Fast and Convergent Method for Large-Scale Sensor Network Localization With Anchor Uncertainty

Eyal Gur, Shoham Sabach

Abstract—**The localization of sensor networks involves determining the spatial coordinates of each sensor in a deployed network. This task is accomplished by utilizing noisy distance measurements between neighboring sensors and a limited number of anchor sensors, which possess known approximate locations.**

In this study, we present a unified maximum-likelihood formulation to address the sensor network localization problem. Our formulation incorporates anchor uncertainty by employing a truncated normal multivariate distribution supported on convex sets. This unified approach surpasses previous formulations as it is applicable to various network architectures and noise levels, and various anchor certainty or uncertainty scenarios.

Additionally, we introduce a novel algorithm that is simple, fast, fully distributed, and parallel. This algorithm converges to critical points of the non-convex, non-smooth, and constrained unified maximum-likelihood formulation. Notably, the algorithm is suitable for both centralized and distributed network architectures, making it well-suited for large-scale networks comprising even thousands of sensors.

Index Terms—**Sensor networks, localization, maximum-likelihood, non-convex non-smooth optimization, first-order methods, global convergence.**

I. Introduction

Sensor Networks (SNs), which include Wireless SNs, consist of small, energy-constrained devices that are deployed in large numbers to monitor physical or environmental conditions [2]. One of the key challenges in deploying these networks is accurately determining the location of each sensor, a process known as localization [11]. One popular technique for SN localization (SNL) relies on Time-of-Arrival (ToA) measurements: assuming all sensors are time-synchronized, each sensor measures the distance to its neighboring sensors based on the travel time of a signal from the sensor to its neighbours (see, for example, [9], [20], [21], [30] and also the recent survey paper [36] that describes aspects of SNs, including ToA and other popular measurement techniques). The task of SNL is then tackled by leveraging the (noisy) distance measurements and by utilizing the location of a limited number of anchor sensors, which are sensors with a measured approximate location.

The anchors acquire knowledge of their approximate (noisy) location either through the use of a GPS device or by means of manual anchor deployment [26], [30]. In real-life SNs, perfect knowledge of anchor locations may not be available due to inaccuracies in GPS measurements [26] or drift in anchor locations over long-term deployments [40]. Consequently, we consider uncertainty in the anchor locations, a concept that is mathematically described in Section II.

E. Gur and S. Sabach are with the Faculty of Data and Decision Sciences, Technion - Israel Institute of Technology, Haifa, Israel.

The SNL problem using the ToA technique can be formulated mathematically as a set of non-linear equations, where each equation describes the estimated difference between the range measurement and the true distance of any two neighbouring sensors. An additional set of equations describes the difference between the approximately measured location and the true location of each of the anchors. The problem of solving this system of equations can be formulated as optimization problems, which can be non-convex and non-smooth. As such, these optimization problems can be very challenging to solve (see, for example, [26], [42], [44] and other works listed below).

A popular mathematical formulation of the SNL problem as an optimization problem is the least squares formulation, which coincides with optimizing the Maximum-Likelihood (ML) function under some noise assumptions. This statistical interpretation is a desirable property when dealing with localization problems, as the optimal solution of the ML function is the solution that makes the observed data most probable. Unfortunately, the ML function is non-convex and non-smooth, and there is no closed-form formula to its optimal solution. Hence, many works tackle the SNL problem with a different formulation that only approximates the ML function. However, some works still directly tackle the original ML function by applying iterative algorithms, which are used to minimize the value of the ML function (see Section III-B).

When applying algorithms for SNL localization, the network's architecture plays a significant role. Some networks may posses a central processor, which collects data from all sensors and utilizes the gathered information in order to execute more complex computational operations. In this case, one can apply centralized algorithms for localization, which are algorithms that consider the entire network's geometry in each update. These type of algorithms generally produce accurate results, however they are computationally expensive and usually can only be implemented for small-scale networks. In the absence of a central processor, one can only apply distributed algorithms for localization, in which a sensor updates its own location using locally available information gathered from its neighboring sensors by some means of communication. While distributed algorithms generally produce less accurate results compared to their centralized counterparts, they consist of computationally inexpensive updates and can be applied for large and very large-scale networks. Some distributed algorithms allow parallel implementation, in which the sensors update their location in parallel rather then sequentially, and the overall running time is reduced (see, for example, the papers [28], [36], [41] for more information about the concept of centralized and distributed techniques in SNs).

In this paper, we present in Section II a unified maximum-likelihood formulation of the SNL problem that also assumes (if applicable) uncertainty in the anchor locations. This unified formulation can be used across all network architectures and it generalizes other existing formulations, as discussed in Section III. We also survey in Section III existing methods found in the literature that tackle the SNL problem. In Section IV, we develop an iterative method, which we call Fast Network Localization (FNL), that directly tackles the unified ML formulation with the assumption of uncertainty in the anchor locations. We also show that FNL can be implemented in a distributed fashion with parallel updates, hence FNL may be applicable for both centralized and distributed network architectures. In Section V, we prove that FNL globally converges to critical points of the unified ML formulation. In Section VI, we conduct numerical experiments to compare FNL with other methods.

II. A Unified Maximum-Likelihood Formulation

Consider a network of K sensors in $\mathbb{R}^m$ (usually $m = 2$ or $m = 3$, but any $m \geq 1$ can be set), where $N < K$ of which are non-anchor sensors and the remaining $K - N$ are anchor sensors. Each sensor has a unique index $1, 2, \ldots, K$, where the indices $1, 2, \ldots, N$ represent the N non-anchors and the indices $N+1, N+2, \ldots, K$ represent the $K - N$ anchors. We denote by $\mathcal{A}$ the set of all anchor indices.

For any two sensors $i, j \in \{1, 2, \ldots, K\}$, sensor j is called a neighbor of sensor i, if sensor i possesses a (noisy) distance measurement between the two, and we denote this relation as (i, j). In this case, the (noisy) distance measurement obtained by i is denoted as $d_{ij} > 0$. The set of all neighboring sensors is denoted by $\mathcal{E}$ (meaning, $(i, j) \in \mathcal{E}$ if and only if j is a neighbor of i and the measurement d_{ij} exists). Usually, a (noisy) distance measurement is gathered if the distance between a sensor to its neighbor is at most some maximal radio range for communication.

The neighboring relation is not necessarily symmetric. That is, if $(i, j) \in \mathcal{E}$ then either $(j, i) \in \mathcal{E}$ or $(j, i) \notin \mathcal{E}$. Indeed, in real-life networks, it is possible that one sensor possesses a distance measurement to its neighbor, but not the other way. If both sensors possess a distance measurement to each other (meaning, $(i, j) \in \mathcal{E}$ and $(j, i) \in \mathcal{E}$), then it could be that $d_{ij} \neq d_{ji}$. Some works (see, for example, [38]) assume that $d_{ij} = d_{ji}$ simply by taking the average of the two measurements. However, calculating the average requires the transfer of measurement data between neighboring sensors, which might increase the overall communication cost in distributed networks. Hence, in this work, we do not make such an assumption.

We denote by $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, 2, \ldots, K$, the vector variable representing the true unknown location of sensor i, and we denote by $\mathbf{x} \in \mathbb{R}^{mK}$ the vector variable obtained by concatenating all variables $\mathbf{x}_i \in \mathbb{R}^m$ into a single column vector. Conversely, for any vector $\mathbf{x} \in \mathbb{R}^{mK}$ and for any index $i = 1, 2, \ldots, K$, we denote by $\mathbf{x}_i \in \mathbb{R}^m$ the vector obtained from $\mathbf{x}$ by taking its $m \cdot (i-1) + 1$ to $m \cdot i$ coordinates (meaning, the coordinates of $\mathbf{x}$ corresponding to sensor i). Now, the true unknown distance between sensor i to its neighboring sensor j is given by $\|\mathbf{x}_i - \mathbf{x}_j\|$, where $\|\cdot\|$ is the Euclidean norm.

For any anchor sensor $l \in \mathcal{A}$, we denote by $\mathbf{a}_l \in \mathbb{R}^m$ the measured (noisy) location of anchor l. Usually, these location measurements are obtained, e.g., using a GPS device attached to each anchor sensor. In addition, for any anchor sensor $l \in \mathcal{A}$, we assume that its true unknown location $\mathbf{x}_l$ resides within an uncertainty set $\mathcal{C}_l \subseteq \mathbb{R}^m$ around its given measured location $\mathbf{a}_l \in \mathbb{R}^m$. The assumption that the true anchor location is within (possibly bounded) uncertainty set around the measured anchor location, addresses the fact that GPS location errors may be bounded [31]. We should mention that the notion of anchor uncertainty sets generalizes other anchor uncertainty and certainty scenarios, as discussed in Section III-A.

In this work, we assume that the uncertainty sets $\mathcal{C}_l$, $l \in \mathcal{A}$, are closed and convex. For example, if the uncertainty set $\mathcal{C}_l$ is a ball with radius $r_l > 0$ centered at the measured location $\mathbf{a}_l$, then the true anchor location $\mathbf{x}_l$ satisfies $\|\mathbf{x}_l - \mathbf{a}_l\| \leq r_l$. Additional examples for possible uncertainty sets are in Section IV-D. It should be noted that different uncertainty sets can be taken for different anchors.

Following the above, the true location of all sensors in the network can be described mathematically as the following set of non-linear equations

$$\begin{cases} d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| + \epsilon_{ij}, & \forall (i,j) \in \mathcal{E}, \\ \mathbf{a}_l = \mathbf{x}_l + \boldsymbol{\epsilon}_l, & \forall l \in \mathcal{A}, \end{cases} \tag{1}$$

where $\epsilon_{ij} \in \mathbb{R}$ and $\boldsymbol{\epsilon}_l \in \mathbb{R}^m$ are some noises. That is, the distance measurements d_{ij}, $(i, j) \in \mathcal{E}$, and the anchor location measurements $\mathbf{a}_l$, $l \in \mathcal{A}$, are viewed as random variables that depend on the unknown parameter to be estimated, which is the true network location.

In the SNL problem, we aim at estimating the location of each sensor in the network by solving the system of equations in (1). In this paper, and following many previous works such as [23], [26], [42], [44], we assume that each ϵ_{ij} follows a normal distribution with mean 0 and variance $\sigma_{ij}^2 > 0$, denoted by $\epsilon_{ij} \sim \mathcal{N}\left(0, \sigma_{ij}^2\right)$. As for the noises $\boldsymbol{\epsilon}_l \in \mathbb{R}^m$, $l \in \mathcal{A}$, recall that we consider in this work the restrictive assumption that the true anchor location to be estimated resides within an uncertainty set around the measured anchor location. That is, $\mathbf{x}_l \in \mathcal{C}_l$. Therefore, a statistical approach is to assume that $\boldsymbol{\epsilon}_l$ follows a multivariate truncated normal distribution supported on the shifted set $\mathcal{C}_l - \mathbf{a}_l \subseteq \mathbb{R}^m$, which will guarantee that the unknown location parameter $\mathbf{x}_l$ to estimated resides in the uncertainty set $\mathcal{C}_l$ around the measured location $\mathbf{a}_l$. By denoting this truncated normal distribution as $\mathcal{N}_l$, $l \in \mathcal{A}$, then $\boldsymbol{\epsilon}_l \sim \mathcal{N}_l\left(\mathbf{0}_m, \boldsymbol{\Sigma}_l\right)$ for some covariance matrix $\boldsymbol{\Sigma}_l \in \mathbb{R}^{m \times m}$.

