Consensus-Based Distributed Total Least Squares Estimation in *Ad Hoc* Wireless Sensor Networks

Alexander Bertrand, *Student Member, IEEE*, and Marc Moonen, *Fellow, IEEE*

Abstract—Total least squares (TLS) is a popular solution technique for overdetermined systems of linear equations, where both the right-hand side and the input data matrix are assumed to be noisy. We consider a TLS problem in an *ad hoc* wireless sensor network, where each node collects observations that yield a node-specific subset of linear equations. The goal is to compute the TLS solution of the full set of equations in a distributed fashion, without gathering all these equations in a fusion center. To facilitate the use of the dual-based subgradient algorithm (DBSA), we transform the TLS problem to an equivalent convex semidefinite program (SDP), based on semidefinite relaxation (SDR). This allows us to derive a distributed TLS (D-TLS) algorithm, that satisfies the conditions for convergence of the DBSA, and obtains the same solution as the original (unrelaxed) TLS problem. Even though we make a detour through SDR and SDP theory, the resulting D-TLS algorithm relies on solving local TLS-like problems at each node, rather than computationally expensive SDP optimization techniques. The algorithm is flexible and fully distributed, i.e., it does not make any assumptions on the network topology and nodes only share data with their neighbors through local broadcasts. Due to the flexibility and the uniformity of the network, there is no single point of failure, which makes the algorithm robust to sensor failures. Monte Carlo simulation results are provided to demonstrate the effectiveness of the method.

Index Terms—Adaptive estimation, distributed estimation, wireless sensor networks (WSNs).

I. INTRODUCTION

AN *ad hoc* wireless sensor network (WSN) [1] consists of spatially distributed sensor nodes that share data with each other through wireless links to perform a certain task, e.g., the estimation of a signal or parameter. Generally, the goal is to implement an estimator that is equal (or close) to an optimal estimator that has access to all observations acquired by

all the nodes in the network. The latter corresponds to a centralized approach, where all observations are gathered in a fusion center to calculate an optimal estimate. A WSN, on the other hand, requires distributed algorithms that allow for in-network processing and only rely on local collaboration between neighboring nodes. It is likely that future data acquisition, control, and physical monitoring, will heavily rely on this type of networks.

Distributed estimation in a linear least squares (LLS) framework has been widely studied in the sensor network literature (see, e.g., [2]–[10]). The LLS framework is applied for linear regression problems, and provides a solution for an overdetermined system of linear equations, i.e., $\mathbf{U}\mathbf{w} = \mathbf{d}$ with $\mathbf{U}$ an $M \times P$ data matrix and $\mathbf{d}$ an M -dimensional data vector with $M \geq P$. The aim is then to find a vector $\mathbf{w}$ that minimizes the squared error between the left- and the right-hand side (LHS) (RHS), i.e.,

$$\mathbf{w}_{LLS} = \arg \min_{\mathbf{w}} \|\mathbf{U}\mathbf{w} - \mathbf{d}\|_2^2 \quad (1)$$

where $\|\cdot\|_2$ denotes the Euclidean norm (L^2 -norm). In a WSN, nodes can either have access to subsets of the columns of $\mathbf{U}$ [2]–[4], e.g., for distributed signal enhancement and beamforming [11], or to subsets of the rows of $\mathbf{U}$ (and $\mathbf{d}$) [5]–[10], e.g., for distributed system identification. Despite this common cost function, both problems are tackled in very different ways. In the former case, each node estimates a node-specific subvector of the global estimator $\mathbf{w}$, whereas in the latter case, each node estimates the full estimator $\mathbf{w}$, which allows for consensus-based approaches. Here, we focus on the latter case, i.e., where each node has access to a node-specific subset of the equations. Distributed LLS estimation amounts to computing the network-wide LLS solution in a distributed way, where the nodes have to reach a consensus on a common network-wide parameter vector $\mathbf{w}$. A motivation for this type of distributed estimation problems, different types of algorithms, and possible applications, can be found in [5]–[10] and references therein.

In a stochastic framework, (1) yields the best linear unbiased estimate (BLUE), if $\mathbf{d}$ is corrupted by additive zero-mean white noise [12]. However, in many practical situations, the input data matrix $\mathbf{U}$ is also noisy. For example, in adaptive filtering, this corresponds to the case where there is additive noise at both the input and output of the filter that is estimated. In [13], it is shown that the resulting LLS estimate (1) is biased in this case. A natural generalization of LLS estimation is total least squares (TLS) estimation, where both $\mathbf{U}$ and $\mathbf{d}$ are indeed assumed to be noisy (cf. [14] for an extensive overview). For the particular case of recursive TLS in adaptive filtering theory, we refer to [13], where it is shown that TLS yields an unbiased estimate if the additive noise at both the input and output of the filter is zero-mean and white.

Manuscript received July 15, 2010; revised November 14, 2010, January 11, 2011; accepted January 19, 2011. Date of publication January 28, 2011; date of current version April 13, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mathini Sellathurai. The work of A. Bertrand was supported by a Ph.D. grant of the I.W.T. (Flemish Institute for the Promotion of Innovation through Science and Technology). This work was carried out at the ESAT Laboratory of Katholieke Universiteit Leuven, in the frame of K.U. Leuven Research Council CoE EF/05/006 (OPTEC, ‘Optimization in Engineering’), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P6/04 (DYSCO, ‘Dynamical systems, control and optimization,’ 2007–2011), and Research Project FWO nr. G.0600.08 (‘Signal processing and network design for wireless acoustic sensor networks’). The scientific responsibility is assumed by its authors.

The authors are with the Department of Electrical Engineering (ESAT-SCD/SISTA), Katholieke Universiteit Leuven, B-3001 Leuven, Belgium (e-mail: alexander.bertrand@esat.kuleuven.be; marc.moonen@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSP.2011.2108651

In this paper, we propose an algorithm to compute the network-wide TLS solution in a distributed way, where the nodes of a WSN have to reach a consensus on a common network-wide parameter vector $\mathbf{w}$. As opposed to the LLS problem, the TLS problem is nonconvex, which makes the derivation of a distributed algorithm nontrivial. To facilitate the use of the dual based subgradient algorithm (DBSA) [15], we transform the TLS problem to an equivalent convex semidefinite program (SDP), based on a technique called semidefinite relaxation (SDR) [16]. This allows us to derive a consensus-based distributed TLS (D-TLS) algorithm, that satisfies the conditions for convergence of the DBSA. Even though we make a detour through SDR and SDP theory, the resulting D-TLS algorithm does not rely on computationally expensive SDP optimization techniques. Instead, we obtain an iterative algorithm that solves local TLS-like problems at each node, which enables the use of robust TLS solvers. Furthermore, even though we solve a relaxed optimization problem, the D-TLS algorithm still obtains the solution of the original TLS problem, which is available at each node (after convergence).

The D-TLS algorithm is flexible and fully distributed, i.e., nodes only share data with their neighbors through local broadcasts (single-hop communication) and the algorithm does not assume a specific network topology or a so-called Hamiltonian cycle (as often assumed in incremental strategies, e.g., [5]), which makes it robust to sensor and link failures. Due to the flexibility and the uniformity of the network (i.e., all nodes perform identical tasks), there is no single point of failure and the network is self-healing.

The outline of the paper is as follows. In Section II, we briefly review the TLS problem statement and we explain how the TLS solution can be computed. Furthermore, we describe the TLS problem in a distributed context for WSNs. In Section III, we review the dual based subgradient algorithm, which forms the backbone of the D-TLS algorithm. In Section IV, we derive the D-TLS algorithm, based on a SDR of the original TLS problem, and we address its convergence properties. In Section V, we provide simulation results. Conclusions are drawn in Section VI.

