A Distributed Adaptive Algorithm for Non-Smooth Spatial Filtering Problems in Wireless Sensor Networks

Charles Hovine and Alexander Bertrand, *Senior Member, IEEE*

***Abstract*—A wireless sensor network often relies on a fusion center to process the data collected by each of its sensing nodes. Such an approach relies on the continuous transmission of raw data to the fusion center, which typically has a major impact on the sensors' battery life. To address this issue in the particular context of spatial filtering and signal fusion problems, we recently proposed the Distributed Adaptive Signal Fusion (DASF) algorithm, which distributively computes a spatial filter expressed as the solution of a smooth optimization problem involving the network-wide sensor signal statistics. In this work, we show that the DASF algorithm can be extended to compute the filters associated with a certain class of non-smooth optimization problems. This extension makes the addition of sparsity-inducing norms to the problem's cost function possible, allowing sensor selection to be performed in a distributed fashion, alongside the filtering task of interest, thereby further reducing the network's energy consumption. We provide a description of the algorithm, prove its convergence, and validate its performance and solution tracking capabilities with numerical experiments.**

***Index Terms*—Wireless sensor networks, distributed signal processing, non-smooth optimization, spatial filtering.**

I. Introduction

THE advent of Cloud computing and the so-called Big Data era has led to a significant increase in the amount of data that is being generated, and subsequently processed. The vision put forth by the Cloud is that data is generated at the "edge", and compute is located at the "center" [2], making it the most extreme form of centralized computing. In the context of Wireless Sensor Networks (WSNs), the center is usually referred to as the fusion center (FC) and is responsible for the analysis and processing of the signals gathered by a possibly large set of wirelessly connected sensor nodes. Although conceptually simple, this "data centralization" approach comes with several challenges. First, it requires significant bandwidth to transfer data from the sensing devices to the central computing location [3]. This can be particularly problematic in the case of low-power WSNs, where centralized data aggregation can have a significant impact on the battery life of the individual sensor nodes, making long term or remote deployments impractical. Second, the back and forth communication between the computing device and sensing devices can prevent real-time computations to be performed due to increased network latency [2]. Time-critical tasks such as speech enhancement in acoustic sensor networks [4], auditory attention decoding in EEG sensor networks [5], [6] or target detection in decentralized radar systems [7], require real-time data analysis to be of any use. Finally, the FC constitutes a single point of failure [8], which can lead to loss of service in case of malfunction. This motivates the development of distributed algorithms that can process the data both locally and in real-time, at the edge of the network, or collaboratively compute optimally compressed representations of the data that can be efficiently offloaded to an external computing device, at a much lower cost than the transmission of the raw sensor data.

Distributed datasets as those sensed by sensor networks can be broadly classified into two categories: those with distributed features, and those with distributed samples. The latter is usually less challenging to process, as the objective function can typically be decomposed as a sum of local node-specific objective functions that only require the samples from one node. In this case, the distributed task can often be solved by first processing the samples of each node locally and independently from the other nodes, and then aggregating the intermediate local results (which typically consist of a parameter vector, rather than individual time samples). Typical examples of algorithms suited for datasets with distributed samples are those put forth by federated learning [9], the Alternating Direction Method of Multipliers (ADMM) [10], [11], and consensus and diffusion strategies [12], [13], amongst others.

Received 13 March 2024; revised 29 August 2024; accepted 2 October 2024. Date of publication 4 October 2024; date of current version 23 October 2024. This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under Grant Agreement 802895 and Grant Agreement 101138304 and in part by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme. An earlier version of this paper was presented at the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [DOI: 10.1109/ICASSP49357.2023.10096408]. The associate editor coordinating the review of this article and approving it for publication was Dr. Renato L. G. Cavalcante. *(Corresponding author: Charles Hovine.)*

The authors are with the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics and with the Leuven AI Institute for Artificial Intelligence, KU Leuven, 3001 Leuven, Belgium (e-mail: charles.hovine@esat.kuleuven.be; alexander.bertrand@esat.kuleuven.be).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2024.3474168, provided by the authors.

Digital Object Identifier 10.1109/TSP.2024.3474168

1053-587X © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

In the case of distributed features, the objective function cannot be decomposed into a sum of local objectives at the individual nodes, making decentralized processing significantly more challenging. For example, in the context of spatial filtering in WSNs, the goal is to linearly combine the different sensor channels of all nodes in an optimal data-driven fashion, which requires measurements of the signal covariance across all node pairs in the WSN. Existing algorithms for distributed-features problem are typically not suited for the online setting of WSNs. Indeed, they typically rely on some form of consensus inner loop that requires the data associated with a single batch of samples to be retransmitted and iterated over multiple times, and often require that the inner iterative loop associated with a single batch of samples converges before moving on to the next batch [10], [14], [15], [16]. [17] is an example of algorithm that could be adapted to the batch-adaptive setting of WSN, but it is limited to solving smooth and unconstrained optimization problems, in star-topology networks only, contrarily to this paper.

This work extends the Distributed Adaptive Signal Fusion (DASF) algorithm [18], [19], which was originally designed to solve such distributed features problems, in particular in the context of spatial filtering in WSNs, where the sensor channels are distributed across the nodes of a WSN. Specifically, the DASF framework provides a generic "meta-algorithm" that computes adaptive spatial filters whose coefficients are the solutions of some smooth optimization problem, where the latter defines the optimal centralized spatial filter based on the network-wide signal statistics (which are assumed to be unknown at start-up). The DASF algorithm relies on the exchange of linearly compressed views of the nodes' data, which are then used by each node to locally solve a subproblem preserving the original problem structure, and iteratively producing a better estimate of the optimal filter coefficients. By spreading iterations of the algorithm over different sample batches, DASF is able to adaptively track the optimal filters based on their evolving statistics, which are estimated online using the most recently collected samples. The DASF framework is applicable to a wide class of spatial filtering problems, including the trace ratio problem [20], principal component analysis [21], generalized eigenvalue problems [22], minimum mean squared error filtering, minimum variance beamforming [23], and single and multi-view canonical correlation analysis [24]. However, the DASF algorithm requires the optimization objective to be smooth, which notably prevents the use of the ℓ_1 norm, which is often used in signal processing to encourage sparsity in the solutions. Our focus in this work, is the extension of the DASF algorithm to non-smooth problems, with the side goal of performing ℓ_1-induced node or sensor selection alongside a given filtering task (rather than performing an a priori node selection based on some task-agnostic criterion as in, e.g., [25]).

Our contribution is three-fold. Firstly, we propose an extension of the original DASF algorithm to certain classes of *non-smooth* adaptive spatial filtering problems, referred to as non-smooth DASF (NS-DASF). Secondly, we provide a convergence and optimality proof for the NS-DASF algorithm based on milder assumptions than the original DASF algorithm. The new assumptions and optimality in the case of non-smooth problems lead to a very different proof strategy compared to the convergence proof of the smooth version of DASF. Finally, we apply our algorithm to the problem of node selection in WSNs, where only a subset of the nodes is required to contribute to the filtering task. In this paper, we show via numerical experiments that the problem of (distributed) node selection can be solved concurrently with the filtering task, by the addition of an appropriate regularizer to the optimization problem.

The outline of the paper is as follows. In Section II we formalize the scope of DASF and NS-DASF in a WSN context. In Section III we describe the NS-DASF algorithm. We first introduce the simpler case of fully-connected connected networks, before extending the description to arbitrary network topologies. In Section IV, we prove the convergence and optimality of NS-DASF. Section V describes several numerical experiments, and Section VI concludes the paper with a brief discussion.

II. Problem Statement

We consider a network of K nodes with labels in $\mathcal{K} = \{1, \dots, K\}$. Each node k senses an M_k-channel stochastic signal $\boldsymbol{y}_k(t)$ with values in $\mathbb{R}^{M_k}$ for each sample $t \in \mathbb{Z}$. We denote the network-wide M-channel signal with values in $\mathbb{R}^M$ as $\boldsymbol{y}(t) \triangleq [\boldsymbol{y}_1(t)^T, \dots, \boldsymbol{y}_K(t)^T]^T$, where $M \triangleq \sum_k M_k$. In this paper, we focus on the distributed computation of an adaptive M-inputs Q-outputs spatial filter $X \in \mathbb{R}^{M \times Q}$, that is optimal in some sense, and structured as $X \triangleq [X_1^T, \dots, X_K^T]^T$, where X_k is the k-th block of X, corresponding to the filter associated with $\boldsymbol{y}_k$.

A. Scope of the Original DASF Framework

The original smooth version of DASF [18], [19], applies to filtering problems of the form

$$\begin{aligned} X^\star \in \underset{X \in \mathbb{R}^{M \times Q}}{\operatorname{argmin}} \ & \varphi(X^T \boldsymbol{y}(t), X^T B) \\ \text{s.t.} \quad & \forall j \in \mathcal{J}_I,\ \eta_j(X^T \boldsymbol{y}(t), X^T D_j) \leq 0, \\ & \forall j \in \mathcal{J}_E,\ \eta_j(X^T \boldsymbol{y}(t), X^T D_j) = 0. \end{aligned} \tag{1}$$

where the matrices B and D_j[1] are deterministic matrices known by every node, φ is a smooth real-valued function encoding some design objective for the filter output, $\mathcal{J}_I$ and $\mathcal{J}_E$ are the sets of inequality and equality constraints indices, respectively, and η_j are smooth functions enforcing some hard constraints on the filter outputs and/or filter coefficients. As $\boldsymbol{y}(t)$ is a stochastic signal, we assume that the aforementioned functions implicitly contain an operator extracting some statistics from the signal, such as e.g. an expectation or covariance operator, hence keeping the problem deterministic[2]. In order for the nodes to be able to evaluate, or at least approximate those quantities,

[1]The non sequitur notation is chosen to stay consistent with the notation introduced in [18], where C_k has another meaning, used later in this paper.

[2]In order to be perfectly rigorous, we could have defined the domain of the above functions as a subset of some Hilbert space of random signals. This would however have introduced unnecessary complexity, as for all intents and purposes, the random signals could be replaced by sample matrices and leave our developments mostly unchanged.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

we assume that $\boldsymbol{y}(t)$ is ergodic, i.e. its statistics can be evaluated using sample averages. We furthermore assume that $\boldsymbol{y}(t)$ is short-time stationary and that its statistics change sufficiently slowly such that they can be tracked by the updates of the DASF algorithm. We emphasize this fact by dropping the time index t for most of the remainder of this paper.

$$X^\star \in \underset{X}{\operatorname{argmax}}\, X^T \mathbb{E}\left\{\boldsymbol{y}\boldsymbol{y}^T\right\} X \tag{2}$$

$$\text{s.t.} \quad X^T X = I \tag{3}$$

is a typical instance of (1). It produces a filter that extracts the principal components of $\boldsymbol{y}$. In this particular case $B = 0$, and the sole constraint is

$$\eta(X^T\boldsymbol{y}, X^T D) = X^T D X = I, \tag{4}$$

with $D = I$. Although the D matrix looks superfluous here, it plays an important algorithmic role, and allows DASF to solve (1) by solving a sequence of lower-dimensional versions of (1), but where D (or B when it is non-zero) is not equal to the identity matrix anymore.

B. Extended Scope of the NS-DASF Framework

The NS-DASF algorithm deals with filters that are solutions of optimization problems of the form

$$X^\star \in \underset{X\in\mathbb{R}^{M\times Q}}{\operatorname{argmin}}\ \varphi(X^T\boldsymbol{y}(t), X^T B) + \gamma(X^T A)$$
$$\text{s.t.} \quad \forall k \in \mathcal{K}, \forall j \in \mathcal{J}_I^k,\ \eta_j(X_k^T\boldsymbol{y}_k(t), X_k^T D_{j,k}) \le 0,$$
$$\forall j \in \mathcal{J}_E^k,\ \eta_j(X_k^T\boldsymbol{y}_k(t), X_k^T D_{j,k}) = 0. \tag{5}$$

The matrices A and $D_{j,k}$[3] are, similarly to B, deterministic matrices known by every node. γ is a possibly non-smooth but convex[4] real-valued function encoding some soft-constraints on the filter coefficients. The η_j's are smooth functions now describing *per-node* constraints on the filter. Note that the block-separability of the constraints is specific to the non-smooth version of DASF, and is not required for its smooth counterpart. The η_j's in (1) can thus possibly introduce multiplicative coupling between the filter coefficients of different nodes, but the η_j's in (5) can only depend on a single block X_k. As described in Subsection IV-F, there is an upper-bound on the maximal allowable number of *per-node* constraints, which depends on the network topology. We also require γ to be *per-node* block separable, i.e. there exist functions γ_k such that

$$\gamma(X^T A) = \sum_{k\in\mathcal{K}} \gamma_k(X_k^T A_k) \tag{6}$$

This implies that A must be a block diagonal matrix, whose blocks we denote A_k. Typical examples of functions satisfying this property are the weighted ℓ_1 and $\ell_{2,1}$ norms, where the later is typically used to introduce group-sparsity in an optimization model, and the per-node sum of nuclear norms, promoting locally low-rank filters. Note that the smooth function φ is not required to be block-separable. Although this block-separability requirement of the constraints can seem arbitrary, it is essential to ensure the optimality of the proposed procedure (see Section IV). As will be seen in Subsection III-B, the separability of the constraints poses an additional challenge in terms of the required data exchanges between the nodes. We denote the parametric optimization problem defined in (5) as $\mathbb{P}(\boldsymbol{y}, A, B, D)$, where $\mathcal{D}$ denotes the collection (i.e. set) of matrices $D_{j,k}$.

