Distributed Blind Source Separation based on FastICA

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

***Abstract*—With the emergence of wireless sensor networks (WSNs), many traditional signal processing tasks are required to be computed in a distributed fashion, without transmissions of the raw data to a centralized processing unit, due to the limited energy and bandwidth resources available to the sensors. In this paper, we propose a distributed independent component analysis (ICA) algorithm, which aims at identifying the original signal sources based on observations of their mixtures measured at various sensor nodes. One of the most commonly used ICA algorithms is known as FastICA, which requires a spatial pre-whitening operation in the first step of the algorithm. Such a pre-whitening across all nodes of a WSN is impossible in a bandwidth-constrained distributed setting as it requires to correlate each channel with each other channel in the WSN. We show that an explicit network-wide pre-whitening step can be circumvented by leveraging the properties of the so-called Distributed Adaptive Signal Fusion (DASF) framework. Despite the lack of such a network-wide pre-whitening, we can still obtain the Q least Gaussian independent components of the centralized ICA solution, where Q scales linearly with the required communication load.**

***Index Terms*—Distributed Optimization, Distributed Spatial Filtering, Independent Component Analysis.**

I. Introduction

INDEPENDENT component analysis (ICA) is a well-known method for reconstructing statistically independent source signals from linear mixtures of these sources measured across multiple sensors [1]–[3]. It has generally been associated with the blind source separation problem [4] and has found applications in various fields including biomedical [5], [6] or acoustic signal processing [7] among others [8], [9]. The emergence and versatility of wireless sensor networks (WSNs) [10], [11] make it attractive to solve such problems in a distributed setting, where different channels of the mixture signal are measured across various sensors placed in different locations. However, the energy and bandwidth bottlenecks of WSNs typically limit the amount of data that can be transmitted between the nodes and therefore make the use of a fusion center often not feasible in practice. Therefore, distributed algorithms that avoid data centralization are necessary for solving the ICA problem in a distributed setting such as WSNs.

Distributed algorithms have been proposed to solve the ICA / blind source separation problem [12], [13]. In [12], the network-wide ICA problem is solved heuristically with a (suboptimal) divide and conquer-type approach based on local neighborhoods. In [13], the ICA problem is solved in a truly distributed fashion, using the well-known alternative direction method of multipliers (ADMM). However, this algorithm iterates over sample batches, requiring multiple retransmissions of (updated versions of) the same batch of samples at each node, which leads to a large communication cost scaling poorly with the network size, often sharing even more data than what was collected at the node. Furthermore, for each new incoming batch of samples, the ADMM iterations have to start from scratch, making ADMM-based algorithms unfit for (adaptive) spatial filtering on streaming data [14]. We note that most distributed estimation methods based on ADMM, consensus, or diffusion [15]–[18] were originally designed for a setting where the data is partitioned in the sample dimension, making them unfit for the distributed ICA problem, where the data is partitioned in the channel dimension. Although some of these methods can be adapted to a channel-partitioning setting, they would face similar challenges as the ADMM-based ICA method in [13], requiring multiple iterations over every new sample batch. Another potential approach would be to first perform a distributed dimensionality reduction algorithm and perform a centralized ICA on the projected data. However, this is not always applicable and might lead to suboptimal results. For example, using a distributed principal component analysis procedure [19], we would select the subspace of sources with the highest variance when reducing dimensions. In low SNR scenarios, this high-variance subspace could mainly capture noise sources, instead of the relevant underlying sources [20].

In this paper, we propose DistrICA, a distributed algorithm that directly solves the network-wide ICA problem in an adaptive setting with streaming data, without first projecting the sensor data in a low-dimensional subspace. By only sharing linearly fused sensor signals between the nodes, the communication burden is significantly reduced, while still obtaining a subset of the sources from the centralized ICA problem.