In the ML estimation (MLE) approach, we utilize the set of equations in (1) in order to find an estimation for the true network location that makes the observed measurements most probable. If all noises are independent, the likelihood function of $\mathbf{x} \in \mathbb{R}^{mK}$ is given by

$$\mathcal{L}(\mathbf{x}) \equiv \mathcal{L}_d(\mathbf{x}) \cdot \mathcal{L}_a(\mathbf{x}), \tag{2}$$

where $\mathcal{L}_d(\mathbf{x})$ is the likelihood function of all distance measurements d_{ij}, $(i,j) \in \mathcal{E}$, and $\mathcal{L}_a(\mathbf{x})$ is the likelihood function of all anchor location measurements $\mathbf{a}_l$, $l \in \mathcal{A}$. For the function $\mathcal{L}_d$ it holds that [23], [26]

$$\mathcal{L}_d(\mathbf{x}) = \prod_{(i,j)\in\mathcal{E}} \frac{1}{\sigma_{ij}\sqrt{2\pi}} \exp\left(-\frac{(\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij})^2}{2\sigma_{ij}^2}\right). \tag{3}$$

As for the function $\mathcal{L}_a$, we first define for any positive-definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{m\times m}$ the induced norm $\|\mathbf{y}\|_{\mathbf{\Sigma}} \equiv \sqrt{\mathbf{y}^T\mathbf{\Sigma}^{-1}\mathbf{y}}$, for any $\mathbf{y} \in \mathbb{R}^m$. Now, it holds that

$$\mathcal{L}_a(\mathbf{x}) = \prod_{l\in\mathcal{A}} \frac{1}{Z_l} \exp\left(-\frac{1}{2}\|\mathbf{x}_l - \mathbf{a}_l\|_{\mathbf{\Sigma}_l}^2\right) \cdot \mathcal{I}_l(\mathbf{x}_l), \tag{4}$$

where $\mathcal{I}_l \colon \mathbb{R}^m \to \{0,1\}$ is the indicator function over the uncertainty set $\mathcal{C}_l$ defined as $\mathcal{I}_l(\mathbf{x}_l) = 1$ if $\mathbf{x}_l \in \mathcal{C}_l$ and 0 otherwise, and the constant $Z_l > 0$ is a normalizing term ensuring that the integral of the probability density function over $\mathcal{C}_l$ indeed equals to 1 (see, for example, the papers [1], [10], [19], [22], [43] for deriving probability density functions of multivariate truncated distributions supported on sets with positive measure).

Now, in order to define the objective function of the unified ML formulation of the SNL problem, we define the non-convex and non-smooth function $\tilde{\mathcal{F}} \colon \mathbb{R}^{mK} \to \mathbb{R}$ as

$$\tilde{\mathcal{F}}(\mathbf{x}) \equiv \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \frac{(\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij})^2}{\sigma_{ij}^2} + \frac{1}{2}\sum_{l\in\mathcal{A}} \|\mathbf{x}_l - \mathbf{a}_l\|_{\mathbf{\Sigma}_l}^2. \tag{5}$$

Notice that the task of maximizing the log of the likelihood function $\mathcal{L}(\mathbf{x})$ in (2) coincides with minimizing the non-convex and non-smooth function $\mathcal{F} \colon \mathbb{R}^{mK} \to \mathbb{R}$ defined as

$$\mathcal{F}(\mathbf{x}) \equiv \tilde{\mathcal{F}}(\mathbf{x}) - \sum_{l\in\mathcal{A}} \log(\mathcal{I}_l(\mathbf{x}_l)). \tag{6}$$

Therefore, we arrive at the following non-convex and non-smooth ML formulation of the SNL problem

$$\boxed{\min_{\mathbf{x}\in\mathbb{R}^{mK}} \quad \mathcal{F}(\mathbf{x}).} \tag{ML}$$

Notice that $-\log(\mathcal{I}_l(\mathbf{x}_l)) = 0$ if $\mathbf{x}_l \in \mathcal{C}_l$ and $+\infty$ otherwise. Hence, Problem (ML) can be written equivalently as the following problem with separable closed and convex constraints

$$\boxed{\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{mK}} \quad & \tilde{\mathcal{F}}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x}_l \in \mathcal{C}_l, \quad \forall l \in \mathcal{A}, \end{aligned}} \tag{ML}$$

where $\tilde{\mathcal{F}}$ is defined in (5).

Remark 1. In real-life SNL scenarios, it could be that the measurement variances, i.e., σ_{ij}^2 for any $(i,j) \in \mathcal{E}$ and $\mathbf{\Sigma}_l$ for any $l \in \mathcal{A}$, are unknown. In this case, one can estimate their value or their ratio (see, for example, [8]). Commonly, it is assumed that $\sigma_{ij} = 1$ and $\mathbf{\Sigma} = \mathbf{I}_m$, where $\mathbf{I}_m$ is the identity matrix of size $m \times m$ (see, for example, the works [15], [34]), which facilitates the objective function $\mathcal{F}$. We should mention that in the case where the measurement variances are known, then these values can be used in the objective function $\mathcal{F}$.

Remark 2. In the special case where $d_{ij} = d_{ji}$ for all $(i,j) \in \mathcal{E}$ and $(j,i) \in \mathcal{E}$ (that is, there exists a single distance measurement for every pair of neighbors), then all pairs $(j,i) \in \mathcal{E}$ for which $j > i$, can be removed from the sum in (5), since these pairs do not provide additional information to the system of equations in (1).

Following the definition of the function $\mathcal{F}$ in (6), its domain is the set

$$\operatorname{dom}(\mathcal{F}) \equiv \mathbb{R}^{mN} \times \left(\bigtimes_{l\in\mathcal{A}} \mathcal{C}_l\right) \subset \mathbb{R}^{mK}. \tag{7}$$

That is, $\mathcal{F}(\mathbf{x}) < \infty$ if and only if $\mathbf{x} \in \operatorname{dom}(\mathcal{F})$.

We point out that the function $\mathcal{F}$ of Problem (ML) is non-convex and non-smooth, hence it is not possible to obtain a closed-form formula to a minimizer of Problem (ML). Still, criticality is a necessary condition for optimality for non-convex and non-smooth problems. Hence, we focus on developing an iterative algorithm that globally converges to critical points of the function $\mathcal{F}$, where by critical point we mean a point $\mathbf{x} \in \mathbb{R}^{mK}$ satisfying $\mathbf{0}_{mK} \in \partial\mathcal{F}(\mathbf{x})$. Since we are dealing with non-convex and non-smooth functions, when we write $\partial\mathcal{F}$ we mean the limiting sub-differential set of $\mathcal{F}$ [24]. By a globally convergent algorithm, we mean that the entire sequence generated by the algorithm converges to a unique point, a result that is stronger than sub-sequence or function value convergence. In the world of non-convex and non-smooth optimization, the state-of-the-art in recent years is to design algorithms that globally converge to critical points of the function at hand.

We conclude this section with a short summary of the main results of this paper.

1) We develop a novel, simple and well-defined iterative algorithm, called FNL (see Section IV), that directly tackles the most general formulation as given in Problem (ML), rather than being tailored to a specific instance.
2) FNL can be implemented in a distributed and parallel fashion, and is therefore suitable for both centralized and distributed network architectures. In addition, FNL is suitable for both high and low noise levels.
3) FNL only uses first-order information, and hence is fast with simple updates, and enjoys low computational, storage and communication burden. As such, FNL can be applied on very large-scale networks that include even tens of thousands of sensors.
4) The sequence generated by FNL is proven to be globally convergent to critical points of Problem (ML).

III. Relations to Other Formulations and Literature Review

Problem (ML), as defined above, can be viewed as a generalization of other ML formulations found in the literature. We outline these connections in the sub-sequent subsection, followed by an exploration of existing literature.

A. Problem (ML) and Other Formulations

By taking $\mathcal{C}_l = \{\mathbf{a}_l\}$ in Problem (ML), for all $l \in \mathcal{A}$, we recover the case where perfect knowledge of anchor location is available. In this case, Problem (ML) reduces to the following formulation that assumes anchor certainty

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^{mN}} \quad & \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} \frac{\left(\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij}\right)^2}{\sigma_{ij}^2} \\ \text{s.t.} \quad & \mathbf{x}_l = \mathbf{a}_l, \quad \forall l \in \mathcal{A}. \end{aligned} \tag{ML-C}$$

Notice that the equality constraints in Problem (ML-C) can be plugged into the objective function, and therefore Problem (ML-C) is an unconstrained optimization problem.

Another instance of the SNL problem that can be obtained from Problem (ML), is the case where the estimated anchor positions $\mathbf{a}_l$, $l \in \mathcal{A}$, are still uncertain, however the true location $\mathbf{x}_l$ is not restricted to reside in any uncertainty set around $\mathbf{a}_l$. Mathematically, this case is obtained from Problem (ML) by taking $\mathcal{C}_l = \mathbb{R}^m$. In this setting, the constraints $\mathbf{x}_l \in \mathcal{C}_l$ in Problem (ML) are now redundant, and Problem (ML) reduces to the following unconstrained with anchor uncertainty formulation

$$\min_{\mathbf{x} \in \mathbb{R}^{mK}} \quad \tilde{\mathcal{F}}(\mathbf{x}), \tag{ML-U}$$

where $\tilde{\mathcal{F}}$ is defined in (5).

Another interesting case that follows from Problem (ML), is when each of the covariance matrices is the identity matrix (or a scalar multiplication of the identity), and the uncertainty sets $\mathcal{C}_l$, $l \in \mathcal{A}$, are balls centered at $\mathbf{a}_l$ with radius $r_l > 0$. Simple algebraic calculations show that in this setting Problem (ML) reduces to the following non-linear least squares with ℓ_2 regularization problem over separable ball constraints

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^{mK}} \quad & \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} \frac{\left(\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij}\right)^2}{\sigma_{ij}^2} + \frac{1}{2} \sum_{l \in \mathcal{A}} \|\mathbf{x}_l - \mathbf{a}_l\|^2 \\ \text{s.t.} \quad & \|\mathbf{x}_l - \mathbf{a}_l\| \leq r_l, \quad \forall l \in \mathcal{A}. \end{aligned} \tag{ML-L}$$

It should be noted that many additional formulations of the SNL problem can be derived from Problem (ML) by considering different values of the given data parameters $\mathbf{\Sigma}_l$, $\mathcal{C}_l$ and σ_{ij}. However, the three instances described above, which are Problems (ML-C), (ML-U) and (ML-L) (or some other closely related variants of these problems), are the main ML formulations found in the literature for the SNL problem.

B. Existing Works for the SNL Problem

There has been a significant research focused on developing optimization algorithms to improve the localization accuracy in sensor networks. These algorithms seek to minimize errors in sensor locations by leveraging some known measurements. Here, we focus on algorithms that use the ToA technique.

Roughly, the existing ToA literature for SNL is categorized into three: (i) methods that tackle convex relaxations of one of the instances of Problem (ML) described in Section III-A, (ii) methods that use a different formulation obtained by taking the square of the measurements, and (iii) methods that directly tackle one of the instances of the ML formulation. We now discuss the three categories and survey some related works.

1) Convex relaxations: due to the non-convexity of the ML formulation, a common approach is to relax it into a convex problem. For example, in [23], [26], [35], [39], [44] and others, Problem (ML-U) or Problem (ML-C) are relaxed into a convex Semi-Definite Programming (SDP) or Second-Order Cone Programming (SOCP) problems, which are solved using SDP solvers that utilize interior-point methods. Some of the SDP and SOCP formulations can even be implemented in a distributed fashion. Though these methods are guaranteed to converge to the optimal solution of the relaxation, this solution is not even a critical point of the original ML formulation (see in this regard the discussion in [32]). In addition, solving SDP problems involves a high computational burden, and hence these method are not always suitable for large-scale networks that contain hundreds or thousands of sensors.

Not all convex relaxations are solved using SDP solvers. In [38], the authors suggest a simple distributed and parallel first-order algorithm, called SF, that solves a convex relaxation of Problem (ML-C). The authors of [38] equivalently rewrite Problem (ML-C) with non-convex sphere constraints, which are then relaxed into convex ball constraints. Then, iterations of the Accelerated Projected Gradient method of Nesterov [27] are applied, and their simplicity allow algorithm SF to be used on very large-scale networks. However, this relaxation is still non-smooth, hence algorithm SF is not necessarily well-defined since it calculates the gradient at each iteration. Moreover, as in other works that tackle convex relaxations, a solution obtained by SF is not even a critical point of the original ML formulation.

2) Squaring the measurements: since Problem (ML) is non-smooth, several works tackle this issue by squaring the measurements. Meaning, taking the square on both sides of the system of equations in (1), which results with the formulation

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^{mK}} \quad & \sum_{(i,j) \in \mathcal{E}} \left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2\right)^2 + \sum_{l \in \mathcal{A}} \|\mathbf{x}_l - \mathbf{a}_l\|^2 \\ \text{s.t.} \quad & \mathbf{x}_l \in \mathcal{C}_l, \quad \forall l \in \mathcal{A}. \end{aligned} \tag{8}$$

We should mention that other problem formulations that follow from the squaring technique can be considered. Though still non-convex, Problem (8) (or some variant of this problem) allows for optimization methods from the world of smooth analysis. For example, in [7] the authors consider the unconstrained variant of Problem (8), and apply iterations of the well-known Gradient Descent method. In [34], the authors use convex relaxation techniques on Problem (8), which results with a distributed algorithm that is solved using SDP solvers.