II. PROBLEM STATEMENT

A. The TLS Problem

We consider an overdetermined system of linear equations in P unknowns (stacked in a vector $\mathbf{w}$), given by

$$\mathbf{U}\mathbf{w} = \mathbf{d} \quad (2)$$

with $\mathbf{U}$ an $M \times P$ noisy data matrix and $\mathbf{d}$ an M -dimensional noisy data vector, with $M \geq P$. Since this is an overdetermined system of equations, its solution set is usually empty. The goal is then to find the TLS solution, i.e., to solve the constrained optimization problem

$$\min_{\mathbf{w}, \Delta \mathbf{U}, \Delta \mathbf{d}} \|\Delta \mathbf{U}\|_F + \|\Delta \mathbf{d}\|_2 \quad (3)$$

$$\text{s.t. } (\mathbf{U} + \Delta \mathbf{U})\mathbf{w} = (\mathbf{d} + \Delta \mathbf{d}) \quad (4)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. In Section V-A, we will demonstrate that the TLS estimate can be significantly better

than the LLS estimate in cases where the input data matrix $\mathbf{U}$ is noisy. In [13], it is shown that the TLS estimate provides an unbiased estimate when both $\mathbf{U}$ and $\mathbf{d}$ are contaminated with white noise (whereas the LLS solution is biased in this case). In order to compute the solution of (3)–(4), we define the matrix

$$\mathbf{U}_+ = [\mathbf{U} \ \mathbf{d}] \quad (5)$$

and the matrix

$$\mathbf{R} = \mathbf{U}_+^T \mathbf{U}_+. \quad (6)$$

Let $\mathbf{x}$ denote the eigenvector of $\mathbf{R}$ corresponding to the smallest eigenvalue, where $\mathbf{x}$ is scaled such that

$$\mathbf{x} = \begin{bmatrix} \mathbf{w}^* \\ -1 \end{bmatrix}. \quad (7)$$

In [17], it is shown that there exists a unique solution for the TLS problem (3)–(4) if the $(P+1)$ th singular value of $\mathbf{U}_+$ is strictly smaller than the P th singular value (where the singular values are sorted in decreasing order). In this case, the first P entries of $\mathbf{x}$, denoted by $\mathbf{w}^*$, are the solution of the TLS problem (3)–(4) [14], [17]. The eigenvector $\mathbf{x}$ can be computed in a robust and efficient way (e.g., by means of an eigenvalue or singular value decomposition [17]).

B. TLS in Ad Hoc Wireless Sensor Networks

Consider an *ad hoc* WSN with the set of nodes $\mathcal{J} = \{1, \dots, J\}$ and with a random (connected) topology, where neighboring nodes can exchange data through a wireless link. We denote the set of neighbor nodes of node k as $\mathcal{N}_k$, i.e., all the nodes that can share data with node k , node k excluded. Node k collects observations of a data matrix $\mathbf{U}_k$ and a data vector $\mathbf{d}_k$. The goal is then to solve a network-wide TLS problem, i.e., to compute a vector $\mathbf{w}$ from

$$\min_{\mathbf{w}, \Delta \mathbf{U}_1, \dots, \Delta \mathbf{U}_J, \Delta \mathbf{d}_1, \dots, \Delta \mathbf{d}_J} \sum_{k \in \mathcal{J}} (\|\Delta \mathbf{U}_k\|_F + \|\Delta \mathbf{d}_k\|_2) \quad (8)$$

$$\text{s.t. } (\mathbf{U}_k + \Delta \mathbf{U}_k)\mathbf{w} = (\mathbf{d}_k + \Delta \mathbf{d}_k), \quad k = 1, \dots, J. \quad (9)$$

We will refer to this problem as the D-TLS problem, since it corresponds to the TLS problem (3)–(4), where each node has access to a subset of the rows of $\mathbf{U}$ and a subvector of $\mathbf{d}$. The goal is to compute $\mathbf{w}$ in a distributed fashion, without gathering all data in a fusion center.

We will develop a consensus-based algorithm, based on dual decomposition of the optimization problem (8)–(9), which we will refer to as the D-TLS algorithm. The dual-based subgradient algorithm (DBSA) [15], which forms the backbone of the D-TLS algorithm, is described in its general form in Section III. Even though many of the required conditions for DBSA are violated in the case of D-TLS, we will explain in Section IV how DBSA can still be applied, without affecting its convergence results.

Remark I: In dynamic scenarios, an adaptive algorithm is required to track changes in the environment. In this case, extra observations become available over time at each node, and old observations may become invalid. This can be easily included in

the problem statement described above and in the D-TLS algorithm described in Section IV, by applying recursive updating and downdating schemes. However, for the sake of an easy exposition, we assume in the sequel that the matrices $\mathbf{U}_k$ do not change over time.

Remark II: The different nodes usually observe data at a different signal-to-noise ratio (SNR), depending on their position (e.g., the distance to a localized source). Therefore, it is often desirable to give a different weight to each term in (8), or even to each specific row of $\mathbf{U}_k$, depending on the local SNR. This is often referred to as generalized total least squares (GTLS) [14]. However, since the algorithmic treatment of GTLS and TLS are the same, and for the sake of an easy exposition, we do not consider GTLS in this paper.

III. DUAL BASED SUBGRADIENT ALGORITHM

In this section, we briefly review the Dual Based Subgradient Algorithm (DBSA), as applied in [18] on the following optimization problem¹ for a connected network with a set of nodes $\mathcal{J}$

$$\min_{\mathbf{x}} \sum_{k \in \mathcal{J}} f_k(\mathbf{x}) \quad (10)$$

$$\text{s.t. } \mathbf{x} \in \mathcal{X} \quad (11)$$

where $\mathcal{X}$ is a convex set with nonempty interior, $\mathbf{x}$ is an N -dimensional vector, and where $f_k, \forall k \in \mathcal{J}$, are convex functions. In this problem, it is assumed that f_k can only be evaluated at node k . The goal is then to solve (10)–(11) in a distributed fashion, where each node eventually obtains an optimal vector $\hat{\mathbf{x}}$ from the solution set of (10)–(11).

One possible way to solve the above problem is to apply dual composition. The main idea is to introduce local copies of $\mathbf{x}$ in each node, and to optimize and update these variables until a consensus is reached amongst all neighbor nodes as follows:

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_J} \sum_{k \in \mathcal{J}} f_k(\mathbf{x}_k) \quad (12)$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X}, \forall k \in \mathcal{J} \quad (13)$$

$$\mathbf{x}_k = \mathbf{x}_q, \forall k \in \mathcal{J}, \forall q \in \mathcal{N}_k, q < k \quad (14)$$

which has the same solution as (10)–(11), if the network is connected. The constraints (14) are denoted as consensus constraints. The condition $q < k$ removes redundancy in the constraints, such that there is a single consensus constraint for each individual link². We denote the set of network links $\mathcal{L} = \{1, \dots, L\}$, where each link contains a pair of nodes in $\mathcal{J}$. We introduce the $NL \times NJ$ linking matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \dots & \mathbf{C}_{1J} \\ \vdots & & \vdots \\ \mathbf{C}_{L1} & \dots & \mathbf{C}_{LJ} \end{bmatrix} \quad (15)$$

¹The problem described in [18] incorporates private variables, i.e., variables that are not common between the nodes, and assumes that the consensus variable is 1-dimensional. For later purpose, we extend the problem here for N -dimensional consensus variables. For the sake of an easy exposition, we omit the private variables.