II. Problem Setting

Let us consider M sensor signals y_m, $m \in \{1, \dots, M\}$, measured at each sensor m and denote by $y_m(t) \in \mathbb{R}$ the observation of y_m at time t. Arranging these signals as $\mathbf{y}(t) = [y_1(t), \dots, y_M(t)]^T \in \mathbb{R}^M$, we make the assumption that

$$\mathbf{y}(t) = A \cdot \mathbf{s}(t), \tag{1}$$

where $A \in \mathbb{R}^{M \times M}$ is the mixture matrix, and $\mathbf{s}(t) \in \mathbb{R}^M$ is the vector containing statistically independent source signals $s_m(t)$, $m \in \{1, \dots, M\}$. In the remaining parts of this paper, we will omit the time index t unless we want to explicitly refer to a specific time sample. Our objective is to recover the sources s_m from observations of $\mathbf{y}$ by applying a linear filter X^* such that $X^{*T}\mathbf{y} = \mathbf{r}$, where the entries of $\mathbf{r}$ are equal to those of $\mathbf{s}$ up to a scaling and permutation ambiguity [2]. Note that this can also be a partial recovery, in the sense that we might be only interested in a subset of the sources s_m, i.e., $\mathbf{r} \in \mathbb{R}^Q$, with $Q \leq M$. There exists various methods for this purpose, such as maximizing non-Gaussianity, maximum

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 and No 101138304). The authors also acknowledge the financial support of the FWO (Research Foundation Flanders) for project G081722N, and the Flemish Government (AI Research Program). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or ERC. Neither the European Union nor the granting authority can be held responsible for them.

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

likelihood estimation, or minimizing the mutual information [2], [21]. In this paper, we focus on the FastICA algorithm [1], which consists of first applying a pre-whitening step on $\mathbf{y}$ to obtain a signal $\mathbf{z}$, followed by an orthogonal transformation that maximizes the "non-Gaussianity" of the demixed signals[1]. Formally, the whitening procedure can be written as

$$\mathbf{z} = ED^{-1/2}E^T\mathbf{y}, \tag{2}$$

such that $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = I$, where E and D are obtained from the eigenvalue decomposition of the covariance matrix of $\mathbf{y}$, i.e., $R_{\mathbf{yy}} = \mathbb{E}[\mathbf{y}\mathbf{y}^T] = EDE^T$, assumed to be full rank[2]. The ICA filters are then obtained by solving the following problem

$$\underset{W=[\mathbf{w}_1,\ldots,\mathbf{w}_Q]}{\text{max.}} \quad \sum_{m=1}^{Q} \mathbb{E}[F(\mathbf{w}_m^T\mathbf{z})] \quad \text{s.t. } W^TW = I. \tag{3}$$

F is often chosen as $F(x) = \log\cosh(x)$ or $F(x) = -\exp(-x^2/2)$, making $\mathbb{E}[F(x)]$ a proxy for the negentropy of the random variable x (also known as the non-Gaussianity) [2]. The Q least Gaussian independent components are then found by applying a solution W^* of (3) to $\mathbf{z}$: $W^{*T}\mathbf{z}$. Equivalently, we can express this result using the original data $\mathbf{y}$ as $X^{*T}\mathbf{y}$, where $X^* = ED^{-1/2}E^TW^*$ from (2). The steps of FastICA are presented in Algorithm 1, where F' and F'' correspond to the first and second order derivative of F.

Let us now consider a sensor network represented by a graph $\mathcal{G}$ with K nodes within the set $\mathcal{K} = \{1,\ldots,K\}$. Each node k measures an M_k-channel signal $\mathbf{y}_k$, such that, defining

$$\mathbf{y} = [\mathbf{y}_1^T,\ldots,\mathbf{y}_K^T]^T, \tag{4}$$

our goal is to solve the ICA problem on the network-wide data $\mathbf{y} \in \mathbb{R}^M$, where $M = \sum_k M_k$. The signal $\mathbf{y}$ is assumed to be ergodic and (short-term) stationary, such that its statistical properties can be estimated through temporal averaging of observed samples. In such a distributed setting, our aim is again to find a spatial filter $X^* \in \mathbb{R}^{M\times Q}$ such that $X^{*T}\mathbf{y}$ provides the Q least Gaussian sources in $\mathbf{s}$. However, the pre-whitening (2) of the data requires the knowledge of the spatial correlation between different channels of $\mathbf{y}$. While there exist distributed algorithms for obtaining eigenvectors of $R_{\mathbf{yy}}$ (see, e.g., [19], [24]), these methods can only extract a few principal eigenvectors (depending on the available communication budget), and therefore can only compute a compressive version of (2) in which the data is projected onto a lower-dimensional subspace, which is not always desirable [20].