The solutions obtained by algorithms that tackle Problem (8), or any other variant of this problem, lack the desired statistical interpretation as the ML estimator. Additionally, by squaring the measurements, the noise terms now depend on the original noises as well as on the true distance between the sensors, hence Problem (8) only approximates the ML formulation for small enough noises [8].

3) Direct methods: In [34], the authors tackle Problem (ML-L) and develop their SGO algorithm. This method applies the Alternating Minimization scheme, where in each sub-problem the objective function is minimized only with respect to a single sensor. For non-anchor sensors, the exact minimizer is found by solving a quadratic convex and unconstrained sub-problem. For the anchors, the unconstrained minimizer is projected onto its corresponding ball constraint to obtain a minimizer of the sub-problem. It should be noted that even though the SGO updates are simple and can be implemented for very large-scale networks, they may not be always well-defined. In addition, SGO cannot be implemented in parallel, which might increase the runtime and communication costs.

In [15], the authors suggest a family of algorithms called AMU, that tackles Problem (ML-C). This first-order algorithmic framework consists of simple and well-defined updates in the range of fully centralized to fully distributed, depending on the network's architecture, by splitting the network into clusters. As SGO of [34], the AMU method also utilizes the Alternating Minimization scheme and each sub-problem is exactly solved (in the fully distributed case, it can be shown that AMU coincides with SGO). In addition, AMU can be implemented on very large-scale networks, even in the fully centralized setting. AMU was the first method to enjoy global convergence guarantees to critical points of the ML formulation of the SNL problem. However, AMU cannot be implemented in parallel, which limits its applicability.

In [37], the authors rewrite Problem (ML-C) with non-convex sphere constraints. Then, instead of relaxing the constraints to ball constraints as in [38], they apply the Accelerated Projected Gradient method of Nesterov on the problem with the sphere constraints, which results with a first-order and simple algorithm that can be applied on large-scale networks. However, since the constraints are non-convex, the resulting algorithm lacks convergence guarantees and is not necessarily well-defined (see, for example, the book [6] for convergence analysis of gradient methods in the non-convex setting).

In [42], the authors tackle Problem (ML-U) by rewriting it as a Difference-of-Convex (DC) programming problem. Then, the DC formulation and its dual are solved iteratively. This well-defined primal-dual algorithm is shown to outperform some SDP algorithms, at least for networks where all sensors are located in the convex hull of the anchors. However, a system of equations needs to be solved in each iteration of this algorithm, which limits its applicability.

In [32], the authors write Problem (ML-C) with equality constraints and develop a fully distributed and parallel two-stage method called ADMM-H, that is based on the well-known Alternating Direction Method of Multipliers in the first stage, and on a Newton's method in the second. It should be noted that ADMM-H requires tuning of hyper-parameters for different networks and its iterations are not always well-defined. In addition, the sequence generated by either the DC algorithm of [42] or ADMM-H of [32], only has sub-sequence convergence to critical points of Problem (ML-C), which is a weaker result than global convergence to critical points.

We should also mention that some heuristic algorithms for Problem (ML-C) also exist in the literature, such as [25] that uses the Nelder-Mead simplex algorithm, and [13], [29] that use genetic algorithms. However, these heuristic approaches have either weak or no theoretical guarantees on the convergence of the generated sequence.

To conclude this sub-section, we summarize in Table I properties of several algorithms found in the literature.

TABLE I: Comparison of algorithms.

Method	Centralized	Distributed		Globally converges
		Sequential	Parallel	
FNL	✓	✓	✓	✓
SF [38]	✓	✓	✓	✗
ADMM-H [32]	✗	✓	✓	✗
SGO [34]	✓	✓	✗	✗
DC [42]	✓	✗	✗	✗
AMFC [15]	✓	✗	✗	✓
AMFD [15]	✓	✓	✗	✓
EML [35]	✓	✗	✗	✗
SOCP [26]	✓	✗	✗	✗

IV. The FNL Algorithm

In this section, we develop the FNL algorithm, which is a distributed with parallel implementation algorithm that directly tackles the non-convex and non-smooth ML formulation of the SNL problem as given in Problem (ML). We will show that FNL always generates well-defined iterates (regardless of the initialization point), that consist of simple update steps and are given by explicit and inexpensive operations. In Section V, we will also prove that FNL always globally converges to critical points of Problem (ML).

To achieve these goals, we first reformulate Problem (ML) as an equivalent problem. While the reformulation remains non-convex and non-smooth, it admits a structure that allows us to develop the FNL algorithm by utilizing the recent Nested Alternating Minimization (NAM) optimization scheme of [16].

A. Problem Reformulation

We first reformulate Problem (ML) into an equivalent non-convex and non-smooth problem. To achieve this, we introduce $\mathcal{B}$ and $\mathcal{S}$ as the unit ball and sphere in $\mathbb{R}^m$ centered at $\mathbf{0}_m$ with a radius of 1, and $\mathcal{B}^{|\mathcal{E}|}$ as the Cartesian product of $|\mathcal{E}|$ unit balls. Then, we define the non-convex quadratic function $\varphi\colon \mathbb{R}^{mK} \times \mathcal{B}^{|\mathcal{E}|} \to \mathbb{R}$ as

$$\varphi(\mathbf{x},\mathbf{u}) \equiv \sum_{(i,j)\in\mathcal{E}} \sigma_{ij}^{-2}\left(\frac{1}{2}\|\mathbf{x}_i-\mathbf{x}_j\|^2 - d_{ij}\mathbf{u}_{ij}^T(\mathbf{x}_i-\mathbf{x}_j)\right) + \frac{1}{2}\sum_{l\in\mathcal{A}}\|\mathbf{x}_l-\mathbf{a}_l\|_{\mathbf{\Sigma}_l}^2 + \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}\sigma_{ij}^{-2}d_{ij}^2, \tag{9}$$

where the variable $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$ is the concatenation of all the variable vectors $\mathbf{u}_{ij} \in \mathcal{B}$, $(i,j) \in \mathcal{E}$, into a single column. We now define the non-convex function $\Psi\colon \mathbb{R}^{mK} \times \mathcal{B}^{|\mathcal{E}|} \to \mathbb{R}$ as

$$\Psi(\mathbf{x},\mathbf{u}) \equiv \varphi(\mathbf{x},\mathbf{u}) - \sum_{l\in\mathcal{A}}\log(\mathcal{I}_l(\mathbf{x}_l)), \tag{10}$$

where the indicator function $\mathcal{I}_l$ is defined in (4).

By employing similar algebraic derivations as in [14], [15], we derive that $\mathcal{F}$ in Problem (ML) adheres to the upper bound

$$\mathcal{F}(\mathbf{x}) \le \Psi(\mathbf{x},\mathbf{u}), \quad \mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}, \tag{11}$$

with equality attained by taking the unit sphere vectors

$$\bar{\mathbf{u}}_{ij}(\mathbf{x}) \equiv \begin{cases} \frac{\mathbf{x}_i-\mathbf{x}_j}{\|\mathbf{x}_i-\mathbf{x}_j\|}, & \mathbf{x}_i \ne \mathbf{x}_j, \\ \mathbf{e}_1, & \mathbf{x}_i = \mathbf{x}_j, \end{cases} \tag{12}$$

for all $(i,j) \in \mathcal{E}$, where $\mathbf{e}_1 \in \mathcal{S}$ the unit vector with 1 in its first coordinate and 0 otherwise (though any other unit ball vector can be taken). In other words, by taking $\mathbf{u} = \bar{\mathbf{u}}(\mathbf{x}) \in \mathcal{S}^{|\mathcal{E}|}$, where $\bar{\mathbf{u}}(\mathbf{x})$ is the concatenation of all $\bar{\mathbf{u}}_{ij}(\mathbf{x})$, $(i,j) \in \mathcal{E}$, into a single column, we express Problem (ML) equivalently as

$$\boxed{\min_{\mathbf{x} \in \mathbb{R}^{mK}} \left\{ \mathcal{F}(\mathbf{x}) = \Psi(\mathbf{x}, \bar{\mathbf{u}}(\mathbf{x})) \right\}.} \tag{ML}$$

Problem (ML) in matrix form. As mentioned in (9), the function $\varphi(\mathbf{x}, \mathbf{u})$ is quadratic with respect to $(\mathbf{x}, \mathbf{u})$. In particular, Problem (ML) can be written in matrix form. This form will enable us to present the suggested FNL algorithm in a centralized implementation as well as in a distributed and parallel implementation (see Algorithms 1 and 2 below). To this end, we define the following matrices and vectors.

- $\tilde{\mathbf{S}} \in \mathbb{R}^{|\mathcal{E}| \times K}$: defined such that if $(i,j) \in \mathcal{E}$ is the n-th element in the set $\mathcal{E}$, then $\tilde{\mathbf{S}}_{ni} = \sigma_{ij}^{-1}$, $\tilde{\mathbf{S}}_{nj} = -\sigma_{ij}^{-1}$ and all other entries are 0 (this is the so-called edge-sensor incidence matrix of the network).
- $\tilde{\mathbf{D}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$: defined such that its main diagonal entries are $\sigma_{ij}^{-1} d_{ij}$, $(i,j) \in \mathcal{E}$, and all other entries are 0.
- Since the dimension of the problem is $m \geq 1$, we define the matrices $\mathbf{S} \equiv \tilde{\mathbf{S}} \otimes \mathbf{I}_m$ and $\mathbf{D} \equiv \tilde{\mathbf{D}} \otimes \mathbf{I}_m$, where $\otimes$ denotes the Kronecker matrix product.
- $\mathbf{A} \in \mathbb{R}^{mK \times mK}$: a block main diagonal matrix with K blocks on the diagonal, each of size $m \times m$, where all entries off of the block main diagonal are 0. For any sensor $i = 1, 2, \ldots, K$, the block on the main diagonal corresponding to i is $\mathbf{0}_{m \times m}$ if $i \notin \mathcal{A}$ and $\mathbf{\Sigma}_i^{-1}$ otherwise.
- $\mathbf{a} \in \mathbb{R}^{mK}$: the concatenation of the vector $\mathbf{0}_{mN}$ followed by all vectors $\mathbf{a}_l \in \mathbb{R}^m$, $l \in \mathcal{A}$, into a single vector.

Using the above notations, one can see after simple calculations that $\varphi\left(\mathbf{x}, \mathbf{u}^k\right)$, as defined in (9), can be written as

$$\varphi(\mathbf{x}, \mathbf{u}) = \frac{1}{2} \mathbf{x}^T \mathbf{P} \mathbf{x} - \mathbf{b}(\mathbf{u})^T \mathbf{x} + \text{const}, \tag{13}$$

where we define the matrix $\mathbf{P} \equiv \mathbf{S}^T \mathbf{S} + \mathbf{A} \in \mathbb{R}^{mK \times mK}$, the vector $\mathbf{b}(\mathbf{u}) \equiv \mathbf{D} \mathbf{S}^T \mathbf{u} + \mathbf{A} \mathbf{a} \in \mathbb{R}^{mK}$ for any $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$, and const $\equiv \frac{1}{2} \mathbf{a}^T \mathbf{A} \mathbf{a} + \sum_{(i,j) \in \mathcal{E}} \sigma_{ij}^{-2} d_{ij}^2$.

Again, and as discussed above, Problem (ML) can be written as $\min_{\mathbf{x} \in \mathbb{R}^{mK}} \left\{ \mathcal{F}(\mathbf{x}) = \Psi(\mathbf{x}, \bar{\mathbf{u}}(\mathbf{x})) \right\}$, where the function Ψ is defined in (10) and the function φ is defined in (13).

In the next sub-section, we develop the FNL algorithm for tackling Problem (ML).

B. Our Algorithm

In (11), we established that for any $\mathbf{x} \in \mathbb{R}^{mK}$, the function $\mathcal{F}(\mathbf{x})$ is bounded from above by the function $\Psi(\mathbf{x}, \mathbf{u})$, $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$, and that this upper bound is attained for $\mathbf{u} = \bar{\mathbf{u}}(\mathbf{x}) \in \mathcal{S}^{|\mathcal{E}|}$. Hence, the task of minimizing $\mathcal{F}$ coincides with the task of minimizing the function Ψ in an alternating fashion: first, update the vector $\bar{\mathbf{u}}(\mathbf{x}) \in \mathcal{S}^{|\mathcal{E}|}$, and then minimize Ψ with respect to $\mathbf{x} \in \mathbb{R}^{mK}$ using some iterative nested algorithm (see below), and repeat the process until a stopping criterion is satisfied. This Nested Alternating Minimization (NAM) was developed in [16].

