data than what it actually collects at its sensors.

Based on the block partitioning of the DSFO problem in Figure 1, a tempting alternative strategy could be to use a nonlinear Gauss-Seidel method [12], [38], or so-called block coordinate descent algorithms. These are iterative algorithms in which a block of variables in X is optimized while keeping all others fixed, and where the fixed blocks change across iterations. By selecting the blocks of X according to the nodes (i.e., the X_k's in Figure 1), each iteration of the nonlinear Gauss-Seidel method can then be "outsourced" to the node that is responsible for optimizing the corresponding coordinates, which makes it a better fit for the class of problems we are interested in. However, the convergence results for such nonlinear Gauss-Seidel methods often require conditions such as convexity assumptions or constraint sets that can be written as Cartesian products, where each factor corresponds to a constraint set for the block X_k [12], [38], which would not be satisfied in many spatial filtering optimization problems. Moreover, when optimizing the selected block of coordinates, forcing the other ones to remain constant leads to a new optimization problem that is often different from the original problem, and which can be significantly more difficult to solve.

In this paper, we introduce a generic distributed algorithm, referred to as the Distributed Adaptive Signal Fusion (DASF) algorithm, which can be used to solve generic linear DSFO problems. In each iteration of the algorithm, a node within the network is selected to receive compressed data from other nodes and to locally solve a lower-dimensional version of the original network-wide problem. A convenient property is that an instance of the same algorithm that solves the centralized network-wide optimization problem can be used to also solve the local (compressed) problems at each iteration. The compression allows to reduce bandwidth and energy usage in the network, while we still achieve convergence to the centralized solution. Moreover, since the locally constructed problem is of lower dimension, the computational cost required to solve it is also smaller compared to solving the network-wide problem. This makes the proposed algorithm also attractive in fully deterministic distributed settings where computational or memory resources are limited.

Various existing distributed algorithms can be shown to be special cases of our proposed DASF algorithm, including distributed algorithms for generalized eigenvalue decomposition (GEVD) [39], spatial principal component analysis (PCA) [40], least squares (LS), minimum mean square error (MMSE) or multi-channel Wiener filtering (MWF) [41], [42], linearly constrained minimum variance (LCMV) beamforming [5], [43], [44], canonical correlation analysis (CCA) [45] and generalized CCA [46]. However, each of these algorithms was previously treated separately, with convergence proofs that were tailored to these specific cases. Our aim is to thoroughly define a unified algorithmic framework that contains these already existing algorithms but also extends to new DSFO problems. We also provide a toolbox available both in Matlab and Python to generate and validate new distributed algorithms within this framework [47].

The outline of this paper is as follows. In Section II, we formally define the framework setting and the assumptions used throughout this paper. We then propose the DASF algorithm for fully-connected networks in Section III, and later generalize it to general topologies in Section IV. Finally, we demonstrate the performance of the algorithm in a few new DSFO examples in Section V, thereby demonstrating the generalization properties of the DASF framework. We refer readers to the companion paper [1] for detailed analyses and proofs.

Notation: Uppercase letters are used to represent matrices and sets, the latter in calligraphic script, while scalars, scalar-valued functions and vectors are represented by lowercase letters, the latter in bold. We use the notation χ_q^i to refer to a certain mathematical object χ (such as a matrix, set, etc.) at node q and iteration i. The notation $\left(\chi^i\right)_{i\in\mathcal{I}}$ refers to a sequence of elements χ^i over every index i in the ordered index set $\mathcal{I}$. If it is clear from the context (often in the case where i is over all natural numbers), we omit the index set $\mathcal{I}$ and simply write $\left(\chi^i\right)_i$. A similar notation $\{\chi^i\}_{i\in\mathcal{I}}$ is used for

[1]In this paper, we adopt the terminology from the field of sensor arrays and multi-channel signal processing although the results are also applicable in a more general context, where $\mathbf{y}$ can be viewed as a generic feature vector in an $M-$dimensional feature space where distributed agents each observe a part of the feature vector.

non-ordered sets. Additionally, I_Q denotes the $Q \times Q$ identity matrix, $\mathbb{E}[\cdot]$ the expectation operator, $\text{tr}(\cdot)$ the trace operator, *BlkDiag* $(\cdot)$ the operator that creates a block-diagonal matrix from its arguments and $|\cdot|$ the cardinality of a set.

II. Problem description

Consider a set of K nodes, where $\mathcal{K} = \{1, \ldots, K\}$ denotes the set of nodes. The nodes are interconnected in a connected graph where an edge between nodes k and q implies that these nodes can share data (e.g., via a wireless link). The set of neighbors of node k, i.e., the nodes that are connected to node k, is denoted by $\mathcal{N}_k$ (which excludes node k itself).

Each node $k \in \mathcal{K}$ measures samples of a local M_k-channel signal $\mathbf{y}_k(t) \in \mathbb{R}^{M_k}$ at every time instance t. We define the network-wide M-channel signal $\mathbf{y}$ as

$$\mathbf{y}(t) = [\mathbf{y}_1^T(t), \ldots, \mathbf{y}_K^T(t)]^T, \tag{1}$$

with $M = \sum_k M_k$. All $\mathbf{y}_k$'s, and therefore also $\mathbf{y}$, are assumed to be (short-term) stationary and ergodic stochastic signals, such that their statistical properties can be properly estimated given a sufficiently large number of samples at different time instances.

The different channels of $\mathbf{y}$ can be spatially correlated across all nodes in the network, and we do not assume this correlation structure to be known. In a centralized setting, the channels of $\mathbf{y}$ can be linearly combined (fused) using a network-wide spatial filter $X \in \mathbb{R}^{M \times Q}$ with Q output signals (with $Q \ll M$), where we typically aim to find an optimal X such that the filter outputs $X^T\mathbf{y} \in \mathbb{R}^Q$ satisfy some optimality conditions. Typical examples of such filter design problems are listed in Table I, which can all be viewed as special cases of a general class of problems that will be formalized in the next subsection, which we refer to as (distributed) signal fusion optimization ((D)SFO) problems. This table is not exhaustive and various other signal fusion problems fit this framework (see e.g. [48], [49]). Note that all of these examples require knowledge of the *full* correlation matrix $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$, which can only be estimated in a centralized setting where the data from all the nodes are collected in a single fusion center, allowing to estimate the correlation between any two channel pairs of $\mathbf{y}$. One of the key strengths of our proposed DASF framework is that it avoids such a data centralization (in the sense that there is never a node which has access to all the channels of $\mathbf{y}$, i.e., $R_{\mathbf{yy}}$ cannot be constructed), while still achieving the solution of the centralized problem.

Additionally, we consider inner products of the form X^TB, where $B \in \mathbb{R}^{M \times L}$ is a deterministic (i.e., fixed and time-independent) matrix. The LCMV example in Table I is an example where such an inner product with a deterministic matrix appears. Similar to $\mathbf{y}$ in (1), this term is defined as

$$B = [B_1^T, \ldots, B_K^T]^T, \tag{2}$$

where we only require that B_k is known to node k. We note that the argument B allows a deterministic representation of $\mathbf{y}$ in which multiple time samples of $\mathbf{y}$ are stored in the columns of B. Nevertheless, we make a distinction between both expressions to emphasize time-adaptive properties of the algorithm (see Section II-B).

TABLE I
(D)SFO problems that are special cases of (3)

$\mathbf{y}$, $\mathbf{v}$ and $\mathbf{d}$ are multi-variate stochastic processes (signals). TRO is the trace ratio optimization problem and RR represents the ridge regression method. In the CCA case, the minimization is done with respect to X and W.

Problem	Cost function to minimize	Constraints
LCMV [5], [43], [44]	$\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$X^TB = H$
PCA [40]	$-\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$X^TX = I_Q$
GEVD [39]	$-\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$\mathbb{E}[X^T\mathbf{v}(t)\mathbf{v}^T(t)X] = I_Q$
TRO [50]	$-\frac{\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]}{\mathbb{E}[\|\|X^T\mathbf{v}(t)\|\|^2]}$	$X^TX = I_Q$
LS/MMSE /MWF [41], [42]	$\mathbb{E}[\|\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|\|^2]$	$X \in \mathbb{R}^{M \times Q}$
RR	$\mathbb{E}[\|\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|\|^2]$	$\text{tr}(X^TX) \leq \alpha^2$
CCA [45]	$-\mathbb{E}[\text{tr}(X^T\mathbf{y}(t)\mathbf{v}^T(t)W)]$	$\mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}^T(t)X] = I_Q$ $\mathbb{E}[W^T\mathbf{v}(t)\mathbf{v}^T(t)W] = I_Q$

It is noted that some of the problems in Table I involve a second signal $\mathbf{v} : \mathbf{v}(t) = [\mathbf{v}_1^T(t), \ldots, \mathbf{v}_K^T(t)]^T$ collected by the WSN (e.g., in the case of GEVD and CCA), and possibly another filter W to be optimized (e.g., in the case of CCA). This additional signal could either be derived from the same set of sensors (e.g., a time-lagged version of $\mathbf{y}$ as in [45], or observing $\mathbf{y}$ during two different regimes as in [39], [51]), or they could come from different types of sensors with which the nodes are equipped. In the remaining parts of this paper, we will typically consider the case of a single filter X, a single observed (multi-channel) sensor signal $\mathbf{y}$ and a single deterministic parameter B, yet all results can be easily generalized to multiple signals, parameters or filters. This generalization will be briefly addressed at the end of the next subsection and in Section III-C.

It is important to note here that every other quantity of the problem that is not represented by an inner product with X is assumed to be available at each node (e.g., H and $\mathbf{d}$ in the LCMV and LS / MMSE / MWF examples in Table I). This means that each node is able to evaluate the objective and constraint functions of the optimization problem solved over the network if it has access to $X^T\mathbf{y}$ and X^TB.

A. Scope of Signal Fusion Optimization Problems

We first provide a generic description of the signal fusion optimization (SFO) problems that will be covered in this paper. While this description may seem rather exotic at first, we will provide several examples throughout the paper to illustrate how it contains many familiar problems as a special case. The SFO problems studied in this paper can be written in the following way:

$$\begin{aligned} \underset{X \in \mathbb{R}^{M \times Q}}{\text{minimize}} \quad & \varphi\left(X^T\mathbf{y}(t), X^TB\right) \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) \leq 0, \quad \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) = 0, \quad \forall j \in \mathcal{J}_E, \end{aligned} \tag{3}$$

where φ and the η_j's are differentiable scalar- and real-valued functions, and the sets $\mathcal{J}_I$ and $\mathcal{J}_E$ represent the index sets of inequality and equality constraints respectively. Additionally, we define $\mathcal{J} = \mathcal{J}_I \cup \mathcal{J}_E$, and the number of constraints in total is given by $J = |\mathcal{J}|$.

An important observation is that X always appears in an inner product with $\mathbf{y}$ or B, which corresponds to a signal fusion or spatial filtering operation. Furthermore, note that the functions that contain a stochastic signal $\mathbf{y}$ as an argument must contain an operator to translate this stochastic variable into a deterministic loss or constraint function (i.e., the functions in instances of Problem (3) are deterministic, as any stochastic variable is converted into a deterministic value), for example through the use of an expectation operator (see Table I). In most practical cases, including those mentioned in Table I, the evaluation of φ requires the knowledge or estimation of the network-wide spatial covariance matrix $R_{\mathbf{yy}} = E[\mathbf{y}(t)\mathbf{y}^T(t)]$. In this work, we assume that this matrix is unknown, in which case the spatial correlation between the nodes should be learned on the fly by the proposed distributed algorithm.