III. The Distributed ICA algorithm

In this section, we present a distributed ICA method based on the Distributed Adaptive Signal Fusion (DASF) framework [14], [25]–[28], which allows solving (adaptive) channel-partitioned distributed optimization problems. DASF uses a data-driven fuse-and-forward approach, where the N most recent samples of all nodes are forwarded towards an updating node, while linearly fusing them along the way. The updating node collects these fused streams and locally solves a compressed version of the original network-wide problem to find the in-network fusion rules for the next iteration. By rotating the updating node at each iteration, convergence to the optimal solution can be achieved [25], [26]. However, the ICA problem does not trivially fit the family of problems that the DASF algorithm can solve. Nevertheless, in Section III-A, we will explain how the two-step problem (2)-(3) can be cast into the DASF framework, and in particular, how the across-node pre-whitening step (2) can be bypassed, such that only per-node pre-whitening operations are required.

[1]The core idea behind this is that a mixture of non-Gaussian sources is closer to a Gaussian distribution than any of the unmixed sources.

[2]This is to guarantee well-posedness [22], [23] of the problem, which can be briefly summarized as requiring that a small change in the inputs of the problem (the data) implies a small change in the output (the optimal solution).

Algorithm 1: FastICA [1]

input : Multi-channel signal $\mathbf{y}$
$i \leftarrow 0$, $W \leftarrow [\,]$
Compute EDE^T as the eigenvalue decomposition of $R_{\mathbf{yy}}$
$\mathbf{z} \leftarrow ED^{-1/2}E^T\mathbf{y}$
for $m \in \{1,\ldots,Q\}$
  $\mathbf{w}$ initialized randomly
  while *convergence criterion not reached*
    $\mathbf{w} \leftarrow \mathbb{E}[\mathbf{z}F'(\mathbf{w}^T\mathbf{z})] - \mathbb{E}[F''(\mathbf{w}^T\mathbf{z})]\mathbf{w}$
    $\mathbf{w} \leftarrow \mathbf{w} - WW^T\mathbf{w}$ if $m > 1$
    $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|$
  end
  $W \leftarrow [W, \mathbf{w}]$
end
$X \leftarrow ED^{-1/2}E^TW$

A. Reformulating ICA as a DASF problem

As described in [14], the general form that the problems fitting the DASF framework take is given by

$$\begin{aligned}\min_X \mathbb{E}[G(X^T\mathbf{y})] \quad \text{s.t. } &\mathbb{E}[H_j(X^T\mathbf{y})] \le 0 \;\; \forall j \in \mathcal{J}_I,\\ &\mathbb{E}[H_j(X^T\mathbf{y})] = 0 \;\; \forall j \in \mathcal{J}_E,\end{aligned} \tag{5}$$

where the H_j's denote constraint functions of the problem and the sets $\mathcal{J}_I$ and $\mathcal{J}_E$ correspond to index sets of inequality and equality constraints respectively. The DASF algorithm can solve problems of the form (5) in a fully distributed and time-adaptive fashion with provable convergence guarantees [25]. Note that the family of the problems fitting the DASF framework is larger than the one represented by the formulation in (5), which is omitted here for conciseness. Let us now cast (2)-(3) into a problem fitting the DASF framework by embedding the pre-whitening step (2) within the ICA optimization problem (3), by making the change of variables: $X = ED^{-1/2}E^TW$, resulting in the equivalent problem

$$\underset{X=[\mathbf{x}_1,\ldots,\mathbf{x}_Q]}{\text{max.}} \quad \sum_{m=1}^{Q} \mathbb{E}[F(\mathbf{x}_m^T\mathbf{y})] \quad \text{s.t. } X^TR_{\mathbf{yy}}X = I. \tag{6}$$

