

Simple and Fast Convex Relaxation Method for Cooperative Localization in Sensor Networks Using Range Measurements

Cláudia Soares, *Student Member, IEEE*, João Xavier, *Member, IEEE*, and João Gomes, *Member, IEEE*

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 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In the node set $\mathcal{V} = \{1, 2, \dots, n\}$ 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 $\mathcal{A} = \{1, \dots, m\}$. For each sensor $i \in \mathcal{V}$, we let $\mathcal{A}_i \subset \mathcal{A}$ be the subset of anchors (if any) whose distance to node i is quantified by a noisy range measurement. The set 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 $d_{ij} = d_{ji}$ ¹. 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 \mathcal{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

$$\underset{x}{\text{minimize}} f(x), \quad (1)$$

where

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

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 \hat{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.

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).

Manuscript received August 08, 2014; revised February 16, 2015 and June 18, 2015; accepted June 19, 2015. Date of publication July 09, 2015; date of current version July 21, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gesualdo Scutari. This work was partially supported by Fundação para a Ciência e a Tecnologia (grant SFRH/BD/72521/2010 and projects PTDC/EMS-CRO/2042/2012, UID/EEA/50009/2013) and EU FP7 project MORPH (grant agreement no. 288704).

The authors are with the Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, Universidade de Lisboa, Portugal (e-mail: csoares@isr.ist.utl.pt; jxavier@isr.ist.utl.pt; jpg@isr.ist.utl.pt).

Digital Object Identifier 10.1109/TSP.2015.2454853

We present an optimal synchronous and parallel algorithm to minimize this convex underestimator — with proven convergence guarantees. We also propose an asynchronous variant of 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 semi-definite 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' nodes. 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 non-linear 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 semidefinite 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 pre-defined 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

$$\underset{x}{\text{minimize}} \sum_{i \sim j} \frac{1}{2} d_{S_{ij}}^2(x_i - x_j) + \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{S_{aik}}^2(x_i), \quad (2)$$

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

$$S_{ij} = \{z : \|z\| = d_{ij}\}, \quad S_{aik} = \{z : \|z - a_k\| = r_{ik}\}.$$

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

$$\frac{1}{2} d_{S_{ij}}^2(z) = \frac{1}{2} \inf_{\|y\|=d_{ij}} \|z - y\|^2. \quad (3)$$

A simple convexification consists in replacing it by

$$\frac{1}{2} d_{B_{ij}}^2(z) = \frac{1}{2} \inf_{\|y\| \leq d_{ij}} \|z - y\|^2 \quad (4)$$

where $B_{ij} = \{z \in \mathbb{R}^p : \|z\| \leq d_{ij}\}$, is the convex hull of S_{ij} . Actually, (4) is the convex envelope² of (3). This fact is illustrated in Fig. 1 with a one-dimensional example; a formal proof for the generic case is given in Appendix A. The terms of

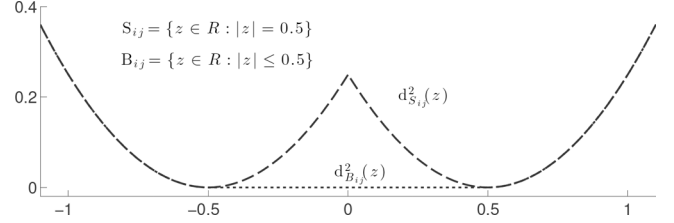


Fig. 1. Illustration of the convex envelope for intersensor terms of the non-convex cost function (2). The squared distance to the ball B_{ij} (dotted line) is the convex hull of the squared distance to the sphere S_{ij} (dashed line). In this one dimensional example the value of the range measurement is $d_{ij} = 0.5$.

(2) associated with anchor measurements are similarly relaxed as

$$d_{B_{aik}}^2(z) = \inf_{\|y - a_k\| \leq r_{ik}} \|z - y\|^2, \quad (5)$$

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

$$\underset{x}{\text{minimize}} \hat{f}(x) = \sum_{i \sim j} \frac{1}{2} d_{B_{ij}}^2(x_i - x_j) + \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{aik}}^2(x_i). \quad (6)$$

The function in Problem (6) is an underestimator 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

$$\underset{x}{\text{minimize}} \frac{1}{2} d_B^2(Ax) + \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{aik}}^2(x_i), \quad (7)$$

where $A = C \otimes I_p$, C is the arc-node incidence matrix of \mathcal{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 \mathcal{E} . We denote the two parcels in (7) as

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

²The convex envelope (or convex hull) of a function γ 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.

where $h_i(x_i) = \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{a_{ik}}}^2(x_i)$. 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

$$\begin{aligned} g(x) &= \frac{1}{2} d_B^2(Ax) \\ &= \frac{1}{2} \inf_{y \in B} \|Ax - y\|^2 \\ &= \frac{1}{2} \inf_{\|y_{ij}\| \leq d_{ij}} \sum_{i \sim j} \|x_i - x_j - y_{ij}\|^2 \end{aligned}$$

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

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

which is the corresponding term in (6).