In order to minimize the function Ψ with respect to $\mathbf{x} \in \mathbb{R}^{mK}$, we see that the partial function $\mathbf{x} \mapsto \Psi(\mathbf{x}, \bar{\mathbf{u}}(\mathbf{x}))$ is given as the sum of non-smooth and convex indicator terms (as defined in (10)) with the quadratic partial function $\mathbf{x} \mapsto \varphi(\mathbf{x}, \bar{\mathbf{u}}(\mathbf{x}))$ (as defined in (9)). Therefore, a natural candidate to minimize $\mathbf{x} \mapsto \Psi(\mathbf{x}, \bar{\mathbf{u}}(\mathbf{x}))$ is the well-known Accelerated Projected Gradient (APG) method of [27]. More precisely, at any iteration $k \geq 0$ of NAM, we apply (inner) iterations of APG as a nested algorithm.

Now, we are ready to present our FNL algorithm.

Algorithm 1 Fast Network Localization (FNL)

```
1: Initialization: $\mathbf{x}^0 \in \mathbb{R}^{mK}$.
2: for $k \geq 0$ do
3:     Update $\mathbf{u}_{ij}^k \equiv \bar{\mathbf{u}}_{ij}\left(\mathbf{x}^k\right)$, $(i,j) \in \mathcal{E}$, according to (12).
4:     Update $\mathbf{x}^{k+1}$ by applying $n_k \in \mathbb{N}$ iterations of APG to
       minimize the function $\mathbf{x} \mapsto \Psi\left(\mathbf{x}, \mathbf{u}^k\right)$ (see Algorithm 2 below),
       starting with $\mathbf{x}^k$.
5: end for
```

Remark 3. (i) In step 3 of FNL, for any pair $(i,j) \in \mathcal{E}$ and $k \geq 0$, the update of $\mathbf{u}_{ij}^k$ has a simple and inexpensive closed-form formula that depends only on the previous iteration $\mathbf{x}^k$. Therefore, the auxiliary vectors $\mathbf{u}_{ij}^k$ can be eliminated from FNL by substituting them in the update of $\mathbf{x}^{k+1}$, and FNL can be written only in terms of the original variable $\mathbf{x} \in \mathbb{R}^{mK}$.

(ii) FNL can be implemented in a *fully distributed* fashion (see also Algorithm 2 below), as the updates of $\mathbf{u}^k$ and $\mathbf{x}^k$, for any $k \geq 0$, only require information that is locally available at each sensor. In addition, we show below that the updates of $\mathbf{u}^k$ and $\mathbf{x}^k$ can be implemented in *parallel* among the sensors.

(iii) In the special case where $d_{ij} = d_{ji}$ for all $(i,j) \in \mathcal{E}$ and $(j,i) \in \mathcal{E}$ (see Remark 2), then only $\mathbf{u}_{ij}^k$ for $(i,j) \in \mathcal{E}$ such that $i < j$, are updated.

(iv) The point $\mathbf{x}^0 \in \mathbb{R}^{mK}$, which is the initialization point of FNL, can be set arbitrarily.

(v) For any (outer) iteration $k \geq 0$ of FNL, the number of APG iterations, denoted by $n_k \in \mathbb{N}$ in step 4 of Algorithm 1, can be set arbitrarily. In order to obtain convergence guarantees of FNL in Section V, we use the following number of inner iterations (see [16, Section 4.2.1])

$$n_k = s + 2^{\lfloor k/r \rfloor} - 1, \tag{14}$$

for any integers $s, r \in \mathbb{N}$. This update rule is quite general due to the flexibility in choosing the parameters s and r.

C. Deriving Explicit APG Updates

For any (outer) iteration $k \geq 0$ of FNL, the explicit update formulas for the APG iterations (see step 4 in Algorithm 1) for minimizing the partial function $\mathbf{x} \mapsto \Psi\left(\mathbf{x}, \mathbf{u}^k\right)$, are presented below in Algorithm 2, and are developed now. Recall that

$$\Psi\left(\mathbf{x}, \mathbf{u}^k\right) = \varphi\left(\mathbf{x}, \mathbf{u}^k\right) - \sum_{l \in \mathcal{A}} \log\left(\mathcal{I}_l\left(\mathbf{x}_l\right)\right). \tag{15}$$

We denote by $\left\{\mathbf{x}^{k,n}\right\}_{n=1}^{n_k}$ the sequence generated by APG at the outer iteration $k \geq 0$ of FNL. Under this notation, the starting point of APG is set to be $\mathbf{x}^{k,1} = \mathbf{x}^k$, and the output point obtained after $n_k \in \mathbb{N}$ inner iterations is $\mathbf{x}^{k,n_k} =$

$\mathbf{x}^{k+1}$. We also denote by $\left\{\mathbf{y}^{k,n}\right\}_{n=1}^{n_k} \subset \mathbb{R}^{mK}$ the auxiliary sequence generated by APG (see Algorithm 2 below for the exact details).

Following (13), we notice that since the partial function $\mathbf{x} \mapsto \varphi\left(\mathbf{x}, \mathbf{u}^k\right)$ is quadratic with respect to $\mathbf{x}$, then

$$L = \lambda_{\max}\left(\mathbf{P}\right), \tag{16}$$

is a Lipschitz constant of its gradient $\nabla_{\mathbf{x}}\varphi$, for any outer iteration $k \geq 0$ of FNL (see, for example, [3]).

The APG iterations can be easily implemented in both centralized and distributed network architectures. We emphasize that these two implementations are not two different variants, and they both implement the exact same algorithm.

Centralized implementation. Following [5], at each inner iteration $n \geq 1$ of APG to minimize the partial function $\mathbf{x} \mapsto \Psi\left(\mathbf{x}, \mathbf{u}^k\right)$ as defined in (15) (for any outer iteration $k \geq 0$ of FNL), the next iteration $\mathbf{x}^{k,n+1}$ is updated by first calculating the gradient step

$$\begin{aligned} \mathbf{c}^{k,n+1} &\equiv \mathbf{y}^{k,n} - \frac{1}{L}\nabla_{\mathbf{x}}\varphi\left(\mathbf{y}^{k,n}, \mathbf{u}^k\right) \\ &= \mathbf{y}^{k,n} - \frac{1}{L}\left(\mathbf{P}\mathbf{y}^{k,n} - \mathbf{b}\left(\mathbf{u}^k\right)\right) \in \mathbb{R}^{mK}, \end{aligned} \tag{17}$$

where the equality follows from (13), and where $L > 0$ is defined in (16). Then, each $\mathbf{c}_l^{k,n} \in \mathbb{R}^m$, for any anchor $l \in \mathcal{A}$, is projected onto the corresponding uncertainty set $\mathcal{C}_l$. Meaning, for any $l \in \mathcal{A}$ we calculate $\mathcal{P}_l\left(\mathbf{c}_l^{k,n}\right)$, where $\mathcal{P}_l$ denotes the orthogonal projection operator onto the closed and convex set $\mathcal{C}_l$. Below, in Section IV-D, we give several explicit examples for projection operators $\mathcal{P}_l$ of several popular uncertainty sets.

Summarizing the above derivations, the APG algorithm applied in any outer iteration $k \geq 0$ of FNL (see step 4 in Algorithm 1), is recorded below in Algorithm 2.

Distributed and parallel implementation. The gradient step in (17) can be implemented in any distributed architecture. Moreover, this distributed update can also be implemented in parallel among the sensors. To see this, notice that for any sensor $i = 1, 2, \ldots, K$, the update step (17) simply reads

$$\mathbf{c}_i^{k,n} = \mathbf{y}_i^{k,n} - \frac{1}{L}\nabla_{\mathbf{x}_i}\varphi\left(\mathbf{y}^{k,n}, \mathbf{u}^k\right) \in \mathbb{R}^m, \tag{18}$$

where $\nabla_{\mathbf{x}_i}\varphi$ denotes the gradient of φ with respect to the variable $\mathbf{x}_i \in \mathbb{R}^m$ (that is, the gradient of the partial function $\mathbf{x}_i \mapsto \varphi\left(\mathbf{x}, \mathbf{u}\right)$). Simple calculations show that

$$\begin{aligned} \nabla_{\mathbf{x}_i}\varphi\left(\mathbf{x}, \mathbf{u}\right) = &\sum_{\{j:\ (i,j)\in\mathcal{E}\}} \sigma_{ij}^{-2}\left(\mathbf{x}_i - \mathbf{x}_j - d_{ij}\mathbf{u}_{ij}\right) \\ &- \sum_{\{j:\ (j,i)\in\mathcal{E}\}} \sigma_{ji}^{-2}\left(\mathbf{x}_j - \mathbf{x}_i - d_{ji}\mathbf{u}_{ji}\right) \\ &+ \mathcal{I}_{\mathcal{A}}\left(i\right) \cdot \mathbf{\Sigma}_i^{-1}\left(\mathbf{x}_i - \mathbf{a}_i\right), \end{aligned} \tag{19}$$

where $\mathcal{I}_{\mathcal{A}}\left(i\right) = 1$ if $i \in \mathcal{A}$ (i.e., if i is an anchor sensor) and 0 otherwise. If the set of indices of the two sums in (19) is empty, then we simply define the corresponding sum to be 0. Now, a distributed and parallel closed-form formula for the gradient $\nabla_{\mathbf{x}_i}\varphi\left(\mathbf{y}^{k,n}, \mathbf{u}^k\right)$ in (18) is obtained from (19).

Hence, the update of $\mathbf{x}^{k,n+1}$ in (17) is given by the formula

$$\mathbf{x}_i^{k,n+1} = \begin{cases} \mathbf{c}_i^{k,n}, & i \notin \mathcal{A}, \\ \mathcal{P}_i\left(\mathbf{c}_i^{k,n}\right), & i \in \mathcal{A}. \end{cases} \tag{20}$$

The update step in (20) can indeed be implemented in a distributed and parallel fashion, as it only requires information that is locally available at each sensor via communication.

The above distributed and parallel derivations of the APG algorithm are summarized below in Algorithm 2.

An important issue in distributed networks, is that the data preprocessing stage will also be performed distributionally. In the case of APG, we therefore require the Lipschitz constant in (16) to be computed in a distributed fashion (we mention that the constant in (16) can be only be calculated in centralized networks, since it requires the maximal eigenvalue of the matrix $\mathbf{P}$). To this end, we recall that any upper bound of L is also a suitable Lipschitz constant. Now, we denote for any two sensors i and j the constant

$$\omega_{ij} \equiv \sigma_{ij}^{-2} \cdot \mathcal{I}_{(i,j)} + \sigma_{ji}^{-2} \cdot \mathcal{I}_{(j,i)} = \omega_{ji},$$

where $\mathcal{I}_{(i,j)} = 1$ if $(i,j) \in \mathcal{E}$ and 0 otherwise. Below, in Proposition 1, we show that

$$L \leq \max_{(i,j)\in\mathcal{E}}\left\{\sum_{p\neq i}\omega_{ip} + \sum_{p\neq j}\omega_{jp}\right\} + \lambda, \tag{21}$$

where the constant $\lambda > 0$ is defined as the maximum among the maximal eigenvalues of the inverse of the covariance matrices $\mathbf{\Sigma}_l$, for all anchors $l \in \mathcal{A}$. That is,

$$\lambda \equiv \max_{l\in\mathcal{A}}\left\{\lambda_{\max}\left(\mathbf{\Sigma}_l^{-1}\right)\right\}. \tag{22}$$

Notice that the upper bound of L in (21) is a constant that can be calculated in a distributed fashion, as it only requires information available at each sensor via communication during preprocessing. Below, in Remark 5, we further discuss ways to calculate the upper bound of L in (21).

Remark 4. In step 6 of the distributed and parallel implementation of Algorithm 2, each sensor broadcasts a vector of size m to its neighbors (the same vector to each neighbor). This vector is then used in the next (inner) iteration. Hence, algorithm FNL is *synchronous*, as it requires a broadcast synchronization between the sensors.

Remark 5. Here we give a few notes on the distributed Lipschitz constant L used in Algorithm 2.

(i) A particular and useful instance of $\lambda > 0$ as defined in (22), is when the covariance matrices $\mathbf{\Sigma}_l$, for any anchor $l \in \mathcal{A}$, are a scalar multiplication of the identity matrix. That is, $\mathbf{\Sigma}_l = \mu\mathbf{I}_m$ for some constant $\mu > 0$ and for all $l \in \mathcal{A}$. In this case, (22) simply reduces to $\lambda = 1/\mu$.

