Distributed Adaptive Signal Fusion for Fractional Programs

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

***Abstract*—The distributed adaptive signal fusion (DASF) is an algorithmic framework that allows solving spatial filtering optimization problems in a distributed and adaptive fashion over a bandwidth-constrained wireless sensor network. The DASF algorithm requires each node to sequentially build a compressed version of the original network-wide problem and solve it locally. However, these local problems can still result in a high computational load at the nodes, especially when the required solver is iterative. In this paper, we study the particular case of fractional programs, i.e., problems for which the objective function is a fraction of two continuous functions, which indeed require such iterative solvers. By exploiting the structure of a commonly used method for solving fractional programs and interleaving it with the iterations of the standard DASF algorithm, we obtain a distributed algorithm with a significantly reduced computational cost compared to the straightforward application of DASF as a meta-algorithm. We prove convergence and optimality of this "fractional DASF" (F-DASF) algorithm and demonstrate its performance via numerical simulations.**

***Index Terms*—Distributed Optimization, Distributed Signal Processing, Spatial Filtering, Signal Fusion, Wireless Sensor Networks, Fractional Programming.**

I. Introduction

WIRELESS sensor networks (WSNs) consist of a set of sensor nodes that are distributed over an area to collect sensor signals at different locations. In a signal processing context, the aim of a WSN is often to find a filter to apply to these signals gathered at the various nodes such that the resulting filtered signals are optimal in some sense, or satisfy certain desired properties. In an adaptive context where signal statistics are varying in time, centralizing the data is very costly in terms of bandwidth and energy resources, as it requires continuously communicating a large number of samples of signals measured at the nodes. Therefore, various methods have been proposed, such as consensus, incremental strategies, diffusion or the alternating direction method of multipliers (ADMM) [2]–[5] to find these optimal filters in a fully distributed fashion, i.e., where the different sensor nodes collaborate with each other to perform a certain inference task.

In this paper, we are interested in signal fusion and spatial filtering problems, commonly encountered in biomedical signal analysis [6]–[8], wireless communication [9]–[11], or acoustics [12]–[14]. More specifically, our goal is to solve such problems over WSNs, by finding an optimal, network-wide spatial filter X, to optimally fuse the channels of the network-wide multi-channel signal $\mathbf{y}(t)$ obtained from stacking the local signals measured at each node of the network. Mathematically, the optimal X results from solving an optimization problem of which the objective function is written in the form $\varrho(X^T\mathbf{y}(t))$.

We focus in particular on so-called fractional programs, i.e., problems where the function ϱ can be written as a fraction of two continuous functions, i.e., $\varrho(X^T\mathbf{y}(t)) = \varphi_1(X^T\mathbf{y}(t))/\varphi_2(X^T\mathbf{y}(t))$. Several spatial filtering problems have such an objective, including signal-to-noise ratio (SNR) maximization filters [15], trace ratio optimization (TRO) [16], regularized total least squares (RTLS) [17], [18], among others, and have found applications in the aforementioned fields. For example, in the context of electroencephalography (EEG) sensor networks, the motor imagery problem aims to find such a filter X by solving the TRO problem to distinguish between imagined left versus right-hand movement using signals collected from various sensors placed on the scalp [7], [16]. Another example consists of solving the RTLS problem in an antenna or microphone array for direction-of-arrival estimation when both the observations and the source signals are noisy [19].

In the existing distributed methods mentioned earlier, the objective ϱ is assumed to be per-node separable, i.e., can be written as a sum $\varrho(X) = \sum_k \varrho_k(X)$ over the nodes k of the network, where ϱ_k depends solely on the data of node k. However, the objective function of the problems of interest, namely fractional spatial filtering problems, is not per-node separable and requires second-order statistics between all the sensor channel pairs of the nodes in the network. Artificially rewriting these problems as consensus-type ones to be able to apply ADMM or consensus is still possible but leads to a large increase in communication costs over the network that does not scale with the network size, even leading to nodes sharing more data than what they collect [20], [21]. This is because these algorithms iterate over a fixed batch of samples (each node transmitting the same batch multiple times), and have to start from scratch for each new batch of incoming samples. Therefore, we are interested in distributed methods that are directly designed to solve spatial filtering problems in

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 grant agreement 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 the granting authorities. Neither the European Union nor the granting authorities 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

A conference precursor of this manuscript has been published in [1].

an adaptive setting with streaming data to avoid having such issues.

For this purpose, an algorithmic framework called distributed adaptive signal fusion (DASF) has been proposed in [20], [22]. The main idea behind this method is to iteratively create a compressed version of the global problem at each node by transmitting only compressed data within the network, therefore greatly reducing the communication cost. These compressed problems are then solved locally at each node, and the process is repeated with a new batch of measured signals to be able to track changes in the signal statistics. A convenient property of the DASF "meta" algorithm is that the local problems to be solved at each node are compressed instances of the original (centralized) problem. The DASF algorithm therefore merely requires to "copy-paste" this solver at each individual node [23].

However, solving the compressed problem can be computationally expensive when an expensive iterative solver is required. This is indeed also the case for fractional programs, which are commonly solved using an iterative algorithm referred to as Dinkelbach's procedure. In this paper, we propose the fractional DASF (F-DASF) algorithm which implements a single iteration of the Dinkelbach procedure within each iteration of the DASF algorithm, therefore interleaving the iterations of both algorithms and avoiding nested iterative loops. This however implies that F-DASF cannot rely on the convergence guarantees and proofs of the original DASF algorithm in [20], [22], as these assume that the nodes fully execute Dinkelbach's procedure until convergence within each iteration of DASF. In contrast, the F-DASF algorithm only partially solves the compressed problem at each iteration by applying only a single step of the Dinkelbach procedure. As we will see, this will imply that various subresults in the convergence proof applicable to the DASF algorithm cannot be applied to the F-DASF algorithm. Results such as the fact that fixed points of the algorithm are stationary, and even the monotonic decrease of the objective have to be proven from scratch. Therefore, the main result of this paper will be to show that the simplified F-DASF algorithm still converges to the solution of the global problem, under similar conditions (such as well-posedness and constraint qualifications) as the original DASF algorithm. In our results, we will see that on top of significantly reducing the computational cost and simplifying the overall method, the F-DASF algorithm has comparable convergence rates to the DASF algorithm.

The resulting F-DASF algorithm can in fact be viewed as a generalization of the distributed trace ratio (TRO) algorithm in [24], [25], which is a particular fractional program where both φ_1 and φ_2 consist of only a single quadratic term. Our proposed F-DASF algorithm can be applied to the entire class of fractional programs. Furthermore, it extends the "vanilla" DASF framework in [20], [22] towards a more efficient class of algorithms for the particular case of fractional programs.

The outline of this paper is as follows. In Section II, we review the basics of fractional programs and Dinkelbach's procedure to solve them. In Section III we formally define the distributed problem setting and the assumptions used throughout this paper. The proposed F-DASF algorithm is provided in Section IV while its convergence proof is given in Section V. Finally, Section VI shows the performance of the algorithm and its comparison with the state-of-the-art.

Notation: Uppercase letters are used to represent matrices and sets, the latter in calligraphic script, while scalars, scalar-valued functions, and vectors are represented by lowercase letters, the latter in bold. We use the notation χ_q^i to refer to a certain mathematical object χ (such as a matrix, set, etc.) at node q and iteration i. The notation $\left(\chi^i\right)_{i\in\mathcal{I}}$ refers to a sequence of elements χ^i over every index i in the ordered index set $\mathcal{I}$. If it is clear from the context, we omit the index set $\mathcal{I}$ and simply write $\left(\chi^i\right)_i$. A similar notation $\{\chi^i\}_{i\in\mathcal{I}}$ is used for non-ordered sets. We define an accumulation point of a sequence $(X^i)_{i\in\mathbb{N}}$ as the limit of a converging subsequence $(X^i)_{i\in\mathcal{I}}$ of $(X^i)_{i\in\mathbb{N}}$, with $\mathcal{I}\subseteq\mathbb{N}$. Additionally, I_Q denotes the $Q\times Q$ identity matrix, $\mathbb{E}[\cdot]$ the expectation operator, $\mathrm{tr}(\cdot)$ the trace operator, and $|\cdot|$ the cardinality of a set.

II. Fractional Programming Review

A fractional program is an optimization problem with an objective function r represented by a ratio of two continuous and real-valued functions f_1 and f_2:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & r(X) \triangleq \frac{f_1(X)}{f_2(X)} \\ \text{subject to} \quad & X \in \mathcal{S}, \end{aligned} \tag{1}$$

where $\mathcal{S}\subset\mathbb{R}^{M\times Q}$ is a non-empty constraint set and $f_2(X)>0$ for $X\in\mathcal{S}$. We define the minimal value of r over $\mathcal{S}$ as $\rho^*\triangleq\min_{X\in\mathcal{S}} r(X)$ and the set of arguments achieving this value as $\mathcal{X}^*\triangleq\{X\in\mathcal{S}\mid r(X)=\rho^*\}$. We denote by $X^*\in\mathcal{X}^*$ one particular solution of (1). To solve fractional programs, there exist two prominent methods based on solving auxiliary problems instead of the original problem (1), for which an overview can be found in [26]. The first method consists of creating an equivalent problem by defining new optimization variables using what is referred to as the Charnes-Cooper transform. It initially was presented in [27] for linear f_1 and f_2 with $\mathcal{S}$ a convex polyhedron, and was then extended to cases where convexity assumptions were imposed on f_1 and f_2 [26], [28]. The second method is a parametric approach and is the one we will focus on in this paper. Initially presented in [29], it is often referred to as Dinkelbach's procedure. Let us define the auxiliary functions $f:\mathcal{S}\times\mathbb{R}\to\mathbb{R}$ and $g:\mathbb{R}\to\mathbb{R}$:

$$f(X,\rho)\triangleq f_1(X)-\rho f_2(X), \tag{2}$$

$$g(\rho)\triangleq\min_{X\in\mathcal{S}} f(X,\rho), \tag{3}$$

where ρ is a real-valued scalar. It can be shown that g is continuous, strictly decreasing with ρ, and has a unique root. We also observe that for any fixed X, the function f in (2) is affine, and therefore g is concave. Note that these results do not require convexity assumptions on f_1 or f_2. It has been shown [29]–[33] that there is a direct relationship between the roots of g and the solution of (1), given in the following lemma.

Lemma 1 (see [32])**.** *Suppose $\mathcal{S}$ is compact. Then, if for a scalar ρ we have $g(\rho)=0$ then $\rho=\rho^*$, where ρ^* is the global minimum of r.*

Algorithm 1: Dinkelbach's procedure [29]

```
output: $\rho^*$
$X^0$ initialized randomly in $\mathcal{S}$, $\rho^0 \leftarrow r(X^0)$, $i \leftarrow 0$
repeat
    1) $X^{i+1} \leftarrow \underset{X\in\mathcal{S}}{\text{argmin}}\ f(X,\rho^i)$
    2) $\rho^{i+1} \leftarrow r(X^{i+1})$ where $r$ is defined in (1)
    $i \leftarrow i+1$
```

It is noted that it is possible to obtain weaker relationships between the roots of g and (1) if $\mathcal{S}$ is not compact [32], [33], but this is beyond the scope of this review, i.e., we will assume that $\mathcal{S}$ is compact in the remaining of this paper, unless mentioned otherwise.

Dinkelbach's procedure is an iterative method aiming to find the unique root ρ^* of g. We start by initializing X^0 randomly and set $\rho^0 = r(X^0)$. Then, at each iteration i, we find the solution set of the auxiliary problem minimizing f given ρ:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X,\rho^i) = f_1(X) - \rho^i f_2(X) \\ \text{subject to} \quad & X \in \mathcal{S}. \end{aligned} \tag{4}$$

The solution of (4) might not be unique but given by a set $\mathcal{X}^{i+1}$, in which case one $X^{i+1} \in \mathcal{X}^{i+1}$ is selected. Finally, the function values are updated as:

$$\rho^{i+1} = r(X^{i+1}). \tag{5}$$

The steps of Dinkelbach's procedure are summarized in Algorithm 1, while convergence results are summarized in the following theorem, which combines results from [32]–[34].

Theorem 1. *Assuming $\mathcal{S}$ is compact, the sequence $(\rho^i)_i$ converges to the finite minimal value ρ^* of (1). Additionally, convergent subsequences of $(X^i)_i$ converge to an optimal solution of (1). If Problem (1) has a unique solution X^*, then $(X^i)_i$ converges to X^*.*

If the constraint set $\mathcal{S}$ is not compact, we can still obtain convergence of the sequence $(\rho^i)_i$ to ρ^* if Problem (1) has a solution, the solution sets of the auxiliary problems (4) are not empty and $\sup_i f_2(X^i)$ is finite [32], with sup denoting the supremum.