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

To simplify notation, let us define the functions:

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

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

$$\nabla \phi_{B_{ij}}(z) = z - P_{B_{ij}}(z), \quad (8)$$

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.*,

$$\|\nabla \phi_{B_{ij}}(x) - \nabla \phi_{B_{ij}}(y)\| \leq \|x - y\|. \quad (9)$$

We show (9) in Appendix B.

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

$$\begin{aligned} \nabla g(x) &= A^\top \nabla \phi_B(Ax) \\ &= A^\top (Ax - P_B(Ax)) \\ &= \mathcal{L}x - A^\top P_B(Ax), \end{aligned} \quad (10)$$

where the second equality follows from (8) and $\mathcal{L} = A^\top A = L \otimes I_p$, 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

$$\begin{aligned} \|\nabla g(x) - \nabla g(y)\| &= \|A^\top (\nabla \phi_B(Ax) - \nabla \phi_B(Ay))\| \\ &\leq \|A\| \|Ax - Ay\| \\ &\leq \|A\|^2 \|x - y\| \\ &= \lambda_{\max}(A^\top A) \|x - y\| \\ &\stackrel{(a)}{=} \lambda_{\max}(L) \|x - y\| \\ &\leq \underbrace{2\delta_{\max}}_{L_g} \|x - y\|, \end{aligned} \quad (11)$$

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 δ_{\max} . A proof of the bound $\lambda_{\max}(L) \leq 2\delta_{\max}$ can be found in [17]³.

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

$$\nabla h_i(x_i) = \sum_{k \in \mathcal{A}_i} \nabla \phi_{B_{a_{ik}}}(x_i). \quad (12)$$

The gradient of h is also Lipschitz continuous. The constants L_{h_i} for ∇h_i are

$$\begin{aligned} \|\nabla h_i(x_i) - \nabla h_i(y_i)\| &\leq \sum_{k \in \mathcal{A}_i} \|\nabla \phi_{B_{a_{ik}}}(x_i) - \nabla \phi_{B_{a_{ik}}}(y_i)\| \\ &\leq |\mathcal{A}_i| \|x_i - y_i\|, \end{aligned} \quad (13)$$

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

$$\begin{aligned} \|\nabla h(x) - \nabla h(y)\| &= \sqrt{\sum_i \|\nabla h_i(x_i) - \nabla h_i(y_i)\|^2} \\ &\leq \sqrt{\sum_i |\mathcal{A}_i|^2 \|x_i - y_i\|^2} \\ &\leq \underbrace{\max(|\mathcal{A}_i| : i \in \mathcal{V})}_{L_h} \|x - y\|. \end{aligned} \quad (14)$$

We are now able to write $\nabla \hat{f}$, the gradient of our cost function, as

$$\nabla \hat{f}(x) = \mathcal{L}x - A^\top P_B(Ax) + \begin{bmatrix} \sum_{k \in \mathcal{A}_1} x_1 - P_{B_{a_{1k}}}(x_1) \\ \vdots \\ \sum_{k \in \mathcal{A}_n} x_n - P_{B_{a_{nk}}}(x_n) \end{bmatrix}. \quad (15)$$

A Lipschitz constant $L_{\hat{f}}$ is, thus,

$$L_{\hat{f}} = 2\delta_{\max} + \max(|\mathcal{A}_i| : i \in \mathcal{V}). \quad (16)$$

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

³A tighter bound would be $\lambda_{\max}(L) \leq \max_{i \sim j} \{\delta_i + \delta_j - c(i, j)\}$ where δ_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 $2\delta_{\max}$ is easier to compute in a distributed way.

In summary, we can compute the gradient of \hat{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 \hat{f} .