The problem description (5) is purposedly kept generic and somewhat abstract in order to cover as many problems as possible. However, this structure is satisfied by several problems of interest. For example, the problem

$$\max_X \operatorname{Tr}\left(X^T \mathbb{E}\left\{\boldsymbol{y}(t)\boldsymbol{y}(t)^T\right\} X\right)$$
$$\text{s.t.} \quad X_k^T \mathbb{E}\left\{\boldsymbol{y}_k(t)\boldsymbol{y}_k(t)^T\right\} X_k = I_Q \quad \forall k \in \mathcal{K} \tag{7}$$

where I_Q is the Q-dimensional identity matrix, and $\operatorname{Tr}(\cdot)$ and $\mathbb{E}\{\cdot\}$ denote the trace and expectation operators, respectively, is typically referred to as the SUMCORR formulation of generalized canonical correlation analysis (GCCA) [26], [27], [28] and can be cast in the form (5). SUMCORR extracts signal components that are highly correlated between the nodes, and can for example be used to find a subspace which is observed by every node in the network [28], [29]. By adding $\gamma(X) = \sum_k \|X_k\|_F$ to the objective function of (7), where $\|\cdot\|_F$ denotes the Frobenius norm, we obtain a sparse version of SUMCORR, which encourages a subset of the nodes to participate in the problem.

As another example, the following problem can be viewed as a sparse Wiener filtering problem [30] with additional power constraints on the per-node filter outputs:

$$\min_X \mathbb{E}\left\{\left\|X^T\boldsymbol{y}(t) - \boldsymbol{d}(t)\right\|_F^2\right\} + \|X\|_\infty$$
$$\text{s.t.} \quad \mathbb{E}\left\{\left\|X_k^T\boldsymbol{y}_k(t)\right\|_F^2\right\} \le P_k \quad \forall k \in \mathcal{K} \tag{8}$$

where $\|\cdot\|_\infty$ is the matrix ∞-norm, corresponding to the largest ℓ_1-norm of its rows, encouraging the least number of input channels to be used. P_k are scalars denoting some power constraint on the filter output, and $\boldsymbol{d}(t)$ is a known target signal taking values in $\mathbb{R}^Q$. The Wiener filter is typically used in denoising applications, for example in acoustic sensor networks, where $\boldsymbol{d}$ could for example be a known speech signal and $\boldsymbol{y}$ the recordings of several microphone arrays [4]. Note that these are only two examples of the many problems that fit in the proposed non-smooth DASF framework.

Our goal is to efficiently track a solution $X^\star \in \mathbb{R}^{M\times Q}$ of (5) and the corresponding filter output $X^{\star T}\boldsymbol{y}(t)$. Although the optimal filter $X^\star$ is required, we are typically more interested in the output signals of the spatial filter, i.e., in the Q-dimensional filtered output $\boldsymbol{z}(t) \triangleq X^{\star T}\boldsymbol{y}(t)$ for each sample time t, which could, for example, correspond to a denoised speech signal that must be available to the network at any time. The DASF algorithm must therefore be such that both the optimal filter and the filtered signal $\boldsymbol{z}(t)$ can be computed in a bandwidth efficient manner.

The NS-DASF algorithm assumes that a centralized solver that would be able to find the solution of (5) if all data would

[3]Most problems are usually formulated with $\gamma(AX)$ rather than $\gamma(X^T A)$. This notation was chosen to stay consistent with the $X^T\cdot$ structure of the remaining functions involved in the problem.

[4]Note that neither strong or strict convexity is required.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

be known at a fusion center is available. Similarly to the original DASF algorithm, the NS-DASF algorithm will use this solver to compute the solution of lower-dimensional versions of (5) that are available at individual nodes. We also assume that computing a solution of this local lower dimensional problem is cheap in comparison to the cost of sharing the data $\boldsymbol{y}(t)$, which motivates the design of a distributed algorithm that can compute $X^\star$ and $\boldsymbol{z}(t)$ while also limiting the amount of data that needs to be exchanged between the nodes, by relying on local computations instead. This is a reasonable assumption, as it is well known that the wireless data exchange is typically an energy bottleneck in WSNs [3], [31].

III. The NS-DASF Algorithm

In this section, we describe the extension of DASF to non-smooth problems (NS-DASF), i.e. we describe an iterative procedure to solve (5) in a distributed fashion, while also tracking the filtered output $\boldsymbol{z}(t)$ at each node. The procedure relies on each node sending a compressed view of its local data to a given node, which we call the *updating node*, and whose role is assumed by a different node at each iteration. Based on the compressed data it received, the updating node will update the current estimate of $X^\star$. For the sake of an easier exposition, we first introduce the algorithm in fully-connected networks before extending the description to arbitrary network topologies at the end of the section.

A. NS-DASF in Fully-Connected Networks

The state of the algorithm at any iteration i is characterized by the (initially random) current estimate of the optimal solution X^i and the updating node index q^i (we use the node index q without any iteration index to refer to the updating node when the iteration is clear from the context). The algorithmic procedure is essentially the same as the one described in [18] for its smooth counterpart, except for the handling of the term γ and the resulting local problems solved by each updating node. However, the convergence analysis (and in particular the optimality results) is substantially impacted by the addition of this non-smooth term (see Section IV).

Each iteration is divided into three phases:

i) Data Aggregation: Each node k collects N new samples of $\boldsymbol{y}_k$ and sends a block of N Q-dimensional *compressed* samples of

$$\widehat{\boldsymbol{y}}_k^i \triangleq X_k^{iT} \boldsymbol{y}_k \tag{9}$$

along with

$$\widehat{D}_{j,k}^i \triangleq X_k^{iT} D_{j,k} \tag{10}$$

$$\widehat{A}_k^i \triangleq X_k^{iT} A_k \tag{11}$$

$$\widehat{B}_k^{iT} \triangleq X_k^{iT} B_k \tag{12}$$

to the updating node q, where B_k is the block-row of B associated with X_k. Note that X_k^i acts both as the compression matrix and as the current estimate of the optimal filter. Upon reception of the compressed data, the updating node q constructs the *local* data

$$\begin{aligned}
\widetilde{\boldsymbol{y}}^i &= \begin{bmatrix} \boldsymbol{y}_q^T & \widehat{\boldsymbol{y}}_1^{iT} & \cdots & \widehat{\boldsymbol{y}}_{q-1}^{iT} & \widehat{\boldsymbol{y}}_{q+1}^{iT} & \cdots & \widehat{\boldsymbol{y}}_K^{iT} \end{bmatrix}^T, \\
\widetilde{A}^i &= \mathrm{BlkDiag}(A_q, \widehat{A}_1^i, \ldots, \widehat{A}_{q-1}^i, \widehat{A}_{q+1}^i, \ldots, \widehat{A}_K^i), \text{ and} \\
\widetilde{B}^i &= \begin{bmatrix} B_q^T & \widehat{B}_1^{iT} & \cdots & \widehat{B}_{q-1}^{iT} & \widehat{B}_{q+1}^{iT} & \cdots & \widehat{B}_K^{iT} \end{bmatrix}^T
\end{aligned} \tag{13}$$

adding node q's own data to the aggregated data. Similarly to $\mathcal{D}$, we denote the collection of the $\widehat{D}_{j,k}^i$'s and the $D_{j,q}$ as $\widetilde{\mathcal{D}}^i$. One can already notice that the "local" data $\widetilde{\boldsymbol{y}}^i$, $\widetilde{A}^i$, $\widetilde{B}^i$ and $\widetilde{\mathcal{D}}^i$ live in a subspace of the original data $\boldsymbol{y}$, A, B and $\mathcal{D}$, and can be interpreted as a low-dimensional "view" (i.e. linear combination of the channels/rows) of the original data, where the view of the node's own data is unaltered (i.e. uncompressed).

ii) Local Solution: In order to update the current estimate of the optimal filter X^i, the updating node q computes a solution of the original problem (5), but using the aggregated data (13) received in the previous step instead of the original, global, data $\boldsymbol{y}$, A, B, and $\mathcal{D}$. More specifically, it solves $\mathbb{P}(\widetilde{\boldsymbol{y}}^i, \widetilde{A}^i, \widetilde{B}^i, \widetilde{\mathcal{D}}^i)$:

$$\begin{aligned}
&\widetilde{X}^\star \in \underset{\widetilde{X}}{\operatorname{argmin}}\ \varphi(\widetilde{X}^T \widetilde{\boldsymbol{y}}^i, \widetilde{X}^T \widetilde{B}^i) + \gamma(\widetilde{X}^T \widetilde{A}^i) \\
\text{s.t.}\quad &\forall k \in \mathcal{K},\ \forall j \in \mathcal{J}_I^k,\ \eta_j(\widetilde{X}_k^T \widehat{\boldsymbol{y}}_k^i, \widetilde{X}_k^T \widehat{D}_{j,k}^i) \leq 0, \\
&\forall j \in \mathcal{J}_E^k,\ \eta_j(\widetilde{X}_k^T \widehat{\boldsymbol{y}}_k^i, \widetilde{X}_k^T \widehat{D}_{j,k}^i) = 0
\end{aligned} \tag{14}$$

where we denote $\widehat{\boldsymbol{y}}_q^i \triangleq \boldsymbol{y}_q$, $\widehat{B}_q^i \triangleq B_q$ and $\widehat{D}_q^i \triangleq D_q$ for the special case of the updating node, and where $\widetilde{X}^\star$ is partitioned as

$$\widetilde{X}^\star = [\widetilde{X}_q^{\star T}, \widetilde{X}_1^{\star T}, \ldots, \widetilde{X}_{q-1}^{\star T}, \widetilde{X}_{q+1}^{\star T}, \ldots, \widetilde{X}_K^{\star T}]^T. \tag{15}$$

with each block associated with the corresponding blocks of the local data (13). Note that $\widetilde{X}_q$ is an $M_q \times Q$ matrix, whereas all the other $\widetilde{X}_k$'s are $Q \times Q$ matrices. Also note that because the node receives blocks of samples, it will not solve $\mathbb{P}(\widetilde{\boldsymbol{y}}^i, \widetilde{A}^i, \widetilde{B}^i, \widetilde{\mathcal{D}}^i)$ exactly, but an approximation thereof, where the implicit statistics involved in φ and the η_j's are approximated using sample averages across a batch of N samples.

As (14) shares the structure of the original problem (5), it can be solved using the same solver. It should be noted that the dimension of this local problem is much smaller than the original problem, and therefore cheaper to solve, making it amenable to run on devices with limited computing capabilities.

iii) Parameters Update: Following the partitioning defined in (15), the updating node q updates its own filter according to

$$X_q^{i+1} \leftarrow \widetilde{X}_q^\star \tag{16}$$

and sends the appropriate blocks of $\widetilde{X}^\star$ to the other nodes, such that they can update their local filters according to

$$X_k^{i+1} \leftarrow X_k^i \widetilde{X}_k^\star. \tag{17}$$

The above rule is related to (9)-(10): At each node, the signal $\boldsymbol{y}_k$ of a node can only be manipulated by the updating node "through" the current estimate of the filter X_k^i, via the parametrization introduced by $\widetilde{X}_k^\star$.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: NS-DASF algorithm in fully-connected networks.

begin
- $i \leftarrow 0$, $q \leftarrow 1$, Randomly initialize X^0
- **loop**
  - **for** $k \in \mathcal{K} \setminus \{q\}$ **do**
    - **At node** k
      - Collect a new batch of N samples of $\boldsymbol{y}_k(t)$ and send the compressed samples $\widehat{\boldsymbol{y}}_k^i(t) = X_k^{iT}\boldsymbol{y}_k(t)$ along with $\widehat{A}_k^i = X_k^{iT}A_k$, $\widehat{B}_k^i = X_k^{iT}B_k$ and $\widehat{D}_k^i = X^{iT}D_k$ to node q.
  - **At node** q
    - Obtain $\widetilde{X}^\star$ by solving and selecting any solution of $\mathbb{P}(\widehat{\boldsymbol{y}}^i(t), \widehat{A}^i, \widehat{B}_k^i, \widehat{\mathcal{D}}^i)$ (see (14)).
    - Extract the $\widetilde{X}_k^\star$'s from $\widetilde{X}^\star$ according to (15).
    - $X_q^{i+1} \leftarrow \widetilde{X}_q^\star$
    - **for** $k \in \mathcal{K} \setminus \{q\}$ **do**
      - Send $\widetilde{X}_k^\star$ to node k.
      - **At node** k
        - $X_k^{i+1} \leftarrow X_k^i \widetilde{X}_k^\star$
  - $i \leftarrow i+1$, $q \leftarrow (q+1) \mod K$

Upon completion of the above steps, the updating node role is passed-on to another node[5] and another iteration begins, using another batch of N samples. At the end of each iteration, the updating node has access to an estimate of the output filtered signal for the latest N-samples block:

$$\boldsymbol{z} \approx \widetilde{X}^{\star T}\widetilde{\boldsymbol{y}}^i = X^{i+1T}\boldsymbol{y}. \tag{18}$$

As a different batch of N samples is used at each iteration, the (NS-)DASF algorithm produces an estimate of the filtered signal for each N-samples block, while at the same time improving the estimate of the optimal spatial filter $X^\star$, such that each new block of the filtered signal is closer to the desired filtered signal (under the stationarity assumption). In other words, (NS-)DASF acts as a time-recursive block-adaptive filter, which continually adapts itself to the (possibly changing) statistics of $\boldsymbol{y}(t)$ [18].