The formulation (3) covers a wide range of popular spatial filtering and signal processing problems, including those shown in Table I. For example, for the LCMV case (first example in Table I), we have $\varphi(X^T\mathbf{y}(t)) = \mathbb{E}[||X^T\mathbf{y}(t)||^2]$, and $\eta_j(X^TB) = [X^TB - H]_j$ for each element $[X^TB - H]_j$ of the matrix $X^TB - H$. Note that quadratic terms of the form X^TX, which appear in some problems in Table I, should be seen as $(X^TB)\cdot(X^TB)^T$ with $B = I_M$. The reader is also referred to Section V in which a few extra examples are provided.

Finally, to simplify notation in various parts of this paper, we also define the differentiable functions f and h_j's, replacing φ and η_j's respectively, to describe Problem (3) as a function of X only, in which case $\mathbf{y}$ and B should be viewed as internal function parameters:

$$\begin{aligned} f(X) &\triangleq \varphi\left(X^T\mathbf{y}(t), X^TB\right), \\ h_j(X) &\triangleq \eta_j\left(X^T\mathbf{y}(t), X^TB\right),\ \forall j \in \mathcal{J}. \end{aligned} \tag{4}$$

Furthermore, we denote the constraint set of (3) as $\mathcal{S}$, the complete solution set as $\mathcal{X}^*$ and a single solution as X^*, i.e., $X^* \in \mathcal{X}^*$.

Note on further generalizations: The problem description (3) considers only one argument of each type $(X^T\mathbf{y}(t), X^TB)$ involving only one filter variable X, one stochastic signal $\mathbf{y}$ and one deterministic matrix B. However, this is merely for conciseness and intelligibility of the description of our framework, i.e., the framework can be straightforwardly generalized to multiple versions of variables and each of the two arguments in (3), e.g., to also cover the cases of GEVD and CCA in Table I. Formally, the full scope of SFO problems we consider is

$$\begin{aligned} \underset{X^{(a)},\forall a}{\text{minimize}}\ & \varphi\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, ...\right),\ \forall a,b,c \\ \text{subject to}\ & \eta_j\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, ...\right) \leq 0\ \ \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^{(a)T}\mathbf{y}^{(b)}(t), X^{(a)T}B^{(c)}, ...\right) = 0\ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{5}$$

Additionally, even though we restrict ourselves to real-valued arguments, the results can be extended to complex-valued arguments (with real-valued cost functions) based on standard techniques such as those explained in [11], [52], [53]. Further extensions are possible where each node minimizes a different function, as in [42], [54], but this is beyond the scope of this paper.

B. Adaptivity and Approximation in Practical Settings

The problems we are interested in typically involve an expectation operator over random signals with unknown distributions. In practical settings, the expectation operators in the objective and constraint functions of (3) are usually approximated using sample averages [55]–[59]. The expectation operator over the distribution of $\mathbf{y}(t)$ for a generic deterministic function g taking $\mathbf{y}(t)$ as an argument is then approximated[2] as

$$\mathbb{E}[g(\mathbf{y}(t))] \approx G(Y(t)) \triangleq \frac{1}{N}\sum_{\tau=t}^{t+N-1} g(\mathbf{y}(\tau)), \tag{6}$$

where $Y(t) = [\mathbf{y}(t), \ldots, \mathbf{y}(t+N-1)]$ denotes a matrix that contains N observations of $\mathbf{y}$, starting with the observation at time sample t. Here, it is assumed that the signal $\mathbf{y}$ is ergodic such that the expectation operators can be accurately approximated using (6). In particular for the second-order statistics, which are commonly encountered in signal processing problems (see Table I), the approximation (6) results in

$$\mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)] \approx \frac{1}{N}Y(t)Y^T(t). \tag{7}$$

Table II gives the practical approximations of some commonly encountered functions in problems of interest (including expressions found in Table I) using (6). Note that the stationarity assumption removes any time dependence from the problems of interest such that any window of contiguous time samples of $\mathbf{y}$ can be used.

Throughout this paper, we assume stationarity for mathematical tractability, which is a common assumption in the analysis of adaptive filters [55], [61]–[64]. However, in practice, we do not require the signals to be fully stationary, as long as the dynamics in the underlying signal statistics are sufficiently slow, such that (6) gives a reasonable approximation. In this case, the solution X^* of (3) becomes time-dependent. When the DASF framework is applied in such an adaptive context, the targeted instance of Problem (3) is effectively replaced by its sample average counterpart, where the statistics are regularly re-estimated in a block-based fashion, i.e., each time a new block of N samples becomes available. As we will explain in Sections III and IV, every new block of N samples will initiate one new iteration of the DASF algorithm, i.e., the iterations of DASF are spread over different sample blocks, such that the algorithm becomes time-recursive (implicitly assuming $G(Y(t)) \approx G(Y(t+N))$). We will demonstrate in Section V-D through an example that the proposed DASF algorithm can indeed track slow changes in the signal statistics, and is able to recover from abrupt changes.

[2]Convergence of the approximation to the true expectation for $N \to +\infty$ is studied in the stochastic optimization literature. We refer the reader to the sample average approximation method in particular [60].

TABLE II
PRACTICAL APPROXIMATIONS OF COMMON FUNCTIONS

Approximations computed in practice to evaluate some commonly encountered functions using (6) and N observations of the stochastic signals, e.g., $\{\mathbf{y}(\tau)\}_{\tau=t}^{t+N-1}$. $Y(t)$ denotes the sample matrix of $\mathbf{y}$ for N observations, $Y = [\mathbf{y}(t), \ldots, \mathbf{y}(t+N-1)]$, while V and D are similarly defined for $\mathbf{v}$ and $\mathbf{d}$ respectively.

$\mathbb{E}[g(\mathbf{y}(t))]$	$G(Y(t))$
$\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]$	$\|\|X^TY(t)\|\|_F^2/N$
$\mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}^T(t)X]$	$X^TY(t)Y^T(t)X/N$
$\mathbb{E}[\|\|\mathbf{d}(t) - X^T\mathbf{y}(t)\|\|^2]$	$\|\|D(t) - X^TY(t)\|\|_F^2/N$
$\mathbb{E}[\text{tr}(X^T\mathbf{y}(t)\mathbf{v}^T(t)W)]$	$\text{tr}(X^TY(t)V^T(t)W)/N$

C. General Assumptions

In order for our DASF algorithm to be applicable and achieve convergence and optimality, the problem (3) must satisfy some sufficient conditions, which are listed in this subsection for completeness, while the technical details are explained in the companion paper [1]. It is noted that these conditions are usually satisfied for all examples listed in Table I (except in some contrived cases) and we refer the reader to [1] for some examples on how these conditions can be checked in practice. In addition to these sufficient conditions, we restate the implicit assumption from Section II-A that the functions f and h_j in (4) are smooth functions, i.e., they are continuously differentiable over the variable X on their respective domain, or equivalently, the functions φ and η_j are continuously differentiable over X for any $\mathbf{y}$ and B.

Assumption 1. *The targeted instance of Problem (3) is well-posed[3], in the sense that the solution set is not empty and varies continuously with a change in the parameters of the problem.*

Note that since in practice, the DASF algorithm will be used for solving a particular instance of Problem (3), Assumption 1 is only required for that particular problem and not all problems within the scope of the framework. As an example, consider the PCA problem of Table I. It can be shown that the PCA problem satisfies Assumption 1 if the covariance matrix of $\mathbf{y}$ is positive definite and its $Q+1$ largest eigenvalues are all distinct.

Assumption 2. *The linear independence constraint qualifications (LICQ) hold at the solutions of Problem (3), i.e., the solutions satisfy the Karush-Kuhn-Tucker (KKT) conditions.*

If X^* is a solution of Problem (3), then Assumption 2 implies that the gradients $\nabla_X h_j(X^*)$, $j \in \mathcal{J}^*$, are linearly independent[4], where $\mathcal{J}^* \subseteq \mathcal{J}$ is the set of all indices j for which $h_j(X^*) = 0$. We refer the reader to [67] for further details on constraint qualifications. If there is no constraint function η_j in Problem (3), Assumption 2 implies that $\nabla_X f(X^*) = 0$.

Assumption 3. *f has compact sublevel sets in $\mathcal{S}$, i.e., for all $m \in \mathbb{R}$, $\{X \in \mathcal{S} \mid f(X) \leq m\}$ is compact.*

Note that in $\mathbb{R}^{M\times Q}$, compactness is equivalent to closedness and boundedness of a set, which is a relatively mild condition. In fact, as is shown in [1], the assumption can be further relaxed by only requiring that at least one sublevel set $\{X \in \mathcal{S} \mid f(X) \leq f(X^0)\}$ is compact, where X^0 is the initialization point of the DASF algorithm.

Moreover, the convergence proof in [1] requires an additional sufficient condition that is akin to the LICQ. We postpone its definition to [1] because of its technical nature, and because it is a relatively mild condition, which is generally satisfied in practice. Nevertheless, an important implication of this condition, which is relevant to disclose at this point, will be that it imposes an upper bound on the number of constraints J the problem is allowed to have (see [1]):

$$J \leq \min\left(\frac{Q^2}{K-1}\sum_{k\in\mathcal{K}}|\mathcal{N}_k|,\ (1+\min_{k\in\mathcal{K}}|\mathcal{N}_k|)Q^2\right). \quad (8)$$

Here, K is the total number of nodes, $|\mathcal{N}_k|$ is the number of neighbors of node k, and Q is the number of columns of X.

We also make the implicit assumption that a (centralized) algorithm is available to solve (with arbitrary accuracy) the targeted problem instance, i.e., the instance of (3) for which we aim to design a distributed algorithm. This is a reasonable premise, since it makes little sense to design a distributed algorithm for a problem that cannot even be solved in a centralized setting. Nevertheless, it is important as our DASF framework will use the same solver to find solutions of compressed versions of the targeted problem at each node, as will be explained next. We note that there are no restrictions on the solver, which can be chosen freely (e.g., closed-form solutions, steepest descent methods, interior point methods, trust region methods, etc.).

III. DISTRIBUTED ADAPTIVE SIGNAL FUSION IN FULLY-CONNECTED NETWORKS (FC-DASF)

For the sake of an easier exposition, let us first consider the special case of a fully-connected network where data transmitted by any node is received by every other node (more general topologies are treated in Section IV). As the optimization problem (3) depends on the full signal $\mathbf{y}$ and its (unknown) spatial correlation across the nodes, the nodes would need to share some information between each other. However, sharing the full signal $\mathbf{y}$ would require significant bandwidth and consume a large amount of power. Instead, each node will linearly compress its local M_k-channel signal into an $R-$channel signal (with $R < M_k$) before broadcasting it to the other nodes in the network. At iteration i, the compression is done by multiplying the signal $\mathbf{y}_k$ at node k with an $M_k \times R$ matrix F_k^i, which we refer to as the local compressor at node

[3]The notion of (generalized Hadamard) well-posedness we require is based on [65], [66]. The main difference is that we require the map from the parameter (inputs of the problem) space to the solution space to be continuous instead of upper semicontinuous, which is required for the convergence proof of the DASF algorithm. These technical details are presented in [1].

