

# Alternating Minimization Based First-Order Method for the Wireless Sensor Network Localization Problem\*

Eyal Gur, Shoham Sabach, and Shimrit Shtern

Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology, Technion city,  
Haifa 3200003, Israel

[eyal.gur@campus.technion.ac.il](mailto:eyal.gur@campus.technion.ac.il), [ssabach@ie.technion.ac.il](mailto:ssabach@ie.technion.ac.il), [shimrit@technion.ac.il](mailto:shimrit@technion.ac.il)

## Abstract

We propose an algorithm for the Wireless Sensor Network localization problem, which is based on the well-known algorithmic framework of Alternating Minimization. We start with a non-smooth and non-convex minimization, and transform it into an equivalent smooth and non-convex problem, which stands at the heart of our study. This paves the way to a new method which is globally convergent: not only does the sequence of objective function values converge, but the sequence of the location estimates also converges to a unique location that is a critical point of the corresponding (original) objective function. The proposed algorithm has a range of fully distributed to fully centralized implementations, which all have the property of global convergence. The algorithm is tested over several network configurations, and it is shown to produce more accurate solutions within a shorter time relative to existing methods.

## 1 Introduction

Sensor networks consist of several wireless sensors located in a given area, for purposes such as environment monitoring, battlefields surveillance, etc. (see [2] for more examples and details). In our setting, each sensor is composed of a low-powered radio transceiver which monitors its immediate surroundings (e.g., temperature, sound, etc.), and a processor that collects and manipulates the data. Therefore, the location of each sensor has a significant role. Since the number of sensors in a network can be large (even thousands of sensors), it is not cost effective to equip each one of them with a GPS device, nor to deploy the sensors in a known logged location.

A network in this context is described as a group of  $K$  sensors, each denoted by an index in the set  $\{1, 2, \dots, K\}$ . The communication between two sensors  $i$  and  $j$  is made available only if the distance

---

\*This research was partially supported by Israel Science Foundation Grant 1460/19.

between the two is at most  $r \geq 0$ , which is a given radio range for communication. In this case, we say that the sensors  $i$  and  $j$  are neighbors. The Wireless Sensor Network (WSN) localization problem aims at finding the location of all sensors in the network based on a few anchors, which are sensors with a known location (e.g., via GPS devices), and noisy distance measurements from each sensor to its neighbors. Mathematically speaking, given  $K$  sensors and anchors in  $\mathbb{R}^n$ , where  $m$  of which ( $m < K$ ) are anchors, we wish to estimate the location of all  $N = K - m$  sensors in  $\mathbb{R}^n$ .

We denote, for simplicity, the set of the  $N$  sensors to be located by  $\mathcal{V} := \{1, 2, \dots, N\}$ , and the set of the  $m$  anchors by  $\mathcal{A} := \{N + 1, N + 2, \dots, K\}$ . For an anchor  $j \in \mathcal{A}$  we denote its given location by  $\mathbf{a}_j \in \mathbb{R}^n$ . Additionally, we denote by  $\mathcal{E}$  the set of all pairs of neighboring sensors, i.e.,  $(i, j) \in \mathcal{E}$  if  $i < j$  and sensors  $i$  and  $j$  are neighbors, which means that the distance between them is at most  $r$ . For each pair  $(i, j) \in \mathcal{E}$ , the (noisy) measurement of their distance is denoted by  $d_{ij}$  (following [32], we assume w.l.o.g. that  $d_{ij} = d_{ji}$ )<sup>1</sup>. We use the notation  $M = |\mathcal{E}|$ . Note that since the locations of the anchors are known exactly, we can ignore all edges between anchor nodes, which means that if  $(i, j) \in \mathcal{E}$  then  $i \in \mathcal{V}$ . Moreover, we denote by  $\mathbf{x}_i \in \mathbb{R}^n$  the true location of sensor  $i \in \{1, 2, \dots, N\}$ , and we denote by  $\mathbf{x} \in \mathbb{R}^{nN}$  the vector obtained by concatenating the vectors  $\mathbf{x}_i$  for all  $i \in \{1, 2, \dots, N\}$  into a single column vector. Using these notations, in this work we adopt the following non-smooth and non-convex formulation of the WSN localization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^{nN}} \left\{ \sum_{(i,j) \in \mathcal{E}_1} (\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij})^2 + \sum_{(i,j) \in \mathcal{E}_2} (\|\mathbf{x}_i - \mathbf{a}_j\| - d_{ij})^2 \right\}, \quad (1)$$

