IMPROVED TRACKING FOR THE DISTRIBUTED SIGNAL FUSION OPTIMIZATION ALGORITHM IN A FULLY-CONNECTED WIRELESS SENSOR NETWORK

Cem Ates Musluoglu, Marc Moonen and Alexander Bertrand

KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium
{cemates.musluoglu, marc.moonen, alexander.bertrand}@esat.kuleuven.be

ABSTRACT

The distributed signal fusion optimization (DSFO) algorithm is a generic algorithm that can be used to solve various spatial signal and feature fusion optimization problems in a distributed setting such as a wireless sensor network. Examples include principal component analysis, adaptive beamforming, and source separation problems. While the DSFO algorithm adaptively learns the relevant second order statistics from the collected sensor data, accuracy problems can arise if the spatial covariance structure of the signals is rapidly changing. In this paper, we propose a method to improve the tracking or convergence speed of the DSFO algorithm in a fully-connected sensor network with a broadcast communication protocol. While the improved tracking increases communication cost, we demonstrate that this tradeoff is efficient in the sense that an L-fold increase in bandwidth results in an R times faster convergence with R $>>$ L.

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

1. INTRODUCTION

A wireless sensor network (WSN) consists of a collection of sensor nodes which are spread out over an area, where each node is equipped with one or more sensors, a processing unit, and a wireless radio. Instead of transferring all the raw sensor data to a central fusion center, a distributed processing of the sensor data is often preferred in terms of bandwidth efficiency. The distributed signal fusion optimization (DSFO) algorithm described in [1] allows to solve a variety of spatial signal or feature fusion optimization problems in such a distributed context. In contrast to various other distributed algorithms [2–6], signal fusion optimization (SFO) problems of interest do not have a separable objective nor a shared optimization variable across nodes. Instead, the objective and the constraint functions consist of spatial filtering operations given by $X^T\mathbf{y}(t)$, where X is the optimization variable and $\mathbf{y}(t)$ is a multi-channel signal containing all the sensor signals of the WSN. In the DSFO setting, each node k in the WSN measures its own signal $\mathbf{y}_k$ and has its own local variable X_k such that $X^T\mathbf{y}(t) = \sum_k X_k^T\mathbf{y}_k(t)$. This type of partitioning has also been studied under the name of feature partitioning or distributed features in previous works, such as [7–13].

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 802895). The authors also acknowledge the financial support of the FWO (Research Foundation Flanders) for project G.0A49.18N, and the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

Table 1. Examples of SFO problems

$\mathbb{P}$	$\min \varphi$	Constraints
LCMV [15, 16]	$\min_{\mathbf{x}} \mathbb{E}[\lvert\mathbf{x}^T\mathbf{y}(t)\rvert^2]$	$\mathbf{x}^T B = \mathbf{c}^T$
PCA [17]	$\min_X -\mathbb{E}[\lvert\lvert X^T\mathbf{y}(t)\rvert\rvert^2]$	$X^T X = I_Q$
CCA [18]	$\min_{(X,W)}$ $-\mathbb{E}[\text{tr}(X^T\mathbf{y}(t)\mathbf{v}^T(t)W)]$	$\mathbb{E}[X^T\mathbf{y}(t)\mathbf{y}^T(t)X] = I_Q$ $\mathbb{E}[W^T\mathbf{v}(t)\mathbf{v}^T(t)W] = I_Q$

In practice, the iterations of the DSFO algorithm are spread over time, i.e., over different batches of signal observations (samples), thereby exploiting the stationarity of the underlying signals. This results in an adaptive algorithm that is able to track the changes in the signal statistics. However, this is a valid argument as long as the changes in the statistical parameters of $\mathbf{y}$ (e.g., their spatial covariance matrix) are slower than the algorithm convergence rate. In this paper, we propose a modified scheme for the DSFO algorithm in a fully-connected WSN, which improves the tracking speed by re-using and re-transmitting previously measured observations over the network, instead of using new data batches at every iteration. Although this leads to additional communication costs, we show that the resulting tradeoff is efficient.