[4]A set of matrices $\{A_j\}_{j\in\mathcal{J}}$ is linearly independent when $\sum_{j\in\mathcal{J}}\alpha_j A_j = 0$ is satisfied if and only if $\alpha_j = 0$, $\forall j \in \mathcal{J}$, or equivalently, when $\{\text{vec}(A_j)\}_{j\in\mathcal{J}}$ is a set of linearly independent vectors, where $\text{vec}(\cdot)$ is the vectorization operator.

k. The $R-$dimensional compressed signal resulting from this compressive spatial filtering can be written as

$$\widehat{\mathbf{y}}_k^i(t) \triangleq F_k^{iT}\mathbf{y}_k(t). \tag{9}$$

The deterministic parameters B_k are similarly compressed to obtain $\widehat{B}_k^i$, which are also broadcast between nodes. The index i emphasizes that these compressors are not constant and will be iteratively updated across time. Note that the iteration index i is also added to the stochastic signal $\widehat{\mathbf{y}}_k$ in order to indicate that the content of the signal (and hence its statistics) changes in each iteration due to an update of the underlying compression matrix. However, this does not imply that we iterate over a single batch of samples of this signal, i.e., an update from $\widehat{\mathbf{y}}_k^i$ to $\widehat{\mathbf{y}}_k^{i+1}$ (or F_k^i to F_k^{i+1}) only affects future samples of $\widehat{\mathbf{y}}_k$ that are collected after performing the update. As a result, the compressor F_k operates as a block-adaptive filter, which updates its coefficients after every new block of N samples, such that F_k^i operates on the samples in the data matrix $Y(iN)$, whereas F_k^{i+1} operates on the samples in $Y(iN+N)$, where $Y(t)$ is defined as in (6). In other words, an update of the compressor F_k affects how future samples of $\mathbf{y}_k$ are compressed, yet previously collected samples will not be re-compressed or re-broadcast.

Equation (9) results in a compression ratio of M_k/R. The compression implies that the nodes do not have full access to the network-wide signal $\mathbf{y}$. Nevertheless, we will show that an optimal solution $X^* \in \mathcal{X}^*$ of Problem (3) can be achieved if $R = Q$, where Q is the number of columns of X, i.e., the number of output signals of the filter X (in many cases only a single-channel output is desired, such that $R = Q = 1$). In a fully-connected network, this implies that we must assume that $Q < M_k$ in order to achieve a bandwidth reduction at node k. However, in the case of more general topologies, we can also achieve this for $Q \geq M_k$ (see Section IV).

After introducing the DASF algorithm for (3) in the next subsection, we will provide insights on the relationship between the network-wide problem and the problems solved at each node in Section III-B. Extensions to the more general form (5) will be presented in Section III-C.

A. Algorithm Derivation

Consider the partitioning of the optimization variable X in per-node sub-blocks, i.e.,

$$X = [X_1^T, \ldots, X_K^T]^T, \tag{10}$$

where $X_k \in \mathbb{R}^{M_k \times Q}$. This way, every X_k has a corresponding local signal $\mathbf{y}_k$, i.e., the part of X that is applied to $\mathbf{y}_k$ in the expression $X^T\mathbf{y}$, such that we can write

$$X^T\mathbf{y}(t) = \sum_{k\in\mathcal{K}} X_k^T\mathbf{y}_k(t). \tag{11}$$

A similar observation for the local deterministic terms B_k implies that we can write the objective φ using the local filters and data:

$$\begin{aligned} f(X) &= \varphi(X^T\mathbf{y}(t), X^TB) \\ &= \varphi\left(\sum_{k\in\mathcal{K}} X_k^T\mathbf{y}_k(t), \sum_{k\in\mathcal{K}} X_k^TB_k\right). \end{aligned} \tag{12}$$

The constraint functions η_j can also be written in a similar way. Therefore, we are able to express the full optimization problem (3) using X_k's, $\mathbf{y}_k$'s and B_k's. The main idea behind the DASF framework is to partially reconstruct and solve a compressed version of Problem (3) at any selected node, which is referred to as the "updating node", and which changes from iteration to iteration. Let us now set $R = Q$ and

$$F_k^i = X_k^i, \tag{13}$$

where X_k^i corresponds to the local estimate of X_k at node k at iteration i. Stacking them together as in (10), we obtain the estimate X^i of the global variable X at iteration i. This means that, within the DASF algorithm, the X_k^i's act both as compressors and as part of the optimization variable. Combining (9) and (13), the compressed signal that is broadcast by node k can be written as

$$\widehat{\mathbf{y}}_k^i(t) \triangleq X_k^{iT}\mathbf{y}_k(t) \in \mathbb{R}^Q, \tag{14}$$

which implies that the network-wide filter output at iteration i can be computed as the sum of the compressed signals in (14), i.e.,

$$\widehat{\mathbf{y}}^i(t) \triangleq X^{iT}\mathbf{y}(t) = \sum_{k\in\mathcal{K}} X_k^{iT}\mathbf{y}_k(t) = \sum_{k\in\mathcal{K}} \widehat{\mathbf{y}}_k^i(t). \tag{15}$$

In a similar way, the compressed deterministic terms at node k are

$$\widehat{B}_k^i \triangleq X_k^{iT}B_k \in \mathbb{R}^{Q\times L}, \tag{16}$$

and we have

$$\widehat{B}^i \triangleq X^{iT}B = \sum_{k\in\mathcal{K}} X_k^{iT}B_k = \sum_{k\in\mathcal{K}} \widehat{B}_k^i. \tag{17}$$

Since the network is assumed to be fully-connected, each node has access to (observations of) all signals $\widehat{\mathbf{y}}_k^i$, such that each node can compute the filter outputs (15). Let q be the updating node at iteration i, and define the stacked vector containing all available signals at node q as

$$\widetilde{\mathbf{y}}_q^i(t) \triangleq [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_1^{iT}(t), \ldots, \widehat{\mathbf{y}}_{q-1}^{iT}(t), \widehat{\mathbf{y}}_{q+1}^{iT}(t), \ldots, \widehat{\mathbf{y}}_K^{iT}(t)]^T, \tag{18}$$

which contains $\widetilde{M}_q \triangleq M_q + Q(K-1)$ channels. Note that node q only has access to uncompressed observations of $\mathbf{y}_q$ and a corresponding batch of compressed observations of all the other nodes. Similarly, the matrix containing all available deterministic terms is obtained by stacking B_q which is available at node q and the compressed $\widehat{B}_k^i$'s received from other nodes:

$$\widetilde{B}_q^i \triangleq [B_q^T, \widehat{B}_1^{iT}, \ldots, \widehat{B}_{q-1}^{iT}, \widehat{B}_{q+1}^{iT}, \ldots, \widehat{B}_K^{iT}]^T, \tag{19}$$

which is an $\widetilde{M}_q \times L$ matrix. Based on (18) and (19), we define a new local variable $\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q\times Q}$ at node q, such that we are able to formulate a local optimization problem using only data available at node q at iteration i:

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad & \varphi(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i) \\ \text{subject to} \quad & \eta_j(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{20}$$

A key observation here is the similarity between (20) and (3). This means that node q can locally apply the same solver as the one used for solving the centralized problem, albeit on a problem of smaller size. Note that this implies that the computational cost required to solve Problem (20) is smaller compared to solving (3).

At iteration i, node q solves the local problem (20), and we denote its solution as $\widetilde{X}_q^{i+1}$, which can be partitioned as

$$\widetilde{X}_q^{i+1} = [X_q^{(i+1)T}, G_1^{(i+1)T}, \ldots, G_{q-1}^{(i+1)T}, G_{q+1}^{(i+1)T}, \ldots, G_K^{(i+1)T}], \tag{21}$$

where X_q^{i+1} is $M_q \times Q$ and each G_k^{i+1} is $Q \times Q$. By comparing (21) with (18), we see that G_k^{i+1} refers to the part of $\widetilde{X}_q$ that is multiplied with the received compressed data $\widehat{\mathbf{y}}_k^i$ from node k in the inner product $\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t)$ in (20). Since $\widehat{\mathbf{y}}_k^i(t) = X_k^{iT}\mathbf{y}_k(t)$, we can instead multiply the compressor X_k^i at node k with this matrix G_k^{i+1}. As a result, each node k in the network updates its local X_k as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q \\ X_k^i G_k^{i+1} & \text{if } k \neq q, \end{cases} \tag{22}$$

where X_q^{i+1} and G_k^{i+1} are obtained from the partitioning (21) of $\widetilde{X}_q^{i+1}$. Since the updating node q does not have access to the X_k^i of the other nodes $k \neq q$, it needs to communicate the matrices G_k^{i+1} to the other nodes, so that they can update their local X_k as well[5]. A block diagram of this process is provided in Figure 2.

If the minimization (20) has multiple solutions, an ambiguity exists on the choice of the local variable, which can be resolved by selecting a specific solution at each iteration. We propose to select the solution $\widetilde{X}_q^{i+1}$ for which the distance $||\widetilde{X}_q^{i+1} - \widetilde{X}_q^i||_F$ is minimal, where $\widetilde{X}_q^i$ is defined as

$$\widetilde{X}_q^i = [X_q^{iT}, I_Q, \ldots, I_Q]^T. \tag{23}$$

The choice of the Frobenius norm $||\cdot||_F$ as a distance metric is arbitrary. Other distance functions d can also be used (and might be better suited for the specific instance of Problem (3) at hand), as long as they are continuous and satisfy $d(X,Y) = 0 \iff X = Y$. These conditions on d are needed for the convergence of the proposed method, as explained in [1].

All the steps of the FC-DASF algorithm as explained above are summarized in Algorithm 1. Algorithm 1 converges under mild technical conditions to an optimal filter X^* solving the network-wide problem (3). The convergence results with detailed analysis and proofs can be found in the companion paper [1].

Remark 1. It is noted that a transmission of the compressed signal $\widehat{\mathbf{y}}_k^i$ at node k corresponds in practice to transmitting a batch with the N most recent time samples of $\widehat{\mathbf{y}}_k^i$. This allows for the receiving node q to estimate the necessary signal statistics to evaluate or optimize (20), where a larger value of N results in a closer approximation of the true value (remember that the objective function in (3) and (20) has a built-in operator to transform the stochastic signal $\mathbf{y}$ into a deterministic loss, which in practice is usually replaced with an average over N samples, as in Table II). Leveraging the (short-term) stationarity assumption, different batches of N samples are used in each iteration, such that the communication bandwidth becomes independent of the number of iterations. Therefore, Algorithm 1 behaves similarly to an adaptive filter which learns over time how to optimally filter

[5]Note that the communication cost to transmit these G_k matrices is negligible compared to the transmission of a batch of observations of $\widehat{\mathbf{y}}_k$'s (see also Remark 1).

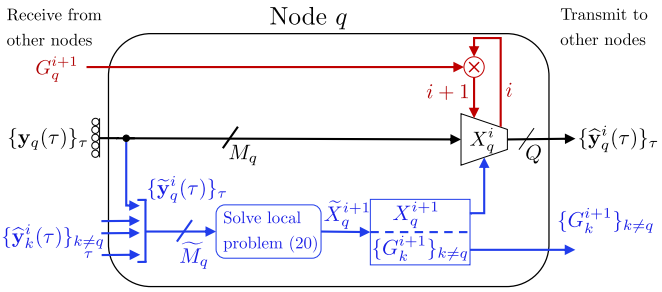

Fig. 2. Block diagram representation of the steps followed by a given node q in the FC-DASF algorithm. The black part is executed for any sample time t at each node. The red and blue parts are only executed at each iteration increment $i \to i+1$, where the blocks in blue are only executed when node q is the updating node. Otherwise, the part in red is carried out. Node q has always access to its own signal samples $\mathbf{y}_q(t)$ measured at its own sensors (represented by rings, in black), while the compressed signal samples $\widehat{\mathbf{y}}_k^i(t)$ are transmitted to node q by the respective nodes k (represented by arrows in blue). For intelligibility, we omitted the data flow of the expression $X^T B$ from the diagram.