The full algorithm description is given by Algorithm 1.

B. NS-DASF in Arbitrary Network Topologies

A fully-connected network topology allows the updating node to receive the compressed data of every other node directly. In an arbitrary network topology, the updating node can only receive data from its neighbors. Applying the same procedure as in the fully-connected case, i.e. requiring the nodes to relay the compressed data of their neighbors to the updating node, would result in a significant communication overhead, most extreme in the case of line topologies. To avoid this, it was proposed in [18] to construct at each iteration a spanning tree that is rooted at the updating node, and let each node fuse (i.e. sum) the compressed data of its neighbors before relaying it to the neighbor closest to the updating node. In other words, the updating node will receive some linear combination of the compressed data of the rest of the network. With this interpretation in mind, the procedure described for fully connected networks can readily be applied, considering each branch at a given iteration *as if it were a single node*. The local variables X_k of each node in a branch are therefore updated with the same linear transformation (i.e. there is a single matrix $\widetilde{X}_{(\cdot)}$ per branch).

As we have required the constraints to be block separable, we would in practice still need to forward all the compressed data $X_k^{Ti}\boldsymbol{y}_k$ in order for the updating node to be able to evaluate each $\eta_j(X_k^{Ti}\boldsymbol{y}_k, X^{Ti}D_k)$. As this would require expensive data relaying, we require the constraints to depend only on the first or second order statistics of $X_k^T\boldsymbol{y}_k$ in the arbitrary topology case. This allows the nodes to relay the compressed data $X_k^{Ti}D_k$ along with $X_k^{Ti}\mathbb{E}\{\boldsymbol{y}_k\boldsymbol{y}_k^T\}X_k$ or $X_k^{Ti}\mathbb{E}\{\boldsymbol{y}_k\}$ instead of a batch of N samples of $X_k^{Ti}\boldsymbol{y}_k$, which is of much larger dimension than the compressed statistics (which have a dimension of the same order of magnitude as the parameter update matrices $\widetilde{X}$). For example, to handle constraints of the form

$$X_k^T\mathbb{E}\{\boldsymbol{y}_k\boldsymbol{y}_k^T\}X_k = I_Q,$$

the nodes would relay (sample-averaged estimates of) $\widehat{R}_k^i \triangleq X_k^{iT}\mathbb{E}\{\boldsymbol{y}_k\boldsymbol{y}_k^T\}X_k^i$ itself and still be able to evaluate the constraints, as then the local constraints in (14) would be of the form

$$\widetilde{X}_k^T\widehat{R}_k^i\widetilde{X}_k = I_Q. \tag{19}$$

In what follows, we denote the per-node covariance matrix $\mathbb{E}\{\boldsymbol{y}_k\boldsymbol{y}_k^T\}$ as R_k, and the per-node first order statistics $\mathbb{E}\{\boldsymbol{y}_k\}$ as r_k. The procedure in arbitrary topologies is as follows.

(i) Data Aggregation: A spanning tree rooted at the updating node and preserving the links with its neighbors is computed in a distributed fashion, using e.g. [25, Algorithm 4]. We denote the set of nodes in the subtree (i.e. branch) containing k and obtained by removing the link between k and q as $\mathcal{B}_{kq}$ (see Fig. 1 for an example). The compressed data $\widehat{D}^i$ and $\widehat{A}_k$ are forwarded to the updating node. Similarly, each node k computes and forwards its compressed first and second-order statistics

$$\widehat{R}_k^i = X_k^{Ti}R_kX_k^i, \tag{20}$$
$$\widehat{r}_k^i \triangleq X_k^{Ti}r_k \tag{21}$$

based on the latest available batch of samples. We denote the compressed data associated with subtree $\mathcal{B}_{kq}$ as

$$\widehat{\boldsymbol{y}}_{kq}^i = \sum_{l\in\mathcal{B}_{kq}} \widehat{\boldsymbol{y}}_l^i, \text{ and}$$
$$\widehat{B}_{kq}^i = \sum_{l\in\mathcal{B}_{kq}} \widehat{B}_l^i. \tag{22}$$

This can be computed recursively by having each node k in the subtree sum the data it receives from its children, and then forward the result to its parent. This ensures that the dimension of the data sent by each node stays equal to Q, independently of the network size and topology, making the communication of the compressed N-sample batches fully scalable. The full aggregation procedure is formally described by Algorithm 2 and an illustrative example is shown in Fig. 1. Upon completion, the updating node q has access to the compressed data $\widehat{\boldsymbol{y}}_{kq}^i$ of each

[5]As will be discussed in Section IV, the order does not matter as long as each node acts as an updating node an infinite number of times.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

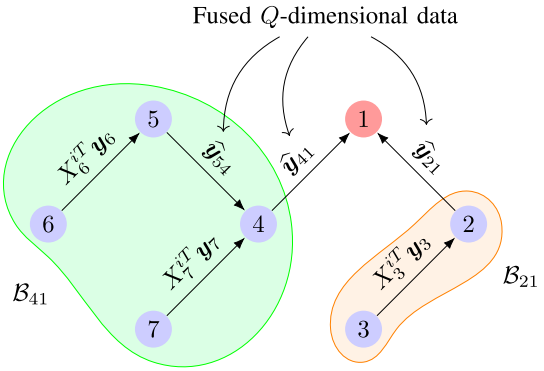


Fig. 1. Example of an aggregation scheme in a spanning tree rooted at node 1. Transmission of $\widehat{R}_k^i$, $\widehat{B}_k^i\widehat{D}_k^i$ and $\widehat{A}_k^i$ omitted.

Algorithm 2: Recursive aggregation procedure in a tree-topology network rooted at node q.

input : Parent node p_k and set of childrens $\mathcal{C}_k$ for each node $k \neq q$ (the parent node is the neighbor closest to the updating node q, $\mathcal{C}_k = \emptyset$ for leaf nodes)

begin

At node k

Collect a new batch of samples of $\boldsymbol{y}_k(t)$

Wait for the aggregate compressed signals received from children $\widehat{\boldsymbol{y}}_{lk}^i$ along with the sets of compressed matrices $\{\widehat{D}_{j,m}^i\}_{m\in\mathcal{B}_{lk}}$, $\{\widehat{A}_m^i\}_{m\in\mathcal{B}_{lk}}$, $\{\widehat{R}_m^i\}_{m\in\mathcal{B}_{lk}}$ and $\{\widehat{r}_m^i\}_{m\in\mathcal{B}_{lk}}$ for $l \in \mathcal{C}_k$

Send $\widehat{\boldsymbol{y}}_{kp_k}^i = \widehat{\boldsymbol{y}}_k^i + \sum_{l\in\mathcal{C}_k}\widehat{\boldsymbol{y}}_{lk}^i$ to p_k and similarly for $\widehat{B}_{kp_k}^i$

Send $\{\widehat{D}_{j,k}^i\} \cup_l \{\widehat{D}_{j,m}^i\}_{m\in\mathcal{B}_{lk}}$, $\{\widehat{A}_k^i\} \cup_l \{\widehat{A}_m^i\}_{m\in\mathcal{B}_{lk}}$, $\{\widehat{R}_k^i\} \cup_l \{\widehat{R}_m^i\}_{m\in\mathcal{B}_{lk}}$ and $\{\widehat{r}_k^i\} \cup_l \{\widehat{r}_m^i\}_{m\in\mathcal{B}_{lk}}$ to p_k

of its neighbors $n \in \mathcal{N}_q$, and the set of compressed matrices $\widehat{R}_k^i$, $\widehat{r}_k^i$, $\widehat{D}_{j,k}^i$ and $\widehat{A}_k$ for every node in the network.

The updating node constructs the local data $\widetilde{\boldsymbol{y}}^i$ as

$$\widetilde{\boldsymbol{y}}^i \triangleq \begin{bmatrix} \boldsymbol{y}_q^T & \widehat{\boldsymbol{y}}_{n_1k}^{iT} & \cdots & \widehat{\boldsymbol{y}}_{n_Lk}^{iT} \end{bmatrix}^T,$$
$$\widetilde{B}^i \triangleq \begin{bmatrix} B_q^T & \widehat{B}_{n_1k}^{iT} & \cdots & \widehat{B}_{n_Lk}^{iT} \end{bmatrix}^T \tag{23}$$

where $\{n_1, \ldots, n_L\} = \mathcal{N}_q$ are the neighbors of node q. The matrix $\widetilde{A}^i$, is constructed as in the fully connected case (13).

(ii) Local Solution: The updating node solves

$$\begin{aligned} \widetilde{X}^\star \in \underset{\widetilde{X}}{\operatorname{argmin}}\ & \varphi(\widetilde{X}^T\widetilde{\boldsymbol{y}}^i, \widetilde{X}^T\widetilde{B}^i) + \gamma(\widetilde{A}^i\widetilde{X}) \\ \text{s.t.}\quad & \forall\, n \in \mathcal{N}_q, \forall l \in \mathcal{B}_{nq} \\ & \forall j \in \mathcal{J}_I^l,\ \eta_j(\widetilde{X}_n^T\widehat{\boldsymbol{y}}_l^i, \widetilde{X}_n^T\widehat{D}_{j,l}^i) \leq 0, \\ & \forall j \in \mathcal{J}_E^l,\ \eta_j(\widetilde{X}_n^T\widehat{\boldsymbol{y}}_l^i, \widetilde{X}_n^T\widehat{D}_{j,l}^i) = 0, \end{aligned} \tag{24}$$

where the dependence on the $\widehat{R}_k^i$ and $\widehat{r}_k^i$ is implicitly embedded in the η_j.

(iii) Parameters Update: $\widetilde{X}^\star$ is now partitioned correspondingly to (23) as

$$\widetilde{X}^\star \triangleq \begin{bmatrix} X_q^{\star T}, \widetilde{X}_{n_1}^{\star T}, \ldots, \widetilde{X}_{n_L}^{\star T} \end{bmatrix}^T. \tag{25}$$

Algorithm 3: NS-DASF algorithm in arbitrary networks.

begin

$i \leftarrow 0$, $q \leftarrow 1$, Randomly initialize $\boldsymbol{X}^0$

loop

Construct a spanning tree rooted at node q.

Aggregate the data according to Alg. 2.

At node q

Obtain $\widetilde{X}^\star$ by solving and selecting any solution of (24).

Extract the $\widetilde{X}_k^\star$'s from $\widetilde{X}^\star$ according to (25).

Update the filter of node q with $\widetilde{X}_q^\star$.

for $n \in \mathcal{N}_q$ **do**

Send $\widetilde{X}_n^\star$ to node n.

for $l \in \mathcal{B}_{nq}$ **do**

At node l

Wait for $\widetilde{X}_n^\star$ and forward it to its children.

$X_l^{i+1} \leftarrow X_l^i\widetilde{X}_n^\star$

$i \leftarrow i+1$, $q \leftarrow (q+1) \mod K$

Similarly to the fully-connected case, the updating node updates its filter with $X_q^\star$ and for each branch $\mathcal{B}_{n_lq}$, the nodes all update their filters according to $X_k^{i+1} \leftarrow X_k^i\widetilde{X}_{n_l}^\star$ for all $k \in \mathcal{B}_{n_lq}$ (Note that $\widetilde{X}_{n_l}^\star$ is the same for every node in the branch). The full algorithmic procedure is described by Algorithm 3.

Remark III.1: The dimension of the local problem (24) directly depends on the number of neighbors of the updating node kept when building the spanning tree. Indeed, the local optimization variable will have dimension $(Q|\mathcal{N}_q| + M_q) \times Q$. We could in practice build a spanning tree ignoring some of the links between the updating node and its neighbors, but this would not yield any savings in required bandwidth (every node still needs to forward its data to some node), and it would be at the expense of convergence speed, as the available degrees of freedom available when minimizing the local problems (24) would then be lower.