It is noted that, while we focus on the general DSFO algorithm from [1] in this paper, our proposed modified scheme is also applicable in other related algorithms such as the distributed adaptive node-specific signal estimation (DANSE) algorithm for signal enhancement [14].

2. REVIEW OF THE DSFO FRAMEWORK

Consider a WSN with K nodes given in the set $\mathcal{K} = \{1, \ldots, K\}$, where each node k measures its own M_k-channel signal $\mathbf{y}_k$. Let:

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

be the stacked vector containing all the sensor signals, where $\mathbf{y}(t) \in \mathbb{R}^M$ with $M = \sum_k M_k$. The signal $\mathbf{y}$ is assumed to be short-term stationary and ergodic. The goal is now to learn a network-wide signal fusion or spatial filter $X \in \mathbb{R}^{M\times Q}$ such that the $Q-$channel output signal $X^T\mathbf{y}(t)$ is optimal in some sense. To this end, each node is responsible for optimizing its corresponding part of X, based on the following per-node

partitioning:

$$X = [X_1^T, \ldots, X_K^T]^T \in \mathbb{R}^{M \times Q}. \tag{2}$$

The generic SFO problem can be written in the form:

$$\begin{aligned} \mathbb{P}: \underset{X}{\text{minimize}} \quad & \varphi\left(X^T\mathbf{y}(t), X^TB, X^T\Gamma X\right) \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB, X^T\Gamma X\right) \leq 0 \ \ \forall j \in \mathcal{I}, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB, X^T\Gamma X\right) = 0 \ \ \forall j \in \mathcal{E}, \end{aligned} \tag{3}$$

where φ is the objective and η_j's are the constraint functions respectively. The sets $\mathcal{I}$ and $\mathcal{E}$ contain the indices j representing inequality and equality constraints respectively. A few practical examples are given in Table 1 (we refer to [1] for a more extensive list). The notation in (3) emphasizes that the optimization variable X and the signal $\mathbf{y}(t)$ exclusively appear in the form $X^T\mathbf{y}(t)$ in the considered SFO problems. The matrix X can also appear in linear forms X^TB (e.g., linearly constrained minimum variance beamforming (LCMV) in Table 1) and quadratic forms $X^T\Gamma X$ (e.g., principal component analysis (PCA) in Table 1, with $\Gamma = I_M$), where B and Γ are a-priori known fixed matrices, and Γ is block-diagonal. Although not represented for conciseness, the considered SFO problems can also have more than one variable, as in the canonical correlation analysis (CCA) example in Table 1. We note that the modifications that we propose in this paper only affect the fusion terms $X^T\mathbf{y}(t)$, therefore in the remaining parts of this paper, we will omit X^TB and $X^T\Gamma X$ from (3) for conciseness.

In practice, $\mathbf{y}$ is a stochastic signal, therefore φ and η_j's are functions of the statistical parameters of $\mathbf{y}$ and hence contain expectation operators $\mathbb{E}[\cdot]$, as can be seen in the examples given in Table 1, which we have omitted in (3). The statistical properties of $\mathbf{y}$ can be estimated using a window, or batch, of N time samples $\{\mathbf{y}(t)\}_{t=0}^{N-1}$, using the short-term stationarity and ergodicity of $\mathbf{y}$, where N is sufficiently large. However, in a distributed setting, communicating $\mathbf{y}_k$'s between nodes to construct $\mathbf{y}$ is costly. The DSFO algorithm solves this issue and guarantees convergence to an optimal solution X^* of the problem by requiring each node k to communicate only a linearly compressed version of $\mathbf{y}_k$, instead of $\mathbf{y}_k$ itself. While the DSFO algorithm can be applied in any connected network topology [1], we specifically focus here on the case where the network is fully-connected, i.e., a signal that is broadcast by any node can be received by all other nodes in the network.