Algorithm 1: Fully-Connected Distributed Adaptive Signal Fusion (FC-DASF) Algorithm
Code available in [47]

output: X^*

Initialize X^0, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) Every node k collects a new batch of N samples of $\mathbf{y}_k$ (see Remark 1), compresses these to N samples of $\widehat{\mathbf{y}}_k^i$ using (14) and transmits them to node q.
  $\widehat{B}_k^i$ is computed using (16) and transmitted to node q.
  at *Node* q **do**
    2a) Compute $\widetilde{X}_q^{i+1}$ as the solution of (20). If the solution is not unique, select the solution which minimizes $||\widetilde{X}_q^{i+1} - \widetilde{X}_q^i||_F$ with $\widetilde{X}_q^i$ defined in (23).
    2b) Partition $\widetilde{X}_q^{i+1}$ as in (21).
    2c) Transmit G_k^{i+1} to node k for every $k \neq q$.
  end
  3) Every node updates X_k^{i+1} according to (22).
  $i \leftarrow i+1$

Note: Each iteration uses a different batch of N samples in step 1, i.e., the iterations can be spread over different time segments in order to avoid retransmitting the same batch of N samples across the network (see also Remark 1). This makes the sample time index t coupled to the iteration index i.

newly observed samples (based on past samples). In a tracking context where the signal statistics of $\mathbf{y}$ change over time, the algorithm can still be applied if the statistics change slower than the convergence speed of the algorithm.

The DASF framework could in principle also be applied in a batch-mode (non-adaptive) framework, in which all operations are performed entirely on a single batch of samples (instead of spreading out the iterations over different sample batches of length N). In this case, the argument $X^T\mathbf{y}$ can be dropped and the argument X^TB can be used to represent the batch of samples, in which all available samples of $\mathbf{y}$ are stored in the columns of B.

Remark 2. Although each node is able to communicate with every other node in a fully-connected network, Algorithm 1 is still distributed in nature. Indeed, the network-wide data is never centralized, i.e., the updating nodes q have never access to $\mathbf{y}$ or B, but only to $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$, and therefore cannot estimate any network-wide statistics such as $\mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$, whereas a centralized solver has access to this information.

B. Link between the Central and Local Problems

In this subsection, we further explain the link between the central problem (3) and the local problems (20) at the updating node q, where the latter can be viewed as a parameterized version of the former. This will provide some additional insights and show some useful properties of the DASF framework. Their relationship can be described by means of the transformation matrix:

$$C_q^i = \left[\begin{array}{c|c|c} 0 & \Theta_{<q}^i & 0 \\ \hline I_{M_q} & 0 & 0 \\ \hline 0 & 0 & \Theta_{>q}^i \end{array}\right] \in \mathbb{R}^{M\times\widetilde{M}_q}, \tag{24}$$

where $\Theta_{<q}^i = \textit{BlkDiag}(X_1^i,\ldots,X_{q-1}^i)$ and $\Theta_{>q}^i = \textit{BlkDiag}(X_{q+1}^i,\ldots,X_{q+1}^i)$. Then, from (14) and (18), one can validate that

$$\widetilde{\mathbf{y}}_q^i(t) = C_q^{iT}\mathbf{y}(t), \tag{25}$$

while (16) and (19) result in

$$\widetilde{B}_q^i = C_q^{iT}B. \tag{26}$$

Using these relationships in (20), we see that

$$\begin{aligned}\varphi(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i) &= \varphi\left(\widetilde{X}_q^T(C_q^{iT}\mathbf{y}(t)),\widetilde{X}_q^T(C_q^{iT}B)\right)\\ &= \varphi\left((C_q^i\widetilde{X}_q)^T\mathbf{y}(t),(C_q^i\widetilde{X}_q)^TB\right)\\ &= f(C_q^i\widetilde{X}_q).\end{aligned} \tag{27}$$

Similarly, we find that, $\forall j\in\mathcal{J}$:

$$\begin{aligned}\eta_j(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i) &= \eta_j\left((C_q^i\widetilde{X}_q)^T\mathbf{y}(t),(C_q^i\widetilde{X}_q)^TB\right)\\ &= h_j(C_q^i\widetilde{X}_q).\end{aligned} \tag{28}$$

This implies that the local optimization variable $\widetilde{X}_q$ defines a parameterization of the global variable X. Indeed, if we define

$$\widetilde{X}_q = [X_q^T, G_1^T,\ldots,G_{q-1}^T,G_{q+1}^T,\ldots,G_K^T]^T, \tag{29}$$

where X_q is $M_q\times Q$ and every G_k is $Q\times Q$, we have at iteration i (for the updating node q)

$$X = C_q^i\widetilde{X}_q = \begin{bmatrix} X_1^i\boxed{G_1} \\ \vdots \\ X_{q-1}^i\boxed{G_{q-1}} \\ \boxed{X_q} \\ X_{q+1}^i\boxed{G_{q+1}} \\ \vdots \\ X_K^i\boxed{G_K} \end{bmatrix}. \tag{30}$$

Note that only the framed variables in (30) appear as optimization variables in the local problem (20), which is clear from comparing (21) with (29). This shows that the updating node q only has the full freedom to update X_q, i.e., its local compressor. The remaining parts of X can only change up to a multiplication from the right by a matrix G_k, $k\neq q$ when it is node q's turn to solve the local problem. Therefore, by sequentially changing the updating node across iterations, we allow every node to fully update its own local compressor while only manipulating the other sub-blocks of X within their respective column spaces.

The solution $\widetilde{X}_q^{i+1}$ of the local problem (20) at node q and iteration i, which can also be written as

$$\begin{aligned}\widetilde{X}_q^{i+1} &\triangleq \underset{\widetilde{X}_q\in\widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}}\ f\left(C_q^i\widetilde{X}_q\right),\\ &= \underset{\widetilde{X}_q\in\widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}}\ \varphi\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right),\end{aligned} \tag{31}$$

where $\widetilde{\mathcal{S}}_q^i$ denotes the constraint set of (20), defines a new point X^{i+1} for the global problem (3) via (30).

In the following lemma, we show that the global variable X^i produced by the DASF algorithm[6] always satisfies the global constraint set $\mathcal{S}$ for any iteration $i>0$.

Lemma 1. *For any iteration $i>0$,*

$$\widetilde{X}_q\in\widetilde{\mathcal{S}}_q^i \iff C_q^i\widetilde{X}_q\in\mathcal{S}. \tag{32}$$

In particular, $X^i\in\mathcal{S}$ and $\widetilde{X}_q^i\in\widetilde{\mathcal{S}}_q^i$ for all $i>0$.

Proof. From (28), it follows automatically that any point $\widetilde{X}_q$ in the constraint set of the local problem (20) has a corresponding point $X=C_q^i\widetilde{X}_q$ in the constraint set of the global problem (3). This implies that, if $\widetilde{X}_q$ is a feasible point of (20), the point X parameterized by $\widetilde{X}_q$, such that $X=C_q^i\widetilde{X}_q$, is a feasible point of (3), and vice versa, which proves (32).

From (29)-(30), we find that the point X^i (before the update at iteration i) is equal to $X^i=C_q^iX_q^i$ with $\widetilde{X}_q^i$ defined in (23). Similarly, we know (by construction) that $X^{i+1}=C_q^i\widetilde{X}_q^{i+1}$. Since $\widetilde{X}_q^{i+1}$ is the solution of (20), X^{i+1} must be a feasible point of (3), which follows from (32). As this holds for all $i\geq 0$, X^i is then also a feasible point of (3), i.e., $X^i\in\mathcal{S}$, if

[6]The proof of Lemma 1 also holds for the general topology-independent DASF algorithm in Section IV.

$i > 0$. Since $X^i = C_q^i \widetilde{X}_q^i$, and using (32), we find that $\widetilde{X}_q^i$ as defined in (23) is a feasible point of (20), i.e., $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$. □

The results of Lemma 1 mean that all points $(X^i)_{i>0}$ generated by the algorithm will be in the constraint set of the global problem (3). Additionally, the final result states that $\widetilde{X}_q^i$ itself is a feasible point of (20), which is important to achieve convergence and a monotonic decrease in f, as it allows the algorithm to stay in the current point X^i if no reduction in f can be obtained at node q in iteration i, in which case $X^{i+1} = X^i$ (a formal proof is given in [1]).

Remark 3. The fact that the local problem (20) inherits the structure from the global problem (3) is one of the key differences between the DASF framework and the nonlinear Gauss-Seidel method (sometimes referred to as the alternating optimization method), which would consist of only updating X_q^i in (3), while freezing the other X_k^i's $\forall k \neq q$. In the latter case, the subproblem that has to be solved in each iteration typically has a different structure than the original one, often leading to problems that are more difficult to solve or for which a straightforward solver might not even exist. Moreover, the extra degrees of freedom to manipulate the X_k's of other nodes through the G_k matrices in the parameterization (30) allow to optimize X^i over a larger subset of $\mathbb{R}^{M\times Q}$ in each individual iteration, leading to larger descents in the loss function f in each iteration. In Gauss-Seidel methods, these G_k matrices do not exist, i.e., all X_k's are fixed except one.

Remark 4. By combining (25) and (30), we find that

$$\widetilde{X}_q^{(i+1)T} \widetilde{\mathbf{y}}_q^i(t) = X^{(i+1)T} \mathbf{y}(t), \tag{33}$$

which means that node q always has access to the filtered signal $X^{(i+1)T}\mathbf{y} = \widehat{\mathbf{y}}^{i+1}$ based on the most recent version of the filter X^{i+1}. If any of the other nodes would act as a data sink, the updating node q has to forward the observations of $\widehat{\mathbf{y}}^{i+1}$ to the data sink.

C. Multiple Signals, Deterministic Terms and Variables

The DASF algorithm can be immediately adapted to the generalized version of (3) defined in (5), i.e., with multiple filters, signals and deterministic terms. In the case of multiple signals (stochastic variables) or deterministic terms appearing in the problem in the forms $X^T\mathbf{y}$ and $X^T B$ respectively, every single object is treated as previously presented, creating new data to be communicated between the nodes for every additional expression. Examples include the GEVD and TRO given in Table I having two signals $\mathbf{y}$ and $\mathbf{v}$. An example of a problem with two deterministic terms is given in Section V.

Similarly, we could also consider cases with multiple optimization variables. Taking the example of CCA given in Table I, the two optimization variables (X, W) appear as $X^T\mathbf{y}$ and $W^T\mathbf{v}$ in the problem. Then, nodes $k \neq q$ compress their signals as $\widehat{\mathbf{y}}_k^i = X_k^{iT}\mathbf{y}_k$ and $\widehat{\mathbf{v}}_k^i = W_k^{iT}\mathbf{v}_k$ and transmit them to node q. Then, node q solves its local problem as

$$\left(\widetilde{X}_q^{i+1}, \widetilde{W}_q^{i+1}\right) = \underset{(\widetilde{X}_q, \widetilde{W}_q)\in\widetilde{\mathcal{S}}_q^i}{\operatorname{argmin}} \varphi\left(\widetilde{X}_q^T \widetilde{\mathbf{y}}_q^i(t), \widetilde{W}_q^T \widetilde{\mathbf{v}}_q^i(t)\right). \tag{34}$$

Partitioning $\widetilde{W}_q$ as

$$\widetilde{W}_q = [W_q^T, H_1^T, \ldots, H_{q-1}^T, H_{q+1}^T, \ldots, H_K^T]^T, \tag{35}$$

similarly to (29) for $\widetilde{X}_q$, node q sends the G_k^{i+1} and H_k^{i+1}'s to corresponding nodes k such that they update their local variables as in (22). The same procedure can be applied for more than two variables.