B. Parallel Method

Since \hat{f} has a Lipschitz continuous gradient we can follow Nesterov's optimal method [20]. Our approach is detailed in Algorithm 1. Step 5 computes the extrapolated points w_i in a standard application of Nesterov's method [21]. Steps 7 and 8, which constitute the core of the algorithm, correspond the i -th entry of $\nabla \hat{f}$ given in (15). Specifically, Step 7 coincides with the i th entry of $\mathcal{L}x - A^\top P_B(Ax)$ in (10) where $c_{(i \sim j, i)}$ denotes the entry $(i \sim j, i)$ in the arc-node incidence matrix C , and δ_i is the degree of node i . The i -th entry of $\mathcal{L}x$ can be computed by node i , from its current position estimate and the position estimates of the neighbors, in particular, it holds $(\mathcal{L}x)_i = \delta_i x_i - \sum_{j \in N_i} x_j$. The less obvious parallel term is $A^\top P_B(Ax)$. We start the analysis by the concatenated projections $P_B(Ax) = \{P_{B_{ij}}(x_i - x_j)\}_{i \sim j \in \mathcal{E}}$. Each one of those projections only depends on the edge terminals and the noisy 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: $L_{\hat{f}}; \{d_{ij} : i \sim j \in \mathcal{E}\}; \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}\};$
Output: \hat{x}

- 1: $k = 0;$
- 2: each node i chooses random $x_i(0) = x_i(-1);$
- 3: **while** some stopping criterion is not met, each node i **do**
- 4: $k = k + 1$
- 5: $w_i = x_i(k-1) + \frac{k-2}{k+1}(x_i(k-1) - x_i(k-2));$
- 6: node i broadcasts w_i to its neighbors
- 7: $\nabla g_i(w_i) = \delta_i w_i - \sum_{j \in N_i} w_j +$
 $+ \sum_{j \in N_i} c_{(i \sim j, i)} P_{B_{ij}}(w_i - w_j);$
- 8: $\nabla h_i(w_i) = \sum_{k \in \mathcal{A}_i} w_i - P_{B_{a_{ik}}}(w_i);$
- 9: $x_i(k) = w_i - \frac{1}{L_{\hat{f}}}(\nabla g_i(w_i) + \nabla h_i(w_i));$
- 10: **end while**
- 11: **return** $\hat{x} = x(k)$

C. Asynchronous Method

The method described in Algorithm 1 is fully parallel but still depends on some synchronization between all the nodes — so that their updates of the gradient are consistent. This requirement can be inconvenient in some applications of sensor networks; to circumvent it, we present a fully asynchronous method, achieved by means of a broadcast gossip scheme (cf. [22] for an extended survey of gossip algorithms).

Nodes are equipped with independent clocks ticking at random times (say, as Poisson point processes). When node i 's clock ticks, it performs the update of its variable x_i and broadcasts the update to its neighbors. Let the order of node activation be collected in $\{\xi_k\}_{k \in \mathbb{N}}$, a sequence of independent random variables taking values on the set \mathcal{V} , such that

$$\mathbb{P}(\xi_k = i) = P_i > 0. \quad (17)$$

Then, the asynchronous update of variable x_i on node i can be described as in Algorithm 2.

Algorithm 2: Asynchronous method

Input: $L_{\hat{f}}; \{d_{ij} : i \sim j \in \mathcal{E}\}; \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}\};$
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: $x_i(k) = \arg \min_{w_i} \hat{f}(x_1(k-1), \dots, w_i, \dots, x_n(k-1))$
- 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

$$\underset{x}{\text{minimize}} \sum_i \left(\sum_{j \in N_i} \frac{1}{4} d_{B_{ij}}^2(x_i - x_j) + \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{a_{ik}}}^2(x_i) \right), \quad (18)$$

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

$$\underset{x_i}{\text{minimize}} \hat{f}_{sl_i}(x_i) := \sum_{j \in N_i} \frac{1}{4} d_{B_{ij}}^2(x_i) + \sum_{k \in \mathcal{A}_i} \frac{1}{2} d_{B_{a_{ik}}}^2(x_i), \quad (19)$$