Consider a fully-connected WSN, where each node k collects a batch of N samples of the M_k-channel signal $\mathbf{y}_k$, denoted as $\{\mathbf{y}_k(t)\}_{t=iN}^{(i+1)N-1}$ where i is the iteration index of the DSFO algorithm. Note that the N sample batch changes with each iteration. Additionally, each node k has an estimation X_k^i of their local variable X_k, which is part of the network-wide X as defined in the partitioning (2). While X_k^i is part of the optimization variable X, it is also used by the DSFO algorithm to compress the M_k-channel signal into a $Q-$channel signal, thereby reducing the communication cost (note that many practical examples have $Q = 1$). The compressed signals are then given by:

$$\widehat{\mathbf{y}}_k^i(t) \triangleq X_k^{iT}\mathbf{y}_k(t). \tag{4}$$

At each iteration, one node (say node q) acts as the "updating node", where the updating node changes in each iteration. Each $k \in \mathcal{K}\backslash\{q\}$ sends $\{\widehat{\mathbf{y}}_k^i(t) = \{X_k^{iT}\mathbf{y}_k(t)\}_{t=iN}^{(i+1)N-1}$ to node q at iteration i. We define the locally available signal at node q after receiving the compressed signals $\widehat{\mathbf{y}}_k^i$ from the other nodes as:

$$\widetilde{\mathbf{y}}_q^i(t) = [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_1^{iT}(t), \ldots, \widehat{\mathbf{y}}_{q-1}^{iT}(t), \widehat{\mathbf{y}}_{q+1}^{iT}(t), \ldots, \widehat{\mathbf{y}}_K^{iT}(t)]^T, \tag{5}$$

which corresponds to stacking node q's own signal with the compressed signals it receives. The mechanism of the DSFO algorithm is to solve a compressed version of the global problem (3) at the updating node by using the locally available signal $\widetilde{\mathbf{y}}_q^i$. We define the local variable $\widetilde{X}_q$ at node q as:

$$\widetilde{X}_q = [X_q^T, G_{1,q}^T, \ldots, G_{q-1,q}^T, G_{q+1,q}^T, \ldots, G_{K,q}^T]^T, \tag{6}$$

where X_q is $M_q \times Q$ and every $G_{k,q}$ is $Q \times Q$. The purpose of $\widetilde{X}_q$ is to be the local fusion matrix for $\widetilde{\mathbf{y}}_q^i$, analogous to X being the network-wide fusion matrix for $\mathbf{y}$. We write:

$$\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t) = X_q^T\mathbf{y}_q(t) + \sum_{k\in\mathcal{K}\backslash\{q\}} G_{k,q}^T\widehat{\mathbf{y}}_k^i(t) \tag{7}$$

$$= X_q^T\mathbf{y}_q(t) + \sum_{k\in\mathcal{K}\backslash\{q\}} (X_k^iG_{k,q})^T\mathbf{y}_k(t), \tag{8}$$

such that the global variable X is parameterized as:

$$\begin{aligned} X = [&(X_1^iG_{1,q})^T, \ldots, (X_{q-1}^iG_{q-1,q})^T, \\ &X_q^T, (X_{q+1}^iG_{q+1,q})^T, \ldots, (X_K^iG_{K,q})^T]^T. \end{aligned} \tag{9}$$

We define the parameterized local problem at node q as:

$$\begin{aligned} \widetilde{\mathbb{P}}_q^i: \underset{\widetilde{X}_q}{\text{minimize}} \quad & \varphi\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\right) \\ \text{subject to} \quad & \eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\right) \leq 0 \ \ \forall j \in \mathcal{I}, \\ & \eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t)\right) = 0 \ \ \forall j \in \mathcal{E}. \end{aligned} \tag{10}$$

Note the similarity between (10) and (3) (omitting X^TB and $X^T\Gamma X$), yet the problems are not equal, as the signals are not the same, namely $\widetilde{\mathbf{y}}_q^i$ and $\mathbf{y}$ respectively. We define the solution of the local problem (10) at node q and iteration i as:

$$\widetilde{X}_q^{i+1} \triangleq \text{argmin}\ \widetilde{\mathbb{P}}_q^i, \tag{11}$$

where $\widetilde{X}_q^{i+1}$ is partitioned into X_q^{i+1} and $G_{k,q}^{i+1}$'s, $k \neq q$, as in (6). If the solution of (11) is not unique, node q chooses the $\widetilde{X}_q$ for which the distance between its consecutive local estimators $||X_q^{i+1} - X_q^i||_F$ is minimized. Based on the parameterization in (9), the local variables X_k are updated as:

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q \\ X_k^iG_{k,q}^{i+1} & \text{if } k \neq q \end{cases}, \tag{12}$$

to have an estimation X^{i+1} of the global variable X at the end of iteration i, where X^{i+1} is obtained by stacking X_k^{i+1}'s, as defined in (2). Since each node k has only access to its own local variable X_k, the updating node q needs to communicate the newly computed $G_{k,q}^{i+1}$'s to nodes $k \neq q$ so that they can update their variable X_k according to (12). The DSFO algorithm then chooses another node to be the updating one (for the next batch of N samples) and the process is repeated until convergence. From (7)-(9), we also see that the network-wide

Algorithm 1: DSFO in a Fully-Connected Network

X^0 initialized randomly, $i \leftarrow 0$.
repeat
Choose the updating node as $q \leftarrow (i \mod K) + 1$.
1) Every node k collects $\{\mathbf{y}_k(t)\}_{t=iN}^{(i+1)N-1}$, compresses it to obtain $\{\widehat{\mathbf{y}}_k^i(t)\}_{t=iN}^{(i+1)N-1}$ using (4) and transmits it to node q.
at *Node* q **do**
2a) $\widetilde{X}_q^{i+1} \leftarrow \text{argmin } \widetilde{\mathbb{P}}_q^i$.
If the solution of (10) is not unique, select the solution which minimizes $||X_q^{i+1} - X_q^i||_F$.
2b) Partition $\widetilde{X}_q^{i+1}$ as in (6).
2c) Transmit $G_{k,q}^{i+1}$ to node k for every $k \neq q$.
end
3) Every node updates X_k^{i+1} according to (12).
$i \leftarrow i + 1$

output $X^T\mathbf{y}(t)$ for the batch of N samples used at iteration i can be computed locally at node q as $X^{iT}\mathbf{y}(t) = \widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i(t)$. Algorithm 1 summarizes the steps of the DSFO algorithm as discussed in this section. Sufficient conditions for the convergence and optimality of this algorithm can be found in [1].

3. RE-USING DATA FOR IMPROVED TRACKING

In the DSFO algorithm described in Section 2, each batch of N samples of $\mathbf{y}_k$ is used in only one iteration i, i.e., the batch $\{\mathbf{y}_k(t)\}_{t=iN}^{(i+1)N-1}$ at iteration i. This might lead to a slow convergence/tracking speed, in particular for large networks (i.e., large K). To see this, note that it takes at least K iterations before each X_k has been updated once, which happens only after sample time $t = KN$ (i.e., K batches of N samples). In order to improve the tracking performance, one could perform $R > 1$ iterations per $N-$sample batch. While this would require extra data transmissions per batch, we will show that a specific re-transmission scheme which exploits the broadcast nature of the transmissions can actually make this tracking-vs-bandwidth tradeoff very efficient.

Let us define $\mathcal{T}_m \triangleq \{(m-1)N, \ldots, mN-1\}$, where m is a strictly positive integer, and let R be the number of iterations over which the signal batch $\{\mathbf{y}(t)\}_{t\in\mathcal{T}_m}$ will be used. This means that the $m-$th batch $\{\mathbf{y}(t)\}_{t\in\mathcal{T}_m}$ will be used over the iterations $\mathcal{I}_m \triangleq \{(m-1)R, \ldots, mR-1\}$ of the DSFO algorithm, hence $m = \lceil (i+1)/R \rceil$. In Algorithm 1, we had $R = 1$ and therefore $m = i + 1$ at every iteration, i.e., the batch index m and iteration i always update at the same pace. If $R > 1$, each new batch of N samples will lead to multiple DSFO iterations, leading to faster convergence/tracking properties, i.e., more nodes can update per batch.