²Note that there is still a lot of redundancy left, since many links can be removed without harming the overall consensus between the nodes.

where the $N \times N$ submatrix $\mathbf{C}_{lk}$ is defined as

$$\mathbf{C}_{lk} = \begin{cases} \mathbf{I}_N, & \text{if } l = \{k, q\} \text{ and } k < q \\ -\mathbf{I}_N, & \text{if } l = \{k, q\} \text{ and } q < k \\ \mathbf{O}_N, & \text{otherwise} \end{cases} \quad (16)$$

with $\mathbf{I}_N$ denoting the $N \times N$ identity matrix and $\mathbf{O}_N$ denoting an all-zero $N \times N$ matrix. With this, we can rewrite problem (12)–(14) as

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_J} \sum_{k \in \mathcal{J}} f_k(\mathbf{x}_k) \quad (17)$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X}, \forall k \in \mathcal{J} \quad (18)$$

$$\mathbf{C}\bar{\mathbf{x}} = \mathbf{0} \quad (19)$$

where $\bar{\mathbf{x}} = [\mathbf{x}_1^T \dots \mathbf{x}_J^T]^T$. This problem is not separable due to the consensus constraints (19). However, by applying dual decomposition [15], [19], the problem can be solved iteratively in a fully distributed fashion. Consider the dual function

$$d(\boldsymbol{\lambda}) = \min_{\bar{\mathbf{x}}} L(\boldsymbol{\lambda}, \bar{\mathbf{x}}) \quad (20)$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X}, \forall k \in \mathcal{J} \quad (21)$$

where $L(\boldsymbol{\lambda}, \bar{\mathbf{x}})$ denotes the so called Lagrangian defined by

$$L(\boldsymbol{\lambda}, \bar{\mathbf{x}}) = \sum_{k \in \mathcal{J}} f_k(\mathbf{x}_k) + \boldsymbol{\lambda}^T \mathbf{C}\bar{\mathbf{x}} \quad (22)$$

and where the variables in the (NL) -dimensional vector $\boldsymbol{\lambda}$ are so called Lagrange multipliers. We denote $\boldsymbol{\lambda}_l$ as the subvector of $\boldsymbol{\lambda}$ that is applied to the rows of $\mathbf{C}$ corresponding to submatrices $\mathbf{C}_{lk}, \forall k \in \mathcal{J}$, i.e., the Lagrange multipliers corresponding to the consensus constraints at link l . If strong duality holds [19], then the optimal value of (12)–(14) is the same as the optimal value of the dual problem

$$\max_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}). \quad (23)$$

Here, the unconstrained optimization problem (23) is referred to as the ‘master problem’, and the optimization problem in (20)–(21) is referred to as the ‘slave problem.’ An important observation is the fact that the slave problem is fully separable

$$d(\boldsymbol{\lambda}) = \sum_{k \in \mathcal{J}} d_k(\boldsymbol{\lambda}) \quad (24)$$

with

$$d_k(\boldsymbol{\lambda}) = \min_{\mathbf{x}_k} f_k(\mathbf{x}_k) + \sum_{l \in \mathcal{L}_k} \boldsymbol{\lambda}_l^T \mathbf{C}_{lk} \mathbf{x}_k \quad (25)$$

$$\text{s.t. } \mathbf{x}_k \in \mathcal{X} \quad (26)$$

where $\mathcal{L}_k$ denotes the set of links that contain node k . Since the dual function $d(\boldsymbol{\lambda})$ is not differentiable in general, the master problem is solved by a subgradient method. A subgradient of $d(\boldsymbol{\lambda})$ in the point $\boldsymbol{\lambda}$ is given by

$$\mathbf{g}(\boldsymbol{\lambda}) = \mathbf{C}\bar{\mathbf{x}}^*(\boldsymbol{\lambda}) \quad (27)$$

where $\bar{\mathbf{x}}^*(\boldsymbol{\lambda})$ denotes an $\bar{\mathbf{x}}$ that is in the solution set of the slave problem (20)–(21) for the given $\boldsymbol{\lambda}$ [15]. It is noted that, if $d(\boldsymbol{\lambda})$

is differentiable in λ , the subgradient $\mathbf{g}(\lambda)$ is equal to the actual gradient $\nabla d(\lambda)$.

We can now solve (12)–(14) with the following algorithm, which is known as the DBSA:

- 1) Let $i = 1$ and $\lambda^0 = \mathbf{0}_N$, where $\mathbf{0}_N$ denotes an all-zero N -dimensional vector.
- 2) Each node $k \in \mathcal{J}$ solves (25)–(26) to get $\mathbf{x}_k^i$.
- 3) Each node $k \in \mathcal{J}$ transmits $\mathbf{x}_k^i$ to the nodes in $\mathcal{N}_k$.
- 4) Each node $k \in \mathcal{J}$ updates the subvectors $\lambda_l^i, \forall l \in \mathcal{L}_k$, according to

$$\lambda_l^{i+1} = \lambda_l^i + \mu (\mathbf{C}_{lk} \mathbf{x}_k^i + \mathbf{C}_{lq} \mathbf{x}_q^i) \quad (28)$$

where q is the node that is connected to node k by link l , and with stepsize $\mu > 0$.

- 5) $i \leftarrow i + 1$.
- 6) return to step 2.

In [15], it is shown that, if the stepsize μ is sufficiently small, the distance of the $\mathbf{x}_k^i$'s to the optimal solution set is reduced in each iteration if the following conditions are satisfied:

- 1) The functions f_k are convex $\forall k \in \mathcal{J}$.
- 2) The set $\mathcal{X}$ is convex with nonempty interior.
- 3) Strong duality holds for (17)–(19).
- 4) All subgradients of $d(\lambda)$ are bounded for all values of λ , i.e., $\exists B \in \mathbb{R}, \forall \lambda : \|\mathbf{g}(\lambda)\|_2 \leq B$.

More specifically, the following theorems hold under the above assumptions [15].

Theorem III.1: If λ^i is not optimal, then for every dual optimal solution λ^* , we have

$$\|\lambda^{i+1} - \lambda^*\| < \|\lambda^i - \lambda^*\| \quad (29)$$

for all stepsizes μ such that

$$0 < \mu < \frac{2(d(\lambda^*) - d(\lambda^i))}{\|\mathbf{g}(\lambda^i)\|_2^2}. \quad (30)$$

Theorem III.2: For a fixed stepsize μ , there is at least asymptotic convergence to a neighborhood of $d(\lambda^*)$, i.e.

$$\limsup_{i \rightarrow \infty} d(\lambda^i) \geq d(\lambda^*) - \frac{\mu B^2}{2} \quad (31)$$

where B is defined in condition 4.

In other words, there is at least asymptotic convergence to a neighborhood of the optimal solution, where the size of this neighborhood shrinks with μ .

IV. D-TLS

In this section, we demonstrate how the DBSA algorithm can be applied to the D-TLS problem (8)–(9), even though the latter is a nonconvex optimization problem. As explained in Section II-A, the solution of the TLS problem (3)–(4) can be found by means of an eigenvalue or a singular value decomposition, i.e., the computation of the eigenvector of $\mathbf{R}$ corresponding to the smallest eigenvalue. We use $\mathbf{x}$ to denote this eigenvector, and we use N to denote the dimension of this vector, i.e., $N =$

$P + 1$. The solution of the TLS problem is then given by the first P entries of $\mathbf{x}$ when scaled such that the last (the $(P + 1)$ th) entry is -1 .

The eigenvector $\mathbf{x}$ corresponding to the smallest eigenvalue of $\mathbf{R}$ is the solution of the following quadratically constrained quadratic program³ (QCQP)

$$\min_{\mathbf{x}} \mathbf{x}^T \mathbf{R} \mathbf{x} \quad (32)$$

$$\text{s.t. } \|\mathbf{x}\|_2^2 = 1. \quad (33)$$

In the case of D-TLS, this becomes

$$\min_{\mathbf{x}} \sum_{k \in \mathcal{J}} \mathbf{x}^T \mathbf{R}_k \mathbf{x} \quad (34)$$

$$\text{s.t. } \|\mathbf{x}\|_2^2 = 1 \quad (35)$$

where $\mathbf{R}_k = \mathbf{U}_{k+}^T \mathbf{U}_{k+}$, with $\mathbf{U}_{k+} = [\mathbf{U}_k \mathbf{d}_k]$.