(ii) A particular and useful instance of the upper bound in (21), is when the values of σ_{ij}, for all $(i,j) \in \mathcal{E}$, coincide and are equal to some constant $\sigma > 0$ (or are all evaluated as such). In this case, the bound (21) reduces to

$$\begin{aligned} L &\leq \sigma^{-2}\max_{(i,j)\in\mathcal{E}}\left\{\deg\left(i\right) + \deg\left(j\right)\right\} + \lambda \\ &\leq 2\sigma^{-2}\max_{i=1,2,\ldots,K}\left\{\deg\left(i\right)\right\} + \lambda, \end{aligned} \tag{23}$$

Algorithm 2 APG in FNL

Input: Set L according to (16) (centralized) or (21) (distributed), set $\mathbf{x}^{k,0} = \mathbf{y}^{k,0} = \mathbf{x}^k \in \mathbb{R}^{mK}$, $t_0 = 1$ and $n_k \in \mathbb{N}$ according to (14).

Centralized implementation:

```
1: for $n = 0, 1, \ldots, n_k - 1$ do
2:     Set $\mathbf{x}^{k,n+1} = \mathbf{c}^{k,n}$ according to (17), and then update $\mathbf{x}_l^{k,n+1} = \mathcal{P}_l\left(\mathbf{c}_l^{k,n}\right)$ for any $l \in \mathcal{A}$.
3:     Set $t_{n+1} = \frac{1+\sqrt{1+4t_n^2}}{2}$.
4:     Set $\mathbf{y}^{k,n+1} = \mathbf{x}^{k,n+1} + \frac{t_n-1}{t_{n+1}}\left(\mathbf{x}^{k,n+1} - \mathbf{x}^{k,n}\right)$.
5: end for
```

Distributed and parallel implementation:

```
1: for $n = 0, 1, \ldots, n_k - 1$ do
2:     for $i = 1, 2, \ldots, K$ do in parallel
3:         Receive all $\mathbf{y}_j^{k,n}$, $j$ satisfying $(i,j), (j,i) \in \mathcal{E}$.
4:         Set $\mathbf{c}_i^{k,n}$ according to (18) and (19), and then update $\mathbf{x}_i^{k,n+1}$ according to (20).
5:         Set $t_{n+1} = \frac{1+\sqrt{1+4t_n^2}}{2}$.
6:         Set $\mathbf{y}_i^{k,n+1} = \mathbf{x}_i^{k,n+1} + \frac{t_n-1}{t_{n+1}}\left(\mathbf{x}_i^{k,n+1} - \mathbf{x}_i^{k,n}\right)$ and broadcast it to the neighbors of $i$.
7:     end for
8: end for
```

where $\deg(i) = \mathrm{indeg}(i) + \mathrm{outdeg}(i)$, which is the sum of the in-degree and out-degree of sensor i (this constant coincides with the constant used in [15], [38]). Notice that the bound in (23) can be further increased, as it holds that

$$L \leq 4\sigma^{-2}K + \lambda, \tag{24}$$

which is a distributed constant that do not require the transfer of local data during preprocessing. In the special case where $d_{ij} = d_{ji}$ for all $(i,j) \in \mathcal{E}$ and $(j,i) \in \mathcal{E}$ (see Remark 2), then $\deg(i) = \mathrm{indeg}(i)$ and (24) now reads $L \leq 2\sigma^{-2}K + \lambda$. Last, we should mention that K represents the number of sensors to be located, hence in the case with complete anchor certainty, then K should be replaced with N (the number of non-anchor sensors).

D. Examples of Uncertainty Sets

For any anchor sensor $l \in \mathcal{A}$, the APG iterations developed above require to calculate the projection of a vector in $\mathbb{R}^m$ onto its corresponding uncertainty set $\mathcal{C}_l \subseteq \mathbb{R}^m$ (see (20)). Here, we mention four popular uncertainty sets that the projections onto them can be calculated explicitly. Recall that $\mathcal{P}_l$, $l \in \mathcal{A}$, is the orthogonal projection operator onto the set $\mathcal{C}_l$.

(i) Perfect anchor knowledge: when $\mathcal{C}_l = \{\mathbf{a}_l\}$, then $\mathcal{P}_l(\mathbf{v}) = \mathbf{a}_l$ for any $\mathbf{v} \in \mathbb{R}^m$. In particular, the anchor locations are constant and are not updated.

(ii) No uncertainty set: if $\mathcal{C}_l = \mathbb{R}^m$, then the anchor location is not restricted to any uncertainty set, and $\mathcal{P}_l(\mathbf{v}) = \mathbf{v}$ for any $\mathbf{v} \in \mathbb{R}^m$.

(iii) Ball uncertainty: the uncertainty set for some $l \in \mathcal{A}$ is assumed to be a ball with radius $r_l > 0$ centered at the measured location $\mathbf{a}_l$. That is, $\mathcal{C}_l = \{\mathbf{v} \in \mathbb{R}^m \colon \|\mathbf{v} - \mathbf{a}_l\| \leq r_l\}$. In this case [4]

$$\mathcal{P}_l(\mathbf{v}) = \begin{cases} \mathbf{v}, & \|\mathbf{v} - \mathbf{a}_l\| \leq r_l, \\ \mathbf{a}_l + r_l \cdot \frac{\mathbf{v} - \mathbf{a}_l}{\|\mathbf{v} - \mathbf{a}_l\|}, & \text{otherwise}. \end{cases} \tag{25}$$

(iv) Ellipsoid uncertainty: here the set $\mathcal{C}_l$ is assumed to be an ellipsoid centered at $\mathbf{a}_l$, for some $l \in \mathcal{A}$. That is, there exist $r_l > 0$ and a symmetric positive-define matrix $\mathbf{Q}_l \in \mathbb{R}^{m \times m}$ such that $\mathcal{C}_l = \left\{\mathbf{v} \in \mathbb{R}^m \colon \|\mathbf{v} - \mathbf{a}_l\|_{\mathbf{Q}_l} \leq r_l\right\}$. In order to find an explicit formula for the projection function $\mathcal{P}_l$, we denote by $\mathbf{D}_l \in \mathbb{R}^{m \times m}$ the diagonal matrix and by $\mathbf{U}_l \in \mathbb{R}^{m \times m}$ the normal matrix that satisfy $\mathbf{Q}_l^{-1} = \mathbf{U}_l^T \mathbf{D}_l \mathbf{U}_l$ (the spectral decomposition of $\mathbf{Q}_l^{-1}$). Then, using the KKT conditions, the orthogonal projection takes the form

$$\mathcal{P}_l(\mathbf{v}) = \begin{cases} \mathbf{v}, & \|\mathbf{v} - \mathbf{a}_l\|_{\mathbf{Q}_l} \leq r_l, \\ \mathbf{U}_l^T\left(\mathbf{w}_l(\mu^*) + \mathbf{a}_l\right), & \text{otherwise}, \end{cases}$$

where

$$\mathbf{w}_l(\mu) \equiv \left(\mathbf{I}_m + \mu \mathbf{D}_l\right)^{-1}\left(\mathbf{a}_l + \mathbf{U}_l \mathbf{v}\right) \in \mathbb{R}^m,$$

and for $\mu^* > -\lambda_{\min}\left(\mathbf{Q}_l^{-1}\right)$ the unique root of the one-dimensional equation $\|\mathbf{w}_l(\mu)\|_{\mathbf{Q}_l} = r_l$.

Remark 6. The projections in Algorithm 2 are only calculated for the anchors. Since the number of anchors is limited, and since each projection is in a lower-dimensional space (usually $m = 2, 3$), then calculating the projections is not a computationally demanding task. Additionally, recall that this task can always be implemented in parallel among the anchors.

V. Global Convergence Analysis of FNL

In this section, we prove that FNL globally converges to critical points of the original non-convex and non-smooth function $\mathcal{F}$ of the unified Problem (ML). The global convergence result is based on a proof methodology introduced in [16] for NAM, which was subsequently extended to handle non-smooth problems in [17]. More precisely, the methodology outlined in [17] establishes that under certain conditions (which we verify below), NAM with APG as a nested algorithm globally converges to critical points of the objective function. We formalize this result for FNL in Theorem 1.

Theorem 1. *Let $\left\{\mathbf{x}^k\right\}_{k \geq 0}$ be a sequence generated by FNL. Then, it converges to some $\mathbf{x}^* \in \mathbb{R}^{mK}$, which is a critical point of the function $\mathcal{F}$ of Problem* (ML).

In order to prove Theorem 1, throughout this section we make the following mild assumption.

Assumption 1. (i) The directed graph induced by the network (where the nodes correspond to the sensors and two neighbors induce a directed edge) is weakly connected.
(ii) There is at least one anchor sensor.

By a weakly connected directed graph we mean that the underlying undirected graph induced by the network (where each directed edge is replaced with an undirected edge) is connected [33]. It should be noted that if Assumption 1(i) does not hold, then the network can be divided into disjoint weakly connected sub-networks and each can be treated separately, as long as each such sub-network contains at least one anchor.

Following [17, Theorem 3.1], in order to prove the global convergence result of Theorem 1, we need to show that the function Ψ as defined in (11), and the sequence $\left\{\mathbf{x}^k\right\}_{k\geq 0}$ generated by FNL, satisfy the following properties.

Proposition 1. (i) *There exists $\rho > 0$ such that the function $\mathbf{x} \mapsto \Psi\left(\mathbf{x}, \mathbf{u}\right)$ is ρ-strongly convex for any $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$.*

(ii) *The function $\mathbf{x} \mapsto \varphi\left(\mathbf{x}, \mathbf{u}\right)$ has an L-Lipschitz continuous gradient, for $L > 0$ defined in* (21)*, for any $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$.*

(iii) *The sequence $\left\{\mathbf{x}^k\right\}_{k\geq 0}$ generated by FNL is bounded.*

Due to its technical nature, we postpone the proof of Proposition 1 to Appendix A.

A useful function that will enable us to prove Theorem 1 is defined as the sum of negative norm terms with some convex function. Mathematically, for any non-zero scalars $\beta_1, \beta_2, \ldots, \beta_M \neq 0$, we define the non-smooth and concave functions $\phi_i \colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, $i = 1, 2, \ldots, M$, as

$$\phi_i\left(\mathbf{y}, \mathbf{z}_i\right) = -\left|\beta_i\right| \cdot \left\|\mathbf{y} - \mathbf{z}_i\right\|. \tag{26}$$

In addition, we define the function $\phi \colon \mathbb{R}^m \times \mathbb{R}^{mM} \to \mathbb{R}$ as

$$\phi\left(\mathbf{y}, \mathbf{z}\right) = \sum_{i=1}^{M} \phi_i\left(\mathbf{y}, \mathbf{z}_i\right) + \delta\left(\mathbf{y}\right), \tag{27}$$

where the function $\delta \colon \mathbb{R}^m \to (-\infty, \infty]$ is some extended real-valued convex function (possibly non-smooth). The following result in Lemma 1 will be useful to relate the critical points of the function Ψ with the the critical points of the function $\mathcal{F}$. The proof of Lemma 1 is given in Appendix B.

Lemma 1. *Let $\left(\mathbf{y}^*, \mathbf{z}^*\right) \in \mathbb{R}^m \times \mathbb{R}^{mM}$ satisfying $\mathbf{y}^* = \mathbf{z}_1^* = \ldots = \mathbf{z}_M^*$, then it holds that*

$$\beta_1 \mathcal{S} + \beta_2 \mathcal{S} + \ldots + \beta_M \mathcal{S} + \partial\delta\left(\mathbf{y}^*\right) \subseteq \partial_{\mathbf{y}}\phi\left(\mathbf{y}^*, \mathbf{z}^*\right),$$

where $\partial_{\mathbf{y}}\phi$ is the limiting sub-differential of the partial function $\mathbf{y} \mapsto \phi\left(\mathbf{y}, \mathbf{z}\right)$.

Before we proceed to the proof of Theorem 1, we define the following functions and scalars. For any sensor $i = 1, 2, \ldots, K$, we define the convex and non-smooth indicator function $\delta_i \colon \mathbb{R}^m \to \{0, +\infty\}$ as

$$\delta_i\left(\mathbf{v}\right) \equiv \begin{cases} 0, & i \notin \mathcal{A}, \\ 0, & i \in \mathcal{A} \text{ and } \mathbf{v} \in \mathcal{C}_i, \\ +\infty, & i \in \mathcal{A} \text{ and } \mathbf{v} \notin \mathcal{C}_i. \end{cases} \tag{28}$$

Notice that $\delta_i\left(\mathbf{v}\right) = -\mathcal{I}_{\mathcal{A}}\left(i\right) \cdot \log\left(\mathcal{I}_i\left(\mathbf{v}\right)\right)$, where $\mathcal{I}_i$ is defined in (4) and $\mathcal{I}_{\mathcal{A}}$ is defined in (19).