Then, we can rewrite the objective function of (6) as $\mathbb{E}[G(X^T\mathbf{y})]$, where $G(X^T\mathbf{y}) = \sum_{m=1}^{Q} F_m(X^T\mathbf{y})$, such that $F_m(X^T\mathbf{y}) = F(\mathbf{e}_m^TX^T\mathbf{y})$, where $\mathbf{e}_m$ is the $m-$th column of the $Q \times Q$ identity matrix, making it fit the formulation in (5). Additionally, note that each entry of the constraint $X^TR_{\mathbf{yy}}X = \mathbb{E}[X^T\mathbf{y}\mathbf{y}^TX] = I$ is written as $\mathbb{E}[\mathbf{x}_m^T\mathbf{y}\mathbf{y}^T\mathbf{x}_n - \mathbb{1}\{m = n\}] = \mathbb{E}[\mathbf{e}_m^TX^T\mathbf{y}\mathbf{y}^TX\mathbf{e}_n - \mathbb{1}\{m = n\}] = 0$, for $m, n \in \{1,\ldots Q\}$, where $\mathbb{1}$ denotes the indicator function. Therefore, taking

$$H_{m,n}(X^T\mathbf{y}) = \mathbf{e}_m^TX^T\mathbf{y}\mathbf{y}^TX\mathbf{e}_n - \mathbb{1}\{m = n\}, \tag{7}$$

we see that the constraints of (6) fit the formulation of the DASF framework in (5). Note that, since (6) is equivalent to (2)-(3), it should be clear that Algorithm 1 also defines a solver for (6). The latter is a seemingly trivial yet important observation, which we will exploit in the next subsections when deriving the proposed DistrICA algorithm.

B. Data flow of DistrICA

Within the distributed problem setting introduced in Section II, let every node $k \in \mathcal{K}$ have an estimate X_k^i at iteration i of the block $X_k \in \mathbb{R}^{M_k \times Q}$ of X, partitioned as $\mathbf{y}$ in (4):

$$X = [X_1^T, \ldots, X_K^T]^T. \tag{8}$$

All X_k^i's are initialized randomly if $i = 0$. Every node then collects N time samples of its own M_k-channel signal $\mathbf{y}_k$ to obtain $\{\mathbf{y}_k(t)\}_{t=iN}^{iN+N-1}$ and linearly compresses every sample using its current estimate[3] of X_k into the $Q-$channel signal

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

Let us select one node among all the nodes of the network to be the updating node for the current iteration i, which we call q. The network is then temporarily pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$ to obtain a unique path between each and every node. The pruning function should not remove any link between the updating node q and its neighbors, but can otherwise be chosen freely [14]. Every node then proceeds to transmit the N compressed samples $\{\widehat{\mathbf{y}}_k^i(t)\}_{t=iN}^{iN+N-1}$, towards this updating node q in an inwards flow in which signals from neighboring nodes are linearly fused throughout their path towards node q:

$$\widehat{\mathbf{y}}_{k\to n}^i(t) \triangleq X_k^{iT}\mathbf{y}_k(t) + \sum_{l \in \mathcal{N}_k \setminus \{n\}} \widehat{\mathbf{y}}_{l\to k}^i(t), \tag{10}$$

where $\mathcal{N}_k$ denotes the set of neighboring nodes of node k after pruning, and $\widehat{\mathbf{y}}_{k\to n}^i \in \mathbb{R}^Q$ is the data transmitted by node k to its neighboring node n. Note that the second term of (10) is recursive and vanishes for the leaf nodes of the tree, which initializes the process such that the inward flow naturally arises without central coordination (a node l sends its data to node n from the moment it has received data from all its neighbors except one, being node n). In this way, each node effectively transmits a $Q-$channel signal, independent of the size of the network, making the communication bandwidth fully scalable.

The updating node q eventually receives N samples of

$$\widehat{\mathbf{y}}_{n\to q}^i(t) = X_n^{iT}\mathbf{y}_n(t) + \sum_{k \in \mathcal{N}_n \setminus \{q\}} \widehat{\mathbf{y}}_{k\to n}^i(t) = \sum_{k \in \mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i(t) \tag{11}$$

from all its neighbors $n \in \mathcal{N}_q$, where $\mathcal{B}_{nq}$ is the branch of $\mathcal{T}^i(\mathcal{G}, q)$ that is connected to q via n (excluding q itself).