For $R > 1$, a straightforward approach would be to apply the same steps as in Algorithm 1 while re-using the same batch R times. However, this would be inefficient communication-wise, as each node would have to re-compress and re-transmit the same batch of samples R times. Instead, we propose that for each *new* batch $\{\mathbf{y}_k(t)\}_{t\in\mathcal{T}_m}$, i.e., for each increment of m, all nodes k first broadcast the compressed data batch $\{X_k^{iT}\mathbf{y}_k(t)\}_{t\in\mathcal{T}_m}$ to the entire network of nodes, where i is the index corresponding to the first iteration where this new batch of samples is used. Assuming a broadcast communication protocol, this requires only one transmission per node, which is the same as in a single iteration of Algorithm 1. Every node in the network then has access to:

$$\{\widehat{\mathbf{y}}_k^i(t)\}_{t\in\mathcal{T}_m} = \{X_k^{iT}\mathbf{y}_k(t)\}_{t\in\mathcal{T}_m},\ \forall k. \tag{13}$$

After this initial broadcast, the first updating node (say node q) solves Problem (10), obtaining $\widetilde{X}_q^{i+1}$. Partitioning $\widetilde{X}_q^{i+1}$ as in (6) gives a new X_q^{i+1} and matrices $G_{k,q}^{i+1}$, to be used in (12), i.e., $X_k^{i+1} = X_k^i G_{k,q}^{i+1}$, $\forall k \in \mathcal{K}\backslash\{q\}$. For the other nodes to be aware of these changes, node q broadcasts its compressed signal samples obtained using its new estimation X_q^{i+1}, as well as the matrices $G_{k,q}^{i+1}$, i.e., node q broadcasts:

$$\{\widehat{\mathbf{y}}_q^{i+1}(t) = X_q^{(i+1)T}\mathbf{y}_q(t)\}_{t\in\mathcal{T}_m} \text{ and } \{G_{k,q}^{i+1}\}_{k\neq q}, \tag{14}$$

so that the next updating node has access to the batch of samples of node q, this time compressed by the new X_q^{i+1} and can update the compressed signals of the other nodes (received in the previous iteration i) using:

$$\forall k \neq q : \{X_k^{(i+1)T}\mathbf{y}_k(t)\}_{t\in\mathcal{T}_m} = \{G_{k,q}^{(i+1)T}X_k^{iT}\mathbf{y}_k(t)\}_{t\in\mathcal{T}_m}. \tag{15}$$

In the remaining $R-1$ iterations in $\mathcal{I}_m$, the only transmissions done in the network are the broadcasts (14) performed by the updating node (which changes at each iteration). Each node therefore needs to re-compress and re-broadcast the same batch of samples only when it becomes an updating node. The modifications are presented in Algorithm 2.

As $X_k^{iT}\mathbf{y}_k$ is a $Q-$channel signal, the initial broadcast has a communication cost in $\mathcal{O}(NQ)$ per node, where $\mathcal{O}$ represents the Landau notation, and $\mathcal{O}(NQK)$ over the full network. Then, for each iteration $i \in \mathcal{I}_m$ (except the last one), the extra data to be transmitted (only by a single node each time) is given in (14), which has a cost in $\mathcal{O}(NQ)$ (assuming N is large[1]). Note that the final updating node does not have to broadcast (14) as the next iteration starts with a new batch. Thus, over the R iterations in $\mathcal{I}_m$, the total communication cost in the network is in $\mathcal{O}(NQ(K+R-1))$ per batch.

Algorithm 2 thus allows to perform $R > 1$ iterations per batch of N samples, albeit with a larger communication bandwidth than Algorithm 1 (i.e., $\mathcal{O}(NQ(R+K-1))$ versus $\mathcal{O}(NQK)$ per batch, respectively). To appreciate why this is an efficient and desirable tradeoff, consider the case where $R = K$. In this case, the bandwidth of Algorithm 2 is doubled compared to Algorithm 1, yet the former can perform K iterations in the time where the latter can only perform a single iteration (note that a batch increment is directly coupled to the sample time of the sensors). As a result, the convergence or tracking speed is K times faster, whereas the bandwidth is only doubled.

[1] In practice, the number of observations in a single batch is large in order to estimate the statistics of $\widetilde{\mathbf{y}}_k$ [1]. As a result, the transmission of the parameters $G_{k,q}$ becomes negligible. We therefore neglect these in the asymptotic communication cost.

