Abstract—We address the sensor network localization problem given noisy range measurements between pairs of nodes. We approach the nonconvex maximum-likelihood formulation via a known simple convex relaxation. We exploit its favorable optimization properties to the full to obtain an approach that is completely distributed, has a simple implementation at each node, and capitalizes on an optimal gradient method to attain fast convergence. We offer a parallel but also an asynchronous flavor, both with theoretical convergence guarantees and iteration complexity analysis. Experimental results establish leading performance. Our algorithms top the accuracy of a comparable state-of-the-art method by one order of magnitude, using one order of magnitude fewer communications.

Index Terms—Convex relaxations, distributed algorithms, distributed iterative sensor localization, maximum likelihood estimation, nonconvex optimization, wireless sensor networks.

I. INTRODUCTION

SENSOR networks are becoming ubiquitous. From environmental and infrastructure monitoring to surveillance, and healthcare networked extensions of the human senses in contemporary technological societies are improving our quality of life, our productivity, and our safety. Applications of sensor networks recurrently need to be aware of node positions to fulfill their tasks and deliver meaningful information. Nevertheless, locating the nodes is not trivial: these small, low cost, low power devices are deployed in large numbers, often with imprecise prior knowledge of their locations, and are equipped with minimal processing capabilities. Such limitations call for localization algorithms which are scalable, fast, and parsimonious in their communication and computational requirements.

A. Problem Statement

The sensor network is represented as an undirected graph $G = (\mathcal{V}, \mathcal{E})$. In the node set $\mathcal{V} = \{1, 2, \ldots, n\}$ we represent the sensors with unknown positions. There is an edge $i \sim j \in \mathcal{E}$ between sensors $i$ and $j$ if a noisy range measurement between nodes $i$ and $j$ is available (at both of them) and nodes $i$ and $j$ can communicate with each other. Anchors are elements with known positions and are collected in the set $\mathcal{A} = \{1, \ldots, m\}$. For each sensor $i \in \mathcal{V}$, we let $\mathcal{A}_i \subseteq \mathcal{A}$ be the subset of anchors (if any) whose distance to node $i$ is quantified by a noisy range measurement. The set $\mathcal{N}_i$ collects the neighbors of node $i$.

Let $\mathbb{R}^p$ be the space of interest ($p = 2$ for planar networks, and $p = 3$ otherwise). We denote by $x_i \in \mathbb{R}^p$ the position of sensor $i$, and by $d_{ij}$ the noisy range measurement between sensors $i$ and $j$, available at both $i$ and $j$. Following [1], we assume $d_{ij} = |x_i - x_j|$. Anchor positions are denoted by $a_k \in \mathbb{R}^p$. We let $r_{ik}$ denote the noisy range measurement between sensor $i$ and anchor $k$, available at sensor $i$.

The distributed network localization problem addressed in this work consists in estimating the sensors’ positions $x = \{x_i : i \in \mathcal{V}\}$, from the available measurements $\{d_{ij} : i \sim j\} \cup \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}_i\}$ and known anchor positions $a_k \in \mathcal{A}$, through collaborative message passing between neighboring sensors in the communication graph $G$.

Under the assumption of zero-mean, independent and identically-distributed, additive Gaussian measurement noise, the maximum likelihood estimator for the sensor positions is the solution of the optimization problem

$$\min_{x} f(x),$$

where

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} (|x_i - x_j| - d_{ij})^2 + \sum_{i=1}^{n} \sum_{k \in \mathcal{A}_i} \frac{1}{2} (|x_i - a_k - r_{ik}|^2).$$

Problem (1) is nonconvex and difficult to solve [2], nevertheless, it is guaranteed to have a global minimum, since function $f$ is continuous and coercive (because, as shown in Lemma 5 ahead, it is lower bounded by a coercive function $f$).

B. Contributions

We set forth a convex underestimator of the maximum likelihood cost for the sensor network localization problem (1) based on the convex envelopes of its parcels.

We present an optimal synchronous and parallel algorithm to minimize this convex underestimator — with proven convergence guarantees. We also propose an asynchronous variant of

1This entails no loss of generality: it is readily seen that, if $d_{ij} \neq d_{ji}$, then it suffices to replace $d_{ij} \leftarrow (d_{ij} + d_{ji})/2$ and $d_{ji} \leftarrow (d_{ij} + d_{ji})/2$ in the forthcoming optimization problem (1).
this algorithm and prove it converges almost surely. Furthermore, we analyze its iteration complexity.

Moreover, we assert the superior performance of our algorithms by computer simulations; we compared several aspects of our method with [2], [3], and [4], and our approach always yields better performance metrics. When compared with the method in [2], which operates under the same conditions, our method outperforms it by one order of magnitude in accuracy and in communication volume.

C. Related Work

With the advent of large-scale networks, the computational paradigm of information processing algorithms — centralized versus distributed — becomes increasingly critical. A centralized method can be less suited for a network with meager communication and computation resources, while a distributed algorithm might not be adequate if the network is supposed to deliver in one place the global result of its computations. Further, none of the available techniques to address Problem (1) claims convergence to the global optimum — due to the nonconvexity, but also due to ambiguities in the network topology which create more than one distant global optimum [5].

1) Centralized Paradigm: The centralized approach to the problem of sensor network localization summoned up a wide body of research. It involves a central processing unit to which all sensor nodes communicate their collected measurements. Centralized architectures are prone to data traffic bottlenecks close to the central node. Resilience to failure, security and privacy issues are, also, not naturally accounted for by the centralized architecture. Moreover, as the number of nodes in the network grows, the problem to be solved at the central node becomes increasingly complex, thus raising scalability concerns.

Focusing on recent work, several different approaches are available, such as the work in [6], where sensor network localization is formulated as a regression problem over adaptive bases. The method has an initialization step using eigendecomposition of an affinity matrix; its entries are functions of squared distance measurements between sensors. The refinement is done by conjugate gradient descent over a discrepancy function of squared distances — which is mathematically more tractable but amplifies measurement errors and outliers and does not benefit from the limiting properties of maximum likelihood estimators. This approach is closely related to multidimensional scaling, where the sensor network localization problem is posed as a least-squares problem, as in [7]. Multidimensional scaling is unreliable in large-scale networks due to their sparse connectivity. Also relying on the well-tested weighted least squares approach, the work in [5] performs successive minimizations of a weighted least squares cost function convolved with a Gaussian kernel of decreasing variance.

Another successfully pursued approach is to perform semidefinite or weaker second-order cone relaxations of the original nonconvex problem (1) [3], [8]. These approaches do not scale well, since the centralized SDP or SOCP problem gets very large even for a small number of nodes. In [3] and [9] the majorization-minimization framework was used with quadratic cost functions to derive centralized approaches to the sensor network localization problem.

2) Distributed Paradigm: In the present work, the expression distributed method denotes an algorithm requiring no central or fusion node where all nodes perform the same types of computations. Distributed approaches for cooperative localization have been less frequent than centralized ones, despite the more suited nature of this computational paradigm to sensor networks, when the target application does not require that the estimate of all sensor positions be available in one place.