Remark 5. The communication burden required to transmit the compressed terms $X_k^{iT}B_k$ is minimal because B_k's are deterministic parameters, as opposed to the signals $\mathbf{y}_k$, which require sending batches of multiple compressed observations in each iteration to estimate the signal statistics at the updating node (see Remark 1). The communication cost of the deterministic part can be further reduced when we have an expression of the form $(X^T B^{(1)}) \cdot (X^T B^{(2)})^T = X^T \Gamma X$, where Γ is a block-diagonal deterministic matrix written as

$$\Gamma = \mathit{BlkDiag}(\Gamma_1, \ldots, \Gamma_K), \tag{36}$$

where each Γ_k is known by node k. Each node could then transmit $X_k^{iT}\Gamma_k X_k^i \in \mathbb{R}^{Q\times Q}$ at iteration i instead of $X_k^{iT}B_k^{(1)}$ and $X_k^{iT}B_k^{(2)}$, which is more efficient when $Q < 2L$. Although expression (36) is quite specific, it is encountered often in spatial filtering, for example for orthogonality constraints such as $X^TX = I_Q$ or ℓ_2-norm regularization terms. For example, consider the PCA constraint $X^TX = I_Q$, where $B^{(1)} = B^{(2)} = I_M$. Then it is sufficient for the nodes to transmit $X_k^{iT}X_k^i$ instead of X_k^i.

IV. Topology-Independent DASF (TI-DASF)

Until this point, we have considered fully-connected WSNs only, where every node in the network is a neighbor of every other node. In this section, we extend our discussions and describe the DASF algorithm for other network topologies. For this purpose, we first consider star topologies which are helpful to introduce the main idea, which will then lead to generalizations to tree topologies and finally to any (connected) network topology.

A. Star Topologies

We keep the same definitions introduced in Sections II and III and consider now that the WSN has a central node $c \in \mathcal{K}$ to which every other node $k \neq c$ is connected (having only node c as a neighbor). In the case where the center node c is the updating node, we have the same setting as the fully-connected case and the steps described in the previous section apply, therefore we present here a strategy when the updating node $q \neq c$. A straightforward approach would be to let node c relay all the data from all other nodes to create a virtually fully-connected network. However, this would put high bandwidth requirements on node c, which would not scale well with respect to network size. Instead, we claim that it is sufficient for node q to have access to the signal defined in (15), which is slightly rewritten here as

$$\widehat{\mathbf{y}}^i(t) = X^{iT}\mathbf{y}(t) = X_q^{iT}\mathbf{y}_q(t) + \sum_{k\neq q} \widehat{\mathbf{y}}_k^i(t). \tag{37}$$

Note that the second term can be computed at node c (which includes node c's own sensor observations $\mathbf{y}_c$, as well as the compressed signals $\widehat{\mathbf{y}}_k^i$ of the nodes $k \neq q$) such that only the sum has to be forwarded instead of the individual terms. The data received by node q from c is then a $Q-$channel signal given by

$$\widehat{\mathbf{y}}_{c\to q}^i(t) \triangleq \sum_{k\in\mathcal{K}\setminus\{q\}} \widehat{\mathbf{y}}_k^i(t), \tag{38}$$

and a similar expression can be written for the deterministic terms:

$$\widehat{B}_{c\to q}^i \triangleq \sum_{k\in\mathcal{K}\setminus\{q\}} \widehat{B}_k^i. \tag{39}$$

From the perspective of node q, the network consists of only itself and node c, so by following analogous steps to Algorithm 1, node q creates its vector of locally available data:

$$\begin{aligned} \widetilde{\mathbf{y}}_q^i(t) &= [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{c\to q}^{iT}(t)]^T, \\ \widetilde{B}_q^i &= [B_q^T, \widehat{B}_{c\to q}^{iT}]^T, \end{aligned} \tag{40}$$

such that the corresponding $\widetilde{X}_q^{i+1}$ is obtained at iteration i by solving the local problem (20) using the data that is available at node q, as in (40). Similar to (21), we define the partitioning:

$$\widetilde{X}_q^{i+1} = [X_q^{(i+1)T}, G_c^{(i+1)T}]^T \in \mathbb{R}^{(M_q+Q)\times Q}, \tag{41}$$

where G_c^{i+1} is analogous to the G_k^{i+1}'s in the previous section. Since node q has only one link, there is only one such matrix resulting from solving the local problem (20), which is sent to node c. Finally, the central node c disseminates this matrix G_c^{i+1} to the other nodes to update their local compressor as in (22), but with $G_k = G_c$ for all k.

B. Tree Topologies

We now consider a network represented by a tree, i.e., a graph without cyclic paths. A leaf node is defined as a node with a single neighbor, i.e., a node k for which $|\mathcal{N}_k| = 1$. Recalling that our objective is to be able to recreate (37) at the updating node q, we perform an in-network fusion across the different tree branches that are rooted in node q. This fusion can be done in a bottom-up fashion without central coordination. Indeed, the strategy for each node $k \neq q$ is to wait until it has received the compressed signals from all its neighbors except one (denoted as node n), sum these and add its own compressed signal $\widehat{\mathbf{y}}_k^i$, and transmit to its remaining neighbor n. Formally, the compressed signal being sent from node $k \neq q$ to n at iteration i is

$$\widehat{\mathbf{y}}_{k\to n}^i(t) = X_k^{iT}\mathbf{y}_k(t) + \sum_{l\in\mathcal{N}_k\setminus\{n\}} \widehat{\mathbf{y}}_{l\to k}^i(t). \tag{42}$$

Note that this is a recursive definition, which is bootstrapped by the leaf nodes, for which the second (recursive) term vanishes. This data fusion flow is illustrated in Figure 3 for an example network. The fused signals will eventually arrive at the updating node q which receives

$$\widehat{\mathbf{y}}_{n\to q}^i(t) = X_n^{iT}\mathbf{y}_n(t) + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{\mathbf{y}}_{k\to n}^i(t) = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i(t), \tag{43}$$

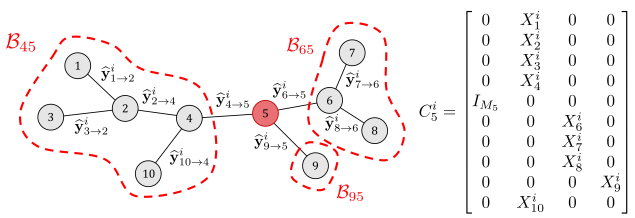

Fig. 3. [50] Example of a tree network where the updating node is node 5. Each neighbor of node 5 creates its own cluster containing the nodes "hidden" from node 5 behind them, shown here as $\mathcal{B}_{45}$, $\mathcal{B}_{65}$, $\mathcal{B}_{95}$. The resulting transition matrix is given by C_5^i.

from each of its neighbors $n \in \mathcal{N}_q$, where we define $\mathcal{B}_{nq}$ to be the connected subgraph containing node n when the link between nodes n and q is removed (see Figure 3). The same process is applied to the deterministic terms such that node q receives

$$\widehat{B}_{n\to q}^i = X_n^{iT}B_n + \sum_{k\in\mathcal{N}_n\setminus\{q\}} \widehat{B}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{B}_k^i \tag{44}$$

from all its neighbors $n \in \mathcal{N}_q$.

Writing $\mathcal{N}_q = \{n_1, \ldots, n_{|\mathcal{N}_q|}\}$, we have the vector of available data at the updating node q:

$$\begin{aligned} \widetilde{\mathbf{y}}_q^i(t) &= [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{n_1\to q}^{iT}(t), \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}(t)]^T, \\ \widetilde{B}_q^i &= [B_q^T, \widehat{B}_{n_1\to q}^{iT}, \ldots, \widehat{B}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T. \end{aligned} \tag{45}$$

We note that the relationship between the network-wide data $\mathbf{y}$, B and the locally available $\widetilde{\mathbf{y}}_q^i$, $\widetilde{B}_q^i$ can again be described by means of a compression matrix C_q^i as in (25)-(26), such that $\widetilde{\mathbf{y}}_q^i = C_q^{iT}\mathbf{y}$ and $\widetilde{B}_q^i = C_q^{iT}B$. An example of such a matrix C_q^i is shown in Figure 3. We recommend the reader to use the example of this figure to appreciate the structure of this matrix, yet we also provide a general definition for completeness. In general, this matrix can be defined as

$$C_q^i = \left[\begin{array}{c|c} 0 & \\ I_{M_q} & \Theta_{-q}^i \\ 0 & \end{array}\right], \tag{46}$$

where I_{M_q} is placed in the $q-$th block-row, and Θ_{-q}^i is a block matrix with K block-rows and $|\mathcal{N}_q|$ block-columns, where the block at the $k-$th block-row and $m-$th block-column is represented by $\Theta_{-q}^i(k,m) \in \mathbb{R}^{M_k\times Q}$. Each block-column corresponds to one of the neighbors $n \in \mathcal{N}_q$ of q, which we re-index as $m_n \in \{1, \ldots, |\mathcal{N}_q|\}$. Then, we have

$$\Theta_{-q}^i(k, m_n) = \begin{cases} X_k^i & \text{if } k \in \mathcal{B}_{nq} \\ 0 & \text{otherwise} \end{cases}. \tag{47}$$

As in the previous cases, the transition from the local variable to the network-wide one is given by $X = C_q^i\widetilde{X}_q$ at node q and iteration i. We can verify that $\widetilde{X}_q$ is a feasible point of the local problem if and only if $C_q^i\widetilde{X}_q$ is a feasible point of the global problem, i.e., (32) and more generally Lemma 1 also holds here. Node q then solves its local problem (20) using the locally available data described in (45), to obtain $\widetilde{X}_q^{i+1}$, partitioned as

$$\widetilde{X}_q^{i+1} = \left[X_q^{(i+1)T}, G_{n_1}^{(i+1)T}, \ldots, G_{n_{|\mathcal{N}_q|}}^{(i+1)T}\right]^T. \tag{48}$$

Each G_n^{i+1} is then disseminated into the corresponding subgraph $\mathcal{B}_{nq}$ through node n (and the nodes behind it in $\mathcal{B}_{nq}$) and every node updates its compressor as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q \\ X_k^i G_n^{i+1} & \text{if } k \in \mathcal{B}_{nq},\, n \in \mathcal{N}_q \end{cases} \tag{49}$$

such that we again have $X^{(i+1)T}\mathbf{y} = \widetilde{X}_q^{(i+1)T}\widetilde{\mathbf{y}}_q^i$ and $X^{(i+1)T}B = \widetilde{X}_q^{(i+1)T}\widetilde{B}_q^i$.