IV. Convergence and Optimality

In this section, we provide convergence and optimality results for the proposed NS-DASF algorithm. The addition of the non-smooth term and the different technical assumptions under which convergence and optimality are obtained do not allow for a straightforward extension of the convergence proofs of the original DASF algorithm (see [19]). These proofs are the main contribution of this paper. In order to keep them accessible, the convergence and optimality are first derived for fully-connected networks in Sections IV-A to IV-D, while Section IV-E briefly describes how the same results obtained for fully-connected networks can be extended to arbitrary network topologies. Section IV-G describes our main result via a single theorem stating the convergence and optimality of DASF for any network topology, including fully-connected networks.

Similarly to the original proofs described in [19] and as described earlier in Section II, we must make two simplifying assumptions on $\boldsymbol{y}(t)$ to ensure that the optimal solution $X^\star$ is time-independent and does not vary across iterations, which

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

would make the convergence analysis of the algorithm mathematically intractable.

Short-Time Stationarity: We assume that the stochastic signal $\boldsymbol{y}(t)$ is stationary, as is typically assumed in the convergence analysis of adaptive filters. However, this should not be viewed as a practical limitation. In order to track the optimal solution $X^\star$, it is in practice sufficient to assume that the changes in the statistics of $\boldsymbol{y}(t)$ are sufficiently slow compared to the convergence time of the algorithm.

Perfect Estimation of the Signal Statistics: As the functions involved in (5) are real-valued, they implicitly depend on the statistics of $\boldsymbol{y}(t)$. In practice, the actual distribution of $\boldsymbol{y}(t)$ is unknown, and its statistics would typically be estimated from sample averages, i.e., the expected value operator in the examples (7) and (8) would be replaced with a sample average over a finite batch of samples. In the (NS-)DASF algorithm, these statistics are estimated on the most recent batch of N samples that were transmitted by the nodes. However, the convergence proof assumes for mathematical tractability that these statistics are estimated perfectly, which means that the convergence results should be viewed as asymptotic results. We refer the reader to the stochastic optimization literature for details on this topic (see, for example, [32]).

A. Relationship Between Local and Global Problems (in Fully-Connected Networks)

Before delving into the actual proof, we give some intuition about the relationship between global and local problems in the case of fully-connected networks (see Section IV-E for the extention to arbitrary network topologies).

From (9)-(13), it can be observed that $\widetilde{\boldsymbol{y}}^i$, $\widetilde{A}^i$, $\widetilde{B}^i$ and $\widehat{D}^i_{j,k}$ are linear (compressive) transformations of $\boldsymbol{y}$, A, B and $D_{j,k}$. Therefore, these variables can be related via a matrix transformation with some matrix C_q^i, such that $\widetilde{\boldsymbol{y}}^i = C_q^{iT}\boldsymbol{y}$ (and similarly for the other matrices, with the exception of $D_{j,k}$, as explained below). From (9)-(13), it is clear that the matrix C_q^i is constructed from the entries in X^i, such that we can define C_q^i as the result of a matrix-valued function $C_q(\cdot)$ such that $C_q^i \triangleq C_q(X^i)$. With this notation (9)-(13) can be written as[6]

$$\begin{aligned}\widetilde{\boldsymbol{y}}^i &= C_q(X^i)^T\boldsymbol{y},\\ \widetilde{B}^i &= C_q(X^i)^T B, \text{and}\\ \widetilde{A}^i &= C_q(X^i)^T A\end{aligned} \tag{26}$$

which is simply the expression in matrix form of the combination of (9)-(13). Note that $C_q(X)$ is here implicitly defined[7].

An update rule naturally emerges from the parametrizations introduced by (26), as by simple associativity,

$$\begin{aligned}\widetilde{X}^{\star T}\widetilde{\boldsymbol{y}}^i &= \widetilde{X}^{\star T}\left(C_q(X^i)^T\boldsymbol{y}\right)\\ &= \left(C_q(X^i)\widetilde{X}^\star\right)^T\boldsymbol{y}\\ &= X^{i+1T}\boldsymbol{y}\end{aligned} \tag{27}$$

(and similarly for B and A). The last equality follows from the structure of C_q, which flows directly from the update rules (16)-(17) (see supplementary materials). This means that we can also re-express (16)-(17) in matrix form as

$$X^{i+1} \leftarrow C_q(X^i)\widetilde{X}^\star. \tag{28}$$

This shows that the local problem (14) at node q can be interpreted as a parametrized version of the centralized problem (5), where the optimization variable is now constrained to a smaller linear subspace defined by the column space of $C_q(X^i)$. We define

$$L(X) \triangleq \varphi(X^T\boldsymbol{y}, X^T B) + \gamma(AX) \tag{29}$$

and the set $\mathcal{X}$ as the set of feasible points of (5), allowing us to express the original global problem (5) as

$$\min_{X\in\mathbb{R}^{M\times Q}} L(X) \quad \text{s.t.} \quad X \in \mathcal{X}. \tag{30}$$

Then, based on (27)-(28), we find that the variable X^{i+1} that is obtained by solving (14) followed by the updates (16)-(17) must be a solution of the following problem:

$$\begin{aligned}X^{i+1} \in \operatorname*{argmin}_X\ & L(X)\\ \text{s.t.}\quad & X \in \mathcal{X}\\ & X \in \operatorname{range} C_q(X^i),\end{aligned} \tag{31}$$

with range denoting the range or column space operator[8]. The block structure[9] of $C_q(X^i)$ allows us to further express this constraint set as

$$\begin{aligned}\mathcal{S}_q(X) &\triangleq \operatorname{range} C_q(X)\\ &= \operatorname{range} X_1 \times \cdots \times \operatorname{range} X_{q-1} \times \mathbb{R}^{I_{M_q}\times Q}\\ &\quad \times \operatorname{range} X_{q+1} \times \cdots \times \operatorname{range} X_K.\end{aligned} \tag{32}$$

The new constraint thus simply restricts each block X_k of the optimization variable X to stay within the range of the corresponding block X_k^i of X^i (at the previous iteration), except for the block associated with the updating node q, which can move freely within the original constraint set. Solving this equivalent problem and updating X^{i+1} with its solution is effectively equivalent to performing the *Local Solution* and *Parameters Update* steps described earlier in Section III-A.

[6]We purposedly omit the matrices $D_{j,k}$ as their case is a bit different. Defining $D'_{j,k} \triangleq [0\ \cdots\ D_{j,k}^T\ \cdots\ 0]^T$, the same relationship can be established by noting that the k-th block of $C_q(X^i)D'_{j,k}$ is $\widehat{D}^i_{j,k}$ and q-th block $C_q(X^i)D'_{j,q}$ is $D^i_{j,q}$,and extending the same reasoning used for $\boldsymbol{y}$. See the explicit structure of C_q in the supplementary materials.

[7]The explicit description and structure of $C_q(X)$ is explained in the supplementary material for the interested reader.

[8]With a slight abuse of notation, as X has Q columns. We thus actually mean the Q-th Cartesian power of the column space.

[9]Owing to the fact that $X_k^{i+1} \leftarrow X_k^i\widetilde{X}_k^\star$ for every $k \neq q$, (see supplementary materials for details).

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

B. Notation and Proof Outline

We can summarize the local problem at any node q by a single set-valued map

$$F_q(X) \triangleq \underset{U \in \mathcal{X} \cap \mathcal{S}_q(X)}{\operatorname{argmin}} \; L(U), \tag{33}$$

such that (31) is equivalent to

$$X^{i+1} \in F_{q^i}(X^i). \tag{34}$$

We will show the convergence of the algorithm by studying the properties of the map (33). We will first show that the successive application of the map converges to the set of its fixed points, defined as

$$\{X \mid \forall k \in \mathcal{K},\ X \in F_k(X)\}. \tag{35}$$

We will then show that the fixed points are solutions, or at least stationary points, of the original problem (5). In order to obtain the most general result, we do not assume any particular order for the sequence of updating nodes q^i, but simply that each node is selected infinitely many times. Formally, this can be expressed as

$$\operatorname{Acc}\ (q^i)_{i \in \mathbb{N}} = \mathcal{K}. \tag{36}$$

where Acc $\cdot$ denotes the set of accumulation points of a sequence[10].

C. Subsequential Convergence (in Fully-Connected Networks)

This first section contains a proof that the NS-DASF algorithm converges to the set of its fixed points, i.e. for any distance δ, we can find an index T, such that for any $i > T$ there is some fixed point X_F of NS-DASF such that $\left\|X_F - X^i\right\|_F < \delta$.

In this first part, we first prove that the procedure results in a monotonic decrease of the objective, and a sequence of feasible points. We then describe the three technical assumptions on which the convergence relies, and finally state the convergence towards fixed points. For readability, the technical details of this first part are located in the appendix.

Let us now show the monotonic decrease of the objective values.

Proposition 1 (Monotonic decrease): Let $(X^i)_{i \in \mathbb{N}}$ be a sequence of iterates generated by (34). Then, the sequence of objective function values $(L(X^i))_{i \in \mathbb{N}}$ is monotonically decreasing. In addition, all the points in $(X^i)_{i \in \mathbb{N}}$ are feasible, i.e. $X^i \in \mathcal{X}$.

Proof: As by the definition of the algorithm (34) X^{i+1} is a solution of (33), it must be feasible (i.e. $X^i \in \mathcal{X}$ for any i). By the definition (32) of $\mathcal{S}_q$, we have that $X \in \mathcal{S}_q(X)$ for every X (as each block of X is trivially in its own range). Combining these two facts yields that $X^i \in \mathcal{X} \cap \mathcal{S}_q(X^i)$, i.e. it is feasible for the local problem (31). As $X^{i+1} \in F_q(X^i)$, it is the solution of (31), and therefore it must be that $L(X^{i+1}) \leq L(X)$ for every X in $\mathcal{X} \cap \mathcal{S}_q(X^i)$. Since we just showed that X^i is also in this set, we find that $L(X^{i+1}) \leq L(X^i)$. □

The remainder of the convergence analysis relies on three technical assumptions, satisfied by a broad class of problems.

Assumption 1 (Continuity, compactness and CCP regularizer): The function $L : \mathcal{X} \mapsto \mathbb{R}$ is continuous and has compact sublevel sets. In addition, γ is closed, convex and proper (CCP).

Assumption 1 is standard in the optimization literature, as lower-semicontinuity and bounded sub-level sets is a sufficient condition for the existence of minimizers.

The closedness and convexity requirements are automatically satisfied for any matrix norm, which are often used as regularizers. Note that the continuity of L on $\mathcal{X}$ prevents γ to encode some constraint via an indicator function, as this would automatically result in a discontinuity. The compactness assumption is required to ensure that the optimization procedure eventually visits point it has already visited, and does not result in a sequence growing or shrinking indefinitely.

Compactness implies both closedness and boundedness. Closedness is automatically satisfied as L is assumed continuous, which is sufficient to ensure closed sub-level sets. When $\boldsymbol{y}$ has linearly independent channels (the covariance matrix $\mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^T\}$ has full rank), boundedness is ensured in the case of a strictly convex φ or γ. Any constraint on the norm of the filter outputs will also result in bounded sub-level sets for L.

Note that the monotonic decrease property from Proposition 1 implies that the compactness of all sublevel sets in Assumption 1 can be relaxed to the compactness of at least the sublevel set of $L(X^0)$ in $\mathcal{X}$, where X^0 is the initialization point of the algorithm.

Assumption 2 (Similar solutions): The solution of $\mathbb{P}(\cdot,\cdot,\cdot,\cdot)$ is unique, or all its solution share the same column space.

This is trivially satisfied by strictly convex problems, or problems that can be cast as subspace problems, i.e., optimization problems where the solution set is a linear subspace[11]. For problems without such a subspace structure, we argue that the occurrence of multiple solutions remains unlikely, as the purpose of the regularizer γ is precisely to discriminate between ambiguous solutions and its introduction will typically produce a single solution [34], [35], [36].

Assumption 2 ensures that in the case of local problems with multiple solutions, all the solutions yield the same local problem at the next iteration, that is the value of $\mathcal{S}_q(X^{i+1})$ is independent of any particular choice of local solution $\widetilde{X}^\star$. Indeed, one may notice from the definition of $\mathcal{S}_q$ in (32) that for any full-rank matrix R, $\mathcal{S}_q(XR) = \mathcal{S}_q(X)$.

The next assumption relies on a parametric description of the constraint set. We define the set-valued map

$$\begin{aligned}\mathcal{X}(\boldsymbol{y}, \mathcal{D}) \triangleq \{X \in \mathbb{R}^{M \times Q} \mid \forall k \in \mathcal{K},\\ \forall j \in \mathcal{J}_I^k,\ \eta_j(X_k^T \boldsymbol{y}_k, X_k^T D_{j,k}) \leq 0,\\ \forall j \in \mathcal{J}_E^k,\ \eta_j(X_k^T \boldsymbol{y}_k, X_k^T D_{j,k}) = 0\}\end{aligned} \tag{37}$$

[10] An accumulation point of a sequence is defined as a point that has infinitely many elements of the sequence in a fixed neighborhood around itself, no matter how small the neighborhood. A converging sequence has a unique accumulation point.

[11] Notable examples are principal component analysis, trace ratio or Rayleigh quotient optimization, canonical correlation analysis, and even SUMCORR under some technical conditions on the covariance matrix (see [33]).