where $B_{s_{ij}} = \{z \in \mathbb{R}^p : \|z - x_j\| \leq d_{ij}\}$. 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: $\xi_k; L_{\hat{f}}; \{d_{ij} : j \in N_i\}; \{r_{ik} : k \in \mathcal{A}_i\};$
Output: $x_i(k)$

```

1: if  $\xi_k$  not  $i$  then
2:    $x_i(k) = x_i(k-1);$ 
3:   return  $x_i(k);$ 
4: end if
5: choose random  $z(0) = z(-1);$ 
6:  $l = 0;$ 
7: while some stopping criterion is not met do
8:    $l = l + 1;$ 
9:    $w = z(l-1) + \frac{l-2}{l+1}(z(l-1) - z(l-2));$ 
10:   $\nabla \hat{f}_{sl_i}(w) = \frac{1}{2} \sum_{j \in N_i} w - P_{B_{S_{ij}}}(w) + \sum_{k \in \mathcal{A}_i} w - P_{B_{a_{ik}}}(w)$ 
11:   $z(l) = w - \frac{1}{L_{\hat{f}}} \nabla \hat{f}_{sl_i}(w)$ 
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 \hat{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

$$x_i(k) = x_i(k-1) - \frac{1}{L_{\hat{f}}} \nabla_i \hat{f}(x(k-1)) \quad (20)$$

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

$$\mathbb{P}(\xi_k = i) = \frac{1}{n}. \quad (21)$$

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 $\phi_{B_{ij}}(z) = \phi_{S_{ij}}(z)$ for $\|z\| \geq d_{ij}$; when the functions differ, for $\|z\| < d_{ij}$, we have that $\phi_{B_{ij}}(z) = 0$. The same applies to the terms related

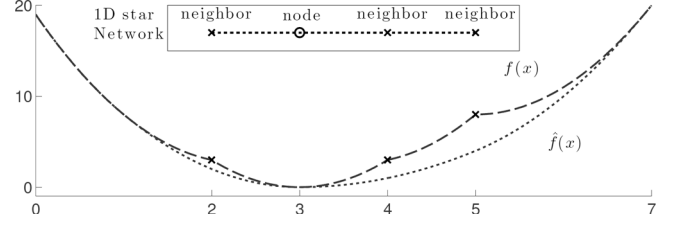


Fig. 2. One-dimensional example of the quality of the approximation of the true nonconvex cost $f(x)$ by the convexified function $\hat{f}(x)$ in a star network. Here the node positioned at $x = 3$ has 3 neighbors.

TABLE I
BOUNDS ON THE OPTIMALITY GAP FOR THE EXAMPLE IN FIG. 2

$f^* - \hat{f}^*$	Equation (22)	Equation (23)
0.0367	0.0487	3.0871

to anchor measurements. The optimal value of function f , denoted by f^* , is bounded by $\hat{f}^* = \hat{f}(x^*) \leq f^* \leq f(x^*)$, where x^* is the minimizer of the convexified problem (6), and $\hat{f}^* = \inf_x \hat{f}(x)$ is the minimum of function \hat{f} . With these inequalities we can compute a bound for the optimality gap, after (6) is solved, as

$$\begin{aligned}
f^* - \hat{f}^* &\leq f(x^*) - \hat{f}^* \\
&= \sum_{i \sim j \in \mathcal{E}} \frac{1}{2} \left(d_{S_{ij}}^2(x_i^* - x_j^*) - d_{B_{ij}}^2(x_i^* - x_j^*) \right) \\
&\quad + \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i} \frac{1}{2} \left(d_{S_{a_{ik}}}^2(x_i^*) - d_{B_{a_{ik}}}^2(x_i^*) \right) \\
&= \sum_{i \sim j \in \mathcal{E}_2} \frac{1}{2} d_{S_{ij}}^2(x_i^* - x_j^*) + \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_{2_i}} \frac{1}{2} d_{S_{a_{ik}}}^2(x_i^*). \quad (22)
\end{aligned}$$

In (22), we denote the set of edges where the distance of the estimated positions is less than the distance measurement by $\mathcal{E}_2 = \{i \sim j \in \mathcal{E} : d_{B_{ij}}^2(x_i^* - x_j^*) = 0\}$, and similarly $\mathcal{A}_{2_i} = \{k \in \mathcal{A}_i : d_{B_{a_{ik}}}^2(x_i^*) = 0\}$. Inequality (22) suggests a simple method to compute a bound for the optimality gap of the solution returned by the algorithms:

- 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).
- Our bound is tighter than the one (available *a priori*) from applying [23, Th. 1], which is

$$f^* - \hat{f}^* \leq \sum_{i \sim j \in \mathcal{E}} \frac{1}{2} d_{ij}^2 + \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{A}_i} \frac{1}{2} r_{ik}^2. \quad (23)$$

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 \mathcal{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:

$$\lim_{k \rightarrow \infty} d_{\mathcal{X}^*}(x(k)) = 0, \quad a.s. \quad (24)$$

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 one single $x^* \in \mathcal{X}^*$, but it does imply that $\lim_{k \rightarrow \infty} \hat{f}(x(k)) = \hat{f}^*$, 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

$$x(k) \rightarrow x^*. \quad (25)$$

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 a 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 $\rho \in (0, 1)$. There exists a constant $b(\rho, x(0))$ such that