Algorithm 2: DSFO in a Fully-Connected Network with Data Re-Use

X^0 initialized randomly, $i \leftarrow 0$, $m \leftarrow 1$.
repeat
  1) Every node k collects $\{\mathbf{y}_k(t)\}_{t\in\mathcal{T}_m}$, compresses it to obtain $\{\widehat{\mathbf{y}}_k^i(t)\}_{t\in\mathcal{T}_m}$ using (4) and broadcasts the compressed signals to all other nodes.
  repeat R times
    $q \leftarrow (i \mod K) + 1$.
    at *Node* q **do**
      2a) $\widetilde{X}_q^{i+1} \leftarrow \operatorname{argmin} \widetilde{\mathbb{P}}_q^i$.
      If the solution of (10) is not unique, select the solution which minimizes $||X_q^{i+1} - X_q^i||_F$.
      2b) Partition $\widetilde{X}_q^{i+1}$ as in (6).
      2c) Broadcast $\{\widehat{\mathbf{y}}_q^{i+1}(t) = X_q^{(i+1)T}\mathbf{y}_q(t)\}_{t\in\mathcal{T}_m}$ and $\{G_{k,q}^{i+1}\}_{k\neq q}$ to all other nodes.
    end
    3) For all k, every node updates X_k^{i+1} according to (12) and recomputes $\{\widehat{\mathbf{y}}_k^{i+1}(t)\}_{t\in\mathcal{T}_m} = \{G_{k,q}^{(i+1)T}\widehat{\mathbf{y}}_k^i(t)\}_{t\in\mathcal{T}_m}$.
    $i \leftarrow i+1$
  $m \leftarrow m+1$

4. SIMULATIONS

In this section, we demonstrate the performance of the proposed method. We consider a WSN with $K = 5$ nodes, each measuring the signal $\mathbf{y}_k$ on $M_k = 5$ channels. The network-wide signal $\mathbf{y}$ has then $M = 25$ channels and is modeled as:

$$\mathbf{y}(t) = \mathbf{c} \cdot d(t) + \mathbf{n}(t), \tag{16}$$

where $\mathbf{c}$ is an $M-$dimensional vector, d is a $1-$dimensional source signal and $\mathbf{n}$ is an additive noise. The values over time of d have been chosen independently at random, following the zero-mean Gaussian distribution with variance 0.5, i.e., $\mathcal{N}(0, 0.5)$. Similarly, each observation and each entry of $\mathbf{n}$ independently follows $\mathcal{N}(0, 0.1)$. The entries of the time-invariant steering vector $\mathbf{c}$ are drawn from the uniform distribution on the interval $[-0.1, 0.1]$. We take $Q = 1$ such that $X = \mathbf{x} \in \mathbb{R}^M$. The problem solved by the WSN is:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \mathbb{E}\big[||\mathbf{x}^T\mathbf{y}(t)||^2\big] = \mathbf{x}^T K_{\mathbf{yy}}\mathbf{x} \\ \text{subject to} \quad & \mathbf{x}^T\mathbf{x} = 1. \end{aligned} \tag{17}$$

Its solution, denoted by $\mathbf{x}^*$, is given by the normalized eigenvector of the covariance matrix $K_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^T(t)]$ corresponding to the largest eigenvalue, which, based on (16), should be $\mathbf{x}^* \in \{\pm\mathbf{c}/||\mathbf{c}||\}$ (we fix $\mathbf{x}^* = \mathbf{c}/||\mathbf{c}||$). This problem has been studied in a DSFO setting in [17], and was later shown to be a special case of Algorithm 1 in [1].

The covariance matrix is estimated using batches of $N = 100000$ samples of $\mathbf{y}$ during the time intervals $\mathcal{T}_m$ using the following estimator:

$$K_{\mathbf{yy}} \approx \frac{1}{N}\sum_{t\in\mathcal{T}_m} \mathbf{y}(t)\mathbf{y}^T(t). \tag{18}$$

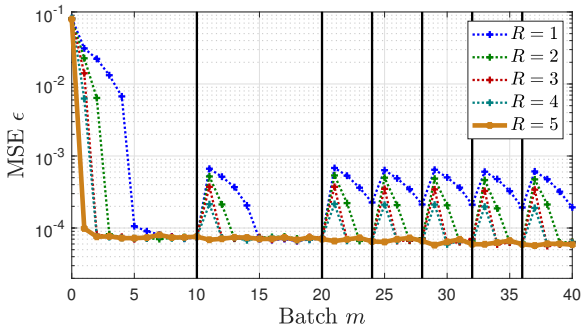


Fig. 1. Comparison of the MSE for different values of R in a network with $K = 5$ nodes.