We consider two main approaches to the distributed sensor network localization problem: 1) one where the nonconvex Problem (1) (or some other nonconvex discrepancy minimization) is attacked directly, and hence the quality of the solution is highly dependent on the quality of the algorithm’s initialization; 2) and another, where the original nonconvex sensor network localization problem is relaxed to a convex problem, whose tightness will determine how close the solution of the convex problem will approximate the global solution of the original problem, not needing any particular initialization.

Initialization Dependent: In reference [10] the authors develop a distributed implementation of multidimensional scaling for solution refinement. These authors base their method on the majorization-minimization framework, but they do not provide a formal proof of convergence for the jacobi-like iteration. The work in [11] puts forward two distributed methods optimizing the discrepancy of squared distances: a gradient algorithm with barzilai-borwein step sizes calculated in a first consensus phase, followed by a gradient computation phase, and a gauss-newton algorithm also with a consensus phase and a gradient computation phase. Both are refinement methods that need good initializations to converge to the global optimum.

Initialization Independent: The work in [12] proposes a parallel distributed algorithm. However, the sensor network localization problem adopts the previously discussed squared distances discrepancy function. Also, each sensor must solve a second order cone program at each algorithm iteration, which can be a demanding task for the simple hardware used in sensor networks’ motes. Furthermore, the formal convergence properties of the algorithm are not established. The work in [13] also considers network localization outside a maximum likelihood framework. The approach proposed in [13] is not parallel, operating sequentially through layers of nodes: neighbors of anchors estimate their positions and become anchors themselves, making it possible in turn for their neighbors to estimate their positions, and so on. Position estimation is based on planar geometry-based heuristics. In [14], the authors propose an algorithm with assured asymptotic convergence, but the solution is computationally complex since a triangulation set must be calculated, and matrix operations are pervasive. Furthermore, in order to attain good accuracy, a large number of range measurement rounds must be acquired, one per iteration of the algorithm, thus increasing energy expenditure. On the other hand, the algorithm presented in [1] and based on the nonlinear Gauss Seidel framework, has a pleasingly simple implementation, combined with the convergence guarantees inherited from the framework. Notwithstanding, this algorithm is sequential, i.e., nodes perform their calculations in turn, not in a parallel fashion. This entails the existence of a network-wide coordination procedure to precompute the processing schedule upon
startup, or whenever a node joins or leaves the network. The sequential nature of the work in [1] was superseded by the work in [2] which puts forward a parallel method based on two consecutive relaxations of the maximum likelihood estimator in (1). The first relaxation is a semi-definite program with a rank relaxation, while the second is an edge based relaxation, best suited for the Alternating Direction Method of Multipliers (ADMM). The main drawback is the amount of communications required to manage the ADMM variable local copies, and by the prohibitive complexity of the problem at each node. In fact, each one of the simple sensing units must solve a semidefinite program at each ADMM iteration and after the update copies of the edge variables must be exchanged with each neighbor. A simpler approach was devised in [4] by extending the source localization Projection Onto Convex Sets algorithm in [15] to the problem of sensor network localization. The proposed method is sequential, activating nodes one at a time according to a predefined cyclic schedule; thus, it does not take advantage of the parallel nature of the network and imposes a stringent timetable for individual node activity.

II. CONVEX RELAXATION

Problem (1) can be written as

$$\text{minimize} \sum_{i,j} \frac{1}{2} d_{S_{ij}}^2(x_i - x_j) + \sum_i \frac{1}{2} \sum_{k \in A_i} d_{S_{akt}}^2(x_i),$$

where $d_{S_{ij}}^2(x)$ represents the squared Euclidean distance of point $x$ to the set $C$, i.e., $d_{S_{ij}}^2(x) = \inf_{y \in C} \|x - y\|^2$, and the sets $S_{ij}$ and $S_{akt}$ are defined as the spheres generated by the noisy measurements $d_{ij}$ and $r_{ik}$

A nonconvexity of (2) follows from the nonconvexity of the building block

$$\frac{1}{2} d_{S_{ij}}^2(x) - \frac{1}{2} \inf_{y \in d_{ij}} \|x - y\|^2.$$  

A simple convexification consists in replacing it by

$$\frac{1}{2} d_{B_{ij}}^2(x) - \frac{1}{2} \inf_{y \in d_{ij}} \|x - y\|^2$$

where $B_{ij} = \{z \in \mathbb{R}^d : \|z\| \leq d_{ij}\}$, is the convex hull of $S_{ij}$. Actually, (4) is the convex envelope of (3). This fact is illustrated in Fig. 1 with a one-dimensional example; a formal proof for the generic case is given in Section I. The terms of (2) associated with anchor measurements are similarly relaxed as

$$\frac{1}{2} d_{S_{akt}}^2(x) - \inf_{y \in a_k \leq r_{ik}} \|x - y\|^2,$$

where the set $B_{akt}$ is the convex hull of $S_{akt}$: $B_{akt} = \{z \in \mathbb{R}^d : \|z - a_k\| \leq r_{ik}\}$. Replacing the nonconvex parcels in (2) by the sums of terms (4) and (5) we obtain the convex problem

$$\text{minimize} \hat{f}(x) = \sum_{i,j} \frac{1}{2} d_{S_{ij}}^2(x_i - x_j) + \sum_i \frac{1}{2} \sum_{k \in A_i} d_{B_{akt}}^2(x_i).$$

The function in Problem (6) is an underestimate of (2) but it is not the convex envelope of the original function. We argue that in our application of sensor network localization it is generally a very good approximation whose sub-optimality can be quantified, as discussed in Section IV-A. The cost function (6) also appears in [4] albeit via a distinct reasoning; our convexification mechanism seems more intuitive. But the striking difference with respect to [4] is how (6) is exploited to generate distributed solution methods. Whereas [4] lays out a sequential block-coordinate approach, we show that (6) is amenable to distributed solutions either via the fast Nesterov’s gradient method (for synchronous implementations) or exact/inexact randomized block-coordinate methods (for asynchronous implementations).

III. DISTRIBUTED SENSOR NETWORK LOCALIZATION

We propose two distributed algorithms: a synchronous one, where nodes work in parallel, and an asynchronous, gossip-like algorithm, where each node starts its processing step according to some probability distribution. Both algorithms require to compute the gradient of the cost function and its Lipschitz constant. In order to achieve this it is convenient to rewrite Problem (6) as

$$\text{minimize} \frac{1}{2} d_{B_{ij}}^2(Ax) + \sum_i \frac{1}{2} \sum_{k \in A_i} d_{B_{akt}}^2(x_i),$$

where $A = C \otimes I_p$, $C$ is the arc-node incidence matrix of $G$, $I_p$ is the identity matrix of size $p$, and $B$ is the Cartesian product of the balls $B_{ij}$ corresponding to all the edges in $E$. We denote the two parcels in (7) as

$$g(x) = \frac{1}{2} d_{B_{ij}}^2(Ax), \quad h(x) = \sum_i h_i(x_i),$$

where $h_i(x_i) = \sum_{k \in A_i} \frac{1}{2} d_{B_{akt}}^2(x_i)$. Problems (6) and (7) are equivalent since $Ax$ is the vector $(x_i - x_j : i \sim j)$ and function $g(x)$ in (7) can be written as

$$g(x) - \frac{1}{2} d_{B_{ij}}^2(Ax) = \frac{1}{2} \inf_{y \in \mathbb{R}^d} \|Ax - y\|^2 = \frac{1}{2} \inf_{y \in B_{ij}} \sum_{i,j} \|x_i - x_j - y_{ij}\|^2$$

and as all the terms are non-negative and the constraint set is a Cartesian product, we can exchange $\inf$ with the summation, resulting in

$$g(x) - \frac{1}{2} \sum_{i,j} \inf_{y_{ij} \leq d_{ij}} \|x_i - x_j - y_{ij}\|^2 = \frac{1}{2} \sum_{i,j} \frac{1}{2} d_{B_{ij}}^2(x_i - x_j),$$

which is the corresponding term in (6).