$$\mathbb{P}(\hat{f}(x(k)) - \hat{f}^* \leq \epsilon) \geq 1 - \rho \quad (26)$$

for all

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

The constant $b(x(0), \rho)$ can be computed from inequality (19) in [26]; it depends only on the initialization and the chosen ρ . Proposition 4 is saying that, with high probability, the function value $\hat{f}(x(k))$ for all $k \geq K$ will be at a distance ϵ of the optimal, and the number of iterations K depends inversely on the chosen ϵ .

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 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 1×1 with 4 anchors in the corners of the square. Network 1 has 10 sensor nodes with an average node degree⁴ 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

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

⁴To characterize the used networks we resort to the concepts of *node degree* k_i , which is the number of edges connected to node i , and *average node degree* $\langle k \rangle = 1/n \sum_{i=1}^n k_i$.

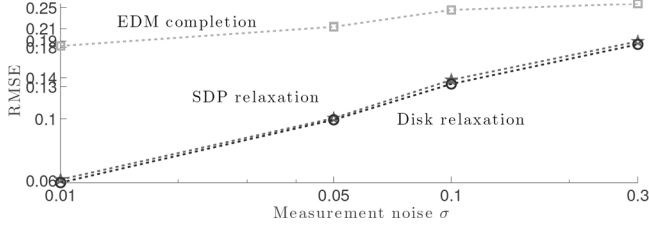


Fig. 3. Relaxation quality: Root mean square error comparison of EDM completion in [3], SDP relaxation in [2] and the disk relaxation (6), used in the present work; measurements were perturbed with noise with different values for the standard deviation σ . The disk relaxation approach in (6) improved 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.

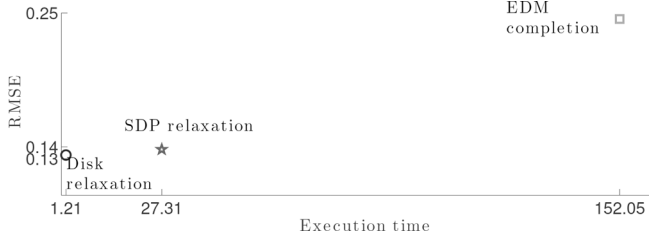


Fig. 4. Relaxation quality: Comparison of the best achievable root mean square error *versus* overall execution time of the algorithms. Measurements were contaminated with noise with $\sigma = 0.1$. Although disk relaxation (6) has a distributed implementation, running it sequentially can be faster by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

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 σ . The accuracy of the algorithms is measured by the original nonconvex 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] eps 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 of 200 runs, equally divided by the 4 noise levels. In Fig. 3 we can see that the disk relaxation in (6) has better performance for all noise levels. Fig. 4 locates the results of optimizing the three convex functions for the same problems in RMSE *versus* execution time, indicating the complexity of the optimization of the

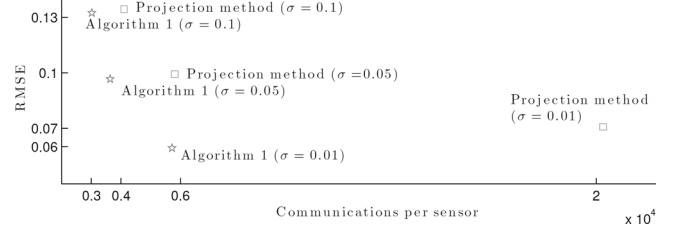


Fig. 5. Performance of the proposed method in Algorithm 1 and of the Projection method presented in [4]. The stopping criterion for both algorithms was a relative improvement of 10^{-6} in the estimate. The proposed method uses fewer communications to achieve better RMSE for the tested noise levels. Our method outperforms the projection method with one fourth of the number of communications for a noise level of 0.01.

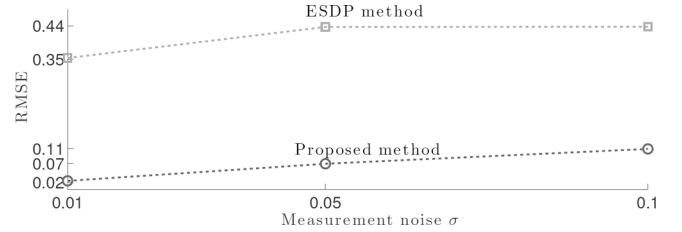


Fig. 6. Performance of the proposed method in Algorithm 1 and of the ESDP method in [2]. The stopping criterion for both algorithms was the number of algorithm iterations. The performance advantage of the proposed method in Algorithm 1 is even more remarkable when considering the number of communications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

considered costs. The convex surrogate (6) used in the present work, combined with our methods, is faster by at least one order of magnitude.

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

TABLE II
NUMBER OF COMMUNICATIONS PER SENSOR FOR THE RESULTS IN FIG. 6

ESDP method	Algorithm 1
21600	2000

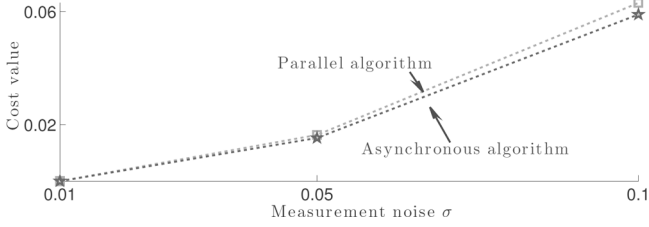


Fig. 7. Final cost of the parallel Algorithm 1 and its asynchronous counterpart in Algorithm 2 with an exact update for the same number of communications. Results for the asynchronous version degrade less than those of the parallel one as the noise level increases. The stochastic Gauss-Seidel iterations prove to be more robust to intense noise.