In addition, for any sensor i we define the scalars

$$\beta_{ij} \equiv -d_{ij} \cdot \sigma_{ij}^{-2}, \quad \forall j \text{ such that } (i, j) \in \mathcal{E},$$
$$\beta_{ji} \equiv d_{ji} \cdot \sigma_{ji}^{-2}, \quad \forall j \text{ such that } (j, i) \in \mathcal{E},$$

and for any pair $(i, j) \in \mathcal{E}$ we define the function $\phi_{ij} \colon \mathbb{R}^{mK} \to \mathbb{R}$ as

$$\phi_{ij}\left(\mathbf{x}\right) \equiv -\left|\beta_{ij}\right| \cdot \left\|\mathbf{x}_i - \mathbf{x}_j\right\|. \tag{29}$$

Last, for any sensor $i = 1, 2, \ldots, K$, we define the non-smooth function $\phi_i \colon \mathbb{R}^{mK} \to \mathbb{R}$ as

$$\phi_i\left(\mathbf{x}\right) \equiv \sum_{\{j \colon (i,j) \in \mathcal{E}\}} \phi_{ij}\left(\mathbf{x}\right) + \sum_{\{j \colon (j,i) \in \mathcal{E}\}} \phi_{ji}\left(\mathbf{x}\right) + \delta_i\left(\mathbf{x}_i\right). \tag{30}$$

Now, equipped with Proposition 1, Lemma 1 and the above notations, we are ready to prove Theorem 1.

Proof of Theorem 1. Following [17] together with Proposition 1, we obtain that the sequence $\left\{\mathbf{x}^k\right\}_{k\geq 0}$ globally converges to some $\mathbf{x}^* \in \mathbb{R}^{mK}$. In addition, if $\mathbf{u}^* \in \mathcal{B}^{|\mathcal{E}|}$ is a limit point of the sequence $\left\{\mathbf{u}^k\right\}_{k\geq 0}$, then $\left(\mathbf{x}^*, \mathbf{u}^*\right)$ is a critical point of the function Ψ, and in particular $\mathbf{0}_{mK} \in \partial_{\mathbf{x}}\Psi\left(\mathbf{x}^*, \mathbf{u}^*\right)$. Additionally, since the sequence $\left\{\mathbf{u}^k\right\}_{k\geq 0}$ is bounded in $\mathcal{B}^{|\mathcal{E}|}$, then by possibly passing to a convergent sub-sequence, we can assume that $\mathbf{u}^k \to \mathbf{u}^*$ as $k \to \infty$.

Now, we are left to show that $\mathbf{x}^* \in \mathbb{R}^{mK}$ obtained by FNL is a critical point of the original function $\mathcal{F}$ of Problem (ML). To this end, it is enough to prove that

$$\partial_{\mathbf{x}_i}\Psi\left(\mathbf{x}^*, \mathbf{u}^*\right) \subseteq \partial_{\mathbf{x}_i}\mathcal{F}\left(\mathbf{x}^*\right), \quad \forall i = 1, 2, \ldots, K, \tag{31}$$

which will guarantee that $\mathbf{x}^*$ is also a critical point of $\mathcal{F}$.

Writing both sides of (31) explicitly, and after eliminating similar differentiable terms from both sides, we see that for any $i = 1, 2, \ldots, K$ the required inclusion in (31) reduces to

$$\sum_{\{j \colon (i,j) \in \mathcal{E}\}} \beta_{ij}\mathbf{u}_{ij}^* + \sum_{\{j \colon (j,i) \in \mathcal{E}\}} \beta_{ji}\mathbf{u}_{ji}^* + \partial\delta_i\left(\mathbf{x}_i^*\right) \subseteq \partial_{\mathbf{x}_i}\phi_i\left(\mathbf{x}^*\right), \tag{32}$$

where $\phi_i\left(\mathbf{x}^*\right)$ is defined in (30).

In case that the function ϕ_i is differentiable at $\mathbf{x}^*$, then $\mathbf{x}_i^* \neq \mathbf{x}_j^*$ for all pairs $(i, j) \in \mathcal{E}$ or $(j, i) \in \mathcal{E}$. In this case, then $\mathbf{x}_i^k \neq \mathbf{x}_j^k$ for all large enough $k \geq 0$ and by taking the limit $k \to \infty$ we have that

$$\beta_{ij}\mathbf{u}_{ij}^* = \nabla_{\mathbf{x}_i}\phi_{ij}\left(\mathbf{x}^*\right) \text{ and } \beta_{ji}\mathbf{u}_{ji}^* = \nabla_{\mathbf{x}_i}\phi_{ji}\left(\mathbf{x}^*\right),$$

and the inclusion in (32) is actually satisfied with equality.

Now, suppose that the function ϕ_i is non-differentiable at $\mathbf{x}^*$. In this case, it is enough to prove the required inclusion in (32) for the harder case in which $\mathbf{x}_i^* = \mathbf{x}_j^*$ for all $(i, j) \in \mathcal{E}$ and $(j, i) \in \mathcal{E}$. Since $\mathbf{u}_{ij}^*, \mathbf{u}_{ji}^* \in \mathcal{S}$, then $\beta_{ij}\mathbf{u}_{ij}^* \in \beta_{ij}\mathcal{S}$ and $\beta_{ji}\mathbf{u}_{ji}^* \in \beta_{ji}\mathcal{S}$. Therefore, we see that the required inclusion in (32) indeed holds true by invoking Lemma 1, and the proof is now completed. $\square$

VI. Numerical Experiments

Considering that our proposed problem formulation encompasses various formulations previously documented in the literature, we have decided to concentrate our efforts in this paper on addressing the primary challenge encountered in the current algorithmic literature: large networks with numerous sensors, irrespective of the problem formulation. To this end, we explore a benchmark network consisting of 1000 sensors and an even larger network comprising 10000 sensors.

Our comparison focuses on our proposed FNL method and algorithms designed for the Problem (ML-U) scenario, which involves unknown anchor locations. This scenario covers realistic situations but is not as constrained as Problem (ML-L), which only one algorithm in the current literature addresses (excluding our FNL algorithm). As Problem (ML-C) involves precisely known anchor locations, it presents a relatively simpler challenge. Therefore, we refrain from comparing our proposed method exclusively with algorithms designed for this scenario. This decision is rooted in the understanding that Problem (ML-C) can be viewed as a specific instance within the broader (ML-U) framework. Consequently, in the subsequent results, our focus remains on algorithms tailored for the Problem (ML-U) scenario, which also demonstrate the capability to handle very-large scale networks of 10000 sensors. Further details are provided below.

Furthermore, to enhance the coherence of this section, we opted to maintain fixed hyper-parameters (such as the number of anchors, starting point, average node degree, etc.) across the compared algorithms. Previous studies such as [15], [32] have indicated that variations in these hyper-parameters tend to impact all algorithms in a proportional manner. Therefore, employing fixed hyper-parameters allows us to gain insights without introducing additional complexities that may not offer significant additional understanding beyond what can be gleaned in a fixed scenario.

A. Computational and Communication Costs of FNL

In Table II, we summarize the computational and communication costs of FNL and other existing methods (see also Table I). We see that FNL enjoys a low computational burden in its centralized and its parallel implementation.

In its centralized implementation, FNL requires the calculation of $|\mathcal{E}|$ norm terms of size m in order to compute the vector $\mathbf{u}^k$ for any iteration $k \geq 0$. In addition, it also performs matrix-vector multiplications of the sizes of the matrices $\mathbf{P}$ and $\mathbf{S}^T$ (see (13)), which ends up with a computational complexity of $\mathcal{O}\left(K^2m^2 + |\mathcal{E}|\, Km^2\right)$.

In its distributed and parallel implementation, each sensor computes $|\mathcal{E}_i|$ norm terms of size m, where $|\mathcal{E}_i|$ denotes the number of neighbors of sensor i. In addition, it computes the sum of $|\mathcal{E}_i|$ vectors of size m, and each anchor performs matrix-vector multiplication of size m. This ends up, for each sensor, with a total complexity of $\mathcal{O}\left(|\mathcal{E}_i|\, m\right)$ for the non-anchors and $\mathcal{O}\left(|\mathcal{E}_i|\, m + m^2\right)$ for the anchors.

Recall that FNL can also deal with anchor uncertainty, therefore the complexity of calculating the anchor projections should be considered in such uncertain scenarios. For example, in the case of ball uncertainty, the complexity of the projection in (25) is $\mathcal{O}\left(m\right)$ for each anchor.

In Table II, we also summarize the communication costs of FNL and other distributed methods. In each iteration, FNL receives $|\mathcal{E}_i|$ messages of size m, one from each of its neighbors (see step 3 in the distributed Algorithm 2). In addition, in each iteration FNL broadcasts exactly one message of size m to be received by its neighbors (step 6 in Algorithm 2). This is in contrary to other distributed and parallel methods, such as ADMM-H of [32], that bear greater communication costs. For more details about the communication costs of distributed algorithms in the literature, see [15].

B. Description of Experiments

We consider two network settings. In the first setting, we examine Stanford's planar benchmark network [45], comprising of $K = 1000$ sensors uniformly distributed in the box $[-0.5, 0.5]^2 \subset \mathbb{R}^2$. Among these sensors, 20 are anchors with approximately known locations. We assume that $\sigma_{ij} \equiv \sigma_{\text{dist}} > 0$ for all $(i,j) \in \mathcal{E}$, logarithmically spaced in the interval $\left[10^{-3}, 10^{-1.5}\right]$, representing environments from low to high noise levels. Each approximate anchor location is drawn from an uncertainty ball with a radius of 0.005 and a noise covariance matrix $\mathbf{\Sigma}_l \equiv 0.0016\mathbf{I}_2$, $l \in \mathcal{A}$, centered around the unknown true anchor location.

TABLE II: Comparison of computational and communication costs. m – problem's dimension; K – number of sensors; $|\mathcal{E}|$ – number of pairs of neighbors; $|\mathcal{E}_i|$ – number of neighbors of sensor i

Method	Computational operations
FNL, centralized	$\mathcal{O}\left(K^2m^2 + \|\mathcal{E}\|\, Km^2\right)$
FNL, distributed	$\mathcal{O}\left(\|\mathcal{E}_i\|\, m\right)$
DC [42]	$\mathcal{O}\left(K^3 + K^2m + \|\mathcal{E}\|\, Km^2\right)$
SGO [34]	$\mathcal{O}\left(\|\mathcal{E}_i\|\, m\right)$
SOCP [26]	$\mathcal{O}\left(\|\mathcal{E}\|^{4.5} + m^2\,\|\mathcal{E}\|^{2.5}\, K^2 + m^2K^{4.5}\right)$

Method	Communication costs			
	In Message		Out Message	
	#	size	#	size
FNL (parallel)	$\|\mathcal{E}_i\|$	m	1	m
SGO [34]	$\|\mathcal{E}_i\|$	m	1	m
ADMM-H [32]	$\|\mathcal{E}_i\|$	$2m$	$\|\mathcal{E}_i\|$	$2m$

In the second setting, we consider a significantly larger planar network comprising $K = 10000$ sensors, as outlined in [15], also uniformly distributed within the box $[-0.5, 0.5]^2 \subset \mathbb{R}^2$. Here, 200 sensors are designated as anchors. We set σ_{dist} to 10^{-4}, indicative of a low to medium noise environment in this context. The anchors are drawn from uncertainty balls with a radius of 0.001 and a covariance matrix of $\mathbf{\Sigma}_l \equiv 0.0003\mathbf{I}_2$, $l \in \mathcal{A}$. For further detailed specifications regarding each of these two network settings, we refer the readers to [15], [32].

For each σ_{dist} value, we generated 25 realizations of distance measurements (Monte Carlo trials) and computed the average values across these trials. We compared the results based on the function value of Problem (ML-U), given that all methods were devised to minimize this function. Thus, a lower function value indicates the effectiveness of a method in solving this model. Additionally, we assessed the bias concerning the true sensor locations, defined as

$$\text{Bias}^k = \left\|\mathbf{x}^k - \mathbf{x}^{\text{real}}\right\|,$$

where $\mathbf{x}^k \in \mathbb{R}^{mK}$ represents the network location obtained by an algorithm at the k-th iteration, and $\mathbf{x}^{\text{real}} \in \mathbb{R}^{mK}$ denotes the true network location. In addition, we also assessed the methods based on their overall runtime.