A. Gradient and Lipschitz Constant of $\hat{f}$

To simplify notation, let us define the functions:

$$\phi_{B_{ij}}(z) = \frac{1}{2} d_{B_{ij}}^2(z), \quad \phi_{B_{akt}}(z) = \frac{1}{2} d_{B_{akt}}^2(z).$$

The convex envelope (or convex hull) of a function $\gamma$ is its best possible convex underestimator, i.e., $\text{conv} \gamma(x) = \sup \{\eta(x) : \eta \leq \gamma, \eta \text{ is convex}\}$, and is hard to determine in general.
Now we call on a key result from convex analysis (see [16, Prop. X.3.2.2, Th. X.3.2.3]): the function in (4), \( \phi_{B_{ij}}(z) - \frac{1}{2} d_{B_{ij}}^2(z) \) is convex, differentiable, and its gradient is

\[
\nabla \phi_{B_{ij}}(z) = z - P_{B_{ij}}(z),
\]

where \( P_{B_{ij}}(z) \) is the orthogonal projection of point \( z \) onto the closed convex set \( B_{ij} \)

\[
P_{B_{ij}}(z) = \arg \min_{y \in B_{ij}} \| z - y \|.
\]

Further, function \( \phi_{B_{ij}} \) has a Lipschitz continuous gradient with constant \( L_\phi = 1 \), i.e.,

\[
\| \nabla \phi_{B_{ij}}(x) - \nabla \phi_{B_{ij}}(y) \| \leq \| x - y \|.
\]

We show (9) in Section II.

Let us define a vector-valued function \( \phi_B \), obtained by stacking all functions \( \phi_{B_{ij}} \). Then, \( g(x) = \phi_B(Ax) \). From this relation, and using (8), we can compute the gradient of \( g(x) \):

\[
\nabla g(x) = A^T \nabla \phi_B(Ax) = A^T (Ax - P_B(Ax)) = \mathcal{L}x - A^T P_B(Ax),
\]

where the second equality follows from (8) and \( \mathcal{L} = A^T A = L \otimes I_p \), with \( L \) being the Laplacian matrix of \( \mathcal{G} \). This gradient is Lipschitz continuous and we can obtain an easily computable Lipschitz constant \( L_g \) as follows

\[
\| \nabla g(x) - \nabla g(y) \| \leq \| A \| \| Ax - Ay \| + \lambda_{\text{max}}(A^T A) \| x - y \|
\]

\[
\leq \| A \| \| x - y \| + \lambda_{\text{max}}(A^T A) \| x - y \| + \frac{\| A \| \| x - y \|}{L_g}
\]

\[
\leq 2\delta_{\text{max}} + \lambda_{\text{max}}(L) \| x - y \|,
\]

where \( \| A \| \) is the maximum singular value norm; equality \( \{ a \} \) is a consequence of Kronecker product properties. In (11) we denote the maximum node degree of \( \mathcal{G} \) by \( \delta_{\text{max}} \). A proof of the bound \( \lambda_{\text{max}}(L) \leq 2\delta_{\text{max}} \) can be found in [17] 3.

3A tighter bound would be \( \lambda_{\text{max}}(L) \leq \max_{i,j} \{ \delta_i + \delta_j - c(i,j) \} \) where \( \delta_i \) is the degree of node \( i \) and \( c(i,j) \) is the number of vertices that are adjacent to both \( i \) and \( j \) [18, Th. 4.13], nevertheless \( 2\delta_{\text{max}} \) is easier to compute in a distributed way.

The gradient of \( h \) is \( \nabla h(x) = (\nabla h_1(x_1), \ldots, \nabla h_n(x_n)) \), where the gradient of each \( h_i \) is

\[
\nabla h_i(x_i) = \sum_{k \in A_i} \nabla \phi_{B_{ik}}(x_i).
\]

The gradient of \( h \) is also Lipschitz continuous. The constants \( L_{h_i} \) for \( \nabla h_i \) are

\[
\| \nabla h(x) - \nabla h(y) \| \leq \sum_{i} \| \nabla \phi_{B_{ik}}(x_i) - \nabla \phi_{B_{ik}}(y_i) \| \leq \lambda_{\text{max}}(\mathcal{A}) \| x_i - y_i \|,
\]

where \( |\mathcal{C}| \) is the cardinality of set \( \mathcal{C} \). We now have an overall constant \( L_h \) for \( \nabla h \),

\[
\| \nabla h(x) - \nabla h(y) \| \leq \sqrt{\sum_i \| \nabla h_i(x_i) - \nabla h_i(y_i) \|^2} \leq \sqrt{\sum_i \| \mathcal{A}_i \| \| x_i - y_i \|^2} \leq \max_{i} \{ \mathcal{A}_i : i \in \mathcal{V} \} \| x - y \|.
\]

We are now able to write \( \nabla \bar{f} \), the gradient of our cost function, as

\[
\nabla \bar{f}(x) = \mathcal{L}x - A^T P_B(Ax) + \sum_{k \in A_\mathcal{C}} x_k - P_B(x_k).
\]

A Lipschitz constant \( L_f \) is, thus,

\[
L_f = 2\delta_{\text{max}} + \max \{ |\mathcal{A}_i| : i \in \mathcal{V} \}.
\]

This constant is easy to precompute by, e.g., a diffusion algorithm — cf. [19, Ch. 9] for more information.

In summary, we can compute the gradient of \( \bar{f} \) using (15) and a Lipschitz constant by (16), which leads us to the algorithms described in Sections III-B and III-C for minimizing \( \bar{f} \).
measurement $d_{ij}$. The product with $A^\top$ will collect, at the entries corresponding to each node, the sum of the projections relative to edges where it intervenes, with a positive or negative sign depending on the arbitrary edge direction agreed upon at the onset of the algorithm. More specifically, $(A^\top P_B(Ax))_i = \sum_{j \in N_i} c_{i \sim j,i} P_{B_{ij}}(x_i - x_j)$, as presented in Step 7 of Algorithm 1. The last summand in (15) is simply $\nabla h(x)$, and the $i$-th entry of $\nabla h(x)$ is given in (12). This can be easily computed independently by each node according to Step 8. The position updates in Step 9 of the algorithm require the computation of the gradient of the cost w.r.t. the coordinates of node $i$, done in the previous steps, evaluated at the extrapolated points $w_i$.