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 \hat{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 ξ_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 α as the function in (3) and β 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 α^* :

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

Thus, α^* is the sum of two closed convex functions: $\alpha^* = g + h$ where $g(s) = \frac{1}{2} \|s\|^2$ and $h(s) = d_{ij} \|s\|$. 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. 2.3.1, 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 $h^*(z_2) = i_{B_{ij}}(z_2)$ [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

$$\begin{aligned}
 \alpha^{**}(z) &= \inf_{z_1 + z_2 = z} \frac{1}{2} \|z_1\|^2 + i_{B_{ij}}(z_2) \\
 &= \inf_{z_2 \in B_{ij}} \frac{1}{2} \|z - z_2\|^2 \\
 &= \beta(z).
 \end{aligned}$$

APPENDIX B
LIPSCHITZ CONSTANT OF $\nabla\phi_{B_{ij}}$

We prove the inequality in (9):

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

where $\nabla\phi_{B_{ij}}(z) = z - P_{B_{ij}}(z)$ and $P_{B_{ij}}(z)$ is the projector onto $B_{ij} = \{z \in \mathbb{R}^p : \|z\| \leq d_{ij}\}$. 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 $P(z) := P_{B_{ij}}(z)$. 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 $w \in B_{ij}$ 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} be 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 $\|x_i\| \rightarrow \infty$ then there are two cases: (1) there is at least one edge $t \sim u$ along the path from i to j where $\|x_t\| \rightarrow \infty$ and $\|x_u\| \not\rightarrow \infty$, and so $d_{B_{tu}}^2(x_t - x_u) \rightarrow \infty$; (2) if $\|x_u\| \rightarrow \infty$ for all u in the path between i and j , in particular we have $\|x_j\| \rightarrow \infty$ and so $d_{B_{a_jk}}^2(x_j) \rightarrow \infty$, and in both cases $\hat{f} \rightarrow \infty$, 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 \hat{f}^* exists and is non-negative. To prove that 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 $\hat{f}^* = p$ is evident.
- 3) \mathcal{X}^* is a sublevel set of a continuous coercive function and, thus, compact. \square

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) $\nabla\hat{f}(x(k)) \rightarrow 0, a.s.$

Proof: Let $\mathcal{F}_k = \sigma(x(0), \dots, x(k))$ be the sigma-algebra generated by all the algorithm iterations until time k . We are interested in $\mathbb{E}[\hat{f}(x(k)) | \mathcal{F}_{k-1}]$, 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 $\phi : \mathbb{R}^p \rightarrow \mathbb{R}$, the slice of \hat{f} along a coordinate direction, $\phi(y) = \hat{f}(x_1, \dots, x_{i-1}, y, x_{i+1}, \dots, x_n)$. As \hat{f} has Lipschitz continuous gradient with constant $L_{\hat{f}}$, so will ϕ : $\|\nabla\phi(y) - \nabla\phi(z)\| \leq L_{\hat{f}}\|y - z\|$, for all y and z , and, thus, it will inherit the property

$$\phi(y) \leq \phi(z) + \langle \nabla\phi(z), y - z \rangle + \frac{L_{\hat{f}}}{2}\|y - z\|^2. \quad (33)$$

Inequality (33) is known as the Descent Lemma [28, Prop. A.24]. The minimizer of the quadratic upper-bound in (33) is $z - \frac{1}{L_{\hat{f}}}\nabla\phi(z)$, which can be plugged back in (33), obtaining

$$\phi^* \leq \phi\left(z - \frac{1}{L_{\hat{f}}}\nabla\phi(z)\right) \leq \phi(z) - \frac{1}{2L_{\hat{f}}}\|\nabla\phi(z)\|^2. \quad (34)$$

In the sequel, for a given $x = (x_1, \dots, x_n)$, we let

$$\hat{f}_i^*(x_{-i}) = \inf\{\hat{f}(x_1, \dots, x_{i-1}, z, x_{i+1}, \dots, x_n) : z\}.$$

Going back to the expectation $\mathbb{E}[\hat{f}(x(k)) | \mathcal{F}_{k-1}] = \sum_{i=1}^n P_i \hat{f}_i^*(x_{-i}(k-1))$, we can bound it from above, resorting to (34), by

$$\begin{aligned} & \sum_{i=1}^n P_i \left(\hat{f}(x(k-1)) - \frac{1}{2L_{\hat{f}}}\|\nabla_i \hat{f}(x(k-1))\|^2 \right) \\ &= \hat{f}(x(k-1)) - \frac{1}{2L_{\hat{f}}} \sum_{i=1}^n P_i \|\nabla_i \hat{f}(x(k-1))\|^2 \\ &\stackrel{(a)}{\leq} \hat{f}(x(k-1)) - \frac{P_{\min}}{2L_{\hat{f}}} \|\nabla\hat{f}(x(k-1))\|^2, \end{aligned} \quad (35)$$