From (49), we observe that we have parameterized the global variable X at node q and iteration i as

$$X = C_q^i \widetilde{X}_q = \begin{bmatrix} X_1^i \boxed{G_{n(1)}} \\ \vdots \\ X_{q-1}^i \boxed{G_{n(q-1)}} \\ \boxed{X_q} \\ X_{q+1}^i \boxed{G_{n(q+1)}} \\ \vdots \\ X_K^i \boxed{G_{n(K)}} \end{bmatrix}, \tag{50}$$

where C_q^i is given in (46) and $n(k)$ is the neighbor $n \in \mathcal{N}_q$ of node q such that $k \in \mathcal{B}_{nq}$. This can be compared with (30) for the fully-connected case. Since the optimization variable of (20) is partitioned as in (48), we can see from (50) that, similarly to the fully-connected case, the updating node q can "freely" update its filter X_q, while the filters of the other nodes can only change up to a right-hand side matrix multiplication with a G_n-matrix (the updating variables are the framed variables in (50)). In contrast to the fully-connected case in (30), some of the G_n's are constrained to be equal due to the network topology since $k, l \in \mathcal{B}_{nq} \Rightarrow G_{n(k)} = G_{n(l)}$. This implies that the degrees of freedom in the updating steps of the tree-based DASF algorithm are determined by the number of neighbors of the updating node.

Remark 6. Sometimes it is possible that a node cannot generate Q linearly independent output channels using (42). For example, this happens if node k is a leaf node and $M_k < Q$. In this case, node k will send its raw uncompressed sensor data $\mathbf{y}_k$ instead of $\widehat{\mathbf{y}}_{k\to n}^i$ defined in (42). The neighbor n that receives the raw data from node k will then treat this data as part of its own sensor signals, i.e., $\mathbf{y}_n$ is stacked with $\mathbf{y}_k$ and its sensor channel count becomes $M_n + M_k$. In other words, the data flow starting at the leaf nodes can initially consist of raw sensor channels until a node has more than Q channels to compress them into a $Q-$channel signal. Therefore, the number of transmitted channels is at most Q per node, but can also be less than Q. An analogous statement applies to the deterministic terms B_k.

C. General Connected Graphs

Suppose the network is represented by a connected graph $\mathcal{G}$, which can potentially contain cycles. The main difference with the previous subsection is that there is more than one choice to forward the compressed data to the updating node q. We therefore propose to prune the graph $\mathcal{G}$ into a (different) tree at each iteration i, so that we can apply the same steps as the ones described for the tree topology case. The resulting tree is denoted as $\mathcal{T}^i(\mathcal{G}, q)$ to highlight the fact that the pruning function $\mathcal{T}^i$ depends on the current updating node q. An example of the data flow in pruned networks for two different updating nodes is given in Figure 4.

The choice of the mapping function $\mathcal{T}^i$ is a free design choice as long as the resulting tree remains connected. However, the convergence results in [1] that define the bound (8) also assume that the pruning function does not remove the links between the updating node q and its neighbors. Indeed, if (8) is satisfied, then this rule ensures that $J \leq (1+|\mathcal{N}_q|)Q^2$ at any updating node q, which is one of the convergence conditions for the DASF algorithm in [1]. Furthermore, disconnecting node q from a neighbor would also reduce the convergence speed, since it would lead to a smaller number of resulting G_n's in (48)-(49) and therefore a lower number of degrees of freedom in the local optimization problem (20),

Algorithm 2: Topology-Independent Distributed Adaptive Signal Fusion (TI-DASF) Algorithm
Code available in [47]

output: X^*

Initialize X^0, $i \leftarrow 0$.
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) Every node k collects a new batch of N samples of $\mathbf{y}_k$. All nodes compress these to N samples of $\widehat{\mathbf{y}}_k^i$ as in (14). $\widehat{B}_k^i$ is computed using (16).
- 3) The nodes sum-and-forward their compressed data towards node q via the recursive rule (42) (and a similar rule for the $\widehat{B}_k^i$'s). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ along with $\widehat{B}_{n\to q}^i$ given in (43)-(44) from all its neighbors $n \in \mathcal{N}_q$.
- **at** *Node* q **do**
  - 4a) Compute $\widetilde{X}_q^{i+1}$ as the solution of (20) where $\widetilde{\mathbf{y}}_q^i$, $\widetilde{B}_q^i$ and $\widetilde{X}_q$ are redefined as in (45) and (48). If the solution of (20) is not unique, select the solution which minimizes $||\widetilde{X}_q^{i+1} - \widetilde{X}_q^i||_F$ with $\widetilde{X}_q^i$ defined as in (23).
  - 4b) Partition $\widetilde{X}_q^{i+1}$ as in (48).
  - 4c) Disseminate G_n^{i+1} to all nodes in $\mathcal{B}_{nq}$, $\forall n \in \mathcal{N}_q$.
- **end**
- 5) Every node updates X_k^{i+1} according to (49).
- $i \leftarrow i + 1$

Note: Each iteration uses a different batch of N samples in step 2, i.e., the iterations can be spread over different time segments in order to avoid retransmitting the same batch of N samples across the network (see also Remark 1). This makes that the sample time index t is coupled to the iteration index i.
Note: As in FC-DASF, the fused output signal $X^T\mathbf{y}(t) = \widehat{\mathbf{y}}(t)$ can be computed as $\widetilde{X}_q^{(i+1)T}\widetilde{\mathbf{y}}_q^i(t)$ at node q without extra transmissions (see also Remark 4).

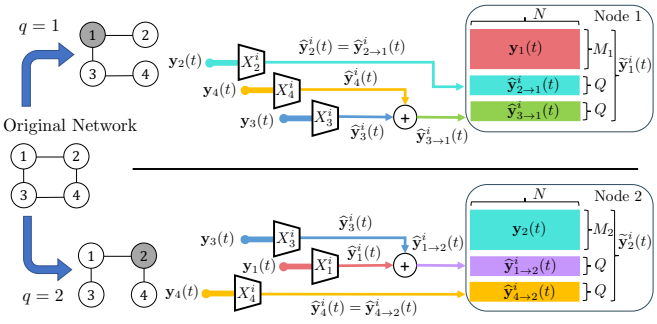

Fig. 4. Data flow for a $4-$node network when nodes $q = 1$ (top) and $q = 2$ (bottom) are the updating node. The updating node has always access to its own data $\mathbf{y}_q$, while receiving fused signals from every other node in the network through its neighboring nodes.

typically leading to a smaller descent of the cost φ or f. In comparison, in the case of fully-connected networks, we were able to use $(K-1)$ of such G_n's in each iteration, which — as we will show in Section V — leads to the fastest convergence.

A simple distributed protocol to establish $\mathcal{T}^i$, i.e., to set up a tree that satisfies the aforementioned rule, is to let the updating node q broadcast a token to each neighbor. Once a node receives a token, it acknowledges a link to the node from which it received it, and it then broadcasts that token to its remaining neighbors from which it did not receive a token. This process continues until all nodes have received a token. If a node receives multiple tokens, it only connects to the parent node from which it received the token first (in case of a tie, it makes a random choice).

The full TI-DASF algorithm is given in Algorithm 2 and convergence analyses are provided in [1]. We note that FC-DASF (Algorithm 1) is a special case of TI-DASF (Algorithm 2). The same generalizations as explained in Section III-C apply for TI-DASF as well.

Remark 7. In the TI-DASF algorithm, each node k transmits only NQ samples of $\mathbf{y}_k$ per iteration, which is independent of the number of neighbors or the total number of nodes in the network. As opposed to the fully-connected case, the TI-DASF algorithm can reduce the total communication burden, even when $Q \geq M_k$. This is because naively relaying all the raw sensor data to a fusion center node for centralized processing would require most nodes to send more than NQ samples as they would also have to forward the data from their neighbors, and their neighbors' neighbors, etc (see Figure 5). Obviously, such a relaying approach does not scale well with the network size, whereas the per-node bandwidth requirements in the TI-DASF algorithm are independent of the number of nodes.

V. Examples and Simulations

In this Section, we present examples of problems that fit the DASF framework and demonstrate the performance of the algorithm in various settings to gain insights into the convergence behavior as a function of Q, K and the topology. The examples also serve as illustrations to familiarize the reader with how to recognize SFO problems of the form (3) or (5), and how to translate these into the DASF framework. It is noted that several existing distributed algorithms can be shown

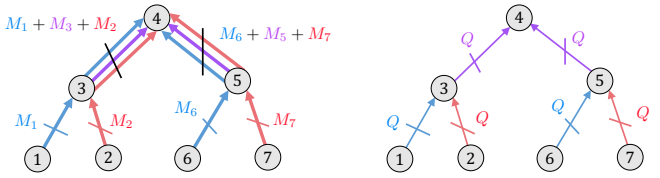

Fig. 5. Comparison between a straightforward relaying approach (left) and the scalable fuse-and-forward approach the DASF algorithm uses (right). In this example, node 4 is the updating node.

to be special cases of the proposed unified DASF framework (Table I). Since these special cases have been validated already, and to show the generalizing properties of the framework, we will validate it on a few new problems that fit our framework. For this purpose, we have also published a companion toolbox [47] which allows to automatically generate and simulate a distributed algorithm for any arbitrary problem of the form (3) or (5). The only requirement is that the user provides a solver for the centralized problem (3) or (5). The software will use this solver to compute the updating step in each iteration of the FC- or TI-DASF algorithm, as the update requires solving a compressed local instance of (3) or (5).

In our experiments, we refer to *randomly generated trees* as trees where each node has between 0 and 4 children with 1.7 children on average[7]. We consider two different sensor signals $\mathbf{y}$ and $\mathbf{v}$ measured at each node throughout this section, following the mixture model given by

$$\mathbf{y}(t) = \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{51}$$

$$\begin{aligned}\mathbf{v}(t) &= \Pi_r \cdot \mathbf{r}(t) + \mathbf{y}(t) \\ &= \Pi_r \cdot \mathbf{r}(t) + \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t),\end{aligned} \tag{52}$$

with $\mathbf{r}(t)$, $\mathbf{s}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_r^2)$, $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_n^2)$ for every entry and time instance t. In the experimental settings of Sections V-A to V-C, the entries of Π_s and Π_r are independent of the time t and drawn from the uniform distribution within the interval $[-0.5, 0.5]$. In Section V-D, we will consider an adaptive setting where Π_s is time-dependent. We assume that both $\mathbf{y}$ and $\mathbf{v}$ are observable at the nodes (this is possible, e.g., if the source $\mathbf{r}$ has an on-off behavior). In all the simulations, we take the number of samples of the signals to be communicated between the nodes to be $N = 10^4$ and each node has an equal number of channels $M_k = M/K$ (where the total number of channels M and the total number of nodes K will vary). The convergence of the DASF algorithm is assessed by tracking the normalized error ϵ:

$$\epsilon(X^i) = \frac{||X^i - X^*||_F^2}{||X^*||_F^2}. \tag{53}$$

The optimal value X^* is computed by solving the problems we present in the following paragraphs using centralized solvers. In case the centralized problem has multiple possible solutions, i.e., $|\mathcal{X}^*| > 1$, we select $X^* \in \mathcal{X}^*$ in (53) that best matches X^i in the final iteration of the simulation. This resolves the ambiguity of the solution, while the plots would still reveal non-convergence in case the algorithm would arrive in a limit

[7]The number of children nodes is selected randomly from $[0, 1, 2, 3, 4]$, which is distributed following the probability vector $[0.2, 0.3, 0.2, 0.2, 0.1]$.

TABLE III
SUMMARY OF PARAMETERS USED IN THE SIMULATIONS.

Experiment	Section V-A	Section V-B	Section V-C
Q	3	5	$\{1,3,5,7\}$
K	$\{10,25,50\}$	30	
M, M_k	$M=450$, $M_k=M/K$, $\forall k\in\mathcal{K}$		
Signal Statistics	$\sigma_r^2=\sigma_n^2=1$	$\sigma_r^2=0.5$, $\sigma_n^2=0.1$	$\sigma_r^2=\sigma_n^2=1$
N	10000		
Monte Carlo Runs	100		

cycle that switches between multiple accumulation points, i.e., we would observe subsequences of $(X^i)_i$ converging to different solutions of the problem. The parameters chosen for each following problem in Sections V-A to V-C are summarized in Table III. In these experiments, we aim to observe the theoretical convergence result of the DASF algorithm and therefore consider stationary and ergodic signals. On the other hand, Section V-D has a slightly different experimental setting as we aim to demonstrate the adaptive properties of the DASF algorithm, in which case stationarity does not hold.