It is noted that Dinkelbach's procedure can be viewed as an instance of Newton's root-finding method, as we can show that [33]:

$$\rho^{i+1} = \rho^i - \frac{g(\rho^i)}{-f_2(X^{i+1})}, \tag{6}$$

which corresponds to Newton's root finding method applied to the function g, since $-f_2(X^{i+1})$ is a subgradient of g at ρ^i. Other algorithms using a different root finding method have also been described to solve fractional programs, for example the bisection method [18], [26]. It is however indicated in [35] that the Newton root finding procedure has faster convergence compared to those in the context of fractional programming.

III. Problem Setting and Preliminaries

We consider a WSN represented by a graph $\mathcal{G}$ consisting of K nodes where the set of nodes is denoted as $\mathcal{K} = \{1,\ldots,K\}$. Each node k measures a stochastic M_k-channel signal $\mathbf{y}_k$ considered to be (short-term) stationary and ergodic. The observation of $\mathbf{y}_k$ at time t is denoted as $\mathbf{y}_k(t) \in \mathbb{R}^{M_k}$. In practice, we will often omit this time index t for notational convenience unless we explicitly want to emphasize the time-dependence of $\mathbf{y}$. We define the network-wide signal to be

$$\mathbf{y} = [\mathbf{y}_1^T,\ldots,\mathbf{y}_K^T]^T, \tag{7}$$

such that the time sample $\mathbf{y}(t)$ at time t is the $M-$dimensional vector obtained by stacking every $\mathbf{y}_k(t)$, where $M = \sum_{k\in\mathcal{K}} M_k$. In our algorithm development, we do not assume the statistics of $\mathbf{y}$ to be known, which implies that the algorithm has to estimate or learn these on the fly based on incoming sensor data at the different nodes.

A. Fractional Problems for Signal Fusion in WSNs

In spatial filtering and signal fusion, we aim to combine the channels of the network-wide signal $\mathbf{y}$ using a linear filter X such that the filter X optimizes a particular objective function. In this work, we are interested in spatial filters that are the solution of some fractional program. Two well-known examples, namely trace ratio optimization (TRO) and regularized total least squares (RTLS), are shown in Table I for illustration purposes.

A generic instance of such fractional programs for spatial filtering and signal fusion can be written as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \varrho(X^T\mathbf{y}(t),X^TB) \triangleq \frac{\varphi_1\left(X^T\mathbf{y}(t),X^TB\right)}{\varphi_2\left(X^T\mathbf{y}(t),X^TB\right)} \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t),X^TB\right) \leq 0, \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t),X^TB\right) = 0, \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{8}$$

The functions η_j represent inequality constraints for $j \in \mathcal{J}_I$ and equality constraints for $j \in \mathcal{J}_E$, while the full index set of constraints is denoted as $\mathcal{J} = \mathcal{J}_I \cup \mathcal{J}_E$, such that we have $J = |\mathcal{J}|$ constraints in total. Throughout this paper, we assume that the constraint set $\mathcal{S}$ of (8) is compact, as is commonly assumed for fractional programs to guarantee convergence of the Dinkelbach procedure. We also assume that the denominator φ_2 is always strictly positive over $\mathcal{S}$ (or strictly negative, in which case we can minimize $-\varphi_1/-\varphi_2$). The matrix B is an $M\times L$ deterministic matrix independent of time. We will see that $\mathbf{y}$ and B are treated similarly in the proposed method, however, we make the distinction between the two to emphasize the adaptive properties of the proposed algorithm. In practice, the matrices B are often required to enforce some constraints and their number of columns L is application dependent. For example, in the TRO problem of Table I, we have $B = I_M$ in the constraints. In a distributed context, we partition B similar to (7):

$$B = [B_1^T,\ldots,B_K^T]^T, \tag{9}$$

where B_k is assumed to be known by node k. Note that the optimization variable X in (8) only appears in the fusion form $X^T\mathbf{y}$ or X^TB, which is a requirement for the DASF framework. Furthermore, every other parameter of the problem that is not fused with X is assumed to be known by every node (such as the signal d in the RTLS example in Table I).

The optimization problems written in the form (8) represent a subclass of the problems considered in the DASF framework [20], namely problems for which the objective is a fractional function. Our goal is to derive a new algorithm for these cases, which is computationally more attractive than straightforwardly applying the generic DASF "meta" algorithm to (8).

Similar to the DASF framework in [20], we also allow Problem (8) to contain multiple signals $\mathbf{y}$ or matrices B, which have been omitted from (8) for notational conciseness. For example, the RTLS problem in Table I requires two different $B-$matrices: $B^{(1)} = I_M$ in the denominator of the objective function since $||\mathbf{x}||^2 = (\mathbf{x}^T \cdot I_M) \cdot (\mathbf{x}^T \cdot I_M)^T = (\mathbf{x}^T \cdot B^{(1)}) \cdot (\mathbf{x}^T \cdot B^{(1)})^T$, and $B^{(2)} = A$ in the constraint function. For the sake of intelligibility, we will continue the derivation and algorithm analysis for a single set of $\mathbf{y}$, and B, yet we emphasize that this is not a hard restriction as the algorithm can be easily generalized to multiple signal variables and multiple deterministic matrices (we refer to the original DASF paper [20] for more details on these cases).

Note that in a centralized context, Problem (8) can be solved using the Dinkelbach procedure, where each auxiliary problem can be written as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \varphi_1\left(X^T\mathbf{y}(t), X^TB\right) - \rho^i\, \varphi_2\left(X^T\mathbf{y}(t), X^TB\right) \\ \text{subject to} \quad & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) \leq 0, \ \ \forall j \in \mathcal{J}_I, \\ & \eta_j\left(X^T\mathbf{y}(t), X^TB\right) = 0, \ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{10}$$

with

$$\rho^i = \frac{\varphi_1\left(X^{iT}\mathbf{y}(t), X^{iT}B\right)}{\varphi_2\left(X^{iT}\mathbf{y}(t), X^{iT}B\right)}. \tag{11}$$

Since the minimization of Problem (8) is done over X only, we define the following functions in order to compactify some of the equations throughout this paper:

$$\begin{aligned} r(X) &\triangleq \varrho(X^T\mathbf{y}(t), X^TB), \\ f_j(X) &\triangleq \varphi_j(X^T\mathbf{y}(t), X^TB),\ j \in \{1,2\}, \\ h_j(X) &\triangleq \eta_j(X^T\mathbf{y}(t), X^TB),\ \forall j \in \mathcal{J}. \end{aligned} \tag{12}$$

These definitions allow us to write Problem (8) as in (1) and the auxiliary problems (10) as in (4), where $\mathcal{S}$ denotes the constraint set defined by the constraint functions h_j. In the remaining parts of this text, we will mention interchangeably (1) and (8) to refer to the problem we aim to solve while interchangeably using (4) and (10) for its auxiliary problems, depending on whether we want to emphasize the specific $X^T\mathbf{y}(t)$, or X^TB, structure or not. We define X^* to be a solution of Problem (8) and $\mathcal{X}^*$ to be the full solution set.

B. General Assumptions

To be able to state convergence guarantees of the proposed algorithm, we require some assumptions on Problem (8). These assumptions are similar to those for the DASF algorithm proposed in [20], [22], the main difference being that some assumptions are required to hold for the auxiliary problems as well, in order for our proposed method to work. In practice, if Problem (8) satisfies these assumptions, the corresponding auxiliary problems typically do so as well.

TABLE I
EXAMPLES OF PROBLEMS WITH FRACTIONAL OBJECTIVES AS IN (8).

In TRO, $\mathbf{v}$ is a second stochastic signal collected by the nodes in addition to $\mathbf{y}$. In RTLS, d is a target signal that is assumed to be known, and A is a deterministic matrix used for Tikhonov regularization.

Problem	Cost function to minimize	Constraints
TRO [16]	$-\frac{\mathbb{E}[\|\|X^T\mathbf{v}(t)\|\|^2]}{\mathbb{E}[\|\|X^T\mathbf{y}(t)\|\|^2]}$	$X^TX = I_Q \Leftrightarrow$ $(X^TB)(X^TB)^T = I_Q$ with $B = I_M$
RTLS [17], [18]	$\frac{\mathbb{E}[\|\|\mathbf{x}^T\mathbf{y}(t)-d(t)\|\|^2]}{1+\|\|\mathbf{x}\|\|^2}$	$\|\|\mathbf{x}^TA\|\|^2 \leq \delta^2$

Assumption 1. *The targeted instance of Problem (8) and its corresponding auxiliary problems (10) are well-posed*[1]*, in the sense that the solution set is not empty and varies continuously with a change in the parameters of the problem.*

This first assumption translates to requiring that an infinitesimal change in the input of the problem, for example in the signal statistics of $\mathbf{y}$, results in an infinitesimal change in its solutions X^*. As an example, consider the "full-rank" versus "rank-deficient" least squares problems, for which Assumption 1 is satisfied for the former but not the latter.

Assumption 2. *The solutions of Problem (8) and its corresponding auxiliary problems (10) satisfy the linear independence constraint qualifications (LICQ), i.e., X^* satisfies the Karush-Kuhn-Tucker (KKT) conditions.*

Assumption 2 is a common assumption in the field of optimization [38]. The LICQ enforces that, if X^* is a solution of Problem (8), then the gradients $\nabla_X h_j(X^*)$, $j \in \mathcal{J}^*$, are linearly independent[2], where $\mathcal{J}^* \subseteq \mathcal{J}$ is the set of all indices j for which $h_j(X^*) = 0$. This is a requirement for the KKT conditions to be necessary for a solution X^*. Further details on constraint qualifications can be found in [38]. If there is no constraint function η_j in Problem (8), Assumption 2 simply implies that $\nabla_X f(X^*) = 0$.

An additional assumption requiring the compactness of the sublevel sets of the objective of (10) is required in the original DASF algorithm [20], [22], its purpose being to ensure that the points generated by the algorithm lie in a compact set. As we will show in Section V, this assumption can be omitted in the case of the proposed F-DASF algorithm due to the fact that we only consider fractional programs with compact constraint sets. Note that the latter assumption should not

[1]The notion of (generalized Hadamard) well-posedness we require is based on [36], [37]. The main difference is that we require the map from the parameter (inputs of the problem) space to the solution space to be continuous instead of upper semicontinuous, which is required for the convergence proof.

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

be viewed as a limiting condition, as we will empirically demonstrate in Section VI-D through a simulation that the F-DASF algorithm introduced in Section IV can still converge to the correct solution for non-compact constraint sets in problems for which the centralized Dinkelbach procedure also converges, yet without a theoretical guarantee.

C. Adaptivity and Approximations of Statistics

As mentioned previously, the signal $\mathbf{y}$ is stochastic, therefore the functions $\varphi_1, \varphi_2, \eta_j$ implicitly contain an operator to transform probabilistic quantities into deterministic values, such that the problem solved in practice is deterministic. The most common operator encountered in signal processing applications is the expectation, i.e., there exist deterministic functions Φ_1, Φ_2, H_j such that

$$\begin{aligned}\varphi_j(X^T\mathbf{y}(t), X^TB) &= \mathbb{E}[\Phi_j(X^T\mathbf{y}(t), X^TB)],\ j\in\{1,2\},\\ \eta_j(X^T\mathbf{y}(t), X^TB) &= \mathbb{E}[H_j(X^T\mathbf{y}(t), X^TB)],\ \forall j\in\mathcal{J}.\end{aligned} \tag{13}$$

A common way to approximate these functions in a practical implementation is to take N time samples of the signal $\mathbf{y}$, say $\{\mathbf{y}(\tau)\}_{\tau=t}^{t+N-1}$, and approximate the expectation through a sample average, for example:

$$\mathbb{E}[\Phi_j(X^T\mathbf{y}(t), X^TB)] \approx \frac{1}{N}\sum_{\tau=t}^{t+N-1}\Phi_j(X^T\mathbf{y}(\tau), X^TB). \tag{14}$$

Under the ergodicity assumption on the signal $\mathbf{y}$, (14) gives an accurate approximation for sufficiently large N. In practical realizations, the objective functions and constraints in (8) will be evaluated on such sample batches of size N.

Furthermore, the stationarity assumption imposed on $\mathbf{y}$ is merely added for mathematical tractability, as it allows us to remove the time-dependence, as it is often done in the adaptive signal processing literature to analyze asymptotic convergence [39], [40]. However, in practice, we do not require the signals to be stationary, but rather short-term stationary and/or with slowly varying statistics. In Section VI, we will demonstrate that the proposed method is indeed able to track changes in the statistical properties of the signal. For ease of notation, we will use the exact functions φ_j and η_j (or f_j and h_j) in the remaining parts of this text, while in practice and in the simulations presented in Section VI they will be replaced by an approximation such as (14).

IV. Cost-Efficient DASF for Fractional Programs

In this section, we propose a new algorithm for solving (8) in a distributed and adaptive fashion, where the required statistics of $\mathbf{y}$ are estimated and learned on the fly based on the incoming sensor observations. As mentioned earlier, the DASF algorithm in [20] can be straightforwardly applied to (8) to obtain such a distributed algorithm, where the nodes would use a solver for the original (centralized) problem to solve locally compressed instances of the network-wide problem. However, this would require a node to fully execute Dinkelbach's procedure (Algorithm 1) within each iteration of the DASF algorithm. Since Algorithm 1 is itself iterative, we would obtain nested iterations of Algorithm 1 within the DASF algorithm's iterations, which would result in a large computational cost at each node. Instead, we propose the fractional DASF (F-DASF) algorithm interleaving the steps of the DASF algorithm with the iterations of Dinkelbach's procedure (Algorithm 1), where each DASF iteration only requires to perform *a single* iteration of Algorithm 1, thereby greatly reducing the computational burden.