A. Transformation Into a Convex Problem

It is noted that (34)–(35) has the same form as (10)–(11) to which the DBSA was applied in Section III. However, we cannot straightforwardly apply the DBSA to (34)–(35) due to the norm constraint, which defines a nonconvex constraint set with empty interior. This violates condition 2 for application of the DBSA, which states that the set $\mathcal{X}$ must be convex with nonempty interior. Furthermore, condition 3 assumes strong duality after addition of the consensus constraints, which is not guaranteed for (34)–(35), again due to the nonconvex norm constraint. However, since (34)–(35) is a QCQP, we can apply SDR to transform the original problem into a convex optimization problem, which has the solution of the original QCQP in its solution set (see [16] for an overview article). Since the new problem is convex, DBSA can be readily applied, and its convergence results will then also hold for the derived algorithm. Remarkably, it will turn out that the SDR yields an algorithm that solves local TLS problems at each node⁴ (see Section IV-B), which enables the use of robust TLS solvers. Furthermore, even though we solve a relaxed problem, we eventually obtain the solution of the original D-TLS problem (34)–(35).

SDR is based on the observation that $\mathbf{x}^T \mathbf{R}_k \mathbf{x} = \text{tr}(\mathbf{x}^T \mathbf{R}_k \mathbf{x}) = \text{tr}(\mathbf{R}_k \mathbf{x} \mathbf{x}^T)$, where $\text{tr}(\mathbf{Q})$ denotes the trace of the matrix $\mathbf{Q}$. By applying the substitution $\mathbf{X} = \mathbf{x} \mathbf{x}^T$, we transform (34)–(35) into the equivalent problem

$$\min_{\mathbf{X}} \sum_{k \in \mathcal{J}} \text{tr}(\mathbf{R}_k \mathbf{X}) \quad (36)$$

$$\text{s.t. } \text{tr}(\mathbf{X}) = 1 \quad (37)$$

$$\mathbf{X} \succeq 0 \quad (38)$$

$$\text{rank}(\mathbf{X}) = 1 \quad (39)$$

³The stationary points of the Lagrangian of (32)–(33) are the eigenvectors of $\mathbf{R}$. Therefore, the eigenvector corresponding to the smallest eigenvalue is the global minimum of (32)–(33).

⁴If DBSA would be directly applied to the original QCQP (34)–(35), this would result in a different distributed algorithm that is not guaranteed to converge, and it cannot rely on robust TLS solvers, since each local cost function will then have linear terms due to the consensus constraints.

where $\mathbf{X} \succeq 0$ is used to denote that $\mathbf{X}$ is symmetric and positive (semi)definite. It is noted that the rank constraint (39) is the only nonconvex constraint. The SDR approach relaxes (36)–(39) to a convex optimization problem by removing this rank constraint, i.e.

$$\min_{\mathbf{X}} \sum_{k \in \mathcal{J}} \text{tr}(\mathbf{R}_k \mathbf{X}) \quad (40)$$

$$\text{s.t. } \text{tr}(\mathbf{X}) = 1 \quad (41)$$

$$\mathbf{X} \succeq 0. \quad (42)$$

This type of problem is known as a SDP, which is studied extensively in [19] and [20]. Obviously, the SDP resulting from the SDR usually yields a new problem which is not equivalent to the original QCQP. However, there is a known upper bound on the rank of the matrix $\mathbf{X}^*$ with lowest rank that is in the solution set of any feasible SDP with an $N \times N$ matrix variable and m linear constraints [16]

$$\text{rank}(\mathbf{X}^*) \leq \frac{\sqrt{8m+1} - 1}{2}. \quad (43)$$

Furthermore, this matrix $\mathbf{X}^*$ can easily be found [21]. It is noted that if $m = 1$, as it is the case for (40)–(42), the rank of $\mathbf{X}^*$ reduces to 1. This is a very important observation, since it implies that the optimal solution of (36)–(39), and consequently of the D-TLS problem (34)–(35), can be found by solving the relaxed (convex) problem (40)–(42).

We can now apply the DBSA to the problem (40)–(42). Condition 1 for convergence of the DBSA is straightforwardly satisfied. Condition 4 is also satisfied due to the boundedness of the constraint set. To satisfy condition 2, the constraint set needs to be convex with a nonempty interior, but the latter is not satisfied in (40)–(42). However, if we would replace the constraint (41) with $1 + \epsilon \geq \text{tr}(\mathbf{X}) \geq 1$ where $\epsilon > 0$, the solution set of (40)–(42) remains the same (due to the minimization), but the constraint set has now a nonempty interior (and it remains bounded and convex). Therefore, condition 2 is also satisfied⁵. Condition 3 is satisfied due to the following theorem⁶ from [22]:

Theorem IV.1: If the solution set of an SDP is nonempty and bounded, then strong duality holds.

This theorem holds for the SDP (40)–(42) (also after including the consensus constraints), because there is at least one solution (corresponding to the solution of (34)–(35)) and boundedness is guaranteed due to the fact that the constraint set is bounded.

B. The D-TLS Algorithm

In the previous section, we have explained how the Distributed Total Least Squares Algorithm (D-TLS) problem can be transformed into an SDP which has the same solution,

⁵The resulting DTLS algorithm as described in subSection IV-B will be exactly the same, whether we use the constraint (41), or the constraint $1 + \epsilon \geq \text{tr}(\mathbf{X}) \geq 1$, even though only the latter satisfies condition 2. Therefore, we keep the constraint (40) in the sequel without affecting the convergence results of DBSA.

⁶It is noted that strong duality still holds if (41) is replaced with $1 + \epsilon \geq \text{tr}(\mathbf{X}) \geq 1$, due to convexity of the problem, and the fact that Slater's condition holds.

and satisfies the conditions for convergence of the DBSA. In this section, we derive the formulas that describe the D-TLS algorithm at each node, based on the algorithm description of the DBSA in Section III.

The SDP (40)–(42) yields the following node-specific DBSA slave problems (to be compared to the general formulation (25)–(26)):

$$d_k(\Lambda) = \min_{\mathbf{X}_k} \text{tr}(\mathbf{R}_k \mathbf{X}_k) + \sum_{l \in \mathcal{L}_k} c_{lk} \text{tr}(\Lambda_l^T \mathbf{X}_k) \quad (44)$$

$$\text{s.t. } \text{tr}(\mathbf{X}_k) = 1 \quad (45)$$

$$\mathbf{X}_k \succeq 0 \quad (46)$$

where c_{lk} is defined as the scalar version of (16), i.e., $c_{lk} = (1/N)\text{tr}(\mathbf{C}_{lk})$. The Λ_l 's are $N \times N$ matrices that contain the Lagrange multipliers corresponding to the consensus constraints $\mathbf{X}_k = \mathbf{X}_q, \forall k \in \mathcal{J}, \forall q \in \mathcal{N}_k$.