**Algorithm 1: Parallel method**

**Input:** $L_f; \{d_{ij} : i \sim j \in \mathcal{E}\}; \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}\}$

**Output:** $\hat{x}$

1: $k = 0$
2: while some stopping criterion is not met, each node $i$ do
3:   $w_i = x_i(0) + \frac{1}{k+1}(x_i(k-1) - x_i(k-2))$;
4:   node $i$ broadcasts $w_i$ to its neighbors
5:   $\nabla h_i(w_i) = \sum_{k \in \mathcal{A}_i} w_i - P_{B_{ik}}(w_i)$;
6: end while

**Algorithm 2: Asynchronous method**

**Input:** $L_f; \{d_{ij} : i \sim j \in \mathcal{E}\}; \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}\}$

**Output:** $\hat{x}$

1: each node $i$ chooses random $x_i(0)$;
2: $k = 0$;
3: while some stopping criterion is not met, each node $i$ do
4:   $k = k + 1$;
5:   if $\xi_k = i$ then
6:     $x_i(k) = \arg \min_{w_i} f(x_i(k-1), \ldots, w_i, \ldots, x_n(k-1))$
7:   else
8:     $x_i(k) = x_i(k-1)$
9: end if
10: end while
11: return $\hat{x} = x(k)$

To compute the minimizer in Step 6 of Algorithm 2 it is useful to recast Problem (7) as

$$\min_x \sum_i \left( \frac{1}{4} \sum_{j \in N_i} d_{B_{ij}}^2(x_i - x_j) + \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{ik}}^2(x_i) \right),$$

where the factor $\frac{1}{4}$ accounts for the duplicate terms when considering summations over nodes instead of over edges. By fixing the neighbor positions, each node solves a single source localization problem; this setup leads to the Problem

$$\min_{x_i} f_{sl_i}(x_i) := \sum_{j \in N_i} \frac{1}{4} d_{B_{ij}}^2(x_i) + \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{ik}}^2(x_i),$$

where $B_{sij} = \{z \in \mathbb{R}^p : \|z - x_j\| \leq d_{ij}\}$. Note that the function in (19) is continuous and coercive; thus, the optimization problem (19) has a solution.

**Algorithm 3: Asynchronous update at each node $i$**

**Input:** $\xi_k; L_f; \{d_{ij} : i \sim j \in \mathcal{N}_i\}; \{r_{ik} : k \in \mathcal{A}_i\}$

**Output:** $x_i(k)$

1: if $\xi_k$ not $i$ then
2:   $x_i(k) = x_i(k-1)$;
3: else
4:   return $x_i(k)$;
5: end if
6: choose random $z(0) = z(-1)$;
7: while some stopping criterion is not met do
8:   $l = l + 1$;
9:   $w = z(l - 1) + \frac{1}{l+1}(z(l-1) - z(l-2))$;
10: $\nabla f_{sl_i}(w) = -\frac{1}{L_f} \sum_{j \in N_i} w - P_{B_{ij}}(w) + \sum_{k \in \mathcal{A}_i} w - P_{B_{ik}}(w)$
11: $z(l) = w - \frac{1}{L_f} \nabla f_{sl_i}(w)$
12: end while
13: return $x_i(k) - z(l)$
We solve Problem (19) at each node by employing Nesterov’s optimal accelerated gradient method as described in Algorithm 3. The asynchronous method proposed in Algorithm 2 converges to the set of minimizers of function \( \tilde{f} \), as established in Theorem 2, in Section IV.

We also propose an inexact version in which nodes do not solve Problem (19) but instead take just one gradient step. That is, simply replace Step 6 in Algorithm 2 by

\[
x_i(k) - x_i(k - 1) = \frac{1}{L_{\tilde{f}}} \nabla_i \tilde{f}(x(k - 1))
\]

where \( \nabla_i \tilde{f}(x_1, \ldots, x_n) \) is the gradient with respect to \( x_i \), and assume

\[
P(\xi_k - i) = \frac{1}{n}.
\]

The convergence terms of the resulting algorithm are established in Theorem 3, Section IV.

IV. THEORETICAL ANALYSIS

A relevant question regarding Algorithms 1 and 2 is whether they will return a good solution to the problem they are designed to solve, after a reasonable amount of computations. Sections IV-B and IV-C address convergence issues of the proposed methods, and discuss some of the assumptions on the problem data. Section IV-A provides a formal bound for the gap between the original and the convexified problems.

A. Quality of the Convexified Problem

While evaluating any approximation method it is important to know how far the approximate optimum is from the original one. In this Section we will focus on this analysis.

It was already noted in Section II that for \( |x| \geq d_{ij} \), when the functions differ, for \( |x| < d_{ij} \), we have that \( \phi_{\Delta ij}(x) = 0 \). The same applies to the terms related to anchor measurements. The optimal value of function \( f \), denoted by \( f^* \), is bounded by \( \hat{f}(x^*) \leq f^* \leq \tilde{f}(x^*) \), where \( x^* \) is the minimizer of the convexified problem (6), and \( \hat{f} = \inf_x \tilde{f}(x) \) is the minimum of function \( \tilde{f} \). With these inequalities we can compute a bound for the optimality gap, after (6) is solved, as

\[
f^* - \hat{f}^* = \sum_{i \neq j \in E} \frac{1}{2} d_{\Delta ij}^2 (x_i^* - x_j^*) + d_{\Delta_{\text{lk}}^2}^2 (x_i^* - x_j^*)
\]

The convergence terms of the resulting algorithm are established in Theorem 3, Section IV.

TABLE I

<table>
<thead>
<tr>
<th>( f^* - \hat{f}^* ) Equation (22)</th>
<th>( f^* - \tilde{f}^* ) Equation (23)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0367</td>
<td>0.0487</td>
</tr>
<tr>
<td></td>
<td>3.0871</td>
</tr>
</tbody>
</table>

1) Compute the optimal solution \( x^* \) using Algorithm 1 or 2;
2) Select the terms of the convexified problem (6) which are zero;
3) Add the nonconvex costs of each of these edges, as in (22).

For the one-dimensional example of the star network costs depicted in Fig. 2 the bounds in (22), and (23) averaged over 500 Monte Carlo trials are presented in Table I. The true average gap \( f^* - \hat{f}^* \) is also shown. In the Monte Carlo trials we sampled a zero mean Gaussian random variable with \( \sigma = 0.25 \) and obtained a noisy range measurement as described later by (28). These results show the tightness of the convexified function and how loose the bound (23) is when applied to our problem.

B. Parallel Method: Convergence Guarantees and Iteration Complexity

As Problem (7) is convex and the cost function has a Lipschitz continuous gradient, Algorithm 1 is known to converge at the optimal rate \( O(k^{-2}) \) [20], [24]: \( \hat{f}(x(k)) - \hat{f}^* \leq \frac{2L_{\hat{f}}}{(k+1)^2} ||x_0 - x^*||^2 \).

C. Asynchronous Method: Convergence Guarantees and Iteration Complexity

To state the convergence properties of Algorithm 2 we only need Assumption 1.

Assumption 1: There is at least one anchor linked to some sensor and the graph \( G \) is connected (there is a path between any two sensors).