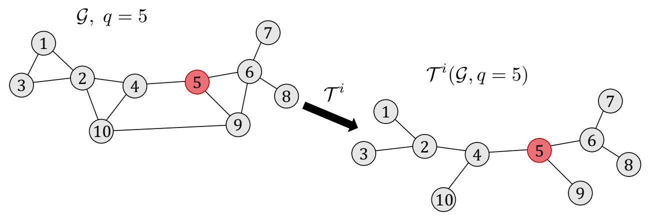


Fig. 1. Example of a pruning when the updating node is $q = 5$.

A. Dynamic Network Pruning

At each iteration i, an updating node $q \in \mathcal{K}$ is selected to be the so-called updating node. As will be described in the next subsections, the other nodes $k \neq q$ of the network will forward and fuse compressed data towards the updating node q which will be responsible for performing the update on the variable X at the current iteration. The selection of the updating node for different iterations will be done in a round-robin fashion. Based on the selected node, the network is pruned so as to obtain a spanning tree $\mathcal{T}^i(\mathcal{G}, q)$, such that there is a unique path connecting each pair of nodes. The pruning's main purpose is to define a spanning tree with a root in the updating node q, allowing to aggregate the data at this node. For networks with arbitrary topologies, feedback loops leading to unwanted effects in the data fusion process would be present if no pruning were implemented. The pruning function $\mathcal{T}^i$ can be chosen freely, however, it should avoid cutting any link between the updating node q and its neighbors $n \in \mathcal{N}_q$ [22], where we denote as $\mathcal{N}_q$ the set of nodes neighboring node q. A visual example of a pruning $\mathcal{T}^i$ for a given graph $\mathcal{G}$ is provided in Figure 1. In practice, the pruning can be implemented in a distributed fashion within the network, e.g., as in Algorithm 4 of [41]. We further note that the dependence of $\mathcal{T}^i$ on the index i is for flexibility purposes, e.g., if the network connectivity graph changes from iteration to iteration. In the remaining parts of this section, the set $\mathcal{N}_k$ refers to the neighbors of node k in the pruned network, i.e., with respect to the graph $\mathcal{T}^i(\mathcal{G}, q)$.

B. Data Flow

This subsection describes the data flow within the F-DASF algorithm, which is similar to the one of the DASF framework in [20], yet is included here for self-containedness.

Let us define X^i to be the estimation of the global filter X at iteration i and partitioned as

$$X^i = [X_1^{iT}, \dots, X_K^{iT}]^T, \tag{15}$$

where X_k is a $M_k \times Q$ matrix such that $X^{iT}\mathbf{y} = \sum_k X_k^{iT}\mathbf{y}_k$ and $X^{iT}B = \sum_k X_k^{iT}B_k$. At the beginning of each iteration i, every node $k \in \mathcal{K}\backslash\{q\}$ uses its current estimate X_k^i to compress its M_k-channel sensor signal $\mathbf{y}_k$ into a $Q-$channel signal, assuming $Q \leq M_k$. A similar operation is done on B_k to obtain

$$\widehat{\mathbf{y}}_k^i \triangleq X_k^{iT}\mathbf{y}_k,\ \widehat{B}_k^i \triangleq X_k^{iT}B_k. \tag{16}$$

Note that given X^i, the objective and constraints of Problems (8) and (10) can be evaluated at $X = X^i$ at a certain node if that node has access to the sums

$$X^{iT}\mathbf{y} = \sum_{k\in\mathcal{K}} \widehat{\mathbf{y}}_k^i,\ X^{iT}B = \sum_{k\in\mathcal{K}} \widehat{B}_k^i. \tag{17}$$

The idea is therefore to reconstruct these sums at the updating node q by fusing and forwarding the signals $\widehat{\mathbf{y}}_k^i$ and matrices $\widehat{B}_k^i$ towards node q. Note that for the signals $\widehat{\mathbf{y}}_k^i$, this means that each node k will need to transmit N samples[3], with N large enough to be able to accurately estimate the statistics required to evaluate the objective and constraints of (8) and (10), implying an $\mathcal{O}(NQ)$ per-node communication cost (assuming N is much larger than the number of columns of B).

The way the data is fused and forwarded towards node q is as follows. Each node $k \neq q$ waits until it receives data from all its neighboring nodes l except one, say node n. Upon receiving this data, node k sums all the data received from its neighbors to its own compressed data (16) and transmits this to node n. The data transmitted to node n from node k is therefore N samples of the signal

$$\widehat{\mathbf{y}}_{k\to n}^i = X_k^{iT}\mathbf{y}_k + \sum_{l\in\mathcal{N}_k\backslash\{n\}} \widehat{\mathbf{y}}_{l\to k}^i. \tag{18}$$

Note that the second term of (18) is recursive, and vanishes for leaf nodes, i.e., nodes with a single neighbor. Therefore, the data transmission is initiated at leaf nodes of the pruned network $\mathcal{T}^i(\mathcal{G},q)$. The data eventually arrives at the updating node q, which receives N samples of the signal

$$\widehat{\mathbf{y}}_{n\to q}^i = X_n^{iT}\mathbf{y}_n + \sum_{k\in\mathcal{N}_n\backslash\{q\}} \widehat{\mathbf{y}}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{\mathbf{y}}_k^i \tag{19}$$

from all its neighbors $n \in \mathcal{N}_q$. The set $\mathcal{B}_{nq}$ in (19) contains the nodes in the subgraph that includes node n after cutting the edge between nodes n and q in $\mathcal{T}^i(\mathcal{G},q)$. An example of this data flow towards node q is provided in Figure 2. A similar procedure is applied for the deterministic matrix B, such that node q receives

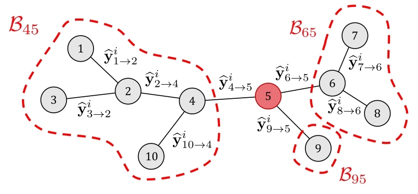
$$\widehat{B}_{n\to q}^i = X_n^{iT}B_n + \sum_{k\in\mathcal{N}_n\backslash\{q\}} \widehat{B}_{k\to n}^i = \sum_{k\in\mathcal{B}_{nq}} \widehat{B}_k^i, \tag{20}$$

from $n \in \mathcal{N}_q$.

Let us now label the neighboring nodes of node q as $\mathcal{N}_q \triangleq \{n_1, \ldots, n_{|\mathcal{N}_q|}\}$. Node q's own observation $\mathbf{y}_q$ and the compressed signals obtained from its neighbors can then be concatenated as

$$\widetilde{\mathbf{y}}_q^i \triangleq [\mathbf{y}_q^T, \widehat{\mathbf{y}}_{n_1\to q}^{iT}, \ldots, \widehat{\mathbf{y}}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T \in \mathbb{R}^{\widetilde{M}_q}, \tag{21}$$

[3]If $Q > M_k$, node k should simply transmit N samples of its raw signal $\mathbf{y}_k$ to one of its neighbors, say $n \in \mathcal{N}_k$, who treats $\mathbf{y}_k$ as part of its own sensor signal. In this way, node k does not actively participate in the algorithm but acts as an additional set of M_k sensors of node n.


$\mathcal{B}_{45}$
$\mathcal{B}_{65}$
$\mathcal{B}_{95}$
1
2
3
4
5
6
7
8
9
10
$\widehat{\mathbf{y}}_{1\to 2}^i$
$\widehat{\mathbf{y}}_{3\to 2}^i$
$\widehat{\mathbf{y}}_{2\to 4}^i$
$\widehat{\mathbf{y}}_{10\to 4}^i$
$\widehat{\mathbf{y}}_{4\to 5}^i$
$\widehat{\mathbf{y}}_{6\to 5}^i$
$\widehat{\mathbf{y}}_{9\to 5}^i$
$\widehat{\mathbf{y}}_{7\to 6}^i$
$\widehat{\mathbf{y}}_{8\to 6}^i$

$$C_5^i = \begin{bmatrix} 0 & X_1^i & 0 & 0 \\ 0 & X_2^i & 0 & 0 \\ 0 & X_3^i & 0 & 0 \\ 0 & X_4^i & 0 & 0 \\ I_{M_5} & 0 & 0 & 0 \\ 0 & 0 & X_6^i & 0 \\ 0 & 0 & X_7^i & 0 \\ 0 & 0 & X_8^i & 0 \\ 0 & 0 & 0 & X_9^i \\ 0 & X_{10}^i & 0 & 0 \end{bmatrix}$$

Fig. 2. (Adopted from [24]) Example of a tree network where the updating node is node 5. The clusters containing the nodes "hidden" from node 5 are shown here as $\mathcal{B}_{45}$, $\mathcal{B}_{65}$, $\mathcal{B}_{95}$. The resulting transition matrix is given by C_5^i.

where $\widetilde{M}_q = |\mathcal{N}_q|\cdot Q + M_q$, and similarly for the deterministic term B:

$$\widetilde{B}_q^i \triangleq [B_q^T, \widehat{B}_{n_1\to q}^{iT}, \ldots, \widehat{B}_{n_{|\mathcal{N}_q|}\to q}^{iT}]^T \in \mathbb{R}^{\widetilde{M}_q\times L}. \tag{22}$$

The quantities $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$ represent the data available at the updating node q during iteration i. In the original DASF algorithm [20], these local data are then used to construct a local (compressed) version of the original global problem at node q. Defining a new variable $\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q\times Q}$ at node q intending to mimic the global variable X locally, node q creates a local version of the global problem (8), given by

$$\begin{aligned} &\underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \ \frac{\varphi_1\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right)}{\varphi_2\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right)} \\ &\text{subject to } \eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right) \leq 0 \ \ \forall j \in \mathcal{J}_I, \\ &\qquad\qquad \eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t), \widetilde{X}_q^T\widetilde{B}_q^i\right) = 0 \ \ \forall j \in \mathcal{J}_E. \end{aligned} \tag{23}$$

At this point, the updating node q can follow the steps of the DASF algorithm [20] and locally solve Problem (23) to obtain the optimal local value $\widetilde{X}_q^*$, after which a new DASF iteration starts with a new updating node. Note that since (23) is a fractional program[4], node q uses the Dinkelbach procedure provided in Algorithm 1 to solve it. However, since the Dinkelbach procedure is itself iterative, using it to solve (23) is computationally expensive, creating nested iterations inside the outer-loop iterations of the DASF algorithm. Therefore, for generic fractional programs solved using Algorithm 1, the DASF algorithm operates at two different time scales. In the next subsection, we propose a modification to the DASF algorithm that avoids this problem by only solving a *single* step of the Dinkelbach procedure at each DASF iteration, greatly reducing the computational burden at each node.

C. The Fractional DASF (F-DASF) Algorithm

Instead of solving a compressed version of the global problem as in (23) at each updating node q, the proposed F-DASF

[4]This is a general property of the DASF framework: the local problems to be solved at each node exhibit the same structure as the centralized problem and can therefore use the same solver.

algorithm is focused on solving the following compressed version of the auxiliary problem (24):

$$\begin{aligned}\underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}}\ &\varphi_1\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right)-\rho^i\varphi_2\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right)\\ \text{subject to } &\eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right)\leq 0\ \ \forall j\in\mathcal{J}_I,\\ &\eta_j\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right)=0\ \ \forall j\in\mathcal{J}_E,\end{aligned}\tag{24}$$

where the term ρ^i appearing in the objective function corresponds to the current value of r for X^i, i.e., $\rho^i=r(X^i)$ and is computed at node q as follows. Let us define $\widetilde{X}_q^i$ as

$$\widetilde{X}_q^i\triangleq[X_q^{iT},I_Q,\ldots,I_Q]^T.\tag{25}$$

From the definitions provided in (17)-(22), we see that $\widetilde{X}_q^i$ is the local equivalent of X^i, such that

$$X^{iT}\mathbf{y}=\widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i,\ X^{iT}B=\widetilde{X}_q^{iT}\widetilde{B}_q^i.\tag{26}$$

Since node q has access to X_q^i, $\widetilde{\mathbf{y}}_q^i$ and $\widetilde{B}_q^i$, the above equations imply that:

$$\rho^i=\varrho\left(\widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^{iT}\widetilde{B}_q^i\right)=\frac{\varphi_1\big(\widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^{iT}\widetilde{B}_q^i\big)}{\varphi_2\big(\widetilde{X}_q^{iT}\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^{iT}\widetilde{B}_q^i\big)}.\tag{27}$$

Note that solving (24) corresponds to executing a single iteration of Dinkelbach's procedure in Algorithm 1. In the case of the standard DASF algorithm, solving the corresponding local problem (23) would require solving multiple instances of (24), each time with a different value for ρ, in order to execute all iterations in Algorithm 1 until convergence.