The slave problem (44)–(46) is solved by each node $k \in \mathcal{J}$ (step 2 in the DBSA algorithm), followed by an exchange of these solutions between neighboring nodes. After receiving its neighbors' solutions, the Lagrange multipliers at node k are updated as follows (step 4 in the DBSA algorithm):

$$\forall l \in \mathcal{L}_k : \Lambda_l^{i+1} = \Lambda_l^i + \mu (c_{lk} \mathbf{X}_k^i + c_{lq} \mathbf{X}_q^i) \quad (47)$$

where q is the node that is connected to node k by link l . It is noted that the nodes communicate $N \times N$ matrices $\mathbf{X}_k^i$, which is not very efficient. Eventually, we are interested in the N -dimensional vector $\mathbf{x}$ that solves the original QCQP (34)–(35) corresponding to the distributed TLS problem (8)–(9). However, based on SDR theory, the problem (44)–(46) must have a rank-1 solution since it is the SDR of the QCQP⁷

$$d_k(\Lambda) = \min_{\mathbf{x}_k} \mathbf{x}_k^T \left(\mathbf{R}_k + \sum_{l \in \mathcal{L}_k} c_{lk} \Lambda_l^i \right) \mathbf{x}_k \quad (48)$$

$$\text{s.t. } \|\mathbf{x}_k\|_2^2 = 1 \quad (49)$$

where we used the substitution $\mathbf{X}_k = \mathbf{x}_k \mathbf{x}_k^T$. We can then solve this QCQP instead of (44)–(46), to obtain the rank-1 solution. Conveniently, the Lagrange multipliers now appear in the quadratic term of the cost function, rather than in a separate linear term as it is the case in (25). Therefore, the above constrained optimization problem again corresponds to computing the eigenvector corresponding to the smallest eigenvalue⁸ of a matrix, i.e., $(\mathbf{R}_k + \sum_{l \in \mathcal{L}_k} c_{lk} \Lambda_l^i)$. This also implies that

$$d_k(\Lambda) = \lambda_{\min} \left(\mathbf{R}_k + \sum_{l \in \mathcal{L}_k} c_{lk} \Lambda_l^i \right) \quad (50)$$

where $\lambda_{\min}(\mathbf{Q})$ denotes the smallest eigenvalue of $\mathbf{Q}$. The nodes now only share N -dimensional vectors with their neighbors. This yields the following algorithm, which we refer to as the D-TLS algorithm.

⁷Note that, since the $\mathbf{X}_k^i$'s are solutions of an SDP, they are symmetric. Therefore the matrices Λ_l^i are also symmetric due to (47), assuming that they are initialized with a symmetric matrix.

⁸It is noted that this eigenvalue can be negative.

The D-TLS Algorithm

- 1) Let $i = 1$ and $\Lambda_l^0 = \mathbf{O}_{N \times N}, \forall l \in \mathcal{L}$.
- 2) Each node $k \in \mathcal{J}$ computes the eigenvector $\mathbf{x}_k^i$ corresponding to the smallest eigenvalue of $\bar{\mathbf{R}}_k^i$ defined by

$$\bar{\mathbf{R}}_k^i = \mathbf{R}_k + \sum_{l \in \mathcal{L}_k} c_{lk} \Lambda_l^i \quad (51)$$

where $\mathbf{x}_k^i$ is scaled such that $\|\mathbf{x}_k^i\|_2 = 1$.

- 3) Each node $k \in \mathcal{J}$ transmits $\mathbf{x}_k^i$ to the nodes in $\mathcal{N}_k$.
- 4) Each node $k \in \mathcal{J}$ updates the Lagrange multipliers $\Lambda_l^i, \forall l \in \mathcal{L}_k$ according to

$$\Lambda_l^{i+1} = \Lambda_l^i + \mu (c_{lk} \mathbf{x}_k^i \mathbf{x}_k^{iT} + c_{lq} \mathbf{x}_q^i \mathbf{x}_q^{iT}) \quad (52)$$

where q is the node that is connected to node k by link l , and with stepsize $\mu > 0$.

- 5) $i \leftarrow i + 1$.
 - 6) return to step 2.
-

In each iteration of the D-TLS algorithm, each node computes the eigenvector corresponding to the smallest eigenvalue of a local symmetric matrix. Hence, although we took a detour through SDR and SDP theory, the resulting D-TLS algorithm eventually can still rely on robust TLS computations, i.e., an eigenvalue decomposition, and there is no need for computationally expensive iterative interior-point algorithms, as mostly used to solve SDP's. Furthermore, if $N \gg |\mathcal{N}_k|$, the $\mathbf{x}_k^{i+1}$ can be efficiently computed from the previous solution $\mathbf{x}_k^i$ by exploiting the rank-1 updates of the Λ_l^i 's [23], [24]. Similar procedures can be used to update (or downgrade) the $\mathbf{R}_k$ matrices when new observations are included or when old observations are removed (e.g., in adaptive scenarios).

Remark I: The multiplier update (52) requires that each node has a unique index to determine the c_{lk} 's, which requires some extra coordination. Furthermore, each node stores a separate $N \times N$ multiplier matrix for each different neighbor. We can eliminate this need by using the substitution

$$\Theta_k^i = \sum_{l \in \mathcal{L}_k} c_{lk} \Lambda_l^i \quad (53)$$

which corresponds to a node-specific variable for node k . Since $c_{lk}^2 = 1$ and $c_{lk}c_{lq} = -1$ for link l that connects nodes k and q , we readily find from (52) and (53) that

$$\Theta_k^{i+1} = \Theta_k^i + \mu \left(|\mathcal{N}_k| \mathbf{x}_k^i \mathbf{x}_k^{iT} - \sum_{q \in \mathcal{N}_k} \mathbf{x}_q^i \mathbf{x}_q^{iT} \right) \quad (54)$$

where we use $|\mathcal{S}|$ to denote the cardinality of the set $\mathcal{S}$. It is noted that each node now only has to store a single multiplier matrix (instead of multiple Λ_l 's, i.e., one for each link). This yields a simplified version of the D-TLS algorithm where the multiplier update (52) is replaced with (54) where $\Theta_k^0 = \mathbf{O}_{N \times N}, \forall k \in \mathcal{J}$, and where $\bar{\mathbf{R}}_k^i$ is redefined as $\bar{\mathbf{R}}_k^i = \mathbf{R}_k + \Theta_k^i$.

Remark II: It is noted that D-TLS is an adaptive algorithm where each node performs a similar task. This uniformity and adaptivity guarantees that there is no single point of failure. This means that the algorithm is self-healing in case of permanent link or sensor failures, assuming that the network graph remains connected. If link l between nodes k and q is permanently removed, both nodes need to remove Λ_l from (51), and the network will still converge to the solution of (34)–(35). If node k fails permanently, its neighbors must remove the Λ_l from (51), $\forall l \in \mathcal{L} : k \in l$. The network will then adapt to find the new solution, i.e., the solution of (34)–(35) with the k th term removed from (34). It is noted that the simplified version of the D-TLS algorithm (see Remark I) does not have this self-healing property, since the Λ_l 's are not stored separately⁹. However, both versions of the algorithm can handle temporary link failures, as long as none of the links or nodes are permanently removed. We will demonstrate in Section V-G that the algorithm still performs well in random graphs, where the links between nodes fail in each iteration with a certain probability $p < 1$.

Remark III: We have transformed the original QCQP (34)–(35) to another problem (44)–(46) that has the original solution in its solution set and for which strong duality holds. However, this does not necessarily imply that strong duality also holds for (34)–(35). Nevertheless, strong duality of the relaxed problem (44)–(46) is sufficient for convergence of the derived D-TLS algorithm to the optimal solution of (44)–(46). This is because the D-TLS algorithm is essentially DBSA applied¹⁰ to (44)–(46), and not to (34)–(35). However, since we select the rank-1 solution out of the solution set of (44)–(46), we eventually obtain the solution of the original D-TLS problem (34)–(35) or (8)–(9). It is noted that, when DBSA is applied to the original QCQP (34)–(35), this would result in a different algorithm, which will probably not converge due to a nonzero duality gap. Furthermore, this algorithm cannot rely on TLS solvers, since linear terms will appear in the local cost functions due to the linear consensus constraints.