where we used $0 < P_{\min} \leq P_i$, for all $i \in \{1, \dots, n\}$ in (a). To alleviate notation, let $g(k) = \nabla\hat{f}(x(k))$; we then have

$$\|g(k)\|^2 = \sum_{i \leq k} \|g(i)\|^2 - \sum_{i \leq k-1} \|g(i)\|^2,$$

and adding $\frac{P_{\min}}{2L} \sum_{i \leq k-1} \|g(i)\|^2$ to both sides of the inequality in (35), we find that

$$\mathbb{E}[Y_k | \mathcal{F}_{k-1}] \leq Y_{k-1}, \quad (36)$$

where $Y_k = \hat{f}(x(k)) + \frac{P_{\min}}{2L} \sum_{i \leq k-1} \|g(i)\|^2$. Inequality (36) defines the sequence $\{Y_k\}_{k \in \mathbb{N}}$ as a supermartingale. As $\hat{f}(x)$ is always non-negative, then Y_k is also non-negative and so [29, Corollary 27.1],

$$Y_k \rightarrow Y, a.s.$$

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, $g(k) \rightarrow 0$, *a.s.* \square

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

$$\hat{f}_i^*(x_{-i}) := \hat{f}\left(x_1, \dots, x_i - \frac{1}{L_{\hat{f}}} \nabla_i \hat{f}(x), \dots, x_n\right).$$

As the second inequality in (34) shows, we have the bound

$$\hat{f}_i^*(x_{-i}(k-1)) \leq \hat{f}(x(k-1)) - \frac{1}{L_{\hat{f}}} \left\| \nabla_i \hat{f}(x(k-1)) \right\|^2$$

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}^* : $x(k_l) \rightarrow y, y \in \mathcal{X}^*$.

Proof: As \hat{f} is coercive, then the sublevel set $\mathcal{X}_{\hat{f}} = \{x : \hat{f}(x) \leq \hat{f}(x(0))\}$ 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 $\mathcal{X}_{\hat{f}}$ we have that there is a convergent subsequence $x(k_l) \rightarrow y$. We evaluate the gradient at this accumulation point, $\nabla \hat{f}(y) = \lim_{l \rightarrow \infty} \nabla \hat{f}(x(k_l))$, 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. \square

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 $\{x(k_l)\}_{l \in \mathbb{N}}$ such that $d_{\mathcal{X}^*}(x(k_l)) > \epsilon$. But, as \hat{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 $\{x(k_l)\}_{l \in \mathbb{N}}$ converging to a point in \mathcal{X}^* , which is a contradiction. \square

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

$$\mathbb{E} [\|x(k) - x^*\|^2 | \mathcal{F}_{k-1}] = \sum_{i=1}^n \frac{1}{n} \left\| x(k-1) - \frac{1}{L_{\hat{f}}} g_i(k-1) - x^* \right\|^2, \quad (37)$$

where $g_i(k-1) = (0, \dots, 0, \nabla_i \hat{f}(x(k-1)), 0, \dots, 0)$ and $\mathcal{F}_k = \sigma(x(1), \dots, x(k))$ is the sigma-algebra generated by all

iterates until time k . Expanding the right-hand side of (37) yields

$$\begin{aligned} & \|x(k-1) - x^*\|^2 + \frac{1}{nL_{\hat{f}}^2} \left\| \nabla \hat{f}(x(k-1)) \right\|^2 \\ & - \frac{2}{nL_{\hat{f}}} (x(k-1) - x^*)^\top \nabla \hat{f}(x(k-1)). \end{aligned}$$

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

$$\mathbb{E} [\|x(k) - x^*\|^2 | \mathcal{F}_{k-1}] \leq \|x(k-1) - x^*\|^2 + \frac{1}{nL_{\hat{f}}^2} \left\| \nabla \hat{f}(x(k-1)) \right\|^2.$$

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

ACKNOWLEDGMENT

The authors would like to thank Pinar Oğuz-Ekim and Andrea 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

- [1] Q. Shi, C. He, H. Chen, and L. Jiang, "Distributed wireless sensor network localization via sequential greedy optimization algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3328–3340, Jun. 2010.
- [2] A. Simonetto and G. Leus, "Distributed maximum likelihood sensor network localization," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1424–1437, Mar. 2014.
- [3] P. Oguz-Ekim, J. Gomes, J. Xavier, and P. Oliveira, "Robust localization of nodes and time-recursive tracking in sensor networks using noisy range measurements," *IEEE Trans. Signal Process.*, vol. 59, no. 8, pp. 3930–3942, Aug. 2011.