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

which describes the feasible set $\mathcal{X}$ as a parametric set depending on the data $\boldsymbol{y}$ and $\mathcal{D}$. We will keep using $\mathcal{X}$ without any argument to denote the feasible set of the global problem (5).

Assumption 3 (Feasible set continuity): The set-valued map $(\boldsymbol{y}, \mathcal{D}) \mapsto \mathcal{X}(\boldsymbol{y}, \mathcal{D})$ is continuous, i.e., it is both upper and lower semicontinuous[12].

Note that this is a significantly more relaxed assumption than the assumption of continuity of the solution set that was used in the proof of the original DASF algorithm [19].

Set continuity can intuitively be understood as requiring that the minimum distance between any point in the set resulting from an infinetisimal change in the inputs of $\mathcal{X}$, i.e., perturbations in the distribution of $\boldsymbol{y}$ or in the entries of $\mathcal{D}$, and the points of the original set can always be made arbitrarily small by choosing a sufficiently small input perturbation (i.e. the set grows and shrinks "smoothly"). To further illustrate this property, we show that the constraints sets most commonly encountered in signal processing problems are indeed continuous with respect to their input parameters.

a) Linear Constraints: Linear constraints of the form

$$\mathcal{X}(R) = \{X \mid RX = P\} \tag{38}$$

vary continuously with R on trajectories where R has constant rank. Indeed, the solution of this linear system can be expressed via the pseudo-inverse of R (used to express the projection on the null-space of R), which is continuous on the set of matrices with constant rank [40], [41]. Intuitively, if the range of R would suddenly lose a dimension, then a new dimension would appear in the solution space of $RX = P$ (corresponding to R's null-space having gained a dimension).

b) Orthogonality Constraints: Quadratic orthogonality constraints of the form

$$\mathcal{X}(\boldsymbol{y}) = \{X \mid X^T \mathbb{E}\left\{\boldsymbol{y}\boldsymbol{y}^T\right\} X = I\} \tag{39}$$

are continuous on trajectories where $\mathbb{E}\left\{\boldsymbol{y}\boldsymbol{y}^T\right\}$ has full rank. A proof of this fact can be found in the supplementary materials.

c) Convex Inequality Constraints: Convex inequality constraints of the form

$$\mathcal{X}(\boldsymbol{y}, \mathcal{D}) = \{X \mid G(\boldsymbol{y}, X, \mathcal{D}) \leq 0\} \tag{40}$$

where $\leq$ is here element-wise, are continuous if the function G is convex and continuous in X for any $\mathcal{D}$ and distribution of $\boldsymbol{y}$, and if for any $\boldsymbol{y}$ and $\mathcal{D}$ there is a strictly feasible X. A proof can be found in [37]. Typical example of such constraints include linear inequality constraints of the form $X^T D \leq R$ or the power constraints of problem (8).

Remark IV.1: In the first two constraint set examples (linear constraints and orthogonality constraints), the continuity of the constraint set requires the data to have constant/full rank. As described in Appendix A, we actually require the continuity of the *local* constraint set, therefore, keeping in mind that we can express the local constraint sets of the local problems as

$$\mathcal{X} \cap \mathcal{S}_q(X^i) = \mathcal{X}(\widetilde{\boldsymbol{y}}^i, \widetilde{\mathcal{D}}^i), \tag{41}$$

this constant rank assumption must also hold for the subblocks of X, because when X_k^i loses rank, the compressed $\widetilde{\boldsymbol{y}}_k$ will have dependent channels. The subset of full-rank matrices is dense within the global set of matrices, i.e. any rank-deficient matrix is arbitrarily close from a full-rank matrix, and it is therefore unlikely that X_k^i becomes exactly rank deficient in practice. However, it can become ill-conditioned, possibly leading to unstable behavior. In [19], some algorithmic modifications have been proposed for such edge cases, which can be applied to NS-DASF as well (we refer to [19] for further details).

Based on those assumptions, we can state the convergence of the prodecude to fixed points of the algorithms. We defer the proof to the appendix.

Proposition 2 (Subsequential convergence): Let $(X^i)_{i\in\mathbb{N}}$ be any sequence generated by (34). Then under Assumptions 1, 2 and 3, any accumulation point of $(X^i)_{i\in\mathbb{N}}$ is a fixed point of the map F_q for any $q \in \mathcal{K}$.

Proof: See Appendix A. □

Note that this result alone is not sufficient to guarantee the convergence to a single point, as the algorithm could have multiple fixed points, and the procedure could eventually result in a sequence oscillating between those fixed points. This behavior is discussed in more details in Section IV-G.

D. Optimality of Fixed Points (in Fully-Connected Networks)

It now remains to show that the aforementioned fixed points the algorithm converges to are optimal in some sense. We first describe the optimality result for the case of fully-connected networks, before describing it for arbitrary network topologies. For ease of notation let us define the functions

$$f(X) \triangleq \varphi(X^T \boldsymbol{y}(t), X^T B) \text{ and } g(X) \triangleq \gamma(X^T A). \tag{42}$$

In the non-smooth case, a stationary point X° is defined as a point satisfying the following inclusion[13]:

$$0 \in \nabla f(X^\circ) + \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ) \tag{43}$$

where $\nabla f(X^\circ)$ is the gradient of f at X°, $\partial g(X^\circ)$ is the subdifferential of g at X° and $N_{\mathcal{X}}(X^\circ)$ is the normal cone to $\mathcal{X}$ at X° [37], [43]. The sum of sets must be understood as a Minkowski sum[14]. Intuitively, the normal cone to a set at a particular point can be understood as the set of directions having no component "pointing inwards" the set (it therefore only contains 0 at a point in the set's interior). The subdifferential is the set of slopes of all the linear underapproximators tangent to the graph of g at a particular point. This inclusion translates the fact that at a stationary point, any direction of improvement musts exit the constraint set, and is known as Fermat's rule. In the smooth case, (43) is strictly equivalent to the usual KKT conditions [44], [45].

[12] This particular concept of set-continuity has different names depending on fields and authors. Authors in variational analysis typically use "inner" and "outer" semicontinuity [37], authors in mathematical economics use upper and lower "hemicontinuity" [38], and authors in set analysis use the terminology we adopt in this paper [39].

[13] A fairly complete and self-contained introduction to the notions of stationarity in the non-smooth case is available in [42].

[14] $\mathcal{A} + \mathcal{B} \triangleq \{a + b \mid a \in \mathcal{A},\ b \in \mathcal{B}\}$.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

In order to prove that the fixed points of the algorithm satisfy (43), we will show that under the proper assumption (or qualification, in optimization parlance), the local optimality at each node (defined by (33)) implies global stationarity (i.e. being a stationary point of the centralized problem (5)).

We first treat the case of fully-connected networks before moving on to arbitrary topologies.

1) Fully-Connected Networks: We start with a technical lemma that ensures that the tangent cone of $\mathcal{R}_q(X)$ is a subset of the sum of the tangent cones of $\mathcal{X}$ and $\mathcal{S}_q(X)$.

Lemma 1: Let X satisfy the following constraint qualification

$$\forall U \in N_{\mathcal{X}}(X), \quad C_k(X)^T U = 0 \Rightarrow U = 0 \quad \forall k \tag{44}$$

Then for every k,

$$N_{\mathcal{R}_k(X)}(X) \subseteq N_{\mathcal{X}}(X) + N_{\mathcal{S}_k(X)}(X).$$

Proof: See supplementary materials. □

Later on, we will provide a more interpretable sufficient condition for the qualification (44), see Proposition 5. The following proposition states the optimality of fixed points in fully-connected networks. Note that the qualification can be ignored in the case of unconstrained problems (as it is trivially satisfied).

Proposition 3 (Optimality in Fully-Connected Networks): Under Assumption 1, fixed points X° of the algorithm (34) satisfying the qualification (44) are stationary points of problem (5).

Proof: Since X° is a fixed point, it is a solution of the local problem (31) at any node q. This local optimality of X° along with the fact that γ and hence g is CCP (Assumption 1) implies that [43, Theorem 4.75] for every k

$$0 \in \nabla f(X^\circ) + \partial g(X^\circ) + N_{\mathcal{R}_k(X^\circ)}(X^\circ). \tag{45}$$

From (45) and Lemma 1, we find that the following must also hold:

$$0 \in \nabla f(X^\circ) + \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ) + N_{\mathcal{S}_k(X^\circ)}(X^\circ). \tag{46}$$

From [43, Prop. 4.44 & 6.43], the block separability of g and $\mathcal{X}$ implies that

$$\partial g(X^\circ) + N_{\mathcal{X}}(X^\circ) = (\partial g_1(X_1^\circ) + N_{\mathcal{X}_1}(X_1^\circ)) \times \cdots \times (\partial g_K(X_K^\circ) + N_{\mathcal{X}_K}(X_K^\circ)). \tag{47}$$

Therefore, as $N_{\mathcal{S}_k(X^\circ)}$ also has a block structure, and in particular, the block $[N_{\mathcal{S}_k(X^\circ)}]_k = \{0\}$ (as the updating block is unconstrained, any direction is feasible, i.e. $[\mathcal{S}_k]_k = \mathbb{R}^{M_k \times Q}$), we have that for any k,

$$-\nabla_k f(X^\circ) \in \partial g_k(X_k^\circ) + N_{\mathcal{X}_k}(X_k^\circ) \tag{48}$$

and therefore

$$-\nabla f(X^\circ) \in \partial g(X^\circ) + N_{\mathcal{X}}(X^\circ), \tag{49}$$

which is equivalent to (43), hence completing the proof. □

We defer the statement of our main result for fully connected networks to Section IV-G, after giving a more interpretable condition for the qualification of Lemma 1 to hold.

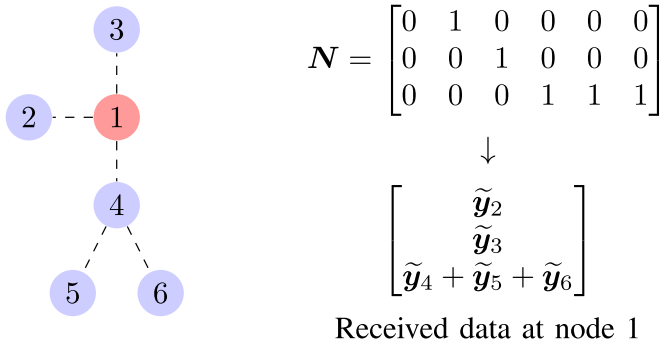


Fig. 2. An example aggregation matrix and received data for a tree rooted at node 1.

E. Extention to Arbitrary Network Topologies

In this subsection, we explain how all previous results, which were derived only for the case of a fully-connected network, can be extended to arbitrary topologies. It only requires a minor change in the description of the in-network fusion process. In the fully-connected case, this fusion process is mathematically described by the matrix C_q. For the case of arbitrary topologies, we have to modify this fusion matrix to describe the per-branch fusion within each of the trees constructed around each updating node q. Indeed, instead of receiving the compressed data $\widehat{\boldsymbol{y}}_k^i = X_k^{iT}\boldsymbol{y}_k$ from all other nodes, the updating node q will receive the linear combination $T_q C_q(X^i)^T \boldsymbol{y}$ where $T_q \in \mathbb{R}^{(M_q+|\mathcal{N}_q|Q)\times(K-1)Q+M_q}$ has the structure

$$T_q = \mathrm{BlkDiag}(I_{M_q}, N \otimes I_Q) \tag{50}$$

where $\otimes$ denotes the Kronecker product, and N is a topology-dependent matrix encoding the aggregation procedure. More specifically, we define some tree graph with the updating node as the root. The matrix N groups the nodes based on the branches $\mathcal{B}_{nq}$ for $n \in \mathcal{N}_q$ (see Fig. 2 for an illustrative example). It has as many rows as the updating node has neighbors, and $K-1$ columns corresponding to all nodes with the column for node q omitted. The element $N_{i,j}$ is 1 if node j is in the i-th branch, and 0 otherwise. The received data $T_q C_q(X^i)^T \boldsymbol{y}^i$ is then a block matrix, where the first block of M_q rows correspond to the updating node's uncompressed data $\boldsymbol{y}_q$, and where each subsequent block of Q rows can be interpreted as the compressed data of a full branch (instead of a single node in the fully-connected setting). Note that in the case of fully-connected networks, $T_q = I$ for any q.

1) Convergence: The results of Sections IV-A to IV-C can be straightforwardly extended to arbitrary network topologies by replacing $C_q(X)$ by $C_q(X)T_q^T$ everywhere it appears in those sections, and thus update the definition of the local range constraint set $\mathcal{S}_q$ in (32) with

$$\mathcal{S}_q'(X) \triangleq \mathrm{range}\, C_q(X) T_q^T. \tag{51}$$

$C_q(X)T_q^T$ shares with C_q the two properties that matters to the proof: its continuity and invariance to full-rank transformations of X. The continuity ensures that the continuity of the local constraint set $\mathcal{X} \cap \mathcal{S}_q'(\cdot)$ in Lemma 2 still applies to arbitrary

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

topologies, and the invariance to full-rank transformation is a key property used in the proof of Proposition 2. The proof of convergence is otherwise identical to the fully-connected case.