We define $\widetilde{X}_q^*$ as the solution of (24), i.e.,

$$\widetilde{X}_q^*\triangleq\underset{\widetilde{X}_q\in\widetilde{\mathcal{S}}_q^i}{\text{argmin}}\ \varphi_1\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right)-\rho^i\varphi_2\left(\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i(t),\widetilde{X}_q^T\widetilde{B}_q^i\right),\tag{28}$$

where $\widetilde{\mathcal{S}}_q^i$ is the constraint set of (24). If (24) does not have a unique solution, $\widetilde{X}_q^*$ is selected as the solution closest to $\widetilde{X}_q^i$ in (25), i.e.,

$$\widetilde{X}_q^*=\underset{\widetilde{X}_q\in\widetilde{\mathcal{X}}_q^i}{\text{argmin}}\ ||\widetilde{X}_q-\widetilde{X}_q^i||_F\tag{29}$$

where $\widetilde{\mathcal{X}}_q^i$ is the set of possible solutions $\widetilde{X}_q$ of (24) and $\widetilde{X}_q^i$ is given in (25). Note that the Frobenius norm to select $\widetilde{X}_q^*$ in (29) is arbitrary, and another distance function can be selected as long as it is continuous.

Finally, let us partition $\widetilde{X}_q^*$ as

$$\widetilde{X}_q^*=[X_q^{(i+1)T},G_{n_1}^{(i+1)T},\ldots,G_{n_{|\mathcal{N}_q|}}^{(i+1)T}]^T,\tag{30}$$

such that G_n is $Q\times Q$, $\forall n\in\mathcal{N}_q$, where the $G-$matrices consist of the part of $\widetilde{X}_q^*$ that is multiplied with the compressed signals of the neighbors of node q in the inner product $\widetilde{X}_q^{*T}\widetilde{\mathbf{y}}_q^i$. Every matrix G_n^{i+1} is then disseminated in the pruned network to the corresponding subgraph $\mathcal{B}_{nq}$ through node $n\in\mathcal{N}_q$, and every node $k\in\mathcal{K}$ can update its local variable X_k as

$$X_k^{i+1}=\begin{cases}X_q^{i+1} & \text{if } k=q\\ X_k^iG_n^{i+1} & \text{if } k\in\mathcal{B}_{nq},\ n\in\mathcal{N}_q.\end{cases}\tag{31}$$