A. Quadratically Constrained Quadratic Problem

In this subsection, we will solve the following problem:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \frac{1}{2}\mathbb{E}[||X^T\mathbf{y}(t)||^2] - \text{tr}(X^TA) \\ \text{subject to} \quad & \text{tr}(X^TX)\leq\alpha^2,\ X^T\mathbf{c}=\mathbf{d}, \end{aligned} \tag{54}$$

where we take $\sigma_n^2=\sigma_r^2=1$ for the noise and signal variance. In this problem, we have three deterministic inner products of the form X^TB where B is known a priori. Two of them are the terms X^TA and $X^T\mathbf{c}$ where $A\in\mathbb{R}^{M\times Q}$ and $\mathbf{c}\in\mathbb{R}^M$. The third one comes from X^TX, which can be written as $(X^TI_M)\cdot(X^TI_M)^T$ which reveals the term X^TB with $B=I_M$ (see also Remark 5). The values of $\alpha\in\mathbb{R}$ and $\mathbf{d}\in\mathbb{R}^Q$ have been chosen randomly while ensuring that $\alpha^2\geq||\mathbf{d}||^2/||\mathbf{c}||^2$ which would otherwise make the problem infeasible. We can write $\mathbb{E}[||X^T\mathbf{y}(t)||^2]=\text{tr}(X^TR_{\mathbf{yy}}X)$ where $R_{\mathbf{yy}}=\mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ is the covariance matrix of the signal $\mathbf{y}$. We note that the matrix $R_{\mathbf{yy}}$ is assumed to be unknown, as the signal statistics are to be learned by the algorithm, so it should not be seen as a deterministic term. Then, the local problem at iteration i that node q needs to solve is

$$\begin{aligned} \underset{\widetilde{X}_q}{\text{minimize}} \quad & \frac{1}{2}\text{tr}(\widetilde{X}_q^TR^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q) - \text{tr}(\widetilde{X}_q^T\widetilde{A}_q^i) \\ \text{subject to} \quad & \text{tr}(\widetilde{X}_q^TC_q^{iT}C_q^i\widetilde{X}_q)\leq\alpha^2,\ \widetilde{X}_q^T\widetilde{\mathbf{c}}_q^i=\mathbf{d}, \end{aligned} \tag{55}$$

where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}=\mathbb{E}[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$ is the correlation matrix of the locally available signal $\widetilde{\mathbf{y}}_q^i$ at node q and iteration i. Note that the compression matrix C_q^i appears in the quadratic constraints, which is indeed what one obtains when computing $\widetilde{B}_q^i$ with $B=I_M$, which directly follows from (26).

For this experiment, we take the number of channels to be $M=200$, take $Q=3$ and look at the behavior of the algorithm for a varying number of nodes in the network with $K\in\{10,25,50\}$ (the number of channels per node $M_k=M/K$ therefore changes for each K). The results are shown in Figure 6. For the fully-connected case, we see that the smaller networks, i.e., when K is small, converge faster in the first iterations. This is because they are able to do a full round update (over all nodes) in a smaller number of iterations compared to larger networks. However, the larger networks can eventually "catch up" and even surpass the convergence speed of their counterparts with smaller K after a certain number of iterations due to the larger number of degrees of freedom (i.e., a larger number of G_k matrices in each iteration). This property is however not observed for randomly generated trees as here the number of G_k matrices at each node is related to its number of neighbors, hence independent of the network size. Overall, the fully-connected topologies lead to faster convergence and the randomly generated trees are the slowest, which is consistent with the expectations based on the amount of degrees of freedom (i.e., the number of G_k matrices in each update). Networks for which the topology is neither a tree nor fully-connected are expected to fall in between these two extreme cases. We add that the high variance observed in Figure 6 over various Monte-Carlo runs is due to the fact that Problem (54) has many parameters chosen randomly.

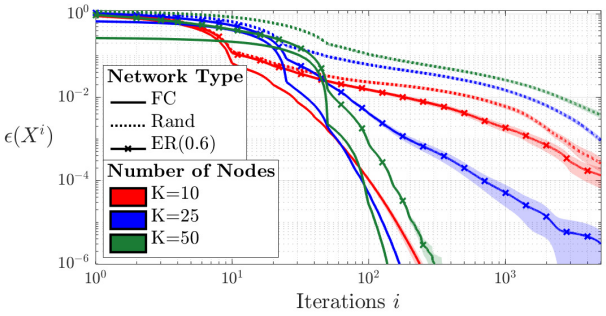

Fig. 6. Convergence comparison of the DASF algorithm solving (54) in fully-connected networks (*FC*), randomly generated trees (*Rand*) and Erdős-Rényi random graphs with connection probability of 0.6 (*ER(0.6)*) for various network sizes. The bold lines represent the mean values across 100 Monte Carlo runs, while the shaded areas delimit the standard error of the mean around them.

B. The Trace Ratio Optimization Problem

We now consider the problem:

$$\begin{aligned} \underset{X}{\text{maximize}} \quad & \frac{\mathbb{E}[||X^T\mathbf{v}(t)||^2]}{\mathbb{E}[||X^T\mathbf{y}(t)||^2]} = \frac{\text{tr}(X^TR_{\mathbf{vv}}X)}{\text{tr}(X^TR_{\mathbf{yy}}X)} \\ \text{subject to} \quad & X^TX=I_Q, \end{aligned} \tag{56}$$

often referred to as the trace ratio or trace quotient optimization (TRO) problem [68]. It is noted that a distributed algorithm for this TRO problem has been proposed in [50] where techniques tailored to the TRO problem have been used. Applying the generic Algorithm 2 to Problem (56) results in an alternative distributed algorithm for the TRO problem, based on the DASF framework. $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ are again spatial correlation matrices of $\mathbf{y}$ and $\mathbf{v}$ respectively, assumed to be unknown to the DASF algorithm. We take the signal and noise variance to be $\sigma_r^2=0.5$ and $\sigma_n^2=0.1$ respectively. Problem (56) can be solved using the solver in [68] by iteratively computing generalized eigenvalue decompositions.

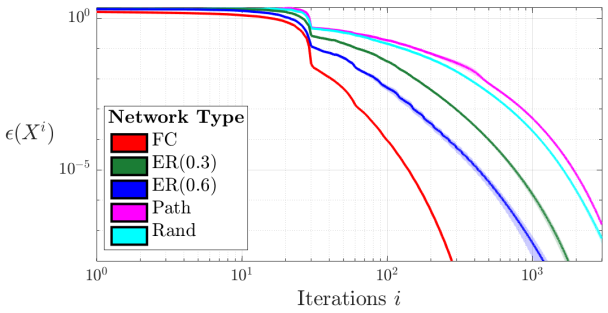

Fig. 7. Convergence comparison of the DASF algorithm solving (56) for various network settings namely fully-connected networks (*FC*), Erdős-Rényi random graphs with connection probability p (*ER(p)*), path or line graphs (*Path*) and randomly generated trees (*Rand*). The bold lines represent the mean values across 100 Monte Carlo runs, while the shaded areas delimit the standard error of the mean around them.

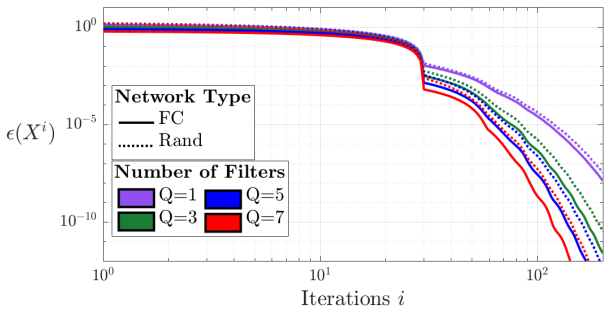

Fig. 8. Convergence comparison of the DASF algorithm solving (58) for various number of filters Q for fully-connected networks (*FC*) and randomly generated trees (*Rand*). The bold lines represent the mean values across 100 Monte Carlo runs, while the shaded areas delimit the standard error of the mean around them.

At updating node q and iteration i, (56) is translated to

$$\begin{aligned} \underset{\widetilde{X}_q}{\text{maximize}} \quad & \frac{\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}\widetilde{X}_q)}{\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q)} \\ \text{subject to} \quad & \widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q = I_Q, \end{aligned} \tag{57}$$

where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ and $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}$ are estimated in the same way as in the previous example. Therefore, the solver in [68] can be applied to solve the local problem (57) in step 4a of Algorithm 2, by replacing $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ by $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}$ and $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q}$ respectively, where q is the updating node at iteration i.

The experimental results are shown in Figure 7, where we highlight the differences in convergence depending on the topology of the network. In particular, we look at fully-connected networks, random trees, line topologies (trees where each node has two neighbors, except leaf nodes which have only one neighbor), and Erdős-Rényi (ER) models, generated using [69]. In each case, we keep the number of nodes, the number of channels and the compression dimension Q constant, with $K = 30$, $M = 450$ and $Q = 5$ respectively. We observe that the more the network is connected the faster the DASF algorithm converges, the fastest being the fully-connected networks, while the slowest convergence is obtained for line graphs. This is again in line with the results and discussion on the degrees of freedom in Section V-A.

C. Quadratic Problem with a Spherical Constraint

Let us now consider:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \frac{1}{2}\mathbb{E}[||X^T\mathbf{y}(t)||^2] + \text{tr}(X^T A) \\ \text{subject to} \quad & \text{tr}(X^T X) = 1, \end{aligned} \tag{58}$$

where $\sigma_n^2 = \sigma_r^2 = 1$ and the elements of A have been chosen independently at random. Note that this problem differs from (54) in the sense that it has a non-convex constraint set due to the non-linear equality constraint. The local problem at node q and iteration i can be written as

$$\begin{aligned} \underset{\widetilde{X}_q}{\text{minimize}} \quad & \frac{1}{2}\text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q) + \text{tr}(\widetilde{X}_q^T \widetilde{A}_q^i) \\ \text{subject to} \quad & \text{tr}(\widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q) = 1. \end{aligned} \tag{59}$$

For this experiment, we fix $K = 30$, $M = 450$ and consider various number of filters $Q \in \{1, 3, 5, 7\}$. We observe in Figure 8 that the larger the number of filters Q, the faster the algorithm converges. This is expected as a larger value for Q implies that more information can be used at every node, however at the expense of a larger communication cost. As for the previous examples, we see that the DASF framework converges faster for fully-connected networks compared to randomly generated trees for Problem (58).