2) Optimality: Proposition 3 must be slightly modified to ensure that the proper qualification is met at each of the local problems, as described hereafter. Note that the qualification can again be ignored in the case of unconstrained problems.

Proposition 4 (Optimality in Arbitrary Network Topologies): Fixed points X° of the procedure (34) satisfying for every k the qualification

$$\forall U \in N_{\mathcal{X}}(X^\circ), \quad T_k C_k(X^\circ)^T U = 0 \Rightarrow U = 0 \tag{52}$$

are stationary points of the problem (5).

Proof: Let $\mathcal{R}'_k(X) \triangleq \mathcal{X}(D) \cap \mathcal{S}'_k(X)$. The qualification is sufficient to ensure the inclusion [37]:

$$N_{\mathcal{R}'_k(X)}(X) \subseteq N_{\mathcal{X}}(X) + N_{\mathcal{S}'_k(X)}(X). \tag{53}$$

The reasoning for the above inclusion is the same as Lemma 1 (see supplementary materials), but where we consider $T_k C_k(X)^T$ instead of $C_k(X)^T$. The rest of the proof is identical to the proof of Proposition 3 with $\mathcal{R}_k$ replaced with $\mathcal{R}'_k$ and $\mathcal{S}_k$ replaced with $\mathcal{S}'_k$. □

Note that Proposition 3 is a special case of Proposition 4 where $T_k = I$.

F. Constraint Qualifications and an Upper Bound on the Number of Constraints

In the case of a constrained optimization problem, the qualifications in Propositions 3 and 4 simply ensure that the local solutions satisfy the optimality conditions of the local problems (see [43] for examples failing to meet the qualification). The following proposition gives a sufficient condition akin to the familiar linear independence constraint qualifications (LICQ) for the qualification to hold in the case of equality and inequality constraints in arbitrary topology networks. In what follows, we define $\vartheta_j^k : X_k \mapsto \eta_j(X_k^T \boldsymbol{y}_k, X_k^T D_k)$.

Proposition 5 (Constraint Qualification): Let $\mathcal{A}^k(X) \triangleq \{j \in \mathcal{J}_I^k \mid \vartheta_j^k(X_k) = 0\}$ denote the set of active inequality constraints for node k and

$$\mathcal{D}_{nk} \triangleq \left\{ X_l^T \nabla \vartheta_j^l(X_l) \mid j \in \mathcal{J}_E^l \cup \mathcal{A}^l(X), l \in \mathcal{B}_{nk} \right\}.$$

Then if the elements of $\mathcal{D}_{nk}$ are linearly independent matrices for every pair of neighboring nodes n, k, then it holds that

$$\forall U \in N_{\mathcal{X}}(X), \quad T_k C_k(X)^T U = 0 \Rightarrow U = 0 \tag{54}$$

for any k.

Proof: See supplementary materials. □

The independence of the elements of $\mathcal{D}_{nk}$ implies that the "compressed" gradients $X_l^T \nabla \vartheta_j^l(X_l)$ of all the constraints, associated with all the nodes of a branch $\mathcal{B}_{nq}$ (for some updating node q), must be independent. In order to ensure that $\mathcal{D}_{nq}$ has independent elements, all the nodes in a single branch can (in total) have at most Q^2 constraints (the dimension of the compressed gradients). This requirement can be taken into account when constructing the spanning tree around the updating node q. In the case of fully-connected networks, each branch contains a single node, and the bound can be relaxed to a maximum of Q^2 constraints *per node*. The difference with the similar bound found for the original smooth DASF algorithm [18], [19] can be attributed to the imposed separability of the constraints in the non-smooth problem (5).

Even though the qualification can appear to be hard to check, especially at the unknown limit points of the algorithm, such assumptions are commonly found in the optimization literature (see e.g. [37], [43], [46], [47], [48], [49], [50]). Unless the problem has a specific structure promoting the violation of the qualification, it is unlikely to be violated for a random instance of the problem. Still, we can in some cases guarantee that the qualification holds globally, i.e. at every point in the constraint set. See Appendix B for a worked-out example.

G. Main Result

We finally summarize the convergence and optimality of NS-DASF with the following theorem:

Theorem 1 (Convergence and optimality): Let $(X^i)_{i\in\mathbb{N}}$ be a sequence generated by Algorithm 1 or 3. Assume that the qualification described in Proposition 5 holds at the accumulation points of $(X^i)_{i\in\mathbb{N}}$, along with Assumptions 1–3. Then $(X^i)_{i\in\mathbb{N}}$ converges to the set of stationary points of problem (5).

Proof: By virtue of Proposition 2 (which can be extended to arbitrary topologies by considering $C_q(X)T_q^T$ instead of $C_q(X)$), the accumulation points of $(X^i)_{i\in\mathbb{N}}$ are fixed points of the algorithm, and the sequence therefore converges to the set of fixed points of the algorithm (see the supplementary materials for the proof that a sequence converges to the set of its accumulation points). Finally, Proposition 5 ensures that the qualification (44) is met, which in turns ensures by Proposition 3 the stationarity of fixed points. □

Theorem 1 only guarantees convergence to a set, which can lead to an unstable filter that "jumps" between different solutions across iterations. In order to ensure that the output filter is stable over time, we must ensure the convergence of the residuals $\left\|X^i - X^{i+1}\right\|_F$. In addition to ensuring a stable filter, the convergence of the residuals guarantees that, if the problem has a finite number of stationary points, the filter will converge to a single point (see [19] for a proof). Convergence of the residuals can be guaranteed when the solution of the local problems is uniquely defined. In this case, the output of the set-valued maps $\mathcal{F}_q$ is a singleton, and the map is therefore a function in the usual sense, resulting in the convergence of the residuals, as stated in the following proposition.

Proposition 6: Let the solution of the local problems (33) be uniquely defined. Then under Assumptions 1 and 3,

$$\lim_{i\to\infty} \left\|X^i - X^{i+1}\right\|_F = 0. \tag{55}$$

Proof: In the proof of Proposition 2, we have shown that for any subsequence generated by the procedure, there is some index set $\mathcal{I}$ such that $(X^i, X^{i+1}, q^i)_{i\in\mathcal{I}}$ converges to some $(\bar{X}, \bar{X}^{+1}, q)$, and such that both $\bar{X}, \bar{X}^{+1} \in \mathcal{F}_q$. But because $\mathcal{F}_q$ is single-valued, it must be that $\bar{X} = \bar{X}^{+1}$. Thus

$$\lim_{i\in\mathcal{I}\to\infty} \left\|X^i - X^{i+1}\right\|_F = 0. \tag{56}$$

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

As this is true for any subsequence, 0 is the sole accumulation point of $(\|X^i - X^{i+1}\|_F^i)_{i\in\mathbb{N}}$, such that this sequence must converge to 0. □

It seems reasonable that, in the context of our target problems (5), the regularizer will most likely favor a particular solution of the local problems to be selected (which is typically the purpose of a regularizer). If that would not be the case, once the algorithm is sufficiently stable in objective values, but without observing convergence in the filters X^i, the weight of the regularization parameter can be progressively increased across iterations until a single solution is obtained. Note that a convex problem, can be turned into a strictly convex one by the addition of Tikhonov regularization, therefore ensuring a single minimizer.

If we cannot guarantee that the local problems have a unique solution, we can enforce the uniqueness by replacing the local problem by a bi-level optimization problem, as originally proposed in [18]. The procedure is essentially the same as the one described for NS-DASF, except that in the case of multiple local solutions of (33), the one minimizing the distance with the previous iterate X^{i-1} is chosen. It is shown in [19], that if the solution set of the global problem is continuous with respect to the distribution of $\boldsymbol{y}$ and the other problem parameters, then the convergence of the residuals is also guaranteed. Note that the continuity of the solution set is a much stronger assumption than the continuity of the constraint set (Assumption 3) considered here.

V. Numerical Experiments

This section provides some numerical results supporting the theoretical claims of Section IV. We consider the problem of computing a sparse Wiener filter. The target filter is the solution of

$$x^\star \triangleq \min_x \mathbb{E}\left\{\left\|x^T\boldsymbol{y}(t) - \boldsymbol{d}(t)\right\|_F^2\right\} + \lambda\|x\|_1, \qquad (57)$$

where $\boldsymbol{d}(t)$ is a single channel known target signal. Note that the regularization term can be written as $\|x^TA\|_1$ with $A = I$, such that it fits (5) (the ℓ_1-norm is computed over a row vector here). We use a lowercase x to emphasize that the filter has a single output channel (i.e. $Q = 1$). λ acts here as a meta-parameter allowing to tune the trade-off between solution accuracy and bandwidth (more nodes will eventually become inactive with a higher value of λ). The local problems associated with (57) will be of the form

$$\min_{\tilde{x}} \mathbb{E}\left\{\left\|\tilde{x}^T\widetilde{\boldsymbol{y}}(t) - \boldsymbol{d}(t)\right\|_F^2\right\} + \lambda\left\|\tilde{x}^T\widetilde{A}\right\|_1. \qquad (58)$$

As $x \mapsto \|Ax\|_1$ is not proximable, using proximal gradient descent [51] for solving the local problems would be very inefficient. We use the Chambolle-Pock [52] algorithm instead, as it was designed specifically for problems such as (58).

A. Synthetic Data

For the following experiment, we considered a fully-connected network where $M = K = 10$. Each entry of a sample of $\boldsymbol{y}(t)$ is sampled from a zero-mean unit-variance Gaussian distribution. $\boldsymbol{d}(t)$ is constructed as

$$\boldsymbol{d}(t) = a^T\boldsymbol{y}(t) + \boldsymbol{n}(t), \qquad (59)$$

where the entries in $\boldsymbol{n}(t)$ and a are also sampled from a zero-mean unit-variance Gaussian distribution.

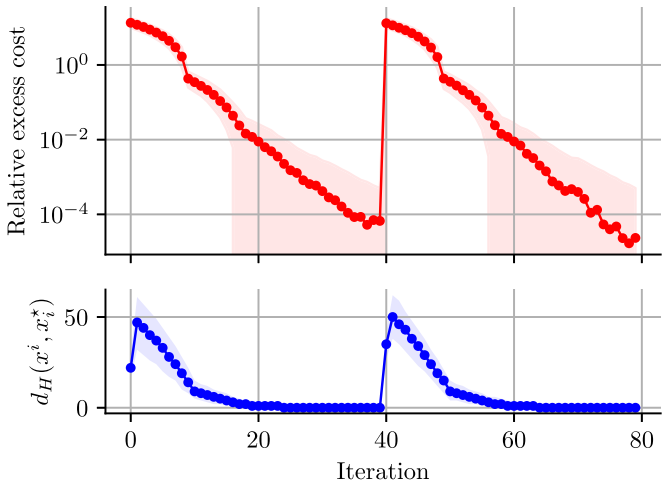


Fig. 3. Transient behavior of NS-DASF. The optimal solution changes at iteration 40. Solid curve depicts the median performance, the shaded area depicts the 5%-95% percentile region and were computed over 10000 Monte-Carlo runs.

Fig. 3 depicts the typical convergence behavior of NS-DASF applied to (57), i.e. error convergence when starting from a random filter. In order to better observe the adaptivity of the algorithm, the solution $x^\star$ was randomly changed at iteration 40. The top plot depicts the relative excess cost computed as

$$1 - \frac{L(x^i)}{L(x^\star)}, \qquad (60)$$

and the bottom plot depicts the hamming distance between the set of active nodes, that is

$$d_H(x_A, x_B) \triangleq H(|x_A| > 0, |x_B| > 0), \qquad (61)$$

where H denotes the actual hamming distance, $|\cdot|$ and $>$ are the element-wise absolute value and "greater than" operators.

NS-DASF applied to (57) appears to exhibit a linear convergence rate. A first dip in excess cost can be observed after the tenth iteration, as it is only then that each node has had the opportunity to freely update its corresponding block of x^i. The set of active channels is correctly identified after at most two full rounds of iterations (i.e. 20 iterations). Fig. 3 depicts this same behavior when changing the solution to a new random point.

Fig. 4 depicts the adaptive property of NS-DASF in a scenario where the solution changes over time for different rates of change. At each iteration, we apply a perturbation with zero-mean Gaussian entries e^i to the ground-truth solution $x^{i\star}$, with the ratio $\|e^i\|_F / \|x^{i\star}\|_F$ kept constant across iterations (this ratio reflects the amount of change in $x^\star$ across iterations, i.e.,

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

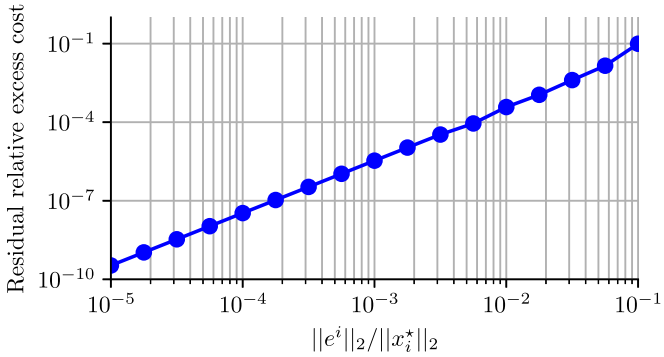


Fig. 4. Tracking performance of NS-DASF. Each data point corresponds to the mean relative excess cost at iteration 200 of 100 Monte-Carlo runs.

a higher ratio is more challenging). The residual excess cost is defined as

$$\lim_{i\to\infty} 1 - \frac{L(x^i)}{L(x^{i\star})}, \tag{62}$$

and is estimated in the simulation by computing the relative excess cost at iteration $i = 200$.