Algorithm 2: F-DASF 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) Every node $k$ collects $N$ samples of $\mathbf{y}_k$. All
       nodes compress these to $N$ samples of $\widehat{\mathbf{y}}_k^i$ and
       also compute $\widehat{B}_k^i$ as in (16).
    3) The nodes sum-and-forward their compressed
       data towards node $q$ via the recursive rule (18)
       (and a similar rule for the $\widehat{B}_k^i$'s). Node $q$
       eventually receives $N$ samples of $\widehat{\mathbf{y}}_{n\to q}^i$ given in
       (19), and the matrix $\widehat{B}_{n\to q}^i$ defined in (20), from
       all its neighbors $n\in\mathcal{N}_q$.
    **at** *Node* $q$ **do**
        4a) Compute $\rho^i$ as in (27).
        4b) Compute (28), i.e., execute a single
            Dinkelbach iteration by solving (24), resulting
            in $\widetilde{X}_q^*$. If the solution is not unique, select a
            solution via (29).
        4c) Partition $\widetilde{X}_q^*$ as in (30).
        4d) Disseminate every $G_n^{i+1}$ in the
            corresponding subgraph $\mathcal{B}_{nq}$.
    **end**
    5) Every node updates $X_k^{i+1}$ according to (31).
    $i \leftarrow i+1$
```

This process is then repeated by selecting different updating nodes at different iterations. Note that every neighbor $n\in\mathcal{N}_q$ of the updating node q has a corresponding matrix G_n. As mentioned in Section IV-A, if the pruning function $\mathcal{T}^i$ were to remove a link between an updating node and one (or multiple) of its neighbors, the convergence speed of the F-DASF algorithm would suffer from it. Indeed, this would translate in less degrees of freedom in the local problem (24) itself, in turn leading to a fewer number of G_n matrices used for updating the global variable X as in (31).

Algorithm 2 summarizes the steps of the proposed F-DASF algorithm presented in this section. It is noted that at each iteration i, different batches of N samples of signals $\mathbf{y}_k$ are used. This means that the iterations of the F-DASF algorithm are spread out over different signal windows across time, making it able to track changes in the signal statistics, similar to an adaptive filter (see Section VI-C for an example). A visual comparison of the difference between the DASF and F-DASF algorithms is provided in Figure 3. The figure highlights how the F-DASF algorithm is computationally much more attractive compared to the original DASF method for fractional programs.

In Section V, we will show that the F-DASF algorithm converges to the solution of the centralized problem (8), despite the fact that none of the nodes have access to the full signal $\mathbf{y}$. In fact, each node only has access to the fused data received from its neighbors, such that the full set of second-order statistics of $\mathbf{y}$ is never observable. While these

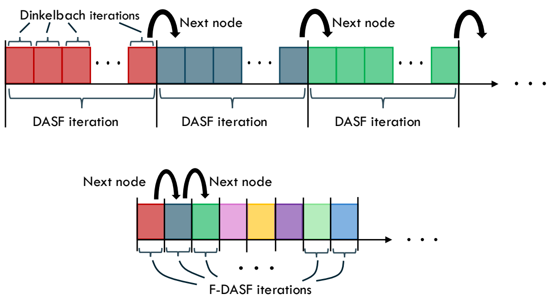


Fig. 3. Visual comparison between the DASF (top) and F-DASF (bottom) algorithms. Each rectangle corresponds to one Dinkelbach iteration in both scenarios.

statements are also valid for the DASF algorithm, the F-DASF method achieves the same result with a significantly reduced computational cost.

Remark 1. It is worth mentioning that although we have presented the F-DASF algorithm as applying a single iteration of Algorithm 1 within a DASF outer loop, F-DASF can also be interpreted as applying a single iteration of the DASF algorithm [20] in step 1 of Algorithm 1. In the latter case, instead of solving the auxiliary problem (10) in a distributed fashion by fixing ρ and applying DASF until convergence, we would only partially solve it by only performing a single iteration of DASF, i.e., a node solves its local problem and immediately updates ρ, which would again result in the F-DASF algorithm presented in this section.

D. Practical Considerations

Although the F-DASF algorithm can directly be implemented in practice by following the steps presented in Algorithm 2, we discuss here some practical considerations to take into account.

1) Non-stationarity: When using real data, the assumptions on the statistical properties of the signals discussed in Section III-C might not always hold in practice. For example, it is well-known that electroencephalography (EEG) signals are highly non-stationary [42], yet fractional programming problems are commonplace in EEG signal processing (for example in the form of the well-known common spatial patterns algorithm [7], [16]). In this case, the resulting filters fit to the longer term average statistics of the signals. Furthermore, the spatial coherence across different channels is often more stationary than the signals produced by the underlying signal sources, in which case the filters will mainly track and exploit these more stable spatial relationships. For example, in microphone array processing, the speech sources themselves are highly non-stationary, but their spatial location is often fixed or only slowly time-varying. If even these more stable spatial relationships turn out to change too fast between different batches of data, one way to improve the tracking properties of the F-DASF algorithm would be to apply multiple iterations over the same signal set. Note that this would increase the communication cost within the network, as the transmitted signals $\widehat{\mathbf{y}}_{k\to n}^i$ in (18) are still updated by the new values of X_k at each iteration. The method described in [43] provides a good tradeoff between the tracking improvement and communication cost increase in such modifications.

2) Effect of problem parameters: Another important aspect to consider is how the problem setting parameters affect the algorithm performance (we also refer the reader to [20] for further details on this). The number of columns Q of X is directly linked to the amount of compression applied to the raw signals $\mathbf{y}_k$ at each node k. A larger ratio Q/M_k implies less compression, i.e., transmission of more informative signals, therefore faster convergence of the F-DASF algorithm, while smaller values of Q lead to smaller communication costs per link at the expense of a slower convergence. Note that in various problems, the value of Q might be imposed by the problem itself, e.g., when the aim is to project the data into a $2-$dimensional space, implying $Q = 2$.

The network topology also affects the convergence speed, as more connected graphs result in larger degrees of freedom when updating the X_k's in (31). This is because the average number of neighbors for each updating node will be higher such that fewer nodes will have to share the same $G-$matrices, i.e., the tree rooted in the updating node will have more branches.

As for the number of nodes in the network, the relationship between K and the convergence is less straightforward than the other parameters discussed above. In general, the convergence speed during the first K iterations is expected to be slower for larger K's, as a full round of update where every node becomes the updating node would take longer to complete. This is especially significant if the starting estimate X^0 of X is chosen randomly, since the random initialization components will be present until a full round of update is done. For large networks, it is therefore beneficial to start from a value X^0 that is fixed, e.g., using prior knowledge of the problem such as an old estimate. Alternatively, the first K iterations of F-DASF could be done in parallel as a faster initialization step, allowing to map the X_k's to the correct scale.

3) Link and node failures: The F-DASF algorithm is robust to link failures as long as there are other paths within the graph where any node can reach any other one (note that a new spanning tree is formed at each iteration), and that a technical condition that relates the number of constraints with the network topology (see equation (68) in Section V) remains satisfied. This also holds in the case where a link connected to the updating node fails at the expense of reduced convergence speed due to the loss of degrees of freedom when updating the X_k's. On the other hand, if a node fails in the long term, the F-DASF algorithm will (re-)converge to the optimal solution using the data from all remaining nodes, assuming that the rest of the network remains connected.

4) Non-ideal Communication Links: In this paper, we have made abstraction of non-idealities in the communication links, i.e., we do not take quantization noise or data loss into account. We also assume a synchronized setting, implying that communication delays accumulate across the network, as nodes would wait until they receive the required data from their neighbors, before fusing and transmitting it to the next node. Furthermore, we assume that all nodes sample at the exact same sampling rate, which in practice requires synchronization protocols to compensate for inter-node sampling rate offsets.

V. Technical Analysis and Convergence

It can be shown that the F-DASF algorithm inherits the same convergence properties as the ones of the DASF algorithm detailed in [22], i.e., under mild technical constraints often satisfied in practice, the F-DASF algorithm converges to the optimal solution of the global problem (1). We note that these convergence results do not trivially follow from the convergence results of the original DASF algorithm. This is because DASF requires the updating nodes to *fully* solve (23). Instead, the updating nodes in F-DASF only perform a *single* iteration of Dinkelbach's procedure[5], which corresponds to solving (24). Although some results from the convergence proof of DASF can be re-used, some non-trivial extensions are required to establish convergence of F-DASF. In this subsection, we will therefore concentrate on the results that differ largely from the DASF case, while similar ones will be briefly mentioned for completeness. In particular, in the DASF scenario, proving the monotonic decrease of the objective and the stationarity of fixed points of the algorithm relied on the fact that each node fully solves a compressed version of the global problem. The main challenge for the F-DASF scenario is to still obtain these results despite this difference, by using fractional programming results presented in Section II. Similar to the proof in [22], for mathematical tractability, we assume that all signals are stationary and that each node is able to perfectly estimate the statistics of its locally available signals, i.e., as if $N \to +\infty$. This means that the proof is only asymptotically valid, i.e., approximating settings with sufficiently large batch sizes N. In practice, estimation errors on these statistics (due to finite N) will result in the algorithm only converging to a neighborhood of the true optimal point, where the solution will "hover" around this optimal point.

A. Preliminaries

We start by describing the relationship between the local problems and the global problem which will be useful for the technical analyses that will be presented in the later parts of this section. From equations (19)-(22), we observe that there exists a matrix $C_q^i \in \mathbb{R}^{M\times \widetilde{M}_q}$ such that

$$\widetilde{\mathbf{y}}_q^i = C_q^{iT}\mathbf{y},\ \widetilde{B}_q^i = C_q^{iT}B, \tag{32}$$

i.e., there exists a linear (compressive) relationship between the local data at the updating node q and the global data $\mathbf{y}$ and B. An example of such a matrix C_q^i is given in Figure 2 for the network given in the same figure. Note that, since the first M_q entries in $\widetilde{\mathbf{y}}_q^i$ are equal to $\mathbf{y}_q$ (see (21)), the matrix C_q^i must have the structure

$$C_q^i = \left[\begin{array}{c|c} 0 & \\ I_{M_q} & * \\ 0 & \end{array}\right], \tag{33}$$

where I_{M_q} is placed in the $q-$th block-row corresponding to the position of the entries $\mathbf{y}_q$ in $\mathbf{y}$.

Similar to (26), we observe that when node q applies a generic spatial filter $\widetilde{X}_q \in \mathbb{R}^{\widetilde{M}_q\times Q}$ to $\widetilde{\mathbf{y}}_q^i$, this is equivalent to applying a generic spatial filter $X \in \mathbb{R}^{M\times Q}$ to the network-wide sensor signal $\mathbf{y}$, i.e., $\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i = X^T\mathbf{y}$. With (32), we then have $\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i = \widetilde{X}_q^T(C_q^{iT}\mathbf{y}) = (C_q^i\widetilde{X}_q)^T\mathbf{y}$, while following similar steps for the deterministic matrix B gives $\widetilde{X}_q^T\widetilde{B}_q^i = \widetilde{X}_q^T(C_q^{iT}B) = (C_q^i\widetilde{X}_q)^TB$, such that

$$X = C_q^i\widetilde{X}_q. \tag{34}$$

When comparing (10) with (24), we observe that their objective and constraint functions are indeed the same if $\widetilde{X}_q^T\widetilde{\mathbf{y}}_q^i = X^T\mathbf{y}$. This implies that if $\widetilde{X}_q$ is a feasible point of the local auxiliary problem (24), the point X parameterized by (34) is a feasible point of the global auxiliary problem (10), and vice versa, as formalized in the following lemma (the proof is equivalent to the proof provided in [20] for the original DASF algorithm and is therefore omitted here).

Lemma 2. *For any iteration $i > 0$ of Algorithm 2,*

$$\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i \iff C_q^i\widetilde{X}_q \in \mathcal{S}. \tag{35}$$

In particular, $X^i \in \mathcal{S}$ and $\widetilde{X}_q^i \in \widetilde{\mathcal{S}}_q^i$ for all $i > 0$, where $\widetilde{X}_q^i$ is defined in (25).

We note that this lemma always holds, independent of whether the initialization point of F-DASF is in $\mathcal{S}$ or not. In particular, the last sentence of the lemma is important as it implies that every output X^i of the F-DASF algorithm is a feasible point of the global auxiliary problem (10) and hence also of the global problem (8) since they have the same constraint set.

The relationship (34) also allows us to compactly write Problem (24) using the functions f_j and h_j defined in (12):

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad & f(C_q^i\widetilde{X}_q,\rho^i) = f_1\left(C_q^i\widetilde{X}_q\right) - \rho^i f_2\left(C_q^i\widetilde{X}_q\right) \\ \text{subject to} \quad & h_j\left(C_q^i\widetilde{X}_q\right) \le 0,\ \ \forall j \in \mathcal{J}_I, \\ & h_j\left(C_q^i\widetilde{X}_q\right) = 0,\ \ \forall j \in \mathcal{J}_E, \end{aligned} \tag{36}$$

where