Each estimation of the covariance matrix is then used R times before using newly collected data to obtain a new estimation of $K_{\mathbf{yy}}$. Additionally, we also change the value of $\mathbf{c}$ at random points in time with an additive noise following $\mathcal{N}(0, 0.01^2)$ to simulate a change in signal statistics and to assess the tracking performances for different values of R.

In our experiments, the metric we use to compare different approaches is the mean-squared error:

$$\epsilon(i) = \frac{1}{M}\left|\left|\mathbf{x}^i - \frac{\mathbf{c}}{||\mathbf{c}||}\right|\right|^2, \tag{19}$$

where we only evaluate $\mathbf{x}^i$ at the last update for the batch, i.e., $i = mR - 1$. Figure 1 shows the results for $R \in \{1, 2, 3, 4, K = 5\}$. The plots represent the median value of ϵ over 100 Monte-Carlo runs. The time instants where $\mathbf{c}$ changes are indicated by black vertical lines.

We observe that for a fixed $\mathbf{c}$, the minimal MSE value is achieved faster for larger values of R, which is eventually reached for every R when the statistics of the signals do not change for a long enough period of time. After every change in statistics, a slight increase in the MSE can be observed but can be corrected if $\mathbf{c}$ does not change too rapidly (as can be seen for $m \in \{10, \ldots, 20\}$). An exception is the case where $R = K = 5$ which is not affected by the changes of $\mathbf{c}$ and is able to track the changes well. On the other hand, if the signal statistics change too fast, a larger value of R is necessary for tracking performances to stay accurate. In the case of $R = 1$, i.e., the original DSFO algorithm (Algorithm 1), the MSE does not attain its minimal value when the signal statistics change very frequently, i.e., after the $20-$th batch, when the statistics change every 4 batch. Note that the bandwidth is only doubled in the case of $R = 5$ (vs. the case $R = 1$), while achieving a $5-$fold faster convergence, thereby allowing to perfectly track the changes in $\mathbf{c}$.

5. CONCLUSION

In this paper, we have proposed a scheme to improve the convergence or tracking speed of the DSFO algorithm for a fully-connected WSN. This is done by re-using the same batch of observations over multiple iterations and adapting the DSFO algorithm so as to keep the communication cost as low as possible. Our claims were confirmed by simulation results. As a future work, an extension of this scheme will be developed for any connected network topology.

6. REFERENCES

[1] Cem Ates Musluoglu and Alexander Bertrand, "A unified framework for distributed signal and feature fusion problems," *Submitted for publication - preprint available at https://homes.esat.kuleuven.be/~abertran/reports/DSFO2021.pdf*, 2021.

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

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

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

[5] Alexandros G Dimakis, Soummya Kar, José MF Moura, Michael G Rabbat, and Anna Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[6] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[7] Bicheng Ying, Kun Yuan, and Ali H Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, 2018.

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

[9] João FC Mota, João MF Xavier, Pedro MQ Aguiar, and Markus Puschel, "Distributed basis pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, 2011.

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

[11] Cristiano Gratton, Naveen KD Venkategowda, Reza Arablouei, and Stefan Werner, "Distributed ridge regression with feature partitioning," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 1423–1427.

[12] Cristiano Gratton, Naveen KD Venkategowda, Reza Arablouei, and Stefan Werner, "Distributed learning over networks with non-smooth regularizers and feature partitioning," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1840–1844.

[13] Bingwen Zhang, Jun Geng, Weiyu Xu, and Lifeng Lai, "Communication efficient distributed learning with feature partitioned data," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2018, pp. 1–6.

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

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

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

[17] Alexander Bertrand and Marc Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Signal Processing*, vol. 104, pp. 120–135, 2014.

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