C. Updating step

With the data exchange described in the previous subsection, node q is now in possession of N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ for $n \in \mathcal{N}_q$ and N samples of its own (uncompressed) signal $\mathbf{y}_q$. Stacking these, we define the data available at node q and iteration i as

$$\widetilde{\mathbf{y}}_q^i(t) \triangleq [\mathbf{y}_q^T(t), \widehat{\mathbf{y}}_{n_1\to q}^{iT}(t), \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}(t)]^T \in \mathbb{R}^{\widetilde{M}_q}, \tag{12}$$

where $\widetilde{M}_q = |\mathcal{N}_q| \cdot Q + M_q$. At this step, the DASF algorithm [14] requires the updating node q to use the available signal, i.e., $\widetilde{\mathbf{y}}_q^i$ to solve the following compressed version of (6):

$$\max_{\widetilde{X}_q = [\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_Q]} \quad \sum_{m=1}^{Q} \mathbb{E}[F(\widetilde{\mathbf{x}}_m^T \widetilde{\mathbf{y}}_q^i)] \quad \text{s.t.} \quad \widetilde{X}_q^T R_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}^i \widetilde{X}_q = I, \tag{13}$$

where $R_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}^i = \mathbb{E}[\widetilde{\mathbf{y}}_q^i \widetilde{\mathbf{y}}_q^{iT}]$ is the covariance matrix of $\widetilde{\mathbf{y}}_q^i$. As seen in (13), a new variable $\widetilde{X}_q$ is defined to act analogously to X locally. Remarkably, (13) has the exact same form as (6), and can therefore be solved locally at node q by applying the FastICA algorithm described in Algorithm 1, based on the batch of N available samples of $\widetilde{\mathbf{y}}_q^i$. We have thus effectively bypassed the network-wide pre-whitening step (2) and replaced it with a local one instead at node q.

While the solution $\widetilde{X}_q^*$ of (13) is only defined up to a scaling and permutation ambiguity, the solver in Algorithm 1 always finds the solution where the sources are scaled to unit-norm and ordered from least to most Gaussian. The remaining sign ambiguity can be easily resolved, e.g., by ensuring that the largest value of each column of $\widetilde{X}_q^*$ is positive, in order to avoid spurious sign flips across iterations.

The solution $\widetilde{X}_q^*$ of (13) is then partitioned as

$$\widetilde{X}_q^* = [X_q^{(i+1)T}, G_{n_1}^{(i+1)T}, \ldots, G_{n_{|\mathcal{N}_q|}}^{(i+1)T}]^T, \tag{14}$$

such that each partition has a corresponding block in the partitioning of $\widetilde{\mathbf{y}}_q^i$ as defined in (12). The Q least Gaussian sources can then be extracted at node q by computing $\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i = X^{(i+1)T}\mathbf{y}$, and be transmitted from node q to other nodes acting as data sinks if necessary. The blocks G_n^{i+1} are disseminated into the corresponding subgraph $\mathcal{B}_{nq}$ through node n, allowing every node to update its local estimator as

$$X_k^{i+1} = \begin{cases} X_q^{i+1} & \text{if } k = q \\ X_k^i G_n^{i+1} & \text{if } k \in \mathcal{B}_{nq},\ n \in \mathcal{N}_q, \end{cases} \tag{15}$$

such that, at the end of iteration i, the new estimate of the network-wide variable X becomes

$$X^{i+1} = [X_1^{(i+1)T}, \ldots, X_K^{(i+1)T}]^T. \tag{16}$$

This procedure is then repeated with a new updating node (e.g., chosen in a round-robin fashion) and a new set of N samples of $\mathbf{y}$ at each iteration, such that changes in the statistical properties can be tracked, making the algorithm adaptive. In particular, X^i is an estimator of $X^*(t)$ for $t = iN$. The steps of the proposed DistrICA algorithm are summarized in Algorithm 2. At the updating node, the