C. Convergence

The convergence of the D-TLS algorithm follows straightforwardly from the convergence of the DBSA. For a fixed stepsize μ , we know from Theorem III.2 that there is at least asymptotic convergence to a neighborhood of the optimal solution, where the size of this neighborhood shrinks with μ . Furthermore, from Theorem III.1, we know that each new iteration of D-TLS gets closer to the optimal solution if μ satisfies (30). In particular, for the case of D-TLS, we find that [to be compared with (30)]

$$d(\lambda^*) = \lambda_{\min} \left(\sum_{k \in \mathcal{J}} \mathbf{R}_k \right) \quad (55)$$

$$d(\lambda^i) = \sum_{k \in \mathcal{J}} \lambda_{\min} (\bar{\mathbf{R}}_k^i) \quad (56)$$

$$\|\mathbf{g}(\lambda^i)\|_2^2 = \sum_{k \in \mathcal{J}} \sum_{q \in \mathcal{N}_k, q < k} \|\mathbf{x}_k^i - \mathbf{x}_q^i\|_F^2 \quad (57)$$

⁹Note that the sum of all Θ_k 's must be zero at all time.

¹⁰Even though D-TLS eventually solves a QP at each node, the solution of this QP is also a solution of (44)–(46) (i.e., the rank-1 solution), hence we implicitly solve (44)–(46).

where $\mathbf{X}_k^i = \mathbf{x}_k^i \mathbf{x}_k^{i,T}$. Here, (55) follows from strong duality, and (56) follows from (50) and (51). Expression (57) follows from (27) and the fact that D-TLS implicitly solves an SDP with substitution $\mathbf{X}_k^i = \mathbf{x}_k^i \mathbf{x}_k^{i,T}$. Using the fact that $\mathbf{X}_k^i = \mathbf{x}_k^i \mathbf{x}_k^{i,T}$ and $\|\mathbf{x}_k^i\|_2 = 1, \forall k \in \mathcal{J}$, we obtain

$$\begin{aligned} \|\mathbf{X}_k^i - \mathbf{X}_q^i\|_F^2 &= \text{tr} \left((\mathbf{X}_k^i - \mathbf{X}_q^i)^2 \right) \\ &= \text{tr} (\mathbf{X}_k^{i,2}) + \text{tr} (\mathbf{X}_q^{i,2}) - 2\text{tr} (\mathbf{X}_k^i \mathbf{X}_q^i) \\ &= \|\mathbf{x}_k^i\|_2^4 + \|\mathbf{x}_q^i\|_2^4 - 2(\mathbf{x}_k^i{}^T \mathbf{x}_q^i)^2 \\ &= 2 - 2(\mathbf{x}_k^i{}^T \mathbf{x}_q^i)^2. \end{aligned} \quad (58)$$

Based on Theorem III.1, this results in the following bound for the stepsize:

$$0 < \mu^i < \frac{\lambda_{\min}(\sum_{k \in \mathcal{J}} \mathbf{R}_k) - \sum_{k \in \mathcal{J}} \lambda_{\min}(\bar{\mathbf{R}}_k^i)}{|\mathcal{L}| - \sum_{k \in \mathcal{J}} \sum_{q \in \mathcal{N}_k, q < k} (\mathbf{x}_k^i{}^T \mathbf{x}_q^i)^2}. \quad (59)$$

Here, the numerator is equal to the difference between the current value of the dual function and its optimal value, which reduces to zero if and only if all $\mathbf{x}_k$ are equal to the solution of (34)–(35). The denominator (57) is the squared consensus error summed over all the links of the network. It is noted that the denominator heavily depends on the number of links $|\mathcal{L}|$, i.e., strongly connected networks require a smaller μ . However, this does not necessarily result in slower convergence, since information diffuses much faster over the network if it is strongly connected (simulations in Section V-C confirm this). The numerator, on the other hand, mainly depends on the number of nodes, i.e., $\lambda_{\min}(\sum_{k \in \mathcal{J}} \mathbf{R}_k)$ increases when $|\mathcal{J}|$ increases (this follows from the fact that all $\mathbf{R}_k$'s are positive definite).

D. Choice of Stepsize μ

A small stepsize μ yields a more accurate estimate of the TLS solution, but generally yields slow convergence. Therefore, it is desirable to adapt μ in each iteration so that it is close to the upper bound (59). This is often not possible in practice since the nodes usually do not have access to most of the variables in (59). However, the second term of both the numerator and the denominator are summations over variables that are locally available in different nodes of the network (in fact, the computation of $|\mathcal{L}|$ can also be viewed as the sum of locally available variables, i.e., $|\mathcal{L}| = (1/2) \sum_{k \in \mathcal{J}} |\mathcal{N}_k|$). Therefore, these summations can be iteratively computed by so called ‘consensus averaging’ algorithms (see e.g., [25]). These procedures compute the average (or sum) of local observations in an iterative fashion, and this average is then eventually known to each node. If N is large, this will not significantly increase the required communication bandwidth that is used by D-TLS. Alternative fast gossip type algorithms for computing the sum of local quantities can be found in [26]. The value $\lambda_{\min}(\sum_{k \in \mathcal{J}} \mathbf{R}_k)$ cannot be computed, and therefore an estimate should be known *a priori* (or at least a tight lower bound should be known).

It is noted that the computation of (59) becomes less robust when the algorithm closely approximates the optimal solution, since both the numerator and the denominator then become very

small. In this case, there will be large fluctuations in the step-sizes of subsequent iterations, and extremely large values may be obtained when the denominator approaches zero. Therefore, it is better to use (59) in combination with an *a priori* fixed upper bound, to avoid instability. An alternative approach is to only use an upper bound for the denominator instead of the full expression, i.e.

$$0 < \mu^i < \frac{\lambda_{\min}(\sum_{k \in \mathcal{J}} \mathbf{R}_k) - \sum_{k \in \mathcal{J}} \lambda_{\min}(\bar{\mathbf{R}}_k^i)}{|\mathcal{L}|} \quad (60)$$

or the less conservative bound

$$0 < \mu^i < \frac{\lambda_{\min}(\sum_{k \in \mathcal{J}} \mathbf{R}_k) - \sum_{k \in \mathcal{J}} \lambda_{\min}(\bar{\mathbf{R}}_k^i)}{\epsilon + |\mathcal{L}| - \sum_{k \in \mathcal{J}} \sum_{q \in \mathcal{N}_k, q < k} (\mathbf{x}_k^i{}^T \mathbf{x}_q^i)^2} \quad (61)$$

with $\epsilon > 0$.

If there is no knowledge available on any of the variables in (59), convergence to the optimal solution can be guaranteed when a variable stepsize is used that satisfies [15]

$$\sum_{i=1}^{\infty} \mu^i = \infty \quad (62)$$

$$\sum_{i=1}^{\infty} (\mu^i)^2 < \infty. \quad (63)$$

A possible (but conservative) choice is $\mu^i = 1/i$. However, in tracking applications, (62)–(63) cannot be used, and then a fixed stepsize is a better alternative. The latter requires some parameter tuning, and it introduces a tradeoff between the speed of convergence (or adaptation speed) and the accuracy of the final solution, as given by Theorem III.2.

V. SIMULATIONS

In this section, we provide numerical simulation results that demonstrate the convergence properties of the D-TLS algorithm. To illustrate the general behavior, we show results that are averaged over multiple Monte Carlo (MC) runs. In each MC run, the following process is used to generate the network and the sensor observations (unless stated otherwise):

- 1) Construct a P -dimensional vector $\mathbf{w}$ where the entries are drawn from a zero-mean normal distribution with unit variance.
- 2) Create a random¹¹ connected network with J nodes, with 3 neighbors per node (on average).
- 3) For each node $k \in \mathcal{J}$:
 - Construct a $2P \times P$ input data matrix $\mathbf{U}_k$ where the entries are drawn from a zero-mean normal distribution with unit variance. The matrix $\mathbf{U}_k$ is then scaled by a random factor drawn from a uniform distribution on the unit interval (this models the different observation SNR at each node).
 - Compute $\mathbf{d}_k = \mathbf{U}_k \mathbf{w}$.