B. EEG Data

This second experiment simulates the context of a wireless encephalography sensor network (WESN) [3]. We used the data associated with subject a of the BCI Competition IV dataset [53]. We considered an eye blink artifact removal task based on Wiener filtering, similar to [3], but with the addition of an ℓ_1 regularizer to obtain a sparse solution. Our goal is to use all the channels in the WESN to estimate the eye blink artifact, such that it can be regressed out from the data. We use a channel near the eyes (channel AF3) as the reference for the artifact. The problem can again be expressed as (57), where $\boldsymbol{d}(t)$ now corresponds to the eye blink artifact in the reference channel AF3. Note that while $\boldsymbol{d}(t)$ itself is not available, the Wiener filter can still be computed by exploting the on-off characteristic of the eye blink artifact (see [3] and [54] for more details).

To illustrate the tracking abilities of the algorithm, an iteration of DASF was performed for each batch of 1000 samples (corresponding to 10 seconds of data), and the relative error power of the estimated signal

$$10\log_{10}\frac{\mathbb{E}\left\{\left\|\boldsymbol{d}(t) - x^T\boldsymbol{y}(t)\right\|^2\right\}}{\mathbb{E}\left\{\left\|\boldsymbol{d}(t)\right\|^2\right\}} \tag{63}$$

was estimated using samples averages. We also recorded the number of non-zero entries of the filters associated with each batch. λ was selected in an ad-hoc fashion, ensuring a good trade-off between reconstruction error and sparsity. In the distributed case, the data was distributed between 4 nodes, each observing 14 channels. Note that, in order to have access to a ground truth, we here define $\boldsymbol{d}(t)$ as the complete signal in channel AF3 (which includes both the eye blink artifact and some neural activity) as we can otherwise not properly compare and quantify the performance of the algorithms. However, we re-iterate that in practice, one should use the method from [3] or [54] to compute the actual Wiener filter that only extracts the blinking artifact while ignoring neural activity.

The performance of the filter over time for both NS-DASF and a centralized solver with access to all the channels of each batch are depicted in Fig. 5. We also considered the case of data-reuse described in [55], where a batch of data is re-used by DASF for 4 iterations with little additional communication cost, thereby allowing to improve the convergence and adaptivity of the filter. On average across all batches, the error associated with the centralized solver is 2.52dB lower than the distributed one, and 1.02dB lower when data-reuse is used. Although all three methods achieve similar performance and sparsity levels, the centralized solver requires 14 times more bandwidth than the distributed solution (8 times in the case of data-reuse). Fig. 6 shows a sample output for each of the three filters.

VI. Discussion

In this paper, we have described an extension of the DASF algorithm for a particular class of non-smooth spatial filtering problems. We have provided theoretical results ensuring the convergence and optimality of this NS-DASF algorithm. Using Monte-Carlo simulations, we have demonstrated the algorithm's transient and stationary behavior when applied to a sparse form of the multichannel Wiener filtering problem. In particular, this shows that NS-DASF can be used to perform channel or node selection alongside a given filtering task, allowing energy savings by omitting the transmission from nodes with 0-norm filters.

Appendix

A. Proof of Proposition 2

This proof is organized as follows.

1) We prove that the continuity of the global constraint set implies the continuity of the local constraint set.
2) We prove that the above implies that the optimal function value of the local problem is an upper semicontinuous function of X^i.
3) Based on the above two results, we show that if we select a subsequence of $(X^i)_{i\in\mathbb{N}}$ over which the updating node q remains the same (i.e. $q^i = q$ for any i in the subsequence), the accumulation points of such a sequence are fixed points of F_q.
4) We finally show that this must also be true for every node q.

We first establish the fact that the continuity of $\mathcal{X}$ also ensures the continuity of the map $\mathcal{R}_q : X \mapsto \mathcal{X} \cap \mathcal{S}_q(X)$, describing the local constraint set, as stated in the following lemma.

Lemma 2 (Continuity of Local Feasible Sets): Let $(\boldsymbol{y}, \mathcal{D}) \mapsto \mathcal{X}(\boldsymbol{y}, \mathcal{D})$ be a continuous map. Then the map $\mathcal{R}_q : X \mapsto \mathcal{X} \cap \mathcal{S}_q(X)$ is continuous.

Proof: See supplementary materials. □

As the sublevel sets of L are compact (Assumption 1), any sequence $(X^i)_{i\in\mathbb{N}}$ satisfying the update rule (34) has at least one accumulation point $\bar{X}$ [56, Theorem 3.6]. In other words, there

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

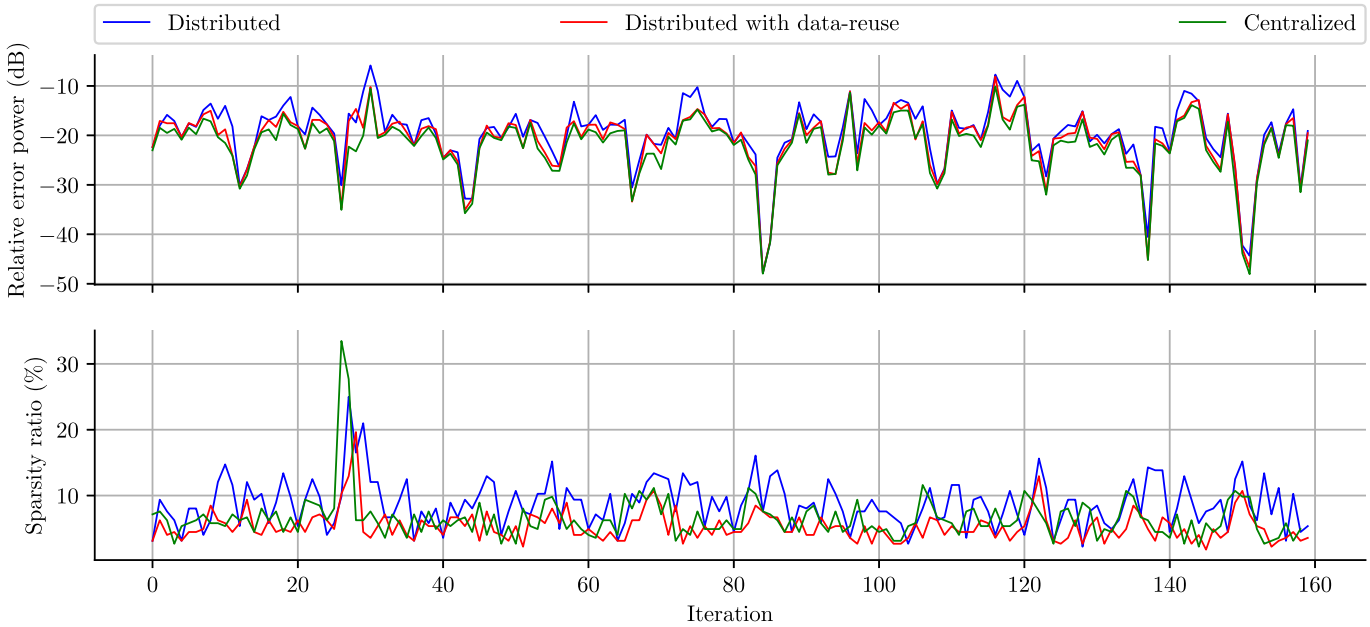


Fig. 5. Relative estimation error power for each batch of 1000 samples. The sparsity ratio corresponds to the ratio between the zero entries and total number of entries of the filter.

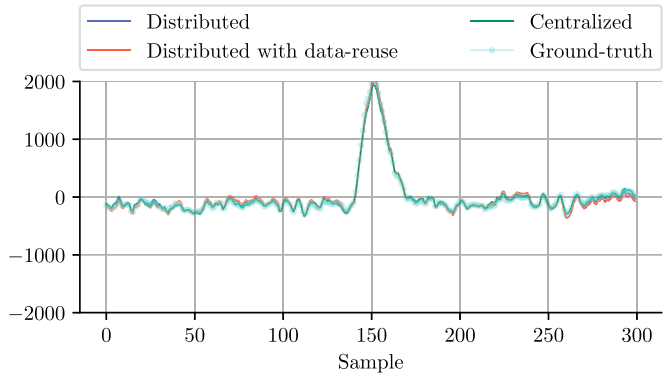


Fig. 6. Example outputs of the filters for the estimation of the AF3 channel.

is some index set $\mathcal{I} \subseteq \mathbb{N}$ such that $(X^i)_{i\in\mathcal{I}}$ converges to $\bar{X}$. Because of (36), we can in particular select $\mathcal{I}$ such that $(q^i)_{i\in\mathcal{I}}$ is a constant subsequence with value q. Our first objective is to show that the accumulation point $\bar{X}$ associated with such a sequence is a fixed point of the map F_q. Before doing so, we first need to show that the value function

$$m_q(X) \triangleq \min_{U\in\mathcal{R}_q(X)} L(U) \tag{64}$$

is upper semicontinuous (in the classical sense, not the set-analysis sense), as stated in the following lemma.

Lemma 3 (Semicontinuity of the Value Function): Under Assumptions 1 and 3, the value function $m_q : \mathcal{X} \to \mathbb{R}$ is upper semicontinuous.

Proof: See supplementary materials. □

We can now relate the accumulation points of $(X^i)_{i\in\mathbb{N}}$ to the fixed points of the maps F_q.

Lemma 4: Let $(X^i)_{i\in\mathbb{N}}$ and $(q^i)_{i\in\mathbb{N}}$ be sequences related by (34) and let $\mathcal{I} \subseteq \mathbb{N}$ be such that the subsequence $(X^i, q^i)_{i\in\mathcal{I}}$ converges to $(\bar{X}, q)$. Then under Assumptions 1 and 3, $\bar{X}$ is a fixed point of F_q.

Proof: Since q^i is part of the finite set $\mathcal{K}$, its convergence implies that it eventually becomes constant. We can therefore proceed as if it was a constant sequence. By (34) and (64), we have $L(X^{i+1}) = m_q(X^i)$. As L is continuous with compact sublevel sets, it attains its minimum value in $\mathcal{X}$ (extreme value theorem) [38], and the sequence $L(X^{i+1})$ is therefore bounded below by $L(X^\star)$ and, as a consequence of its monotonic decrease (Proposition 1), it must therefore converge to some value $\bar{L}$ [56]. From the continuity of L, it must be that $\bar{L} = L(\bar{X})$. Since $L(X^{i+1})$ converges to $L(\bar{X})$, and since $m_q(X^i) = L(X^{i+1})$, it must hold that $m_q(X^i)$ also converges to $L(\bar{X})$. From Lemma 3, it must be that

$$L(\bar{X}) \leq m_q(\bar{X}). \tag{65}$$

Since $\bar{X}$ is the accumulation point of a sequence $(X^i)_i$ living in the closed[15] set $\mathcal{X}$, it must be that $\bar{X} \in \mathcal{X}$ (by the definition of closedness), and since $\bar{X} \in \mathcal{S}_q(\bar{X})$ by definition (see (32)), we conclude that $\bar{X} \in \mathcal{R}_q(\bar{X}) = \mathcal{X} \cap \mathcal{S}_q(\bar{X})$. Hence, we have by definition of the value function,

$$m_q(\bar{X}) \leq L(\bar{X}). \tag{66}$$

Combining this fact with (65), we conclude that

$$m_q(\bar{X}) = L(\bar{X}), \tag{67}$$

and hence $\bar{X} \in F_q(\bar{X}) = \operatorname{argmin}_{U\in\mathcal{R}_q(\bar{X})} L(U)$. □

Based on this sub-result, we can show that accumulation points of the full sequence $(X^i)_{i\in\mathbb{N}}$ are fixed points of the map F_q for any q. Let $(X^i)_{i\in\mathbb{N}}$ be a sequence satisfying (34), by Assumption 1, there must be an index set $\mathcal{I} \subseteq \mathbb{N}$ such that

[15]The closedness of $\mathcal{X}$ follows from the continuity of the η_j.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

the subsequence $(X^i, X^{i+1}, q^i, q^{i+1})_{i\in\mathcal{I}}$ converges to some $(\bar{X}, \bar{X}^{+1}, q, q')$ (we also used the fact that the cartesian product of compact sets is compact [38]). We first prove that $\bar{X}^{+1} \in F_q(\bar{X})$. From Assumption 3 and Lemma 2, we have[16] that $\bar{X}^{+1} \in \mathcal{R}_q(\bar{X})$. From the continuity of L (Assumption 1), we know that

$$L(\bar{X}^{+1}) = L(\bar{X}) = \lim_{i\to\infty} L(X^i), \tag{68}$$

and by Lemma 4 (see (67))

$$L(\bar{X}^{+1}) = L(\bar{X}) = m_q(\bar{X}),$$

we find that $\bar{X}^{+1} \in F_q(\bar{X})$ (as $\bar{X}^{+1}$ is in the local feasible set and minimizes the local objective). We will use this result to show that $\bar{X}^{+1}$ is also a fixed point of F_q (we already know that $\bar{X}$ is a fixed point from Lemma 3).