Algorithm 2: DistrICA Algorithm

X^0 initialized randomly, $i \leftarrow 0$.
repeat
  Choose the updating node as $q \leftarrow (i \mod K) + 1$.
  1) The network $\mathcal{G}$ is pruned into a tree $\mathcal{T}^i(\mathcal{G}, q)$.
  2) All nodes k collect N samples of $\mathbf{y}_k$ and compress them into $\widehat{\mathbf{y}}_k^i$.
  3) The nodes sum-and-forward $\widehat{\mathbf{y}}_k^i$ towards node q via the recursive rule (10). Node q eventually receives N samples of $\widehat{\mathbf{y}}_{n\to q}^i$ given in (11), from all its neighbors $n \in \mathcal{N}_q$.
  at *Node* q **do**
    4a) Solve Problem (13) to obtain $\widetilde{X}_q^*$ using Algorithm 1 on $\widetilde{\mathbf{y}}_q^i$ defined in (12).
    4b) Extract the Q least Gaussian sources by computing $\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i$.
    4c) Partition $\widetilde{X}_q^*$ as in (14) and disseminate every G_n^{i+1} in the corresponding subgraph $\mathcal{B}_{nq}$.
  end
  5) Every node updates X_k^{i+1} according to (15).
  $i \leftarrow i + 1$

[3]Note that the amount of compression depends on Q, which is defined by the number of independent components that we wish to extract.

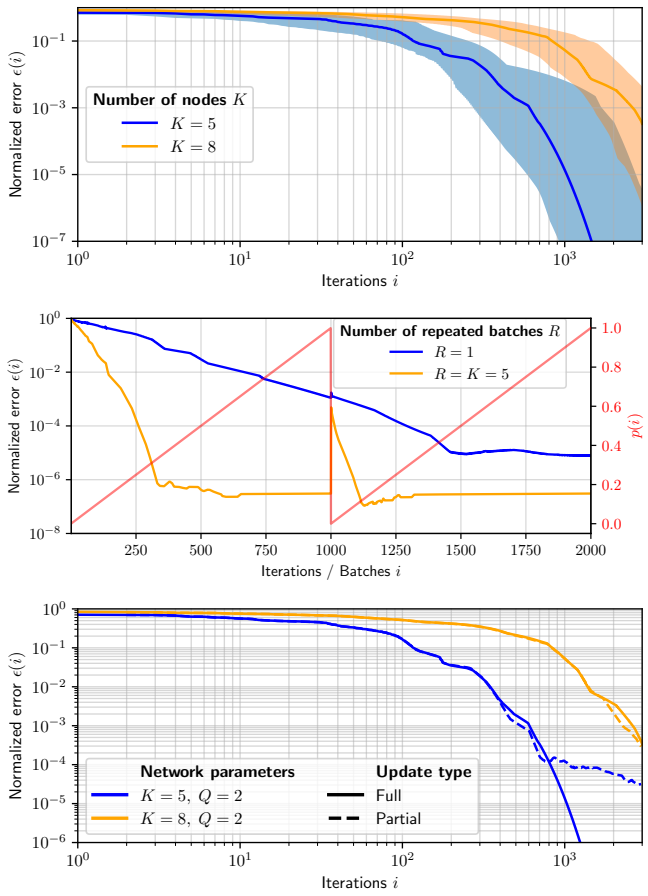

Fig. 1: (Top) Normalized error for varying number of nodes K. (Middle) Convergence in an adaptive setting. (Bottom) Comparison of the error for DistrICA with full and partial solutions.

computational complexity of DistrICA is in the order of the one of FastICA applied on the compressed data $\widetilde{\mathbf{y}}_q$, namely $\mathcal{O}(\widetilde{M}_q^2(N+\widetilde{M}_q)+QT(\widetilde{M}_qN+\widetilde{M}_q^2Q))$, where T corresponds to the number of iterations of the FastICA loop.

Since DistrICA is derived according to the technical principles of the DASF framework, its convergence is guaranteed by the theoretical results in [25]. Note that [25] imposes some mild – yet highly technical – conditions, which can be shown to hold for the case of DistrICA, but which are omitted here for conciseness. One crucial condition is that the number of constraints in (5) has to be smaller than Q^2, which is satisfied since there are $Q(Q+1)/2$ constraints in (6) due to the symmetry of $R_{\mathbf{yy}}$. Additionally, we exploit the fact that the solution set of FastICA is finite, where solutions can only differ up to a sign change of their columns, instead of an invariance to scaling and permutations, guaranteeing convergence to a single point [25, Theorem 5]. As (6) is a non-convex problem, the results in [25] only guarantee convergence to a stationary point of (3), which is also the case for the centralized FastICA algorithm. Nevertheless, FastICA almost always converges to a global optimum in practice [21], such that the same holds for DistrICA.