This assumption holds generally as one needs \( p + 1 \) anchors to eliminate translation, rotation, and flip ambiguities while performing localization in \( \mathbb{R}^p \), which exceeds the assumption requirement. We present two convergence results, — Theorem 2, and Theorem 3 — and the iteration complexity analysis for Algorithm 2 in Proposition 4. Proofs of the Theorems are detailed in Appendix D.
The following Theorem establishes the almost sure (a.s.) convergence of Algorithm 2.

**Theorem 2 (Almost Sure Convergence of Algorithm 2):** Let \( \{x(k)\}_{k \in \mathbb{N}} \) be the sequence of points produced by Algorithm 2, or by Algorithm 2 with the update (20), and let \( \mathcal{X}^* = \{x^* : \hat{f}(x^*) = \hat{f}^*\} \) be the set of minimizers of function \( \hat{f} \) defined in (6). Then it holds:

\[
\lim_{k \to \infty} d_{\mathcal{X}}(x(k)) = 0, \quad \text{a.s.} \quad (24)
\]

In words, with probability one, the iterates \( x(k) \) will approach the set \( \mathcal{X}^* \) of minimizers of \( \hat{f} \); this does not imply that \( \{x(k)\}_{k \in \mathbb{N}} \) will converge to a single \( x^* \in \mathcal{X}^* \), but it does imply that \( \lim_{k \to \infty} \hat{f}(x(k)) = \hat{f}^* \), since \( \mathcal{X}^* \) is a compact set, as proven in Appendix C, Lemma 5.

**Theorem 3 (Almost Sure Convergence to a Point):** Let \( \{x(k)\}_{k \in \mathbb{N}} \) be a sequence of points generated by Algorithm 2, with the update (20) in Step 6, and let all nodes start computations with uniform probability. Then, with probability one, there exists a minimizer of \( \hat{f} \), denoted by \( x^* \in \mathcal{X}^* \), such that

\[
x(k) \to x^*. \quad (25)
\]

This result tells us that the iterates of Algorithm 2 with the modified Step 6 stated in (20) not only converge to the solution set, but also guarantees that they will not be jumping around the solution set \( \mathcal{X}^* \) (unlikely to occur in Algorithm 2, but not ruled out by the analysis). One of the practical benefits of Theorem 3 is that the stopping criterion can safely probe the stability of the estimates along iterations. To the best of our knowledge, this kind of strong type of convergence (the whole sequence converges to a point in \( \mathcal{X}^* \)) was not established previously in the context of randomized approaches for convex functions with Lipschitz continuous gradients, though it was derived previously for randomized proximal-based minimizations of a large number of convex functions, cf. [25, Proposition 9]. We emphasize that what prevents the latter to apply to the exact version of Algorithm 2 is the ambiguity in choosing estimates when \( \mathcal{X}^* \) is not singleton. A possible approach to circumvent non-uniqueness of minimizers in (19) is to add a proximal term (as this makes the function strictly convex). However, the proximal terms tend to slow down convergence. Although overall strong convergence is still an open issue with this device, we saw in preliminary experiments that the proximal terms slowed down the speed of convergence (up to one order of magnitude of degradation in the iteration count).

**Proposition 4 (Iteration Complexity for Algorithm 2):** Let \( \{x(k)\}_{k \in \mathbb{N}} \) be a sequence of points generated by Algorithm 2, with the update (20) in Step 6, and let the nodes be activated with equal probability. Choose \( 0 < \epsilon < \hat{f}(x(0)) - \hat{f}^* \) and \( \rho \in [0, 1) \). Then there exists a constant \( b(\rho, x(0)) \) such that

\[
P\left( \hat{f}(x(k)) - \hat{f}^* \leq \epsilon \right) \geq 1 - \rho \quad (26)
\]

for all

\[
k > K = \frac{2nb(\rho, x(0))}{\epsilon} + 2 - n. \quad (27)
\]

The constant \( b(x(0), \rho) \) can be computed from inequality (19) in [26]; it depends only on the initialization and the chosen \( \rho \).

Proposition 4 is saying that, with high probability, the function value \( \hat{f}(x(k)) \) for all \( k > K \) will be at a distance \( \epsilon \) of the optimal, and the number of iterations \( K \) depends inversely on the chosen \( \epsilon \).

**Proof of Proposition 4:** As \( \hat{f} \) is differentiable and has Lipschitz gradient, the result is trivially deduced from [26, Th. 2]. \( \square \)

### V. Numerical Experiments

In this Section we present experimental results that demonstrate the superior performance of our methods when compared with four state of the art algorithms: Euclidean Distance Matrix (EDM) completion presented in [3], Semidefinite Program (SDP) relaxation and Edge-based Semidefinite Program (ESDP) relaxation, both implemented in [2], and a sequential projection method (PM) in [4] optimizing the same convex underestimator as the present work, with a different algorithm. The first two methods — EDM completion and SDP relaxation — are centralized, whereas the ESDP relaxation and PM are distributed.

1) **Methods:** We conducted simulations with two uniquely localizable geometric networks with sensors randomly distributed in a two-dimensional square of size 1x1 with 4 anchors in the corners of the square. Network 1 has 10 sensor nodes with an average node degree of 4.3, while network 2 has 50 sensor nodes and average node degree of 6.1. The ESDP method was only evaluated in network 1 due to simulation time constraints, since it involves solving an SDP at each node, and each iteration. The noisy range measurements are generated according to

\[
d_{ij} = ||x_i^* - x_j^*|| + \nu_{ij}, \quad r_{ik} = ||x_i^* - a_k|| + \nu_{ik}, \quad (28)
\]

where \( x_i^* \) is the true position of node \( i \), and \( \{\nu_{ij} : i \sim j \in \mathcal{E}\} \cup \{\nu_{ik} : i \in \mathcal{V}, k \in \mathcal{A}_i\} \) are independent Gaussian random variables with zero mean and standard deviation \( \sigma \). The accuracy of the algorithms is measured by the original non-convex cost value in (1) and by the Root Mean Squared Error (RMSE) per sensor, defined as

\[
\text{RMSE} = \sqrt{\frac{1}{n} \left( \frac{1}{M} \sum_{m=1}^{M} ||x^* - \hat{x}(m)||^2 \right)}, \quad (29)
\]

where \( M \) is the number of Monte Carlo trials performed.

#### A. Assessment of the Convex Underestimator Performance

The first experiment aimed at exploring the performance of the convex underestimator in (6) when compared with two other state of the art convexifications. For the proposed disk relaxation (6), Algorithm 1 was stopped when the gradient norm \( \nabla \hat{f}(x) \) reached \( 10^{-6} \) while both EDM completion and SDP relaxation were solved with the default SeDuMi solver [27] with value of \( 10^{-9} \), so that algorithm properties did not mask the real quality of the relaxations. Figs. 3 and 4 report the results of the experiment with 50 Monte Carlo trials over network 2 and measurement noise with \( \sigma = \{0.01, 0.05, 0.1, 0.3\} \); so, we had a total...
Fig. 3. Relaxation quality: Root mean square error comparison of EDM completion in [3], SDP relaxation in [2] and the disk relaxation (6), used in the present work; measurements were perturbed with noise with different values for the standard deviation $\sigma$. The disk relaxation approach in (6) improved on the RMSE values of both EDM completion and SDP relaxation for all noise levels, even though it does not rely on the SDP machinery. The performance gap to EDM completion is substantial.