$$\rho^i = \frac{f_1\big(C_q^i\widetilde{X}_q^i\big)}{f_2\big(C_q^i\widetilde{X}_q^i\big)}. \tag{37}$$

In the remaining parts of this section, we will often refer to the compact notation in equations (1), (4) and (36) to refer to the global problem, its auxiliary problems, and the compressed auxiliary problem, respectively (instead of (8), (10) and (24)) for notational convenience.

B. Convergence in Objective

The following theorem establishes the convergence of the auxiliary parameter ρ^i, which then also implies that there is convergence in the objective of (1) / (8) via the relationship in (27).

[5]Alternatively, as mentioned in Remark 1, this can also be viewed as a global Dinkelbach procedure in which a *single* DASF iteration is performed at each iteration.

Theorem 2. *The sequence $(\rho^i)_i$ generated by Algorithm 2 is a non-increasing, converging sequence.*

Proof. From (37), we have $f\big(C_q^i\widetilde{X}_q^i,\rho^i\big) = f_1\big(C_q^i\widetilde{X}_q^i\big) - \rho^i f_2\big(C_q^i\widetilde{X}_q^i\big) = 0$. Since $\widetilde{X}_q^*$ solves (36), we have that $f\big(C_q^i\widetilde{X}_q^*,\rho^i\big) \leq f\big(C_q^i\widetilde{X}_q,\rho^i\big)$ for any $\widetilde{X}_q \in \widetilde{\mathcal{S}}_q^i$. We note that $\widetilde{X}_q^i$ as defined in (25) is a feasible point of (24), i.e., belongs to $\widetilde{\mathcal{S}}_q^i$ as is shown in Lemma 2. We then write $f\big(C_q^i\widetilde{X}_q^*,\rho^i\big) \leq f\big(C_q^i\widetilde{X}_q^i,\rho^i\big) = 0$. After reordering the terms of $f\big(C_q^i\widetilde{X}_q^*,\rho^i\big)$, we obtain $\frac{f_1\left(C_q^i\widetilde{X}_q^*\right)}{f_2\left(C_q^i\widetilde{X}_q^*\right)} = \rho^{i+1} \leq \rho^i$. Therefore, the sequence $(\rho^i)_i$ is monotonic non-increasing and since it is lower bounded by ρ^*, it must converge. □

We note that convergence of $(\rho^i)_i$ does not necessarily imply convergence of the underlying sequence $(X^i)_i$, nor the optimality of the resulting accumulation point, which is the topic of the next two subsections.

C. Stationarity of Fixed Points

A fixed point of the F-DASF algorithm is defined as a point $\overline{X} \in \mathcal{S}$ that is invariant under any F-DASF update step (for any updating node q), i.e., $X^i = \overline{X}$, $\forall i > 0$ when initializing Algorithm 2 with $X^0 = \overline{X}$. We will now present results that guarantee that such fixed points of the F-DASF algorithm are stationary points of the global problem (8) under mild technical conditions that are akin to the standard LICQ conditions in the optimization literature [38].

Condition 1a. *For a fixed point $\overline{X}$ of Algorithm 2, the elements of the set $\{\overline{X}^T\nabla_X h_j(\overline{X})\}_{j\in\mathcal{J}}$ are linearly independent.*

Condition 1a requires the linear independence of a set of J matrices of size $Q\times Q$, which can only be satisfied if

$$J \leq Q^2. \tag{38}$$

Although Condition 1a requires checking linear independence at fixed points, which are usually unknown, the cases where Condition 1a would be violated (while (38) is satisfied) are very contrived and highly improbable [22]. Additionally, this condition can be verified beforehand for some commonly encountered constraint sets, such as the generalized Stiefel manifold. When Condition 1a is satisfied, we obtain a first result on the stationarity of the fixed points of the F-DASF algorithm.

Theorem 3. *If Condition 1a is satisfied for a fixed point $\overline{X}$ of Algorithm 2, then $\overline{X}$ is a stationary point of both (i) the auxiliary problem (4) (or (10)) for $\rho = r(\overline{X})$ and (ii) the global problem (1) (or (8)), satisfying their KKT conditions.*

Proof. We will first show that a fixed point of the F-DASF algorithm is a KKT point of the auxiliary problem (4). Using this result, we will derive the steps to show that such a point is also a KKT point of the global problem (1).

The KKT conditions of the auxiliary problem (4) can be written as:

$$\nabla_X\mathcal{L}_\rho(X,\rho,\boldsymbol{\lambda}) = 0, \tag{39}$$

$$h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I,\ h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E, \tag{40}$$

$$\lambda_j \geq 0 \ \ \forall j \in \mathcal{J}_I, \tag{41}$$

$$\lambda_j h_j(X) = 0 \ \ \forall j \in \mathcal{J}_I, \tag{42}$$

where

$$\mathcal{L}_\rho(X,\rho,\boldsymbol{\lambda}) \triangleq f(X,\rho) + \sum_{j\in\mathcal{J}}\lambda_j h_j(X) \tag{43}$$

is the Lagrangian of (4). We use $\boldsymbol{\lambda}$ in bold as a shorthand notation for the set of all Lagrange multipliers $\lambda_j \in \mathbb{R}$ corresponding to the constraint h_j.

At iteration i, the updating node q solves the local auxiliary problem (36) which has the following Lagrangian

$$\widetilde{\mathcal{L}}_\rho(\widetilde{X}_q,\rho^i,\widetilde{\boldsymbol{\lambda}}(q)) \triangleq f(C_q^i\widetilde{X}_q,\rho^i) + \sum_{j\in\mathcal{J}}\lambda_j(q)h_j(C_q^i\widetilde{X}_q), \tag{44}$$

where the $\lambda_j(q)$'s are the Lagrange multipliers at updating node q and iteration i corresponding to the local problem (36) and $\widetilde{\boldsymbol{\lambda}}(q)$ denotes their collection. Since $\widetilde{X}_q^*$ obtained in step 4b of Algorithm 2 solves the local problem at node q and iteration i, it must satisfy the KKT conditions of the local problem. In particular, the stationarity condition is given as

$$\nabla_{\widetilde{X}_q}\widetilde{\mathcal{L}}_\rho(\widetilde{X}_q^*,\rho^i,\widetilde{\boldsymbol{\lambda}}(q)) = 0. \tag{45}$$

From the parameterization $X = C_q^i\widetilde{X}_q$ in (34), we have

$$\widetilde{\mathcal{L}}_\rho(\widetilde{X}_q,\rho^i,\widetilde{\boldsymbol{\lambda}}(q)) = \mathcal{L}_\rho(C_q^i\widetilde{X}_q,\rho^i,\widetilde{\boldsymbol{\lambda}}(q)), \tag{46}$$

allowing us to apply the chain rule on (45) to obtain

$$C_q^{iT}\nabla_X\mathcal{L}_\rho(C_q^i\widetilde{X}_q^*,\rho^i,\widetilde{\boldsymbol{\lambda}}(q)) = 0. \tag{47}$$

Using (43), and the parameterized notation in (36), we find the KKT conditions of the local problem:

$$C_q^{iT}\nabla_X\Big[f\left(C_q^i\widetilde{X}_q^*,\rho^i\right) + \sum_{j\in\mathcal{J}}\lambda_j(q)h_j\left(C_q^i\widetilde{X}_q^*\right)\Big] = 0, \tag{48}$$

$$h_j\left(C_q^i\widetilde{X}_q^*\right) \leq 0 \ \ \forall j \in \mathcal{J}_I,\ h_j\left(C_q^i\widetilde{X}_q^*\right) = 0 \ \ \forall j \in \mathcal{J}_E, \tag{49}$$

$$\lambda_j(q) \geq 0 \ \ \forall j \in \mathcal{J}_I, \tag{50}$$

$$\lambda_j(q)h_j\left(C_q^i\widetilde{X}_q^*\right) = 0 \ \ \forall j \in \mathcal{J}_I, \tag{51}$$

satisfied by $X^{i+1} = C_q^i\widetilde{X}_q^*$ and its corresponding set of Lagrange multipliers $\widetilde{\boldsymbol{\lambda}}(q)$. Our aim is now to show that at a fixed point of the F-DASF algorithm, i.e., a point such that $X^{i+1} = X^i = \overline{X}$, the KKT conditions (48)-(51) of the local problem are equivalent to the KKT conditions (39)-(42) of the global problem.

Let us first look at the local stationarity condition under the fixed point assumption $X^{i+1} = X^i = \overline{X}$, which allows us to replace $C_q^i\widetilde{X}_q^* = X^{i+1}$ with $C_q^i\widetilde{X}_q^i = X^i = \overline{X}$, also implying $\rho^{i+1} = \rho^i = \bar{\rho} = r(\overline{X})$. Making these changes in (48) result in

$$C_q^{iT}\nabla_X\Big[f(\overline{X},\bar{\rho}) + \sum_{j\in\mathcal{J}}\lambda_j(q)h_j(\overline{X})\Big] = 0. \tag{52}$$

From the structure of C_q^i as displayed in (33), we observe that the first M_q rows of (52) will be equal to

$$\nabla_{X_q} f(\overline{X}, \bar{\rho}) + \sum_{j\in\mathcal{J}} \lambda_j(q) \nabla_{X_q} h_j(\overline{X}) = 0. \tag{53}$$

Additionally, the fixed point assumption is independent of the updating node q, i.e., whichever node q is selected to be the updating node, the expression (52) will be true. Therefore, (52) and hence (53) simultaneously hold for any node q. Stacking the variations of (53) for each node q then leads to

$$\begin{bmatrix} \nabla_{X_1} f(\overline{X}, \bar{\rho}) \\ \vdots \\ \nabla_{X_K} f(\overline{X}, \bar{\rho}) \end{bmatrix} = \nabla_X f(\overline{X}, \bar{\rho}) = - \begin{bmatrix} \sum_{j\in\mathcal{J}} \lambda_j(1) \nabla_{X_1} h_j(\overline{X}) \\ \vdots \\ \sum_{j\in\mathcal{J}} \lambda_j(K) \nabla_{X_K} h_j(\overline{X}) \end{bmatrix}. \tag{54}$$

Let us now return to (52) and multiply it by $\widetilde{X}_q^{iT}$ from the left, where $\widetilde{X}_q^i$ is defined in (25). From the fact that $C_q^i \widetilde{X}_q^i = X^i = \overline{X}$, we can write

$$\overline{X}^T \nabla_X f(\overline{X}, \bar{\rho}) = - \sum_{j\in\mathcal{J}} \lambda_j(q) \overline{X}^T \nabla_X h_j(\overline{X}). \tag{55}$$

From the linear independence of the set $\{\overline{X}^T \nabla_X h_j(\overline{X})\}_j$ implied by Condition 1a, we obtain the result that the Lagrange multipliers $\{\lambda_j(q)\}_j$ satisfying (55) are unique. Note that the left-hand side of (55) does not depend on the node q, therefore the multipliers are the same for every node, i.e., $\lambda_j(q) = \lambda_j$ for any node q. This result can be used to re-write (54) as

$$\nabla_X f(\overline{X}, \bar{\rho}) = - \sum_{j\in\mathcal{J}} \lambda_j \nabla_X h_j(\overline{X}). \tag{56}$$

Therefore, a fixed point $\overline{X}$ satisfies the global stationarity condition (39) of the auxiliary problem (4) with corresponding Lagrange multipliers $\{\lambda_j\}_j$.

We now look at the three other KKT conditions. Note that the local primal feasibility condition (49) is satisfied globally, since (49) and (40) are the same, as was already shown in (35) for any point, so it must also hold for a fixed point. Additionally, $(\overline{X}, \{\lambda_j\}_j)$ satisfies the local versions of both the dual feasibility (50) and the complementary slackness condition (51), where we replace $C_q^i \widetilde{X}_q^* = X^{i+1}$ by $\overline{X}$ from the fixed point assumption, and $\lambda_j(q)$'s by λ_j's for all $j \in \mathcal{J}_I$. Therefore, $(\overline{X}, \{\lambda_j\}_j)$ also satisfies their global counterparts (41) and (42). The pair $(\overline{X}, \{\lambda_j\}_j)$ hence satisfies all the KKT conditions of the auxiliary problem (4).

We now proceed with proving that $\overline{X}$ also satisfies the KKT conditions of the original fractional program (1), of which the KKT conditions are given by

$$\nabla_X \mathcal{L}(X, \boldsymbol{\mu}) = 0, \tag{57}$$
$$h_j(X) \leq 0 \ \ \forall j \in \mathcal{J}_I,\ h_j(X) = 0 \ \ \forall j \in \mathcal{J}_E, \tag{58}$$
$$\mu_j \geq 0 \ \ \forall j \in \mathcal{J}_I, \tag{59}$$
$$\mu_j h_j(X) = 0 \ \ \forall j \in \mathcal{J}_I, \tag{60}$$

where

$$\mathcal{L}(X, \boldsymbol{\mu}) = r(X) + \sum_{j\in\mathcal{J}} \mu_j h_j(X) \tag{61}$$

is the Lagrangian of Problem (1). The gradient of r with respect to X can be shown to be equal to

$$\nabla_X r(X) = \frac{1}{f_2(X)} \left(\nabla_X f_1(X) - r(X) \nabla_X f_2(X) \right). \tag{62}$$

By plugging in $\overline{X}$ and using the fact that $r(\overline{X}) = \bar{\rho}$, we obtain

$$\nabla_X r(\overline{X}) = \frac{1}{f_2(\overline{X})} \nabla_X f(\overline{X}, \bar{\rho}), \tag{63}$$

and by substituting (56) in this equation, we eventually find

$$\nabla_X r(\overline{X}) = - \sum_{j\in\mathcal{J}} \frac{\lambda_j}{f_2(\overline{X})} \nabla_X h_j(\overline{X}). \tag{64}$$

Then, taking $\mu_j \triangleq \lambda_j / f_2(\overline{X})$ for every $j \in \mathcal{J}$, we observe that we satisfy the stationarity condition (57). Additionally, the primal feasibility condition (58) is automatically satisfied for $\overline{X}$ as it is identical to (40). Finally, since $f_2(X) > 0$ for any $X \in \mathcal{S}$, the multipliers $\mu_j = \lambda_j / f_2(\overline{X})$ satisfy (59) and (60) from the fact that λ_j's satisfy (41) and (42). The pair $(\overline{X}, \{\mu_j\}_j)$ therefore satisfies the KKT conditions of the global problem (1). □

For the multi-input single-output (MISO) case, the filter X is in fact a vector such that $Q = 1$. In this case, (38) implies that Condition 1a can only be satisfied if we have at most one constraint in Problem (8). Similar to [22], we propose an alternative condition for such cases in order to relax the upper bound on the number of constraints.

Condition 1b. *For a fixed point $\overline{X}$ of Algorithm 2, the elements of the set $\{D_{j,q}(\overline{X})\}_{j\in\mathcal{J}}$ are linearly independent for any q, where*

$$D_{j,q}(\overline{X}) \triangleq \begin{bmatrix} \overline{X}_q^T \nabla_{X_q} h_j(\overline{X}) \\ \sum_{k\in\mathcal{B}_{n_1 q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \\ \vdots \\ \sum_{k\in\mathcal{B}_{n_{|\mathcal{N}_q|} q}} \overline{X}_k^T \nabla_{X_k} h_j(\overline{X}) \end{bmatrix}, \tag{65}$$

which is a block-matrix containing $(1 + |\mathcal{N}_q|)$ blocks of $Q \times Q$ matrices.

The size of the matrices $D_{j,q}$ depends on the number of neighbors of node q, therefore Condition 1b depends on the topology of the network. In order to satisfy the linear independence of the matrices (65), an upper bound analogous to the one in (38) for Condition 1a can be found to be

$$J \leq (1 + \min_{k\in\mathcal{K}} |\mathcal{N}_k|) Q^2, \tag{66}$$

where it is assumed that the pruning function $\mathcal{T}^i(\cdot, q)$ preserves all the links of the updating node q. Additionally, we can show that the number of constraints J also needs to satisfy

$$J \leq \frac{Q^2}{K-1} \sum_{k\in\mathcal{K}} |\mathcal{N}_k| \tag{67}$$

for Condition 1b to be satisfied. This second upper bound is related to specific interdependencies between the $D_{j,q}$'s across different nodes q, and has been derived in [22]. Combining both (66) and (67), we have

$$J \leq \min\left(\frac{Q^2}{K-1}\sum_{k\in\mathcal{K}}|\mathcal{N}_k|,\ (1+\min_{k\in\mathcal{K}}|\mathcal{N}_k|)Q^2\right). \quad (68)$$

This is a more relaxed bound than the one in (38), yet it requires to know the full topology of the network. As in the case of the previous condition, Condition 1b is merely a technical condition, as it is highly likely to be satisfied in practice when (68) holds. Condition 1b leads to a result analogous to Theorem 3, given below.

Theorem 4. *If Condition 1b is satisfied for a fixed point $\overline{X}$ of Algorithm 2, then $\overline{X}$ is a stationary point of both (i) the auxiliary problem (4) (or (10)) for $\rho = r(\overline{X})$ and (ii) the global problem (1) (or (8)), satisfying their KKT conditions.*