¹¹Unless stated otherwise, we start from a random tree to guarantee that the network is connected. Links are then randomly added until the average number of links per node equals 3.

 Fig. 1. Comparison of different techniques for the estimation of $\mathbf{w}$.

- Add zero-mean white Gaussian noise to the entries in $\mathbf{U}_k$ and $\mathbf{d}_k$, with a standard deviation of 0.5.

In each experiment, we choose $J = 20$, $N = P + 1 = 10$, and we run the D-TLS algorithm for 400 iterations (unless stated otherwise). It is noted that both algorithms (i.e., D-TLS and its simplified version, as described in Remark I in Section IV-B) are exactly equivalent in each of the experiments in the sequel, except for the experiment in Section V-G.

To assess the convergence and optimality of the algorithm, we use the error between the centralized TLS solution and the local estimate, averaged over the J nodes in the network

$$\frac{1}{|J|} \sum_{k \in J} \|\mathbf{x}_k - \mathbf{x}_{\text{TLS}}\|_2 \quad (64)$$

where $\mathbf{x}_{\text{TLS}}$ is the solution of (34)–(35).

A. TLS Versus LLS

To motivate the use of D-TLS, we first compare different techniques to estimate $\mathbf{w}$. The results are given in Fig. 1, showing the exact entries of the 9-dimensional vector $\mathbf{w}$, together with the following estimates:

- The centralized TLS solution.
- The D-TLS solution at node 1 (with fixed stepsize $\mu = 1$).
- The local TLS solution at node 1, without sharing any data with neighboring nodes.
- The centralized LLS solution.

Fig. 1 demonstrates that the TLS procedure indeed provides a significantly better estimate than the LLS procedure, since the latter ignores the fact that the input data matrix is noisy. Furthermore, it is demonstrated that the solution of the D-TLS algorithm is very close the centralized TLS solution. A last observation is that nodes indeed benefit from sharing their data among each other, since only using the data of node 1 yields a very poor TLS estimate.

B. Influence of Stepsize μ

For the convergence properties of the D-TLS algorithm, we compare the following strategies to choose the stepsize μ :

- Strategy 1: a fixed stepsize $\mu = 1$.
- Strategy 2: an adaptive stepsize, based on upper bound (59), but clipped to always stay smaller than 5.
- Strategy 3: a more conservative adaptive stepsize, based on upper bound (60).

 Fig. 2. Convergence properties of D-TLS for different strategies in choosing μ , averaged over 1000 MC runs.

 Fig. 3. Convergence properties of D-TLS for different values of the fixed stepsize μ , averaged over 1000 MC runs.

For each strategy, 1000 MC runs are performed, and the mean values (over all MC runs) of the error curves are shown in Fig. 2. The gray-colored areas cover one standard deviation of the error curves over all MC runs (on both sides of the mean curve). It is observed that the stepsize based on (60) (strategy 3) has a fast initial convergence, but becomes extremely slow when it gets close to the optimal solution. This can be explained by the fact that the denominator is fixed in (60), and becomes very large compared to the numerator when reaching the optimal solution, yielding an extremely conservative stepsize μ . The fixed stepsize and the adaptive stepsize based on (59) (strategies 1 and 2), have a similar convergence speed.

In the next experiment, we use different values for the fixed stepsize. The results are shown in Fig. 3 (the standard deviation of the MC runs is given for $\mu = 1$ and $\mu = 2.5$). Fig. 3 shows that $\mu = 1$ was actually a lucky guess. Indeed, the convergence of the D-TLS algorithm heavily depends on the stepsize. The stepsizes that are smaller than 1 all yield slower convergence. For the stepsizes larger than 1 ($\mu = 2.5$ and $\mu = 5$), convergence becomes a vague concept, due to the large excess error, i.e., the $\mathbf{x}_k^i$'s vary significantly over the different iterations (this causes the large standard deviation in the experiments with $\mu = 2.5$). The adaptive stepsize based on (59) seems to provide a good convergence speed when prior tuning of the stepsize is not possible, as it is almost identical to the $\mu = 1$ experiment (compare with Fig. 2).

Fig. 4. Convergence properties of D-TLS for different degrees of connectivity, averaged over 200 MC runs.

C. Influence of Connectivity of the Network Graph

In this experiment, we investigate the influence of the connectivity of the network, where the number of nodes is fixed to $J = 20$. Two extremes are of interest: a network with a ring topology (i.e., each node has 2 neighbors) and a fully connected network (i.e., each node has $J - 1$ neighbors). We also investigate networks in between those extremes, by adding links between random node pairs. In particular, we simulated (random) networks with $|\mathcal{L}| \in \{20, 25, \dots, 40, 50, \dots, 100, 190\}$.

200 MC runs are performed for each type of network (the links are chosen differently in each run). The stepsize is fixed, but depends on the number of links, i.e., $\mu = (10)/(|\mathcal{L}|)$, inspired by expression (60), as the denominator increases linearly with the number of links. The results are shown in Fig. 4. Not surprisingly, it is observed that increasing the number of links increases the convergence speed, even though a smaller μ is used. However, this effect becomes less significant for large $|\mathcal{L}|$. In the case of a ring topology, the convergence speed can be greatly improved by only adding a few extra links.

D. Influence of Dimension N

In this experiment, we increase the dimension N of the vector $\mathbf{x}$. When N is large, the ratio between the smallest and second smallest eigenvalue of the matrix $\mathbf{R}_k$ might be close to one, especially at nodes with very low SNR. This makes the eigenvalue problem in step 2 of the D-TLS algorithm ill-conditioned, which may affect the stability or the convergence time of the D-TLS algorithm¹². Therefore, the input data matrix $\mathbf{U}$ is not scaled with a random variable in this experiment, to avoid nodes with very low SNR.

In particular, we simulated 100 MC runs for each value of $N \in \{10, 20, \dots, 100\}$. The stepsize is fixed, but depends on the dimension N , i.e., $\mu = (N)/(10)$, inspired by expression (60), as the numerator increases linearly with N . The results¹³ are shown in Fig. 5. It is observed that the value of N significantly influences the convergence speed of the algorithm.

¹²In practice, low SNR nodes should therefore be removed from the network when using D-TLS for high-dimensional regression problems.

¹³The reason why the variance over the different MC runs is smaller than in previous experiments, is the fact that the SNR is equal in every node and in every MC run.

Fig. 5. Convergence properties of D-TLS for different dimensions N , averaged over 100 MC runs.

Fig. 6. Convergence properties of D-TLS for different sizes of the network, averaged over 100 MC runs.

E. Influence of Size of the Network

In this experiment, we increase the number of nodes J , but the average links per node remains fixed to 3. We simulated 100 MC runs for a network with $J \in \{20, 100, 700\}$ nodes, and with the stepsize fixed to $\mu = 1$. The result is shown in Fig. 6. It is observed that the size of the network has almost no influence on the convergence speed.

F. Random Graphs

In this experiment, each link can fail in each iteration, with a certain probability p . This models packet loss in the communication between nodes. Basically, this means that $\mathcal{N}_k$ in (54) changes with the iteration index i . We simulated 200 MC runs for each value of $p \in \{0, 0.1, 0.2, \dots, 0.9\}$. The results are shown in Fig. 7 for a fixed stepsize $\mu = 1$. It is observed that the algorithm still performs pretty well under significant packet loss. However, high packet loss significantly decreases convergence speed, especially when close to the optimal solution.

G. Self-Healing Property

In this experiment, we demonstrate the self-healing property of the D-TLS algorithm, i.e., its capability to adapt to permanent changes in the network topology. Notice that this is different from the previous experiment with random graphs, where each link is active in an infinite number of iterations. Here, we

Fig. 7. Convergence properties of D-TLS for random network graphs with different link failure probabilities, averaged over 200 MC runs.