Fig. 4. Relaxation quality: Comparison of the best achievable root mean square error versus overall execution time of the algorithms. Measurements were contaminated with noise with $\sigma = 0.1$. Although disk relaxation (6) has a distributed implementation, running it sequentially can be faster by one order of magnitude than the centralized methods.

Fig. 5. Performance of the proposed method in Algorithm 1 and of the Projection method presented in [4]. The stopping criterion for both algorithms was a relative improvement of $10^{-5}$ in the estimate. The proposed method uses fewer communications to achieve better RMSE for the tested noise levels. Our method outperforms the projection method with one forth of the number of communications for a noise level of 0.01.

Table II

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Communications per Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESDP method</td>
<td>21600</td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>2000</td>
</tr>
</tbody>
</table>

Fig. 6. Performance of the proposed method in Algorithm 1 and of the ESDP method in [2]. The stopping criterion for both algorithms was the number of algorithm iterations. The performance advantage of the proposed method in Algorithm 1 is even more remarkable when considering the number of communications presented in Table II.

B. Performance of Distributed Optimization Algorithms

To measure the performance of the presented Algorithm 1 in a distributed setting we compared it with the state of the art methods in [4] and the distributed algorithm in [2]. The results are shown, respectively, in Figs. 5 and 6. The experimental setups were different, since the authors proposed different stopping criteria for their algorithms and, in order to do a fair comparison, we ran our algorithm with the specific criterion set by each benchmark method. Also, to compare with the distributed ESDP method in [2], we had to use a smaller network of 10 sensors because of simulation time constraints — as the ESDP method entails solving an SDP problem at each node, the simulation time becomes prohibitively large, at least using a general purpose solver. The number of Monte Carlo trials was 32, with 3 noise levels, leading to 96 realizations for each noisy measurement. So, in the experiment illustrated in Fig. 5, the stopping criterion for both the projection method and the presented method was the relative improvement of the solution; we stress that this is not a distributed stopping criterion, we adopted it just for algorithm comparison. We can see that the proposed method fares better not only in RMSE but, foremost, in communication cost. The experiment comprised 120 Monte Carlo trials and two noise levels.

From the analysis of both Fig. 6 and Table II we can see that the ESDP method is one order of magnitude worse in RMSE performance, using one order of magnitude more communications, than Algorithm 1.

C. Performance of the Asynchronous Algorithm

A second experiment consisted on testing the performance of the parallel and the asynchronous flavors of our method, presented respectively in Algorithms 1 and 2, the latter with the exact update. The metric was the value of the convex cost function $\tilde{f}$ in (6) evaluated at each algorithm’s estimate of the minimum. To have a fair comparison, both algorithms were allowed to run until they reached a preset number of communications. In Fig. 7 we present the effectiveness of both algorithms in optimizing the disk relaxation cost in (6), with the same amount of communications. We chose the uniform probability law for the random variables $\xi_k$ representing the sequence of updating nodes in the asynchronous version of our method. Again, we ran 50 Monte Carlo trials, each with 3 noise levels, thus leading to 150 samplings of the noise variables in (28).
VI. CONCLUDING REMARKS

Experiments in Section V show that our method is superior to the state of the art in all measured indicators. While the comparison with the projection method published in [4] is favorable to our proposal, it should be further considered that the projection method has a different nature when compared to ours: it is sequential, and such algorithms will always have a larger computation time than parallel ones, since nodes run in sequence; moreover, this computation time grows with the number of sensors while parallel methods retain similar speed, no matter how many sensors the network has.

When comparing with a distributed and parallel method similar to Algorithm 1, like the ESDP method in [2] we can see one order of magnitude improvement in RMSE for one order of magnitude fewer communications of our method—and this score is achieved with a simpler, easy-to-implement algorithm, performing simple computations at each node that are well suited to the kind of hardware commonly found in sensor networks.

There are some important questions not addressed here. For example, it is not clear what influence the number of anchors and their spatial distribution can have in the performance of the proposed and state of the art algorithms. Also, an exhaustive study on the impact of varying topologies and number of sensors could lead to interesting results. Some preliminary experiments show that all convex relaxations experience some performance degradation when tested for robustness to sensors outside the convex hull of the anchors. This issue has been noted by several authors, but a more exhaustive study exceeds the scope of this paper.

But with the data presented here one can already grasp the advantages of our fast and easily implementable distributed method, where the optimality gap of the solution can also be easily quantified, and which offers two implementation flavours for different localization needs.

APPENDIX A

CONVEX ENVELOPE

We show that the function in (4) is the convex envelope of the function in (3). Refer to $\alpha$ as the function in (3) and $\beta$ as the function in (4). We show that $\alpha^{**} = \beta$ where $f^*$ denotes the Fenchel conjugate of a function $f$, cf. [16, Cor. 1.3.6, p. 45, v. 2].

We start by computing $\alpha^*$:

$$
\alpha^*(s) = \sup_z s^\top z - \alpha(z) = \sup_z s^\top z - \left( \frac{1}{2} \inf_{y \in S} \| y - d_{ij} \| \inf_{\alpha} \| z - y \|^2 \right) = \sup_z \sup_{y \in S} s^\top z - \frac{1}{2} \| z - y \|^2 = \sup_{y \in S} \sup_z s^\top z - \frac{1}{2} \| z - y \|^2 = \sup_{y \in S} \frac{1}{2} \| z \|^2 + s^\top y - \frac{1}{2} \| s \|^2 + d_{ij} \cdot |s|.
$$

Thus, $\alpha^*$ is the sum of two closed convex functions: $\alpha^* = g + h$ where $g(s) = \frac{1}{2} \| s \|^2$ and $h(s) = d_{ij} \cdot |s|$. Note that $h(s) = \sigma_B(0, d_{ij}) (s)$ where $\sigma_C(s) = \sup \{ s^\top x : x \in C \}$ denotes the support function of a set $C$. Thus, using [16, Th. 3.1.3, p. 61, v. 2], we have

$$
\alpha^{**}(z) = \inf_{z_1 + z_2 = z} g^*(z_1) + h^*(z_2).
$$

Since $g^*(z_1) = \frac{1}{2} \| z_1 \|^2$ [16, Ex. 1.1.3, p. 38, v. 2] and $h^*(z_2) = i_{B_{d_{ij}}}(z_2)$ [16, Ex. 1.1.5, p. 39, v. 2] where $i_C(x) = 0$ if $x \in C$ and $i_C(x) = +\infty$ if $x \notin C$ denotes the indicator of a set $C$, we conclude that

$$
\alpha^{**}(z) = \inf_{z_1 + z_2 = z} \frac{1}{2} \| z_1 \|^2 + i_{B_{d_{ij}}}(z_2) = \inf_{z_2 \in B_{d_{ij}}} \frac{1}{2} \| z_2 \|^2.
$$