IV. Simulations

We consider a wireless sensor network where each node has $M_k = 5$ sensors, and therefore measure a $5-$channel local signal $\mathbf{y}_k$. Throughout this section, we consider networks randomly generated using the Erdős-Rényi model with connection probability equal to 0.8 and take $F(x) = \log\cosh(x)$ in the objective of (6). The signal model is as given in (1) and generated as follows. The elements of the mixture matrix $A \in \mathbb{R}^{M\times M}$ are drawn independently at random from the standard Gaussian distribution, i.e., $\mathcal{N}(0,1)$. We consider M independent sources in $\mathbf{s}$, where s_1 is a sinusoid, and s_2 a square signal (i.e., $Q = 2$), while s_m, $2 < m \leq M$ are convex combinations of uniformly and normally distributed noise, i.e., $s_m(t) = \alpha_m u_m(t) + (1-\alpha_m)n_m(t)$, where $u_m \sim \mathcal{U}[-0.5, 0.5]$, $n_m \sim \mathcal{N}(0,1)$, and $\alpha_m \in [0,1]$ is different for each source m. The resulting mixtures observed at the different sensors are normalized to unit variance. The goal is to separate s_1 and s_2 from the near-Gaussian (noise) sources. Note that all sources $\mathbf{s}$ have more or less the same variance, hence a principal component analysis will not be able to separate the target subspace from the noise subspace.

To assess the convergence of DistrICA to the centralized solution, we first consider a stationary setting, where at each iteration, $N = 10^4$ samples are measured at the sensor nodes, and the necessary statistical parameters of the stochastic signals are approximated using a temporal averaging over these samples. We use the normalized squared error to measure the accuracy of the estimator $\{X^i\}_i$ obtained through DistrICA over iterations i, given by $\varepsilon(i) = \text{median}\left(\frac{\|X^i - X^*\|^2}{\|X^*\|^2}\right)$, where the median is taken over 30 Monte-Carlo runs under the same experimental settings. An optimal solution X^* of the centralized ICA problem (6) is used for comparison and obtained through FastICA in a centralized setting. All simulations have been performed using the DASF toolbox [29].

Fig. 1 (Top) shows convergence plots for two different network sizes: $K = 5$ and $K = 8$. We observe that, although both settings lead to convergence towards an optimal solution of ICA, the setting with fewer nodes does so faster. This is expected, as the per-node updating frequency is lower in a larger network due to the sequential updating procedure.

In Fig. 1 (Middle), we show the convergence of DistrICA in an adaptive setting, where the mixture matrix A changes in time. For each new incoming batch of samples, A is updated to $A + \Delta \cdot p(i)$, where p is a scalar function defined by the red curve in Fig. 1, changing at each iteration i, i.e., every batch of N samples will have a different mixture matrix. The entries of Δ are drawn from $\mathcal{N}(0,1)$ first, and then scaled such that $\|\Delta\|_F = 0.005 \cdot \|A\|_F$. To make the DistrICA algorithm more adaptive, we now also implement the alternative strategy from [30], which reuses sample batches R times over multiple iterations (see [30] for details, $R = 1$ corresponds to the original algorithm). Note that, in contrast to the stationary setting, we now observe a tracking error that saturates at a non-zero value due to changing signal statistics.

Fig. 1 (Bottom) presents results of DistrICA when the FastICA solver of the compressed problem (13) is not exact, i.e., where we only let the nodes partially solve their local compressed ICA problem. In this experiment, we stop the local FastICA algorithm when it reaches an error of 10^{-3} in the normed difference between two consecutive filter values or a maximum of 10 iterations. This allows reducing the computational burden at the nodes while still guaranteeing convergence [26]. We see that until a certain error level, the convergence is not affected by this partial updating scheme and follows similar convergence properties while greatly reducing the number of computations performed at each node.

V. Conclusion

We have proposed the DistrICA algorithm to solve the ICA problem in a distributed and adaptive setting without requiring centralization of the data measured at different sensor nodes. After presenting its technical aspects, we have provided extensive simulation results comparing different problem settings

and validating its convergence towards the centralized solution of the ICA problem.