The proof of this theorem can be obtained by combining the proof of Theorem 3 provided earlier and the proof of an analogous result for the DASF algorithm [22, Appendix B], which is why we omit it here. We note that the bound (68) should only be satisfied *before* pruning the network using $\mathcal{T}^i$. As long as the pruning function $\mathcal{T}^i$ preserves all links between the updating node q and its neighbors, it can be shown that the result from Theorem 4 holds [22]. This is also the case where the topology of the network would change dynamically across iterations, such as when link failures happen, as long as (68) is still satisfied during any of these failures.

Theorems 3 and 4 only guarantee that a fixed point of Algorithm 2 is a stationary point of Problem (1). A stronger result can be obtained if the fixed point minimizes the auxiliary problem (4).

Theorem 5. *(see [30]) If a point $\overline{X}$ minimizes the auxiliary problem (4) for $\rho = r(\overline{X})$, then it is a global minimizer of Problem (1).*

A proof of this theorem can be found in [30]. Combining Theorems 3, 4, and 5 we can obtain the following result.

Corollary 1. *Let $\overline{X}$ be a fixed point of Algorithm 2 satisfying either Condition 1a or 1b. Then, for $\rho = r(\overline{X})$, if the auxiliary problem (4) has a unique minimum and no other KKT points, we have $\overline{X} = X^*$, where X^* is a solution of (1).*

Proof. From Theorems 3 and 4, we saw that $\overline{X}$ is a KKT point of both (1) and its auxiliary problem (4) for $\rho = r(\overline{X})$. Since the only KKT point of (4) is its unique minimum, $\overline{X}$ solves (4) for $\rho = r(\overline{X})$. From Theorem 5, we conclude that $\overline{X}$ also solves the global problem (1). □

D. Convergence to Stationary Points and Global Minima

Conditions 1a and 1b allowed us to show that fixed points of the F-DASF algorithm are stationary points of (1). It remains to show that the sequence $(X^i)_i$ converges to a global minimum of Problem (1), or at least to a stationary point. As the results provided in this subsection, and their proof, are similar to the ones of the original DASF algorithm, we will omit technical details and refer the reader to [22]. To show convergence of $(X^i)_i$ to a single point, we require the following two conditions.

Condition 2. *The local auxiliary problems (36) satisfy Assumptions 1 and 2.*

Condition 3. *The number of stationary points of each local auxiliary problem (24) is finite, or the solver used by Algorithm 2 to solve the local auxiliary problems (24) can only find a finite subset of the solutions of (24).*

Condition 2 translates to requiring the local auxiliary problems to inherit the same properties of the centralized auxiliary problems (4). This condition is usually satisfied in practice since we already assumed that the centralized auxiliary problems (4) satisfy the assumptions presented in Section III-B, while the local auxiliary problems are compressed versions of these. When Condition 2 is satisfied, any accumulation point $\overline{X}$ of the sequence $(X^i)_i$ is a fixed point of Algorithm 2 for any q and $\lim_{i\to+\infty}||X^{i+1}-X^i|| = 0$. In addition to Condition 2, if Condition 3 is satisfied, then $(X^i)_i$ converges to a single fixed point $\overline{X}$. For the proofs in [22] to work in the case of F-DASF as well, we have to establish that all the points of the sequence $(X^i)_i$ lie in a compact set. This can be easily shown to be true for the F-DASF algorithm. Indeed, since the set $\mathcal{S}$ of Problem (1) is compact, and from the fact that all points of the sequence $(X^i)_i$ generated by the F-DASF algorithm lie in $\mathcal{S}$ (Lemma 2), the sequence $(X^i)_i$ lies in a compact set.

Remark 2. Note that in various fractional problems, the number of stationary points is not finite, for example in the TRO problem in Table I. It can be shown that a solution of the auxiliary problems of the TRO problem is given by an X containing in its columns the Q eigenvectors of $R_{\mathbf{vv}} - \rho^i \cdot R_{\mathbf{yy}}$ corresponding to its Q largest eigenvalues. However, any matrix XU, where U is an orthogonal matrix, is also a solution of the auxiliary problem of the TRO. For these cases, Condition 3 can be relaxed so as to require that the solver used for solving the auxiliary problems (4) can only obtain a finite set of the solutions of (4). For the TRO example, a solver that outputs the Q principal unit norm eigenvectors of $R_{\mathbf{vv}} - \rho^i \cdot R_{\mathbf{yy}}$ is therefore sufficient, as the possible outputs are only different up to a sign change in each column, making the set of possible solutions finite (except for the contrived degenerate case where these Q eigenvalues are not all distinct).

The final step is to combine the previous results to be able to state the convergence guarantees of the F-DASF algorithm to a stationary point of (1). From Condition 1 (either the form a or b), we know that fixed points of the F-DASF algorithm are stationary points of (1) satisfying its KKT conditions in Theorems 3 and 4. Combining this with Conditions 2 and 3, we can establish that the sequence $(X^i)_i$ converges and $\lim_{i\to+\infty} X^i = \overline{X}$, where $\overline{X}$ is a stationary point of (1) satisfying its KKT conditions.

Additionally, we can obtain stronger results if certain uniqueness assumptions are satisfied. Namely, if the global problem (1) has a unique minimum X^* and no other stationary points, or if the conditions of Corollary 1 are satisfied, we have $\overline{X} = X^*$, implying that $(X^i)_i$ converges to X^*. Finally, even in cases where these uniqueness assumptions are not met, we can still expect the F-DASF algorithm to converge to a global minimum if all minima are global minima. This is because of the monotonic decrease of $(r(X^i))_i = (\rho^i)_i$ (see Lemma 2), which implies that the sequence $(X^i)_i$ is kicked out of a potential equilibrium point which is not a minimum and cannot return to it, making the fixed points of Algorithm 2 that are in $\mathcal{X}^*$ the only stable ones.

In general, we see that the F-DASF algorithm converges under similar technical conditions as the DASF algorithm, except for the fact that some of these conditions are required to hold for the auxiliary problems (instead of the original problem) in the F-DASF case.

TABLE II
SUMMARY OF PARAMETERS USED IN THE SIMULATIONS.

Experiment	Section VI-A	Section VI-B	Section VI-D
Q	1	2	2
K	10		
M	$M = 50$	$M = 50$	$M = 100$
Signal Statistics	$\sigma_r^2 = 0.5$, $\sigma_n^2 = 0.2$, $\sigma_\Pi^2 = 0.3$	$\sigma_r^2 = 0.5$, $\sigma_n^2 = 0.1$, $\sigma_\Pi^2 = 0.1$	$\sigma_r^2 = 0.5$, $\sigma_n^2 = 0.2$, $\sigma_\Pi^2 = 0.2$
N	10000		
Monte Carlo Runs	100		

VI. SIMULATIONS

In this section, we demonstrate the performance of the F-DASF algorithm in various experimental settings and compare it to the DASF algorithm for various fractional programs. Implementations are provided in [23]. Throughout these experiments, we consider network topologies generated randomly using the Erdős-Rényi model with connection probability 0.8, and where the pruning function $\mathcal{T}^i(\cdot, q)$ is chosen to be the shortest path. Excluding Section VI-C, we consider stationary signals $\mathbf{y}$ and $\mathbf{v}$, following a mixture model:

$$\mathbf{y}(t) = \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{69}$$

$$\begin{aligned}\mathbf{v}(t) &= \Pi_r \cdot \mathbf{r}(t) + \mathbf{y}(t) \\ &= \Pi_r \cdot \mathbf{r}(t) + \Pi_s \cdot \mathbf{s}(t) + \mathbf{n}(t),\end{aligned} \tag{70}$$

with $\mathbf{r}(t)$, $\mathbf{s}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_r^2)$, $\mathbf{n}(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_n^2)$ for every entry and time instance t. The entries of the mixture matrices Π_s and Π_r are independent of time and are independently drawn from $\mathcal{N}(0, \sigma_\Pi^2)$. For each experiment, the number of channels M_k of the signals measured at node k are equal for each node, and given by M/K, while each node measures $N = 10^4$ samples of its local signals at each iteration of the algorithms. Table II gives an overview of the different parameters selected for the simulations presented below. The main performance metric we use to assess convergence of the F-DASF algorithm is the median squared error (MedSE) ϵ:

$$\epsilon(i) = \text{median}\left(\frac{||X^i - X^*||_F^2}{||X^*||_F^2}\right), \tag{71}$$

where the median is taken over multiple Monte Carlo runs and X^* denotes an optimal solution of the respective problem, obtained from a centralized solver implementing Dinkelbach's procedure. If the centralized problem has multiple solutions, we select X^* a posteriori as the one that best matches X^i in the final iteration of the F-DASF algorithm. The results we present next have been obtained by taking the median over 100 Monte Carlo runs, and a stopping criterion for Dinkelbach's procedure used in DASF has been set to be a threshold of 10^{-8} on the norm of the difference of two consecutive $\widetilde{X}_q$'s and a maximum number of iterations of 10.

We first consider two problems with compact constraint sets in a stationary setting in Sections VI-A and VI-B. In Section VI-C, we will consider time-varying mixture matrices to simulate non-stationarity in an adaptive context, while Section VI-D demonstrates convergence for a problem with a non-compact constraint set.

A. Regularized Total Least Squares

The regularized total least squares (RTLS) problem [17], [18] is given as

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^M} \ & \frac{\mathbb{E}[|\mathbf{x}^T\mathbf{y}(t) - d(t)|^2]}{1 + \mathbf{x}^T\mathbf{x}} = \frac{\mathbf{x}^T R_{\mathbf{yy}}\mathbf{x} - 2\mathbf{x}^T\mathbf{r}_{\mathbf{y}d} + r_{dd}}{1 + \mathbf{x}^T\mathbf{x}} \\ \text{s. t. } & ||\mathbf{x}^T L||^2 \leq 1,\end{aligned} \tag{72}$$

where the variable $X = \mathbf{x}$ is a vector, i.e., $Q = 1$. The matrix $R_{\mathbf{yy}} = E[\mathbf{y}(t)\mathbf{y}^T(t)]$ is the covariance matrix of $\mathbf{y}$, and similarly, we have $\mathbf{r}_{\mathbf{y}d} = \mathbb{E}[d(t)\mathbf{y}(t)]$ and $r_{dd} = \mathbb{E}[d^2(t)]$. In this example, (69) is of the form $\mathbf{y}(t) = \mathbf{p}_s \cdot s(t) + \mathbf{n}(t)$, where $\Pi_s = \mathbf{p}_s$ is a vector and s is a scalar. Moreover, d represents a noisy version of s, given by $d(t) = s(t) + w(t)$, where each time sample of w is drawn from $\mathcal{N}(0, 0.02)$. Finally, L is a diagonal matrix where each element of the diagonal follows $\mathcal{N}(1, 0.1)$. For a fixed ρ, the auxiliary problem of (72) is

$$\begin{aligned}\underset{\mathbf{x}\in\mathbb{R}^M}{\text{minimize}} \quad & \mathbf{x}^T R_{\mathbf{yy}}\mathbf{x} - 2\mathbf{x}^T\mathbf{r}_{\mathbf{y}d} + r_{dd} - \rho\cdot(1 + \mathbf{x}^T\mathbf{x}) \\ \text{subject to} \quad & ||\mathbf{x}^T L||^2 \leq 1.\end{aligned} \tag{73}$$

The local auxiliary problem at iteration i solved at the updating node q in the F-DASF algorithm is then given by

$$\begin{aligned}\min_{\widetilde{\mathbf{x}}_q\in\mathbb{R}^{\widetilde{M}_q}} \ & \widetilde{\mathbf{x}}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{\mathbf{x}}_q - 2\widetilde{\mathbf{x}}_q^T\mathbf{r}^i_{\widetilde{\mathbf{y}}_q d} + r_{dd} - \rho^i(1 + \widetilde{\mathbf{x}}_q^T\widetilde{I}_q^i\widetilde{I}_q^{iT}\widetilde{\mathbf{x}}_q) \\ \text{s. t. } & ||\widetilde{\mathbf{x}}_q^T\widetilde{L}_q^i||^2 \leq 1,\end{aligned} \tag{74}$$

where we have $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = \mathbb{E}[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$ and $\mathbf{r}^i_{\widetilde{\mathbf{y}}_q d} = \mathbb{E}[\widetilde{\mathbf{y}}_q^i(t)d(t)]$, with $\widetilde{\mathbf{y}}_q^i$ the locally available signal defined in (21). The matrices $\widetilde{I}_q^i$ and $\widetilde{L}_q^i$ are the locally available versions of I_M and L, respectively, obtained by taking $B = I_M$ and $B = L$ and computing $\widetilde{B}_q^i$ as in (22). Problem (74) is a quadratic problem with quadratic constraints and can be solved by a solver implementing, for example, an interior-point method. While F-DASF will only solve one instance of (74), the DASF algorithm will solve multiple problems of the form (74) at each iteration, following the steps of the Dinkelbach algorithm. Figure 4 shows the results of the comparison between F-DASF and DASF. We see that, by construction, the F-DASF algorithm requires solving fewer

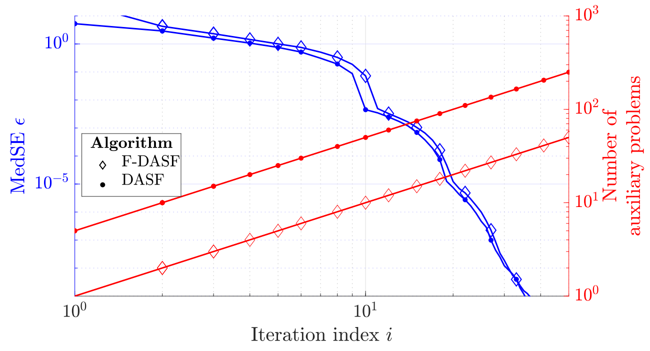


Fig. 4. Convergence and cumulative computational cost comparison between the proposed F-DASF algorithm and the DASF algorithm when solving Problem (72).

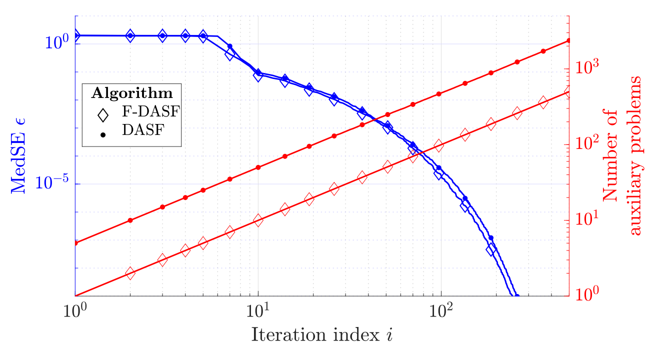


Fig. 5. Convergence and cumulative computational cost comparison between the proposed F-DASF algorithm and the DASF algorithm when solving Problem (75).

problems, yet the convergence rates are similar between F-DASF and DASF. More specifically, the F-DASF algorithm requires solving 5 times fewer auxiliary problems (on average over iterations of median values) than the DASF algorithm.

Note that the sharp change in the convergence plot at iteration $i = 10$ corresponds to the number of nodes $K = 10$, and is due to the random initialization of X^0, as explained in Section IV-D2.

B. *Trace Ratio Optimization*

In this experiment, we consider the trace ratio optimization (TRO) problem and compare an F-DASF implementation with a DASF implementation[6]. Here, we compare its performance to the DASF method. The TRO problem can be written as

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \frac{\mathbb{E}[||X^T\mathbf{v}(t)||^2]}{\mathbb{E}[||X^T\mathbf{y}(t)||^2]} = \frac{\text{tr}(X^T R_{\mathbf{vv}} X)}{\text{tr}(X^T R_{\mathbf{yy}} X)} \\ \text{subject to} \quad & X^T X = I_Q, \end{aligned} \tag{75}$$

with $R_{\mathbf{vv}} = E[\mathbf{v}(t)\mathbf{v}^T(t)]$ and $R_{\mathbf{yy}} = E[\mathbf{y}(t)\mathbf{y}^T(t)]$, while for a given ρ, its auxiliary problem takes the form

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \text{tr}(X^T R_{\mathbf{vv}} X) - \rho \cdot \text{tr}(X^T R_{\mathbf{yy}} X) \\ \text{subject to} \quad & X^T X = I_Q. \end{aligned} \tag{76}$$

Considering $R_{\mathbf{yy}}$ and $R_{\mathbf{vv}}$ to be positive definite, a solution of (76) is given by

$$X_\rho = \text{EVD}_Q(R_{\mathbf{vv}} - \rho \cdot R_{\mathbf{yy}}), \tag{77}$$