APPENDIX B

LIPSCHITZ CONSTANT OF $\nabla \phi_{B_{d_{ij}}}$

We prove the inequality in (9):

$$
\left\| \nabla \phi_{B_{d_{ij}}}(x) - \nabla \phi_{B_{d_{ij}}}(y) \right\| \leq |x - y| \quad (30)
$$

where $\nabla \phi_{B_{d_{ij}}}(z) = z - P_{B_{d_{ij}}}(z)$ and $P_{B_{d_{ij}}}(z)$ is the projector onto $B_{d_{ij}} = \{ z \in \mathbb{R}^p : |z| \leq d_{ij} \}$. Squaring both sides of (30) gives the equivalent inequality

$$
2(P(x) - P(y))^\top (x - y) - |P(x) - P(y)|^2 \geq 0 \quad (31)
$$

where, to simplify notation, we let $P(z) := P_{B_{d_{ij}}}(z)$. Inequality (31) can be rewritten as

$$
(P(x) - P(y))^\top (x - y) + (P(x) - P(y))^\top (P(y) - y) + (P(x) - P(y))^\top (x - P(x)) \geq 0. \quad (32)
$$

By the properties of projectors onto closed convex sets, $(z - P(z))^\top (w - P(z)) \leq 0$, for any $w \in B_{d_{ij}}$ and any $z$, cf. [16, Th. 3.1.1, p. 117, v. 1]. Thus, the last two terms on the left-hand side of (32) are nonnegative. Moreover, the first term is nonnegative due to [16, Prop. 3.1.3, p. 118, v. 1]. Inequality (32) is proved.
Appendix C
AUXILIARY LEMMAS

In this Section we establish basic properties of Problem (7) in Lemma 5 and also two technical Lemmas, instrumental to prove our convergence results in Theorem 2.

Lemma 5 (Basic Properties): Let \( \hat{f} \) as defined in (6). Then the following properties hold.
1) \( \hat{f} \) is coercive;
2) \( \hat{f}^* \geq 0 \) and \( \mathcal{X}^* \neq \emptyset \);
3) \( \mathcal{X}^* \) is compact;

Proof:
1) By Assumption 1 there is a path from each node \( i \) to some node \( j \) which is connected to an anchor \( k \). If \( x_i \to \infty \) then there are two cases: (1) there is at least one edge \( t \sim u \) along the path from \( i \) to \( j \) where \( x_i \to \infty \) and \( x_u \to \infty \), and so \( d_{ij}^2(x_i, x_u) \to \infty \); (2) if \( x_i \to \infty \) for all \( u \) in the path between \( i \) and \( j \), in particular we have \( x_j \to \infty \) and so \( d_{R_{i \to j}}(x_j) \to \infty \), and in both cases \( \hat{f} \to \infty \), thus, \( \hat{f} \) is coercive.
2) Function \( \hat{f} \) defined in (6) is a sum of squares, it is continuous, convex and a real valued function, lower bounded by zero; so, the infimum \( f^* \) exists and is non-negative. To prove this infimum is attained and \( \mathcal{X}^* \neq \emptyset \), we consider the set \( T = \{ x : \hat{f}(x) \leq \alpha \} ; \) \( T \) is a sublevel set of a continuous, coercive function and, thus, it is compact.
As \( \hat{f} \) is continuous, by the Weierstrass Theorem, the value \( p = \inf_{x \in T} \hat{f}(x) \) is attained; the equality \( f^* = p \) is evident.
3) \( \mathcal{X}^* \) is a sublevel set of a continuous coercive function and, thus, compact.

Lemma 6: Let \( \{ x(k) \}_{k \in \mathbb{N}} \) be the sequence of iterates of Algorithm 2, or of Algorithm 2 with the update (20), and \( \nabla \hat{f}(x(k)) \) be the gradient of function \( \hat{f} \) evaluated at each iterate. Then,
1) \( \sum_{k \geq 1} \| \nabla \hat{f}(x(k)) \|^2 < \infty, a.s. \);
2) \( \nabla \hat{f}(x(k)) \to 0, a.s. \).

Proof: Let \( \mathcal{F}_k = \sigma \{ x(0), \ldots, x(k) \} \) be the sigma-algebra generated by all the algorithm iterations until time \( k \). We are interested in \( \mathbb{E} \left[ \hat{f}(x(k)) \mathcal{F}_{k-1} \right] \), the expected value of the cost value of the \( k \)th iteration, given the knowledge of the past \( k - 1 \) iterations. Firstly, let us examine function \( \phi : \mathbb{R}^p \to \mathbb{R} \), the slice of \( \hat{f} \) along a coordinate direction, \( \phi(y) = \hat{f}(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) \). As \( \hat{f} \) has Lipschitz continuous gradient with constant \( L_{ij} \), we have \( \| \nabla \phi(y) \| \leq L_{ij} \| y - z \| \), for all \( y \) and \( z \), and, thus, it will inherit the property
\[
\phi(y) \leq \phi(z) + \langle \nabla \phi(z), y - z \rangle + \frac{L_{ij}}{2} \| y - z \|^2. \tag{33}
\]
Inequality (33) is known as the Descent Lemma [28, Prop. A.24]. The minimizer of the quadratic upper-bound in (33) is \( z = -\frac{1}{L_{ij}} \nabla \phi(z) \), which can be plugged back in (33), obtaining
\[
\phi^* \leq \phi(z) - \frac{1}{L_{ij}} \nabla \phi(z) \leq \phi(z) - \frac{1}{2L_{ij}} \| \nabla \phi(z) \|^2. \tag{34}
\]
The sequel, for a given \( x = (x_1, \ldots, x_n) \), we let
\[
\tilde{f}^*_{x_1}(x_{-i}) = \inf \{ \hat{f}(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_n) : z \}.
\]
Going back to the expectation \( \mathbb{E} \left[ \hat{f}(x(k)) \mathcal{F}_{k-1} \right] = \sum_{i=1}^n P_i \hat{f}_{x_i}(x_{-i}(k - 1)) \), we can bound it from above, recurring to (34), by
\[
\sum_{i=1}^n P_i \left( \hat{f}(x(k - 1)) - \frac{1}{2L_{ij}} \| \nabla \hat{f}(x(k - 1)) \|^2 \right) = \hat{f}(x(k - 1)) - \frac{1}{2L_{ij}} \sum_{i=1}^n P_i \| \nabla \hat{f}(x(k - 1)) \|^2 \leq \hat{f}(x(k - 1)) - \frac{P_{\min}}{2L_{ij}} \| \nabla \hat{f}(x(k - 1)) \|^2, \tag{35}
\]
and adding \( \frac{P_{\min}}{2L_{ij}} \sum_{i=1}^n \| \nabla \hat{f}(x(k - 1)) \|^2 \) to both sides of the inequality in (35), we find that
\[
\mathbb{E} \left[ Y_k | \mathcal{F}_{k-1} \right] < Y_{k-1}; \tag{36}
\]
where \( Y_k = \hat{f}(x(k)) + \frac{P_{\min}}{2L_{ij}} \sum_{i=1}^n \| \nabla \hat{f}(x(k - 1)) \|^2 \). Inequality (36) defines the sequence \( \{ Y_k \}_{k \in \mathbb{N}} \) as a supermartingale. As \( \tilde{f}(x) \) is always non-negative, then \( Y_k \) is also non-negative and so [29, Corollary 27.1],
\[
Y_k \to Y, a.s.
\]
In words, the sequence \( Y_k \) converges almost surely to an integrable random variable \( Y \). This entails that \( \sum_{k \geq 1} \| g(k) \|^2 < \infty, a.s. \), and so, \( g(k) \to 0, a.s. \)

The previous arguments show that Lemma 6 holds for Algorithm 2. To show that Lemma 6 also holds for Algorithm 2 with the update (20) it suffices to redefine
\[
\tilde{f}^*_{x_1}(x_{-i}) = \hat{f}(x_1, x_{i+1}, \ldots, x_n) - \frac{1}{L_{ij}} \nabla \hat{f}(x_1, \ldots, x_n).
\]
As the second inequality in (34) shows, we have the bound
\[
\tilde{f}^*_{x_1}(x_{-i}(k - 1)) \leq \tilde{f}(x(k - 1)) - \frac{1}{L_{ij}} \| \nabla \hat{f}(x(k - 1)) \|^2
\]
and the rest of the proof holds intact.