To address the larger-scale network efficiently, which cannot be handled within a reasonable time frame using current SDP methods, we employed first-order algorithms, specifically SGO from [34] and DC from [42]. Additionally, for the benchmark network, we also applied the second-order SOCP algorithm of [34]. The algorithms FNL, SGO, and DC, were executed for precisely 10000 total iterations (including inner iterations) and were initialized with the same random starting point drawn from $[-0.5, 0, 5]^2$ [15], [32].

For FNL, we selected $s = 40$ and $r = 1000$ (see (14)), although any other integers could have been chosen. This decision primarily influences the relationship between the running time per iteration and the total number of iterations. Importantly, it does not impact the convergence or rate of the algorithm. Larger values of $s \in \mathbb{N}$ result in a lesser decrease in function value between two consecutive inner iterations, yet each iteration runs faster. As a result, similar outcomes are attained across different values of s for the same execution time or cost. This phenomenon has been discussed in detail in the papers [14], [16], [17].

Furthermore, in the case of FNL, we have explored both its centralized implementation and its distributed and parallel implementation, which vary in their Lipschitz constant calculation, determining the step-size (see Remark 5).

All experiments were ran on Intel(R) Xeon(R) Gold 6254 CPU@3.10GHz, 300GB RAM, 72 threads, MATLAB 2022a.

C. Results

In Figure 1, we illustrate the average function and bias output values for the benchmark network as a function of the noise σ_{dist}. The output values represent the results obtained at the final iteration. Regarding function values (left plot), we observe that the iterative first-order methods, namely FNL, DC, and SGO, outperform the computationally intensive SOCP across all noises. This suggests that the first-order methods excel in solving this non-convex and non-smooth problem. It is also evident that both the centralized and distributed implementations of FNL outperform DC and SGO.

In terms of bias (right plot), SOCP outperforms all first-order methods at low noises. However, for larger noise levels, SOCP's performance deteriorates. This phenomenon, wherein second-order algorithms perform well in low noise scenarios but struggle in high noise scenarios for ML formulations of sensor networks, has been previously discussed in [8]. It is worth noting that SOCP is a fully centralized algorithm with high computational complexity, limiting its practicality in real-life scenarios. Conversely, first-order methods are more practical for computationally demanding tasks. Among these methods, FNL emerges as the top performer, achieving both low function values and bias across all noise levels.

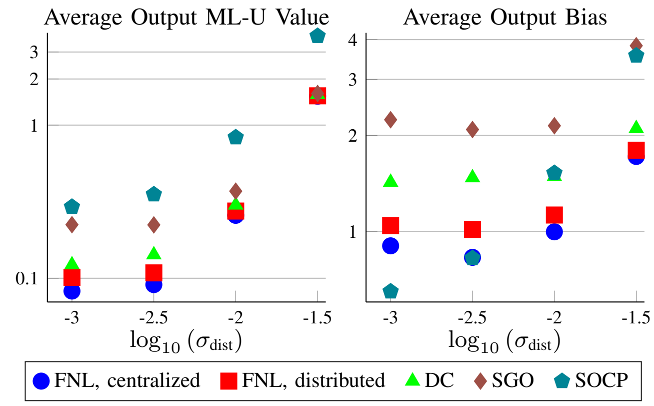

Fig. 1: Benchmark network: average ML-U function value (left) and norm of bias (right) across all realizations, in a logarithmic scale (y-axis), plotted against σ_{dist} (x-axis). The values denote the average output achieved at the final iteration.

To visualize the algorithms' progress in minimizing the function value and bias, we depict in Figure 2 the average decrease in function value and bias against the iterations for the benchmark network, focusing on $\sigma_{\text{dist}} = 10^{-3}$ (similar trends apply to other noise levels). Since SOCP is not an iterative method, we solely present its output values in the plots. The figure reveals that although DC and SGO initially yield lower values than FNL, overall they converge to an inferior network location. Furthermore, the centralized implementation of FNL outperforms its distributed counterpart, primarily because the centralized implementation utilizes complete network information to compute the step-size (see Remark 5).

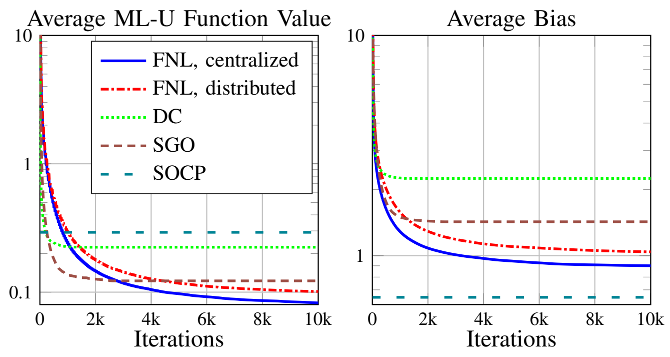

Fig. 2: Benchmark network: average ML-U value (left) and norm of bias (right) across all realizations, in a logarithmic scale (y-axis), plotted against the total number of iterations (x-axis) for a given $\sigma_{\text{dist}} = 10^{-3}$. All lines have been capped from above for clarity of representation. For SOCP, only the output value is plotted.

Now, we discuss the larger-scale network consisting of 10000 sensors. It is worth noting that, to our knowledge, such a large network has rarely been addressed in published literature, with [15] being an exception as it tackles Problem (ML-C). In Figure 3, we depict the decrease in function value and bias for this network. It is important to mention that due to the high computational burden, the second-order method SOCP was not tested on this large network. The results demonstrate that FNL outperforms other methods in terms of minimizing the function, indicating its efficacy in solving the non-convex and non-smooth problem formulation. Regarding bias, DC performs well; however, it is essential to consider that DC is a centralized method and is not feasible for implementation in many real-life network scenarios. Therefore, FNL, whether in its centralized or distributed and parallel implementations, emerges as a favorable choice across all network architectures.

Finally, Table III presents the average total runtime for each method concerning both networks. Importantly, to note that, for the sake of a fair comparison, all methods were implemented without loops and using concise "one-liner" code updates. Since the noise level does not affect the runtime per iteration (only the quality of the solution), the figures were extracted from the low noise level scenario.

Unquestionably, FNL exhibits outstanding performance, taking less than a second to locate the benchmark network with 10k iterations. In contrast, the other methods exhibited much slower performance, with runtime ranging from 55 to 190 seconds. It is worth mentioning that the runtime provided in the table for the distributed implementation of FNL reflects a non-parallel implementation, thus it should be roughly

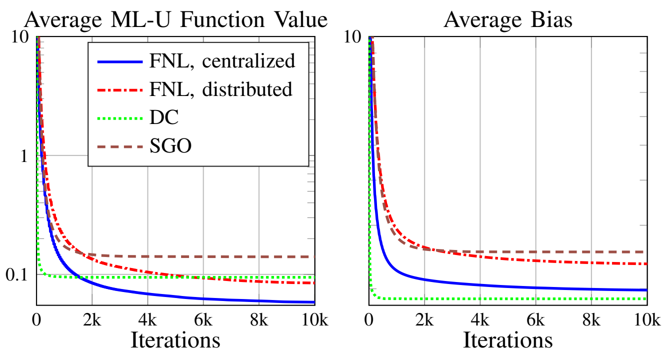

Fig. 3: 10k network: average ML-U value (left) and norm of bias (right) across all realizations in a logarithmic scale (y-axis), plotted against the total number of iterations (x-axis) for a given $\sigma_{\text{dist}} = 10^{-4}$. All lines have been capped from above for clarity.

scaled by the number of sensors in the network to estimate its parallel implementation runtime. This implies that in a parallel implementation, FNL achieves an exceptional runtime of ≈ 0.0008 seconds overall for the benchmark network.

In the larger-scale network, FNL demonstrates remarkable performance, requiring ≈ 11 seconds to locate 10000 sensors with 10k iterations. In contrast, the others experience significantly longer runtimes. In a parallel implementation, FNL locates the network in ≈ 0.0012 seconds for 10k iterations.

TABLE III: Runtime in seconds.

Method	Total runtime	
	Benchmark	10k net
FNL, centralized	0.7136	11.1
FNL, distributed	0.7156	10.8
DC	190.3	2641.3
SGO	55.8	616.6
SOCP	186.6	–

VII. Conclusion

In this paper, we addressed the non-convexity and non-smoothness of the Sensor Network Localization problem by adopting the maximum-likelihood approach and proposed a generalized and unified SNL formulation that considers distance and anchor location measurements, and anchor uncertainty. This unified formulation encompasses previous approaches found in the literature. To effectively solve this formulation, we developed FNL, which is a fast, first-order, well-defined and globally convergent algorithm with low computational complexity. It utilizes the APG method as a nested algorithm, making it applicable in both centralized and distributed network architectures. In its distributed implementation, FNL exhibits low communication costs and can be executed in parallel among the sensors. Our numerical results demonstrated advantages of FNL compared to other methods for large-scale networks, considering its fast convergence and low computational and communication burden.

Appendix

A. Proof of Proposition 1

Proof. We first prove that $\mathbf{P}$ in (13) is positive definite (PD). Obviously, $\mathbf{P}$ is positive semi-definite as a sum of two such matrices. By the structure of $\mathbf{P}$ it follows that $\mathbf{x} \in \text{Kernel}\left(\mathbf{P}\right)$ if and only if $\mathbf{x} \in \text{Kernel}\left(\mathbf{S}\right) \cap \text{Kernel}\left(\mathbf{A}\right)$. So, to prove that $\mathbf{P}$ is PD, we will show that the intersection $\text{Kernel}\left(\mathbf{S}\right) \cap \text{Kernel}\left(\mathbf{A}\right)$ is trivial.

Since every row of $\mathbf{S}$ has exactly one σ_{ij}^{-1} and one $-\sigma_{ij}^{-1}$, and since the network is weakly connected (see Assumption 1(i)), then $\text{Kernel}\left(\mathbf{S}\right) \subseteq \text{span}\left(\{\mathbf{1}_{mK}\}\right)$, where $\mathbf{1}_{mK}$ is the vector of all ones. Hence, any non-zero vector $\mathbf{x} \in \text{Kernel}\left(\mathbf{S}\right)$ is of the form $\mathbf{x} = \alpha \mathbf{1}_{mK}$ for some $\alpha \neq 0$. However, since each covariance matrix $\boldsymbol{\Sigma}_l$, $l \in \mathcal{A}$, is PD by its definition, then the kernel of each such matrix is trivial. Therefore, $\alpha \mathbf{A} \mathbf{1}_{mK}$ is the vector of all zeros except for the coordinates corresponding to the anchors (recall that in Assumption 1(ii) we assume that at least one anchor exists). It now follows that $\text{Kernel}\left(\mathbf{S}\right) \cap \text{Kernel}\left(\mathbf{A}\right)$ is indeed trivial and hence $\mathbf{P}$ is PD.

Now, item (i) follows since $\mathbf{x} \mapsto \varphi\left(\mathbf{x}, \mathbf{u}\right)$ is $\lambda_{\min}\left(\mathbf{P}\right)$-strongly convex for any fixed $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$ (see [4]) and since the function $\psi\left(\mathbf{x}\right)$ is convex.

To prove item (ii), recall that the quadratic function $\mathbf{x} \mapsto \varphi\left(\mathbf{x}, \mathbf{u}\right)$ has $\lambda_{\max}\left(\mathbf{P}\right)$-Lipschitz continuous gradient for any fixed $\mathbf{u} \in \mathcal{B}^{|\mathcal{E}|}$ (see (16)). From Weyl's theorem [18, Theorem 4.3.1] we have $\lambda_{\max}\left(\mathbf{P}\right) \leq \lambda_{\max}\left(\mathbf{S}^T\mathbf{S}\right) + \lambda_{\max}\left(\mathbf{A}\right)$. By the structure of the matrix $\mathbf{A}$ we have $\lambda_{\max}\left(\mathbf{A}\right) = \lambda$ where λ is defined in (22). Now, following [12, Theorem 2.3] it indeed follows that $\lambda_{\max}\left(\mathbf{P}\right) \leq L$, where L is defined in (21).