REFERENCES

[1] Aapo Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[2] Aapo Hyvärinen and Erkki Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[3] Aapo Hyvärinen, "Independent component analysis: recent advances," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1984, pp. 20110534, 2013.

[4] Seungjin Choi, Andrzej Cichocki, Hyung-Min Park, and Soo-Young Lee, "Blind source separation and independent component analysis: A review," *Neural Information Processing-Letters and Reviews*, vol. 6, no. 1, pp. 1–57, 2005.

[5] Christopher J James and Christian W Hesse, "Independent component analysis for biomedical signals," *Physiological measurement*, vol. 26, no. 1, pp. R15, 2004.

[6] Leonid Zhukov, David Weinstein, and Chris Johnson, "Independent component analysis for EEG source localization," *IEEE Engineering in Medicine and Biology Magazine*, vol. 19, no. 3, pp. 87–96, 2000.

[7] Futoshi Asano, Shiro Ikeda, Michiaki Ogawa, Hideki Asoh, and Nobuhiko Kitawaki, "Combined approach of array processing and independent component analysis for blind separation of acoustic signals," *IEEE transactions on speech and audio processing*, vol. 11, no. 3, pp. 204–215, 2003.

[8] Ganesh R Naik and Dinesh K Kumar, "An overview of independent component analysis and its applications," *Informatica*, vol. 35, no. 1, 2011.

[9] D Jouan-Rimbaud Bouveresse and DN Rutledge, "Independent components analysis: theory and applications," in *Data Handling in Science and Technology*, vol. 30, pp. 225–277. Elsevier, 2016.

[10] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[11] D Praveen Kumar, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu, "Machine learning algorithms for wireless sensor networks: A survey," *Information Fusion*, vol. 49, pp. 1–25, 2019.

[12] Yusuke Hioka and W Bastiaan Kleijn, "Distributed blind source separation with an application to audio signals," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 233–236.

[13] SR Mir Alavi and W Bastiaan Kleijn, "Distributed linear blind source separation over wireless sensor networks with arbitrary connectivity patterns," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 3171–3175.

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

[15] 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.

[16] Federico S. Cattivelli and Ali H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.

[17] Hadi Zayyani, "Robust minimum disturbance diffusion LMS for distributed estimation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 521–525, 2021.

[18] Domenico Ciuonzo, Pierluigi Salvo Rossi, and Pramod K. Varshney, "Distributed detection in wireless sensor networks under multiplicative fading via generalized score tests," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9059–9071, 2021.

[19] 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.

[20] Fiorenzo Artoni, Arnaud Delorme, and Scott Makeig, "Applying dimension reduction to EEG data by principal component analysis reduces the quality of its subsequent independent component decomposition," *NeuroImage*, vol. 175, pp. 176–187, 2018.

[21] Erkki Oja and Zhijian Yuan, "The FastICA algorithm revisited: Convergence analysis," *IEEE transactions on Neural Networks*, vol. 17, no. 6, pp. 1370–1381, 2006.

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

[23] Yong-hui Zhou, Jian Yu, Hui Yang, and Shu-wen Xiang, "Hadamard types of well-posedness of non-self set-valued mappings for coincide points," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 63, no. 5-7, pp. e2427–e2436, 2005.

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

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

[26] Charles Hovine and Alexander Bertrand, "Distributed adaptive spatial filtering with inexact local solvers," *arXiv preprint arXiv:2405.03277*, 2024.

[27] Charles Hovine and Alexander Bertrand, "A distributed adaptive algorithm for non-smooth spatial filtering problems," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[28] Cem Ates Musluoglu and Alexander Bertrand, "A distributed adaptive algorithm for node-specific signal fusion problems in wireless sensor networks," in *2023 31st European Signal Processing Conference (EUSIPCO)*, 2023, pp. 1654–1658.

[29] Cem Ates Musluoglu and Alexander Bertrand, "DASF Toolbox," https://github.com/AlexanderBertrandLab/DASF_toolbox, 2022.

[30] Cem Ates Musluoglu, Marc Moonen, and Alexander Bertrand, "Improved tracking for distributed signal fusion optimization in a fully-connected wireless sensor network," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 1836–1840.