Lemma 7: Let \( \{ x(k) \}_{k \in \mathbb{N}} \) be one of the sequences generated with probability one according to Lemma 6.

Then,
1) The function value decreases to the optimum: \( \hat{f}(x(k)) \downarrow \hat{f}^* \);
2) There exists a subsequence of \( \{ x(k) \}_{k \in \mathbb{N}} \) converging to a point in \( \mathcal{X}^* : x(k) \to y \), \( y \in \mathcal{X}^* \).

Proof: As \( \hat{f} \) is coercive, then the sublevel set \( \mathcal{X} = \{ x : \hat{f}(x) \leq \hat{f}(0) \} \) is compact and, because \( \hat{f}(x(k)) \) is non increasing, all elements of \( \{ x(k) \}_{k \in \mathbb{N}} \) belong to this set. From the compactness of \( \mathcal{X} \) we have that there is a convergent subsequence \( x(k_i) \to y \); we evaluate the gradient at this accumulation point, \( \nabla \hat{f}(y) = \lim_{i \to \infty} \nabla \hat{f}(x(k_i)) \), which, by assumption, vanishes, and we therefore conclude that \( y \) belongs
to the solution set $\mathcal{X}^*$. Moreover, the function value at this point is, by definition, the optimal value.

**APPENDIX D**

**PROOFS OF THEOREMS IN SECTION IV**

Equipped with the previous lemmas, we are now ready to prove the Theorems stated in Section IV.

**Proof of Theorem 2:** Suppose the distance does not converge to zero. Then, there exists an $\epsilon > 0$ and some subsequence $\{x(\ell_i)\}_{\ell_i \in \mathbb{N}}$ such that $d_{\mathcal{X}}(x(\ell_i)) > \epsilon$. But, as $\tilde{f}$ is coercive (by Lemma 5), continuous, and convex, and whose gradient, by Lemma 6, vanishes, then by Lemma 7, there is a subsequence of $\{x(\ell_i)\}_{\ell_i \in \mathbb{N}}$ converging to a point in $\mathcal{X}^*$, which is a contradiction. □

**Proof of Theorem 3:** Fix an arbitrary point $\mathbf{x}^* \in \mathcal{X}^*$. We start by proving that the sequence of squared distances to $\mathbf{x}^*$ of the estimate produced by Algorithm 2, with the update defined in (20), converges almost surely; that is, the sequence $\{x(k) - \mathbf{x}^*\}^2_{k \in \mathbb{N}}$ is convergent with probability one. We have

$$E \left[ \parallel x(k) - \mathbf{x}^* \parallel^2 \mathcal{F}_k \right] = 
\sum_{i=1}^{n} \frac{1}{n} \parallel x(k) - \frac{1}{L_j} g_i(k - 1) - \mathbf{x}^* \parallel^2$$

where $g_i(k - 1) = (0, \ldots, 0, \nabla_i \tilde{f}(x(k - 1)), 0, \ldots, 0)$ and $\mathcal{F}_k = \sigma(x(1), \ldots, x(k))$ is the sigma-algebra generated by all iterates until time $k$. Expanding the right-hand side of (37) yields

$$\parallel x(k - 1) - \mathbf{x}^* \parallel^2 + \frac{1}{L_j^2} \parallel \nabla \tilde{f}(x(k - 1)) \parallel^2$$

$$- \frac{2}{L_j} (x(k - 1) - \mathbf{x}^*)^T \nabla \tilde{f}(x(k - 1)).$$

Since $(x(k - 1) - \mathbf{x}^*)^T \nabla \tilde{f}(x(k - 1)) = (x(k - 1) - \mathbf{x}^*)^T (\nabla \tilde{f}(x(k - 1)) - \nabla \tilde{f}(\mathbf{x}^*)) \geq 0$, we conclude that

$$E \left[ \parallel x(k) - \mathbf{x}^* \parallel^2 \mathcal{F}_k \right] \leq \parallel x(k - 1) - \mathbf{x}^* \parallel^2 + \frac{1}{n L_j^2} \parallel \nabla \tilde{f}(x(k - 1)) \parallel^2.$$

Now, as proved in Lemma 6, the sum $\sum_k \parallel \nabla \tilde{f}(x(k)) \parallel^2$ converges almost surely. Thus, invoking the result in [30], we get that $\parallel x(k) - \mathbf{x}^* \parallel^2$ converges almost surely.

We can now invoke the technique at the end of the proof of [25, Prop. 9] to conclude that $x(k)$ converges to some optimal point $\mathbf{x}^*$. □

**ACKNOWLEDGMENT**

The authors would like to thank Pinar Oguz-Ekim and Andreea Simonetto for providing the Matlab implementation of the methods in their papers. We also thank the reviewers for the very interesting questions raised, and productive recommendations.

**REFERENCES**


[23] [AU: more information on source ans where to find]


Cláudia Soares (S’10) received the M.S. degree in electrical and computer engineering from Instituto Superior Tecnico (IST), Lisbon, Portugal, in 2007. She is currently working toward the Ph.D. degree in electrical and computer engineering in the Signal and Image Processing Group of the Institute for Systems and Robotics, IST. Her research interests include distributed optimization and sensor networks.

João Xavier (S’97–M’03) received the Ph.D. degree in Electrical and Computer Engineering from Instituto Superior Tecnico (IST), Lisbon, Portugal, in 2002. Currently, he is an Assistant Professor in the Department of Electrical and Computer Engineering, IST. He is also a Researcher at the Institute of Systems and Robotics (ISR), Lisbon, Portugal. His current research interests are in the area of optimization and statistical inference for distributed systems.

João Gomes (S’95–M’03) received the Diploma, M.S., and Ph.D. degrees in electrical and computer engineering from Instituto Superior Técnico (IST), Lisbon, Portugal, in 1993, 1996 and 2002, respectively. He joined the Department of Electrical and Computer Engineering of IST in 1995, where he is presently an Assistant Professor. Since 1994 he has also been a researcher in the Signal and Image Processing Group of the Institute for Systems and Robotics, in Lisbon. He currently serves as an Associate Editor for signal processing and communications in the IEEE JOURNAL OF OCEANIC ENGINEERING. His research interests include channel identification and equalization in wireless communications, underwater communications and acoustics, fast algorithms for adaptive filtering, and sensor networks.