Fig. 8. Self-healing property of the D-TLS algorithms after removal of nodes.

demonstrate that the algorithm can recover the optimal TLS solution when nodes are permanently removed from the network. It is noted that the simplified algorithm, as described in Remark I at the end of Section IV.B, does not have this self-healing property.

In the experiment, 2 random nodes are permanently removed¹⁴ after 800 iterations, and another 2 after 1600 iterations. When these nodes are removed, the topology of the network changes significantly since many links disappear, and also the TLS solution changes since two terms in the cost function (34) are removed. The Λ_l 's corresponding to the disappearing links l are removed from the expressions (51) and (52). The result is shown in Fig. 8 for a fixed stepsize $\mu = 1$, and a single run. After every 800 iterations, the error increases significantly, but the algorithm swiftly recovers from the changes in the network.

VI. CONCLUSION

In this paper, we have considered the TLS problem in an *ad hoc* wireless sensor network, where each node collects observations that yield a node-specific subset of linear equations. We have derived a D-TLS algorithm that computes the centralized TLS solution of the full set of equations in a distributed fashion, without gathering the data in a fusion center. To facilitate the use of the DBSA, we have transformed the TLS problem to an equivalent convex SDP that satisfies the convergence conditions of DBSA, and yields the same solution as in the original problem. Even

¹⁴Nodes can only be removed if the network graph remains connected after the removal.

though we have made a detour through SDR and SDP theory, the resulting D-TLS algorithm relies on solving local TLS-like problems at each node, rather than on computationally expensive SDP optimization techniques. The algorithm is flexible and fully distributed, i.e., it does not make any assumptions on the network topology and nodes only share data with their direct neighbors through local broadcasts. Due to the flexibility and the uniformity of the network, there is no single point of failure, which makes the algorithm robust to sensor failures. We have provided MC simulation that demonstrate the effectiveness of the algorithm.

ACKNOWLEDGMENT

The authors would like to thank Prof. M. Diehl, Dr. P. Tsiakakis, and the OPTEC and DSP co-workers at the E.E. Department of K.U. Leuven for the interesting discussions with respect to the distributed TLS problem, during the GOA-MaNet seminar. The authors also want to thank the anonymous reviewers, whose comments significantly improved this manuscript.

REFERENCES

- [1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'01)*, 2001, vol. 4, pp. 2033–2036, vol. 4.
- [2] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks—Part I: Sequential node updating," *IEEE Trans. Signal Process.*, vol. 58, pp. 5277–5291, Oct. 2010.
- [3] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks—Part II: Simultaneous and asynchronous node updating," *IEEE Trans. Signal Process.*, vol. 58, pp. 5292–5306, Oct. 2010.
- [4] A. Bertrand and M. Moonen, "Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology," *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 2196–2210, May 2011.
- [5] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, pp. 4064–4077, Aug. 2007.
- [6] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, pp. 3122–3136, July 2008.
- [7] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, pp. 1865–1877, May 2008.
- [8] G. Mateos, I. Schizas, and G. Giannakis, "Closed-form MSE performance of the distributed LMS algorithm," in *Proc. IEEE 13th Digit. Signal Process. Workshop (DSP/SPE)*, 2009, pp. 66–71, 4–7.
- [9] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2009, p. 19, 2009, Article ID 981030.
- [10] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4583–4588, 2009.
- [11] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP J. Adv. Signal Process.*, vol. 2009, p. 14, 2009, Article ID 530435.
- [12] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [13] C. E. Davila, "An efficient recursive total least squares algorithm for FIR adaptive filtering," *IEEE Trans. Signal Process.*, vol. 42, pp. 267–280, 1994.
- [14] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal Processing*, vol. 87, no. 10, pp. 2283–2302, 2007, Special Section: Total Least Squares and Errors-in-Variables Modeling.
- [15] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex Analysis and Optimization*. New York: Athena Scientific, 2003, ch. 8.2.
- [16] Z. Quan Luo, W. Kin Ma, A.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, pp. 20–34, May 2010.
- [17] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

- [18] B. Johansson, C. Carretti, and M. Johansson, "On distributed optimization using peer-to-peer communications in wireless sensor networks," in *Proc. 5th Ann. IEEE Commun. Soc. Conf. Sens., Mesh and Ad Hoc Commun. Netw. (SECON'08)*, 2008, pp. 497–505, 16–20.
- [19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [20] H. Wolkowicz, R. Saigal, and L. Vandenberghe, *Handbook of Semidefinite Programming*. Boston, MA: Kluwer Academic, 2000.
- [21] G. Pataki, "On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues," *Math. Oper. Res.*, vol. 23, pp. 339–358, 1998.
- [22] M. Trnavska, "Strong duality conditions in semidefinite programming," *J. Elect. Eng.*, vol. 56, pp. 87–89, 2005.
- [23] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48.
- [24] K.-B. Yu, "Recursive updating the eigenvalue decomposition of a covariance matrix," *IEEE Trans. Signal Process.*, vol. 39, pp. 1136–1145, May 1991.
- [25] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Inf. Process. Sens. Netw. (IPSN'05)*, Piscataway, NJ, 2005, p. 9, IEEE Press.
- [26] D. Shah, "Gossip algorithms," *Found. Trends in Netw.*, vol. 3, pp. 1–125, 2009.

Alexander Bertrand (S'08) was born in Roeselare, Belgium, in 1984. He received the M.Sc. degree in electrical engineering from Katholieke Universiteit Leuven, Belgium, in 2007.

Since 2007, he has been working towards the Ph.D. degree under the supervision of Prof. M. Moonen, at the Electrical Engineering Department (ESAT), Katholieke Universiteit Leuven. In 2010, he was a visiting researcher at the Adaptive Systems Laboratory, University of California, Los Angeles (UCLA), under the supervision of Prof. A. H. Sayed.

His research interests are in multi-channel signal processing, ad hoc sensor arrays, wireless sensor networks, distributed signal enhancement, speech enhancement, and distributed estimation.

Mr. Bertrand received a Ph.D. scholarship of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) from 2008–2011, and an FWO (Research Foundation Flanders) travel grant for a Visiting Research Collaboration at UCLA in 2010.

Marc Moonen (M'94–SM'06–F'07) received the electrical engineering degree and the Ph.D. degree in applied sciences from Katholieke Universiteit Leuven, Belgium, in 1986 and 1990 respectively.

Since 2004, he has been a Full Professor with the Electrical Engineering Department, Katholieke Universiteit Leuven, where he heads a research team working in the area of numerical algorithms and signal processing for digital communications, wireless communications, DSL, and audio signal processing.

Dr. Moonen received the 1994 KU Leuven Research Council Award, the 1997 Alcatel Bell (Belgium) Award (with P. Vandaele), the 2004 Alcatel Bell (Belgium) Award (with R. Cendrillon), and was a 1997 "Laureate of the Belgium Royal Academy of Science." He received a journal best paper award from the IEEE TRANSACTIONS ON SIGNAL PROCESSING (with G. Leus) and from *Elsevier Signal Processing* (with S. Doclo). He was chairman of the IEEE Benelux Signal Processing Chapter (1998–2002), and is currently President of the European Association for Signal Processing (EURASIP). He served as Editor-in-Chief for the *EURASIP Journal on Applied Signal Processing* (2003–2005), and has been a member of the editorial board of *Integration*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2002–2003) and IEEE SIGNAL PROCESSING MAGAZINE (2003–2005) and *Integration, the VLSI Journal*. He is currently a member of the editorial board of *EURASIP Journal on Advances in Signal Processing*, *EURASIP Journal on Wireless Communications and Networking*, and *Signal Processing*.