Now we prove item (iii). From [4, Lemma 2.42] it follows that the function φ is coercive. Since $\Psi \geq \varphi$, it follows that Ψ is also coercive, which implies that Ψ has bounded level sets. Now, from [17] it follows that for the number of inner iterations as taken in (14), the sequence of function values $\left\{\Psi\left(\mathbf{x}^k, \mathbf{u}^k\right)\right\}_{k\geq 0}$ generated by FNL is non-increasing. Hence, the sequence $\left\{\left(\mathbf{x}^k, \mathbf{u}^k\right)\right\}_{k\geq 0}$ is contained in the level set of Ψ at level $\Psi\left(\mathbf{x}^0, \mathbf{u}^0\right)$, and therefore is bounded. $\square$

B. Proof of Lemma 1

Proof. First, if $\mathbf{y}^* \notin \text{dom}\left(\delta\right)$, then $\partial\delta\left(\mathbf{y}^*\right) = \varnothing$. Therefore, it is enough to prove the required inclusion for the case in which $\mathbf{y}^* \in \text{dom}\left(\delta\right)$.

Let $\mathbf{g} \in \beta_1\mathcal{S} + \beta_2\mathcal{S} + \ldots + \beta_M\mathcal{S} + \partial\delta\left(\mathbf{y}^*\right)$, and we will show directly from the definition of limiting sub-gradients that $\mathbf{g} \in \partial_{\mathbf{y}}\phi\left(\mathbf{y}^*, \mathbf{z}^*\right)$. By the structure of $\mathbf{g}$, there exist $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M \in \mathcal{S}$ and $\mathbf{v} \in \partial\delta\left(\mathbf{y}^*\right)$ such that $\mathbf{g} = \sum_{i=1}^{M}\beta_i\mathbf{v}_i + \mathbf{v}$.

Now, we define the sequence $\left\{\left(\mathbf{y}^k, \mathbf{z}^k\right)\right\}_{k\geq 0} \subset \mathbb{R}^m$ as

$$\mathbf{y}^k \equiv \mathbf{y}^* \quad \text{and} \quad \mathbf{z}_i^k \equiv \mathbf{y}^* + \frac{\text{sign}\left(\beta_i\right)\cdot\mathbf{v}_i}{k}, \quad \forall k \geq 0.$$

Notice that $\left(\mathbf{y}^k, \mathbf{z}^k\right) \to \left(\mathbf{y}^*, \mathbf{z}^*\right)$ and that $\phi\left(\mathbf{y}^k, \mathbf{z}^k\right) \to \phi\left(\mathbf{y}^*, \mathbf{z}^*\right) = 0$ as $k \to \infty$. In addition, since $\mathbf{v}_i \in \mathcal{S}$, then $\mathbf{v}_i \neq \mathbf{0}_m$ and hence $\mathbf{y}^k \neq \mathbf{z}_i^k$ for all $k \geq 0$ and for all $i = 1, 2, \ldots, M$, which implies that the functions ϕ_i, as defined in (26), are continuously differentiable in a small enough neighbourhood of $\left(\mathbf{y}^k, \mathbf{z}^k\right)$, for any $k \geq 0$. Therefore, it holds that (recall that $\hat{\partial}$ denotes the Fréchet sub-differential)

$$\hat{\partial}_{\mathbf{y}}\phi\left(\mathbf{y}^k, \mathbf{z}^k\right) = \sum_{i=1}^{M}\nabla_{\mathbf{y}}\phi_i\left(\mathbf{y}^k, \mathbf{z}_i^k\right) + \hat{\partial}\delta\left(\mathbf{y}^k\right). \tag{33}$$

Since $\mathbf{y}^k = \mathbf{y}^*$ for all $k \geq 0$ and since the Fréchet sub-differential and the limiting sub-differential sets coincide for convex functions, we have that $\mathbf{v} \in \partial\delta(\mathbf{y}^*) = \hat{\partial}\delta(\mathbf{y}^k)$. Hence, it follows from (33) that

$$\mathbf{g}^k \equiv \sum_{i=1}^{M} \nabla_{\mathbf{y}}\phi_i\left(\mathbf{y}^k, \mathbf{z}_i^k\right) + \mathbf{v} \in \hat{\partial}_{\mathbf{y}}\phi\left(\mathbf{y}^k, \mathbf{z}^k\right). \tag{34}$$

Now, taking the limit $k \to \infty$ in (34), it follows by simple calculations that (recall that $\|\mathbf{v}_i\| = 1$ for any $i = 1, 2, \ldots, M$)

$$\lim_{k\to\infty} \mathbf{g}^k = \sum_{i=1}^{M} \frac{|\beta_i| \cdot \operatorname{sign}(\beta_i)\,\mathbf{v}_i}{\|\operatorname{sign}(\beta_i)\,\mathbf{v}_i\|} + \mathbf{v} = \sum_{i=1}^{M} \beta_i \mathbf{v}_i + \mathbf{v} = \mathbf{g},$$

which implies that $\mathbf{g} \in \partial_{\mathbf{y}}\phi(\mathbf{y}^*, \mathbf{z}^*)$, as required. $\square$

REFERENCES

[1] M. P. Adams. Integral, mean and covariance of the simplex-truncated multivariate normal distribution. *PLoS ONE*, 17(7)(e0272014), 2022.

[2] H. M. Ammari. *The Art of Wireless Sensor Networks*, volume 1. Springer-Verlag, Berlin, 2014.

[3] A. Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*, volume 19. SIAM, 2014.

[4] A. Beck. *First-Order Methods in Optimization*, volume 25. SIAM, 2017.

[5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[6] A. Beck and M. Teboulle. A fast dual proximal gradient algorithm for convex minimization and applications. *Operations Research Letters*, 42(1):1–6, 2014.

[7] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks (TOSN)*, 2(2):188–220, 2006.

[8] Y.-T. Chan and K. Ho. A simple and efficient estimator for hyperbolic location. *IEEE Transactions on signal processing*, 42(8):1905–1915, 1994.

[9] L. Cheng, C. Wu, Y. Zhang, H. Wu, M. Li, and C. Maple. A survey of localization in wireless sensor network. *International Journal of Distributed Sensor Networks*, 8(12):962523, 2012.

[10] Y. Cong, B. Chen, and M. Zhou. Fast simulation of hyperplane-truncated multivariate normal distributions. *Bayesian Analysis*, 12(4):1017–1037, 2017.

[11] W. Dargie and C. Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. John Wiley & Sons, 2010.

[12] K. C. Das and R. Bapat. A sharp upper bound on the largest laplacian eigenvalue of weighted graphs. *Linear algebra and its applications*, 409:153–165, 2005.

[13] S. Goyal and M. S. Patterh. Modified bat algorithm for localization of wireless sensor network. *Wireless Personal Communications*, 86(2):657–670, 2016.

[14] E. Gur, A. Amar, and S. Sabach. Direct, fast and convergent solvers for the non-convex and non-smooth TDoA localization problem. *Digital Signal Processing*, 139:104074, 2023.

[15] E. Gur, S. Sabach, and S. Shtern. Alternating minimization based first-order method for the wireless sensor network localization problem. *IEEE Transactions on Signal Processing*, 68:6418–6431, 2020.

[16] E. Gur, S. Sabach, and S. Shtern. Convergent nested alternating minimization algorithms for nonconvex optimization problems. *Mathematics of Operations Research*, 48(1):53–77, 2022.

[17] E. Gur, S. Sabach, and S. Shtern. Nested alternating minimization with FISTA for non-convex and non-smooth optimization problems. *Journal of Optimization Theory and Applications*, 199(3):1130–1157, 2023.

[18] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge university press, 2012.

[19] M. Kanter and H. Proppe. Reduction of variance for gaussian densities via restriction to convex sets. *Journal of Multivariate Analysis*, 7(1):74–81, 1977.

[20] A. Kulaib, R. Shubair, M. Al-Qutayri, and J. W. Ng. An overview of localization techniques for wireless sensor networks. In *2011 international conference on innovations in information technology*, pages 167–172. IEEE, 2011.

[21] X. Li, Z. D. Deng, L. T. Rauchenstein, and T. J. Carlson. Contributed review: Source-localization algorithms and applications using time of arrival and time difference of arrival measurements. *Review of Scientific Instruments*, 87(4):041502, 2016.

[22] Y. Li and S. K. Ghosh. Efficient sampling methods for truncated multivariate normal and student-t distributions subject to linear inequality constraints. *Journal of Statistical Theory and Practice*, 9(4):712–732, 2015.

[23] K. W. K. Lui, W.-K. Ma, H.-C. So, and F. K. W. Chan. Semi-definite programming algorithms for sensor network node localization with uncertainties in anchor positions and/or propagation speed. *IEEE Transactions on Signal Processing*, 57(2):752–763, 2008.

[24] B. S. Mordukhovich. *Variational Analysis and Generalized Differentiation I: Basic Theory*, volume 330. Springer Science & Business Media, 2006.

[25] K. Mridula and P. Ameer. Localization under anchor node uncertainty for underwater acoustic sensor networks. *International Journal of Communication Systems*, 31(2):e3445, 2018.

[26] G. Naddafzadeh-Shirazi, M. B. Shenouda, and L. Lampe. Second order cone programming for sensor network localization with anchor position uncertainty. *IEEE transactions on wireless communications*, 13(2):749–763, 2013.

[27] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269(3):543–547, 1983.

[28] D. Niculescu. Positioning in ad hoc sensor networks. *IEEE network*, 18(4):24–29, 2004.

[29] E. Niewiadomska-Szynkiewicz and M. Marks. Optimization schemes for wireless sensor network localization. *International Journal of Applied Mathematics and Computer Science*, 19(2):291–302, 2009.

[30] A. K. Paul and T. Sato. Localization in wireless sensor networks: A survey on algorithms, measurement techniques, applications and challenges. *Journal of Sensor and Actuator Networks*, 6(4):24, 2017.

[31] L. Paull, S. Saeedi, M. Seto, and H. Li. AUV navigation and localization: A review. *IEEE Journal of oceanic engineering*, 39(1):131–149, 2013.

[32] N. Piovesan and T. Erseghe. Cooperative localization in WSNs: A hybrid convex/nonconvex solution. *IEEE Transactions on Signal and Information Processing over Networks*, 4(1):162–172, 2018.

[33] M. S. Rahman et al. *Basic graph theory*, volume 9. Springer, 2017.

[34] Q. Shi, C. He, H. Chen, and L. Jiang. Distributed wireless sensor network localization via sequential greedy optimization algorithm. *IEEE Transactions on Signal Processing*, 58(6):3328–3340, 2010.

[35] A. Simonetto and G. Leus. Distributed maximum likelihood sensor network localization. *IEEE Trans. Signal Processing*, 62(6):1424–1437, 2014.

[36] V. Sneha and M. Nagarajan. Localization in wireless sensor networks: a review. *Cybernetics and Information Technologies*, 20(4):3–26, 2020.

[37] C. Soares, J. Xavier, and J. Gomes. Distributed, simple and stable network localization. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 764–768. IEEE, 2014.

[38] C. Soares, J. Xavier, and J. Gomes. Simple and fast convex relaxation method for cooperative localization in sensor networks using range measurements. *IEEE Transactions on Signal Processing*, 63(17):4532–4543, 2015.

[39] S. Srirangarajan, A. H. Tewfik, and Z.-Q. Luo. Distributed sensor network localization with inaccurate anchor positions and noisy distance information. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 3, pages III–521. IEEE, 2007.

[40] R. Su, D. Zhang, C. Li, Z. Gong, R. Venkatesan, and F. Jiang. Localization and data collection in AUV-aided underwater sensor networks: Challenges and opportunities. *IEEE Network*, 33(6):86–93, 2019.

[41] X. Su, I. Ullah, X. Liu, and D. Choi. A review of underwater localization techniques, algorithms, and challenges. *Journal of Sensors*, 2020.

[42] C. Wu, C. Li, and Q. Long. A DC programming approach for sensor network localization with uncertainties in anchor positions. *Journal of Industrial & Management Optimization*, 10(3):817, 2014.

[43] R. Wu, W.-K. Ma, Y. Li, A. M.-C. So, and N. D. Sidiropoulos. Probabilistic simplex component analysis. *IEEE Transactions on Signal Processing*, 70:582–599, 2021.

[44] Y. Yan, G. Yang, H. Wang, and X. Shen. Robust multiple sensor localization via semidefinite relaxation in wireless sensor networks with anchor position uncertainty. *Measurement*, 196:111193, 2022.

[45] Y. Ye. Computational optimization laboratory. Universities of Stanford and Iowa. Online: https://web.stanford.edu/˜yyye/Col.html, visited August 15, 2022.