where EVD_Q returns the Q eigenvectors of $R_{\mathbf{vv}} - \rho \cdot R_{\mathbf{yy}}$ corresponding to its Q largest eigenvalues. If the $Q+1$ eigenvalues are all distinct, problem (76) satisfies the convergence assumptions. The local auxiliary problem solved using the F-DASF algorithm at each iteration i and updating node q is

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{maximize}} \quad & \text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} \widetilde{X}_q) - \rho^i \cdot \text{tr}(\widetilde{X}_q^T R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} \widetilde{X}_q) \\ \text{subject to} \quad & \widetilde{X}_q^T C_q^{iT} C_q^i \widetilde{X}_q = I_Q, \end{aligned} \tag{78}$$

[6]We note that the F-DASF implementation of TRO leads to the so-called DTRO algorithm in [24], which can be viewed as a special case of the more general F-DASF algorithm.

with $R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} = E[\widetilde{\mathbf{v}}^i_q(t)\widetilde{\mathbf{v}}^{iT}_q(t)]$ and $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = E[\widetilde{\mathbf{y}}^i_q(t)\widetilde{\mathbf{y}}^{iT}_q(t)]$, where $\widetilde{\mathbf{y}}^i_q$ and $\widetilde{\mathbf{v}}^i_q$ are defined as in (21). A solution $\widetilde{X}^*_q$ of (78) is given by $\text{GEVD}_Q(R^i_{\widetilde{\mathbf{v}}_q\widetilde{\mathbf{v}}_q} - \rho^i \cdot R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, C_q^{iT}C_q^i)$, where $\text{GEVD}_Q(C_1, C_2)$ returns the Q generalized eigenvectors of the pair (C_1, C_2) corresponding to its largest generalized eigenvalues. The DASF algorithm solves (78) multiple times at each updating node until the stopping criterion of the Dinkelbach procedure has been achieved. A comparison of the F-DASF and DASF algorithms is provided in Figure 5, where we again observe similar MedSE values per iteration for both methods, while the DASF algorithm requires solving 4.74 times more auxiliary problems on average (over iterations of median values) than the F-DASF method.

C. *F-DASF in an Adaptive Setting*

We now consider the same settings as in Section VI-B, however, the signal models are now given by

$$\mathbf{y}(t) = \Pi_s(t) \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{79}$$

$$\mathbf{v}(t) = \Pi_r(t) \cdot \mathbf{r}(t) + \Pi_s(t) \cdot \mathbf{s}(t) + \mathbf{n}(t), \tag{80}$$

i.e., the mixture matrices Π_s and Π_r are now time-dependent. In particular, we have $\Pi_s(t) = \Pi_{s,0} \cdot (1 - p(t)) + (\Pi_{s,0} + \Delta_s) \cdot p(t)$, where p is given in Figure 6 and the elements of $\Pi_{s,0}$ and Δ_s are independently drawn from $\mathcal{N}(0, 0.1)$ and $\mathcal{N}(0, 10^{-3})$ respectively, while $\Pi_r(t)$ is defined in the same way as $\Pi_s(t)$, except the fact that the elements of $\Pi_{r,0}$ are independently drawn from $\mathcal{N}(0, 0.5)$. Therefore, the signals $\mathbf{y}$ and $\mathbf{v}$ are not stationary anymore, which implies that X^* is time-dependent. In particular, at each iteration i, we estimate the solution X^{*i} at iteration i using $N = 10^3$ time samples of $\mathbf{y}$ and $\mathbf{v}$ by solving

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{maximize}} \quad & \frac{\text{tr}(X^T \widehat{R}^i_{\mathbf{vv}} X)}{\text{tr}(X^T \widehat{R}^i_{\mathbf{yy}} X)} \\ \text{subject to} \quad & X^T X = I_Q, \end{aligned} \tag{81}$$

using the Dinkelbach procedure, where

$$\widehat{R}^i_{\mathbf{yy}} = \frac{1}{N} \sum_{\tau=0}^{N-1} \mathbf{y}(\tau + iN)\mathbf{y}^T(\tau + iN), \tag{82}$$

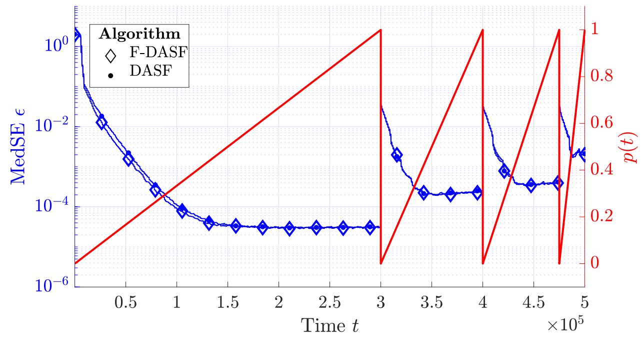


Fig. 6. MedSE of the DASF and F-DASF algorithms when solving Problem (75) in an adaptive setting. The relationship between the time t and the iterations i is given by $i = \lfloor t/N \rfloor$.

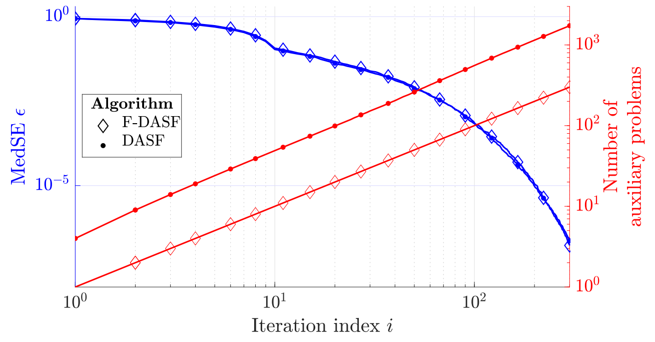


Fig. 7. Convergence and cumulative computational cost comparison between the proposed F-DASF algorithm and the DASF algorithm when solving Problem (84).

and similarly for $\widehat{R}^i_{\mathbf{vv}}$. The MedSE ϵ is then given by

$$\epsilon(i) = \text{median}\left(\frac{||X^i - X^{*i}||_F^2}{||X^{*i}||_F^2}\right). \tag{83}$$

Figure 6 shows the value of ϵ over time t with $i = \lfloor t/N \rfloor$, where we see that the F-DASF algorithm is able to track slow changes in the signal statistics of $\mathbf{y}$ and $\mathbf{v}$ and can adapt to abrupt changes as well, which is characterized by an initial jump in the MedSE value that gradually decreases. Note that ϵ settles around certain values due to the fact that X^* is time-dependent, with a higher MedSE settling value for faster rates of change (i.e., steeper slope of p) in the signal statistics.

D. Quadratic over Linear

To demonstrate the generic nature of the (F-)DASF algorithm, let us consider the following arbitrary toy problem

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \frac{\mathbb{E}[||X^T\mathbf{y}(t)||^2] + \text{tr}(X^TA)}{\text{tr}(X^TB)+c} \\ \text{subject to} \quad & \text{tr}(X^TB)+c > 0, \end{aligned} \tag{84}$$

where $R_{\mathbf{yy}} = E[\mathbf{y}(t)\mathbf{y}^T(t)]$ is the covariance matrix of $\mathbf{y}$, assumed to be positive definite. In this example, every entry of the matrices $A \in \mathbb{R}^{M\times Q}$ and $B \in \mathbb{R}^{M\times Q}$ have been independently drawn from $\mathcal{N}(0,1)$, while c is taken such that the problem is feasible[7]. It can be shown that the solution of Problem (84) has the form

$$X^* = -\frac{1}{2}R_{\mathbf{yy}}^{-1}(A + \mu\cdot B), \tag{85}$$

where μ is a scalar depending on $R_{\mathbf{yy}}$, A, B, and c. Note that the constraint $\text{tr}(X^TB)+c>0$ enforces the requirement $f_2(X) > 0$, but does not constitute a compact set since it is open and unbounded, implying that neither the Dinkelbach procedure nor the F-DASF algorithm have a guarantee of convergence to an optimal point. Therefore, the convergence of the F-DASF (and DASF) algorithm will be assessed by comparing X^i's to X^* given in (85).

[7]It can be shown that the problem is feasible, i.e., the solution is real, if c satisfies either $2c \geq \text{tr}(A^TR_{\mathbf{yy}}^{-1}B) + \sqrt{\text{tr}(A^TR_{\mathbf{yy}}^{-1}A)\text{tr}(B^TR_{\mathbf{yy}}^{-1}B)}$ or $2c \leq \text{tr}(A^TR_{\mathbf{yy}}^{-1}B) - \sqrt{\text{tr}(A^TR_{\mathbf{yy}}^{-1}A)\text{tr}(B^TR_{\mathbf{yy}}^{-1}B)}$.

For a given ρ, the auxiliary problem of (84) is

$$\begin{aligned} \underset{X\in\mathbb{R}^{M\times Q}}{\text{minimize}} \quad & \text{tr}(X^TR_{\mathbf{yy}}X) + \text{tr}(X^TA) - \rho\cdot(\text{tr}(X^TB)+c) \\ \text{subject to} \quad & \text{tr}(X^TB)+c > 0. \end{aligned} \tag{86}$$

If $R_{\mathbf{yy}}$ is positive definite, problem (86) is convex and has a unique solution given by

$$X_\rho = \frac{1}{2}R_{\mathbf{yy}}^{-1}(\rho\cdot B - A). \tag{87}$$

At each iteration i of the F-DASF algorithm, the updating node q solves its local auxiliary problem (24) given by

$$\begin{aligned} \underset{\widetilde{X}_q\in\mathbb{R}^{\widetilde{M}_q\times Q}}{\text{minimize}} \quad & \text{tr}(\widetilde{X}_q^TR^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}\widetilde{X}_q) + \text{tr}(\widetilde{X}_q^T\widetilde{A}_q^i) - \rho^i\cdot(\text{tr}(\widetilde{X}_q^T\widetilde{B}_q^i)+c) \\ \text{subject to} \quad & \text{tr}(\widetilde{X}_q^T\widetilde{B}_q^i)+c > 0, \end{aligned} \tag{88}$$

where $R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q} = E[\widetilde{\mathbf{y}}_q^i(t)\widetilde{\mathbf{y}}_q^{iT}(t)]$ is the covariance matrix of the locally available stochastic signal $\widetilde{\mathbf{y}}_q^i$ at node q defined in (21), while $\widetilde{A}_q^i$ and $\widetilde{B}_q^i$ are the locally available deterministic matrices, as defined in (22). The solution $\widetilde{X}_q^*$ of (88) is obtained by replacing $(\rho, R_{\mathbf{yy}}, A, B)$ by $(\rho^i, R^i_{\widetilde{\mathbf{y}}_q\widetilde{\mathbf{y}}_q}, \widetilde{A}_q^i, \widetilde{B}_q^i)$ in (87), since both (86) and (88) have the same form. In contrast, the DASF algorithm will solve multiple problems (88) at each iteration i and node q as it will apply Dinkelbach's procedure on a compressed version of (84). Figure 7 shows a comparison of both the MedSE values and the cumulative computational cost between the proposed F-DASF algorithm and the existing DASF method. Despite the fact that Problem (84) does not have a compact set, we see that convergence can still be achieved to the optimal solution X^* given in (85), showing that the F-DASF algorithm can still be used to solve problems with non-compact constraint sets. The F-DASF algorithm solves an average value (over iterations of median values) of 5.77 times fewer auxiliary problems than DASF for the case of Problem (84), while again obtaining similar convergence results.

VII. Conclusion

We have proposed the F-DASF algorithm which exploits the structure and properties of a fractional programming method to significantly reduce the computational cost of the DASF method applied to problems with fractional objectives. In particular, the DASF method requires solving a fractional program at each iteration while the F-DASF method only requires a partial solution, implying significantly fewer computations. Despite this reduced number of computations, the proposed method is shown to converge under similar assumptions as the DASF algorithm, and at similar convergence rates. The results obtained in this paper also show that partial solutions can be sufficient for the DASF method to converge and open the way for further studies on other families of problems fitting the DASF framework where computational cost reductions can be obtained by solving optimization problems only partially at each iteration.

References

[1] C. A. Musluoglu and A. Bertrand, "A computationally efficient algorithm for distributed adaptive signal fusion based on fractional programs," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

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

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

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

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

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

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

[8] H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial EEG during imagined hand movement," *IEEE transactions on rehabilitation engineering*, vol. 8, no. 4, pp. 441–446, 2000.

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

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

[11] E. Nayebi, A. Ashikhmin, T. L. Marzetta, and B. D. Rao, "Performance of cell-free massive MIMO systems with MMSE and LSFD receivers," in *2016 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016, pp. 203–207.

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

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

[14] J. Benesty, J. Chen, and Y. Huang, *Microphone array signal processing*. Springer Science & Business Media, 2008, vol. 1.

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

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

[17] D. M. Sima, S. Van Huffel, and G. H. Golub, "Regularized total least squares based on quadratic eigenvalue problem solvers," *BIT Numerical Mathematics*, vol. 44, no. 4, pp. 793–812, 2004.

[18] A. Beck, A. Ben-Tal, and M. Teboulle, "Finding a global optimal solution for a quadratically constrained fractional quadratic problem with applications to the regularized total least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 2, pp. 425–445, 2006.

[19] H. Zhu, G. Leus, and G. B. Giannakis, "Sparse regularized total least squares for sensing applications," in *2010 IEEE 11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2010, pp. 1–5.

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

[21] C. Hovine and A. Bertrand, "A distributed adaptive algorithm for non-smooth spatial filtering problems in wireless sensor networks," *IEEE Transactions on Signal Processing*, pp. 1–16, 2024.

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

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

[24] ——, "Distributed adaptive trace ratio optimization in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653–3670, 2021.

[25] ——, "Distributed trace ratio optimization in fully-connected sensor networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1991–1995.

[26] S. Schaible and T. Ibaraki, "Fractional programming," *European journal of operational research*, vol. 12, no. 4, pp. 325–338, 1983.

[27] A. Charnes and W. W. Cooper, "Programming with linear fractional functionals," *Naval Research logistics quarterly*, vol. 9, no. 3-4, pp. 181–186, 1962.

[28] S. Schaible, "Parameter-free convex equivalent and dual programs of fractional programming problems," *Zeitschrift für Operations Research*, vol. 18, no. 5, pp. 187–196, 1974.

[29] W. Dinkelbach, "On nonlinear fractional programming," *Management science*, vol. 13, no. 7, pp. 492–498, 1967.

[30] R. Jagannathan, "On some properties of programming problems in parametric form pertaining to fractional programming," *Management Science*, vol. 12, no. 7, pp. 609–615, 1966.

[31] S. Schaible, "Fractional programming. ii, on dinkelbach's algorithm," *Management science*, vol. 22, no. 8, pp. 868–873, 1976.

[32] J. Crouzeix, J. Ferland, and S. Schaible, "An algorithm for generalized fractional programs," *Journal of Optimization Theory and Applications*, vol. 47, no. 1, pp. 35–49, 1985.

[33] J.-P. Crouzeix and J. A. Ferland, "Algorithms for generalized fractional programming," *Mathematical Programming*, vol. 52, no. 1, pp. 191–207, 1991.

[34] J. Crouzeix, J. Ferland, and S. Schaible, "A note on an algorithm for generalized fractional programs," *Journal of Optimization Theory and Applications*, vol. 50, no. 1, pp. 183–187, 1986.

[35] J.-P. Crouzeix, J. A. Ferland, and H. Van Nguyen, "Revisiting Dinkelbach-type algorithms for generalized fractional programs," *Opsearch*, vol. 45, no. 2, pp. 97–110, 2008.

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

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

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

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

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

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

[42] S. Siuly, Y. Li, and Y. Zhang, *EEG Signal Analysis and Classification*. Springer International Publishing, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-47653-7

[43] C. A. Musluoglu, M. Moonen, and A. 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.