From Assumption 2 and since both $\bar{X}^{+1}, \bar{X} \in F_q(\bar{X})$, we have $\mathcal{S}_k(\bar{X}) = \mathcal{S}_k(\bar{X}^{+1})$ and thus

$$F_k(\bar{X}) = F_k(\bar{X}^{+1}) \tag{69}$$

for any k, because, by Assumption 2, they share the same column space. As any pair of successive accumulation points $(\bar{X}, \bar{X}^{+1})$ share the same column space, we can inductively deduce that any sequence of accumulation points $(\bar{X}, \ldots, \bar{X}^{+Q})$ (for the right selection of $\mathcal{I}$) share the same column space. In addition, by (68), we also have

$$L(\bar{X}) = L(\bar{X}^{+1}) = \cdots = L(\bar{X}^{+Q}). \tag{70}$$

Because we assumed that every node is selected infinitely many times, i.e. Acc $(q^i)_{i\in\mathbb{N}} = \mathcal{K}$, for any i, we can select $Q < \infty$ such that the sequence of accumulation points associated with $(X^i, \ldots, X^{i+Q})_{i\in\mathcal{I}}$ is such that all nodes are selected at least once between iterations (see supplementary materials for details). For any q we can therefore find some $\bar{X}^{+a}$, with $a \leq Q$, such that $\mathcal{S}_q(\bar{X}) = \mathcal{S}_q(\bar{X}^{+a})$ and hence because $L(\bar{X}) = L(\bar{X}^{+a})$,

$$\bar{X} \in F_q(\bar{X}), \tag{71}$$

where q was selected arbitrarily in $\mathcal{K}$. We therefore find that $\bar{X}$ is a fixed point for any q and therefore a fixed point of the full algorithm.

B. Checking the Qualification for Linear Constraints

We can check that for linear equality constraints of the form

$$X_k^T D_k = R_k \quad \forall k \in \mathcal{K} \tag{72}$$

where $R_k \in \mathbb{R}^{M_k \times L_k}$ and $D_k \in \mathbb{R}^{L_k \times Q}$, the qualification of Proposition 5 holds globally.

There is a total of QL_k constraints at each node k, each constraint constraining the dot product of a column of X_k with a column of D_k to be equal to the corresponding entry of R_k. The compressed gradients in the qualification of Proposition 5 associated with the constraint for entry (m, l) of R_k in (72) can be expressed as the $Q \times Q$ matrix

$$X_k^T \left[0_{M_k} \ \cdots \ d_{l,k} \ \cdots \ 0_{M_k}\right] \tag{73}$$

[16]This is the definition of set upper semicontinuity.

where 0_{M_k} denotes the 0 vector of dimension M_k. $d_{l,k}$ denotes the l-th column of D_k, and where $d_{l,k}$ is in the m-th column. There is such a compressed gradient for every combination of l, m and k. Checking the independence of the compressed gradients therefore amounts to ensuring that the matrices $X_k^T d_{l,k}$ are independent for any pairs of (m, k), when the nodes referred by k are in a common branch. From the constraints (72), this is equivalent to ensuring that the matrix

$$\left[R_{l_1} \ \cdots \ R_{l_N}\right], \tag{74}$$

where $\{l_1, \ldots, l_N\} = \mathcal{B}_{nk}$, has linearly independent columns for any choice of n, k. If the network topology is not known in advance, it is sufficient to check that the concatenation of all the R_k's has independent columns. Note that (74) has $\sum_{j\in\mathcal{B}_{nk}} L_j$ columns, where L_j is the number of columns of R_j, such that a necessary condition for (74) to have linearly independent columns is that $\sum_{j\in\mathcal{B}_{nk}} L_j \leq Q$. Since the total number of constraints associated with (74) is $\sum_{j\in\mathcal{B}_{nk}} QL_j$, this indeed confirms that the total number of constraints can not be larger than Q^2, as already explained in Subsection IV-F.

In fully-connected networks, $\mathcal{B}_{nk} = \{k\}$, such that the constraint qualifications can be checked on a per-node basis. For example, constraints of the form

$$X_k^T \mathbb{E}\left\{\boldsymbol{y}_k(t)\boldsymbol{y}_k(t)^T\right\} X_k = I_Q \quad \forall k \in \mathcal{K} \tag{75}$$

can be shown to always satisfy the qualification, for any X_k when applied in a fully-connected network, since the per-node compressed gradients can be shown to be independent (This is explained in [19, Example 1] and is not repeated here.

ACKNOWLEDGMENT

Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or ERC. Neither the European Union nor the ERC can be held responsible for them.

REFERENCES

[1] C. Hovine and A. Bertrand, "A distributed adaptive algorithm for non-smooth spatial filtering problems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–5.

[2] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[3] A. Bertrand, "Distributed signal processing for wireless EEG sensor networks," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 6, pp. 923–935, Nov. 2015.

[4] A. Bertrand, J. Callebaut, and M. Moonen, "Adaptive distributed noise reduction for speech enhancement in wireless acoustic sensor networks," in *Proc. Int. Workshop Acoustic Echo Noise Control (IWAENC)*, 2010.

[5] A. M. Narayanan, R. Zink, and A. Bertrand, "EEG miniaturization limits for stimulus decoding with EEG sensor networks," *J. Neural Eng.*, vol. 18, no. 5, 2021, Art. no. 056042.

[6] A. M. Narayanan, P. Patrinos, and A. Bertrand, "Optimal versus approximate channel selection methods for EEG decoding with application to topology-constrained neuro-sensor networks," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 92–102, Nov. 2020.

[7] N. A. Goodman and D. Bruyere, "Optimum and decentralized detection for multistatic airborne radar," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 2, pp. 806–813, Apr. 2007.

[8] S. Haykin and K. R. Liu, *Handbook on Array Processing and Sensor Networks*, vol. 63. Hoboken, NJ, USA: Wiley, 2010.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[10] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends® Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[11] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1185–1197, Mar. 2014.

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

[13] A. H. Sayed et al., "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, nos. 4–5, pp. 311–801, 2014.

[14] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 977–992, Feb. 2019.

[15] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Model-distributed solution of regularized least-squares problem over sensor networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 3821–3825.

[16] C. Gratton, N. K. Venkategowda, R. Arablouei, and S. Werner, "Distributed ridge regression with feature partitioning," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 1423–1427.

[17] Y. Liu et al., "FedBCD: A communication-efficient collaborative learning framework for distributed features," *IEEE Trans. Signal Process.*, vol. 70, pp. 4277–4290, 2022.

[18] C. A. Musluoglu and A. Bertrand, "A unified algorithmic framework for distributed adaptive signal and feature fusion problems—Part I: Algorithm derivation," *IEEE Trans. Signal Process.*, vol. 71, pp. 1863–1878, 2023.

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

[20] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Piscataway, NJ, USA: IEEE Press, 2007, pp. 1–8.

[21] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 24, no. 6, 1933, Art. no. 417.

[22] G. H. Golub and C. F. Van Loan, "Matrix computations," 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2013.

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

[24] J. D. Carroll, "Generalization of canonical correlation analysis to three or more sets of variables," in *Proc. 76th Annu. Conv. Amer. Psychol. Assoc.*, vol. 3. Washington, DC, USA, 1968, pp. 227–228.

[25] C. Hovine and A. Bertrand, "MAXVAR-based distributed correlation estimation in a wireless sensor network," *IEEE Trans. Signal Process.*, vol. 70, pp. 5533–5548, 2022.

[26] J. R. Kettenring, "Canonical analysis of several sets of variables," *Biometrika*, vol. 58, no. 3, pp. 433–451, 1971.

[27] C. I. Kanatsoulis, X. Fu, N. D. Sidiropoulos, and M. Hong, "Structured SUMCOR multiview canonical correlation analysis for large-scale data," *IEEE Trans. Signal Process.*, vol. 67, no. 2, pp. 306–319, Jan. 2019.

[28] M. Sørensen, C. I. Kanatsoulis, and N. D. Sidiropoulos, "Generalized canonical correlation analysis: A subspace intersection approach," *IEEE Trans. Signal Process.*, vol. 69, pp. 2452–2467, 2021.

[29] C. Hovine and A. Bertrand, "Distributed maxvar: Identifying common signal components across the nodes of a sensor network," in *Proc. 29th Eur. Signal Process. Conf. (EUSIPCO)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 2159–2163.

[30] S. V. Vaseghi, "Wiener filters," in *Advanced Signal Processing and Digital Noise Reduction*, Hoboken, NJ, USA: John Wiley & Sons, 1996, pp. 140–163.

[31] T. Strypsteen and A. Bertrand, "Bandwidth-efficient distributed neural network architectures with application to neuro-sensor networks," *IEEE J. Biomed. Health Inform.*, vol. 27, no. 2, pp. 933–943, Feb. 2023.

[32] S. Kim, R. Pasupathy, and S. G. Henderson, "A guide to sample average approximation," in *Handbook of Simulation Optimization*, New York, NY, USA: Springer, 2015, pp. 207–243.

[33] L.-H. Zhang, L.-Z. Liao, and L.-M. Sun, "Towards the global solution of the maximal correlation problem," *J. Global Optim.*, vol. 49, no. 1, pp. 91–107, 2011.

[34] R. J. Tibshirani, "The lasso problem and uniqueness," *Electron. J. Statist.*, vol. 7, pp. 1456–1490, 2013.

[35] B. Kibria and S. Banik, "Some ridge regression estimators and their performances," *J. Modern App. Stat. Methods,* vol. 15, pp. 206–238, 2016.

[36] D. A. Lorenz and E. Resmerita, "Flexible sparse regularization," *Inverse Problems*, vol. 33, no. 1, 2016, Art. no. 014002.

[37] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, vol. 317. Springer-Verlag, 2009.

[38] D. Charalambos and B. Aliprantis, *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Berlin, Germany: Springer-Verlag, 2013.

[39] J.-P. Aubin and H. Frankowska, *Set-Valued Analysis*. Springer-Verlag, 2009.

[40] V. Rakocevic, "On continuity of the Moore–Penrose and Drazin inverses," *Matematichki Vesnik*, vol. 49, no. 3, pp. 163–172, 1997.

[41] G. Forchini, "A note on the continuity of projection matrices with application to the asymptotic distribution of quadratic forms," *Appl. Math.*, vol. 1, no. 32, pp. 51–55, 2005.

[42] J. Li, A. M.-C. So, and W.-K. Ma, "Understanding notions of stationarity in nonsmooth optimization: A guided tour of various constructions of subdifferential for nonsmooth functions," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 18–31, Sep. 2020.

[43] J. O. Royset and R. J. Wets, *An Optimization Primer*. Springer-Verlag, 2021.

[44] W. Karush, "Minima of functions of several variables with inequalities as side constraints," M.Sc. Dissertation. Dept. Math., Univ. Chicago, Chicago, IL, USA, 1939.

[45] H. Kuhn and A. Tucker, "Nonlinear programming," in *Proc. 2nd Berkeley Symp. Math. Statist. Probabilistics*, 1951, pp. 481–492.

[46] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[47] A. Ruszczynski, *Nonlinear Optimization*. Princeton, NJ, USA: Princeton Univ. Press, 2011.

[48] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 334–334, 1997.

[49] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Hoboken, NJ, USA: Wiley, 2006.

[50] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Philadelphia, PA, USA: SIAM, 1990.

[51] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM, 2017.

[52] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *J. Math. Imag. Vis.*, vol. 40, no. 1, pp. 120–145, 2011.

[53] M. Tangermann et al., "Review of the BCI competition IV," *Frontiers Neurosci.*, vol. 6, 2012, Art. no. 55.

[54] B. Somers, T. Francart, and A. Bertrand, "A generic EEG artifact removal algorithm based on the multi-channel wiener filter," *J. Neural Eng.*, vol. 15, no. 3, 2018, Art. no. 036007.

[55] C. A. Musluoglu, M. Moonen, and A. Bertrand, "Improved tracking for distributed signal fusion optimization in a fully-connected wireless sensor network," in *Proc. 30th Eur. Signal Process. Conf. (EUSIPCO)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 1836–1840.

[56] W. Rudin et al., *Principles of Mathematical Analysis*, vol. 3. New York, NY, USA: McGraw-Hill, 1976.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on January 28,2025 at 12:22:41 UTC from IEEE Xplore. Restrictions apply.