- [4] M. Gholami, L. Tetrashvili, E. Strom, and Y. Censor, "Cooperative wireless sensor network positioning via implicit convex feasibility," *IEEE Trans. Signal Process.*, vol. 61, no. 23, pp. 5830–5840, Dec. 2013.
- [5] G. Destino and G. Abreu, "On the maximum likelihood approach for source and network localization," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4954–4970, Oct. 2011.
- [6] Y. Keller and Y. Gur, "A diffusion approach to network localization," *IEEE Trans. Signal Process.*, vol. 59, no. 6, pp. 2642–2654, Jun. 2011.
- [7] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 11, pp. 961–974, Nov. 2004.
- [8] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 4, pp. 360–371, Oct. 2006.
- [9] S. Korkmaz and A.-J. van der Veen, "Robust localization in sensor networks with iterative majorization techniques," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Apr. 2009, pp. 2049–2052.
- [10] J. Costa, N. Patwari, and A. Hero III, "Distributed weighted-multiplicative scaling for node localization in sensor networks," *ACM Trans. Sensor Netw. (TOSN)*, vol. 2, no. 1, pp. 39–64, 2006.
- [11] G. Calafiore, L. Carlone, and M. Wei, "Distributed optimization techniques for range localization in networked systems," in *49th IEEE Conf. Decision Control (CDC)*, Dec. 2010, pp. 2221–2226.
- [12] S. Srirangarajan, A. Tewfik, and Z.-Q. Luo, "Distributed sensor network localization using SOCP relaxation," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 4886–4895, Dec. 2008.
- [13] F. Chan and H. So, "Accurate distributed range-based positioning algorithm for wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 4100–4105, Oct. 2009.

- [14] U. Khan, S. Kar, and J. Moura, "DILAND: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1940–1947, Mar. 2010.
- [15] D. Blatt and A. Hero, "Energy-based sensor network source localization via projection onto convex sets," *IEEE Trans. Signal Process.*, vol. 54, no. 9, pp. 3614–3619, Sep. 2006.
- [16] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*. New York, NY, USA: Springer-Verlag, 1993.
- [17] F. R. Chung, *Spectral Graph Theory*. Providence, RI, USA: Amer. Math. Soc., 1997, vol. 92.
- [18] R. B. Bapat, *Graphs and Matrices*. New York, NY, USA: Springer, 2010.
- [19] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2010.
- [20] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Sov. Math. Doklady*, vol. 27, no. 2, pp. 372–376, 1983.
- [21] L. Vandenbergh, "Fast proximal gradient methods," EE236C Course Notes, 2014 [Online]. Available: <http://www.seas.ucla.edu/vandenbe/236C/lectures/fgrad.pdf>
- [22] D. Shah, *Gossip Algorithms*. Norwell, MA, USA: Now Publishers, 2009.
- [23] M. Udell and S. Boyd, "Bounding duality gap for problems with separable objective," 2014, ArXiv Preprint ArXiv:1410.4158v1 [Online]. Available: <http://arxiv.org/abs/1410.4158v1>
- [24] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Norwell, MA, USA: Kluwer, 2004.
- [25] D. Bertsekas, "Incremental proximal methods for large scale convex optimization," *Math. Programm.*, vol. 129, pp. 163–195, 2011.
- [26] Z. Lu and L. Xiao, "On the complexity analysis of randomized block-coordinate descent methods," 2013, Arxiv Preprint ArXiv:1305.4723 [Online]. Available: <http://arxiv.org/abs/1305.4723>
- [27] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Softw.* vol. 11-12, pp. 625–653, 1999 [Online]. Available: <http://fewcal.kub.nl/sturm>
- [28] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.
- [29] J. Jacod and P. Protter, *Probability Essentials*. New York, NY, USA: Springer, 2003, vol. 1.
- [30] H. Robbins and D. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications," in *Herbert Robbins Selected Papers*. New York, NY, USA: Springer, 1985, pp. 111–135.



Cláudia Soares (S'10) received the M.S. degree in electrical and computer engineering from Instituto Superior Técnico (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 Técnico (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.