D. DASF in a Tracking Problem

In this final experiment, we consider the MMSE problem

$$\underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad \mathbb{E}[|s(t) - \mathbf{x}^T\mathbf{y}(t)|^2], \tag{60}$$

where s is a one-dimensional signal, implying $Q = 1$. The problem is solved using the DASF algorithm on a randomly generated Erdős-Rényi network with connection probability 0.8. The network contains $K = 10$ nodes, each measuring a signal $\mathbf{y}_k$ with $M_k = 4$ channels. At each iteration i, $N = 10^4$ new times samples of $\mathbf{y}_k$ are used, such that $i = \lfloor t/N \rfloor$. The network-wide signal is given by

$$\mathbf{y}(t) = \mathbf{p}(t) \cdot s(t) + \mathbf{n}(t), \tag{61}$$

for each t, with $s(t) \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$ and $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1)$ for every entry and time instance t. The main difference with (51) is that now the steering vector $\mathbf{p}$ (corresponding to the mixture matrix Π_s in (51)) changes with time, implying that the statistical properties of $\mathbf{y}$ are time-dependent as well. At each time instance t, the solution of (60) is given by $\mathbf{x}^*(t) = (R_{\mathbf{yy}}(t))^{-1}\mathbf{r}_{\mathbf{y}s}(t)$, where $R_{\mathbf{yy}}(t) = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ and $\mathbf{r}_{\mathbf{y}s}(t) = \mathbb{E}[\mathbf{y}(t)s(t)]$. For each t, we have $\mathbf{p}(t) = (1 - \lambda(t)) \cdot \mathbf{p}_0 + \lambda(t) \cdot (\mathbf{p}_0 + \Delta)$, where $\mathbf{p}_0$ and Δ are time-independent vectors of which the entries are drawn from $\mathcal{N}(0,1)$ and $\mathcal{N}(0, 0.01)$ respectively. The function λ used in this experiment is represented in Figure 9, where it can be observed that smooth, as well as abrupt, changes are modeled. At each iteration, the ground-truth solution of (60) is estimated as

$$\mathbf{x}^{*i} = (R^i_{\mathbf{yy}})^{-1}\mathbf{r}^i_{\mathbf{y}s}, \tag{62}$$

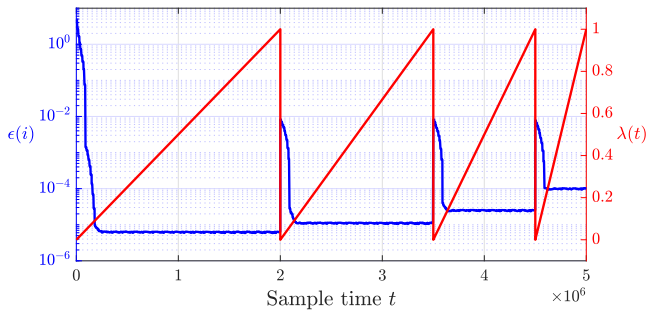

Fig. 9. Error ϵ over time of the DASF algorithm in an adaptive setting (blue). The signal statistics change over time and depend on the function λ, represented in red. The relationship between the time t and the iterations i is given by $i = \lfloor t/N \rfloor$.

where

$$R^i_{\mathbf{yy}} = \sum_{\tau=iN}^{iN+N-1} \mathbf{y}(\tau)\mathbf{y}^T(\tau), \; \mathbf{r}^i_{\mathbf{y}s} = \sum_{\tau=iN}^{iN+N-1} \mathbf{y}(\tau)s(\tau). \quad (63)$$

Figure 9 shows the median value over 100 Monte Carlo runs of the error ϵ:

$$\epsilon(i) = \frac{||\mathbf{x}^i - \mathbf{x}^{*i}||^2}{||\mathbf{x}^{*i}||^2} \quad (64)$$

over time, where $i = \lfloor t/N \rfloor$. The algorithm can adapt to abrupt changes in signal statistics, i.e., when the value of λ suddenly changes, which translates to an initial jump in the error followed by a decrease. We observe that the DASF algorithm is also able to track slow changes in the statistical properties of the signal, shown by constant error values ϵ over the iterations, despite changes in the value of λ. Since the optimal solution $\mathbf{x}^{*i}$ changes at each iteration, the error ϵ settles around a certain threshold, which is higher for larger rates of change in λ and due to the approximation error.

VI. Conclusion

In this paper, we have proposed the DASF framework which contains a large number of well-known distributed spatial filter design problems and algorithms as a special case. For intelligibility purposes, we have first addressed the case of fully-connected networks (FC-DASF) and then generalized it to any network represented by a connected graph (TI-DASF). In order to reduce the communication burden, the nodes only communicate compressed data across the network. An interesting property of the resulting distributed algorithm is that the local problem to be solved at an updating node has the same structure as the original network-wide centralized problem, such that the same solver can be used. The convergence properties of the algorithm have been illustrated by means of four example instances of the (D)SFO problem (3) or its more general form in (5), one of the examples focusing on the adaptive properties of the DASF algorithm. A formal convergence analysis with convergence and optimality proofs, including examples on how the convergence conditions can be checked, are provided in a companion paper [1]. We have also provided a toolbox to automatically design and test the DASF algorithm for any user-defined instances of the (D)SFO problem (3) or (5), available in [47].

VII. Acknowledgments

The authors would like to thank Charles Hovine for the brainstorming sessions and help with Sections II-B and V-D.

References

[1] C. A. Musluoglu, C. Hovine, and A. Bertrand, "A unified algorithmic framework for distributed adaptive signal and feature fusion problems — Part II: Convergence properties," 2022.

[2] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A survey on wireless body area networks," *Wireless networks*, vol. 17, no. 1, pp. 1–18, 2011.

[3] 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.

[4] ——, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in *2011 18th IEEE symposium on communications and vehicular technology in the Benelux (SCVT)*. IEEE, 2011, pp. 1–6.

[5] 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.

[6] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 13–24.

[7] M. F. Othman and K. Shazali, "Wireless sensor network applications: A study in environment monitoring system," *Procedia Engineering*, vol. 41, pp. 1204–1210, 2012.

[8] C. Albaladejo, P. Sánchez, A. Iborra, F. Soto, J. A. López, and R. Torres, "Wireless sensor networks for oceanographic monitoring: A systematic review," *Sensors*, vol. 10, no. 7, pp. 6948–6968, 2010.

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[10] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[11] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014, http://dx.doi.org/10.1561/2200000051.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[13] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 6698–6703.

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

[15] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM Journal on Optimization*, vol. 7, no. 4, pp. 913–926, 1997.

[16] 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.

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

[18] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE transactions on signal processing*, vol. 58, no. 3, pp. 1035–1048, 2009.

[19] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012.

[20] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[21] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[23] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 5445–5450.

[24] A. B. Gershman, N. D. Sidiropoulos, S. Shahbazpanahi, M. Bengtsson, and B. Ottersten, "Convex optimization-based beamforming," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 62–75, 2010.

[25] S. A. Vorobyov, "Principles of minimum variance robust adaptive beamforming design," *Signal Processing*, vol. 93, no. 12, pp. 3264–3277, 2013.

[26] S. Doclo and M. Moonen, "GSVD-based optimal filtering for single and multimicrophone speech enhancement," *IEEE Transactions on signal processing*, vol. 50, no. 9, pp. 2230–2244, 2002.

[27] B. Somers, T. Francart, and A. Bertrand, "A generic EEG artifact removal algorithm based on the multi-channel Wiener filter," *Journal of neural engineering*, vol. 15, no. 3, p. 036007, 2018.

[28] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[29] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis." *Journal of machine learning research*, vol. 8, no. 5, 2007.

[30] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, 2018.

[31] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: models, limits, and algorithms," *IEEE Transactions on Signal processing*, vol. 59, no. 1, pp. 386–398, 2010.

[32] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, 2011.

[33] C. Manss, D. Shutin, and G. Leus, "Distributed splitting-over-features sparse bayesian learning with alternating direction method of multipliers," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 3654–3658.

[34] C. Gratton, N. K. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning over networks with non-smooth regularizers and feature partitioning," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1840–1844.

[35] B. Zhang, J. Geng, W. Xu, and L. Lai, "Communication efficient distributed learning with feature partitioned data," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2018, pp. 1–6.

[36] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 725–738, 2011.

[37] Y. Zeng and R. C. Hendriks, "Distributed delay and sum beamformer for speech enhancement via randomized gossip," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 260–273, 2013.

[38] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss–Seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.

[39] A. Bertrand and M. Moonen, "Distributed adaptive generalized eigenvector estimation of a sensor signal covariance matrix pair in a fully connected sensor network," *Signal Processing*, vol. 106, pp. 209–214, 2015.

[40] ——, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Signal Processing*, vol. 104, pp. 120–135, 2014.

[41] 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.

[42] 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.

[43] 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.

[44] A. Bertrand and M. Moonen, "Distributed LCMV beamforming in a wireless sensor network with single-channel per-node signal transmission," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3447–3459, 2013.

[45] ——, "Distributed canonical correlation analysis in wireless sensor networks with application to distributed blind source separation," *IEEE Transactions on Signal Processing*, vol. 63, no. 18, pp. 4800–4813, 2015.

[46] C. Hovine and A. Bertrand, "Distributed MAXVAR: Identifying common signal components across the nodes of a sensor network," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 2159–2163.

[47] C. A. Musluoglu and A. Bertrand, "DASF Toolbox," 2022. [Online]. Available: https://github.com/AlexanderBertrandLab/DASF_toolbox

[48] H. Cox, R. Zeskind, and M. Owen, "Robust adaptive beamforming," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 10, pp. 1365–1376, 1987.

[49] J. Wouters, P. Patrinos, F. Kloosterman, and A. Bertrand, "Multi-pattern recognition through maximization of signal-to-peak-interference ratio with application to neural spike sorting," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6240–6254, 2020.

[50] C. A. Musluoglu and A. Bertrand, "Distributed adaptive trace ratio optimization in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653–3670, 2021.

[51] Y. Wang, S. Gao, and X. Gao, "Common spatial pattern method for channel selelction in motor imagery based brain-computer interface," in *2005 IEEE engineering in medicine and biology 27th annual conference*. IEEE, 2006, pp. 5392–5395.

[52] A. Hjorungnes and D. Gesbert, "Complex-valued matrix differentiation: Techniques and key results," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2740–2746, 2007.

[53] T. Adali and P. J. Schreier, "Optimization and estimation of complex-valued signals: Theory and applications in filtering and blind source separation," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 112–128, 2014.

[54] 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.

[55] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[56] 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.

[57] I. S. Reed, J. D. Mallett, and L. E. Brennan, "Rapid convergence rate in adaptive arrays," *IEEE Transactions on Aerospace and Electronic Systems*, no. 6, pp. 853–863, 1974.

[58] S. Valaee, B. Champagne, and P. Kabal, "Localization of wideband signals using least-squares and total least-squares approaches," *IEEE Transactions on Signal Processing*, vol. 47, no. 5, pp. 1213–1222, 1999.

[59] C. E. Davila, "An efficient recursive total least squares algorithm for fir adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 42, no. 2, pp. 268–280, 1994.

[60] S. Kim, R. Pasupathy, and S. G. Henderson, "A guide to sample average approximation," *Handbook of simulation optimization*, pp. 207–243, 2015.

[61] P. S. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Springer Nature, 2019.

[62] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2011.

[63] ——, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[64] H.-F. Chen and G. Yin, "Asymptotic properties of sign algorithms for adaptive filtering," *IEEE transactions on automatic control*, vol. 48, no. 9, pp. 1545–1556, 2003.

[65] J. Hadamard, "Sur les problèmes aux dérivées partielles et leur signification physique," *Princeton university bulletin*, pp. 49–52, 1902.

[66] Y.-h. Zhou, J. Yu, H. Yang, and S.-w. Xiang, "Hadamard types of well-posedness of non-self set-valued mappings for coincide points," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 63, no. 5-7, pp. e2427–e2436, 2005.

[67] D. W. Peterson, "A review of constraint qualifications in finite-dimensional spaces," *Siam Review*, vol. 15, no. 3, pp. 639–654, 1973.

[68] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[69] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.