where  $\mathcal{E}_1$  is the subset of  $\mathcal{E}$  with all pairs  $(i, j)$  for which both sensors  $i$  and  $j$  are non-anchors, while  $\mathcal{E}_2$  is the subset of all pairs  $(i, j)$  in  $\mathcal{E}$  for which sensor  $i$  is a non-anchor and sensor  $j$  is an anchor.

This formulation enjoys a statistical interpretation as of finding the maximum-likelihood estimator of the locations, given that the measurement noises are independently normally (Gaussian) distributed (see, for example, [28]).

It should be noted that a closely related problem is the Single Source Localization (SSL), which can be seen as a particular case of the WSN problem for  $N = 1$ . However, the network variant possess several challenges which do not exist in the SSL problem both from an algorithmic and a computational standpoints as we highlight below (for instance, distributed and/or parallel implementations).

---

<sup>1</sup>Note that, in this paper, we use the standard assumption that all distance measurements for sensor pairs which are within the communication radius are available.

## 1.1 Existing Methods for Convex Relaxations

One common approach for tackling the non-convex Problem (1), is by solving certain convex relaxations of the problem<sup>2</sup>. For example, in [32], the authors show that the first summed terms in Problem (1) are just the squared distance in  $\mathbb{R}^n$ , of the point  $\mathbf{x}_i - \mathbf{x}_j$  to the sphere with radius  $d_{ij}$  centered at the origin (similarly for the other sum in the objective). Thus, the convexification presented in [32] is derived by taking the squared distance to the ball of radius  $d_{ij}$ , instead to the sphere. This yields a smooth and convex problem with separable ball constraints that can be solved by classical techniques of convex optimization, especially the well-known Accelerated Projected Gradient method of Nesterov [23].

Another popular relaxation, which stands at the basis of many works in this area, is based on the classical "lifting" technique, in which the non-convexity in the objective is replaced by quadratic equality constraints. These constraints are then relaxed to inequality constraints, and the problem is reformulated as a Semi-Definite Programming (SDP) problem (see, for instance, [12, 31, 21] and references therein). These SDP problems can then be solved using off-the-shelf external solvers that utilize interior-point methods, and are known to be computationally expensive. Thus, these techniques may suffer from an increased computational complexity, which is less desirable when dealing with large-scale problems, where thousands of sensors are deployed (it should be remembered that the "lifting" process significantly increases the dimension of the problem to be solved). Thus, in many cases a further relaxation of the SDP constraints, called edge-based SDP [34], is used in order to obtain a distributed algorithm, which is less computationally demanding (see, for instance, [31] and [21]). However, this still requires the implementation of an interior-point method on each sensor, which could be a major limitation due to the small computational power of individual sensors.

While convex relaxations guarantee convergence to a global minimum point, this point is not necessarily a global minimizer, or even a critical point, of the original (non-relaxed) formulation. See [27] for a discussion on this phenomenon. In order to demonstrate this phenomenon, we have also conducted a numerical comparison between our method, which is applied to the original non-convex and non-smooth formulation of the problem, to the relaxation approach suggested in [32] (see Section 6). Indeed, due to the superior estimation quality of our method compared to the relaxation, as well as similar evidences in previous papers, we do not compare our method to other methods which aim at solving any kind of convex relaxation of the problem.

An alternative formulation used in the literature, that also allows for inaccurate location estimates of

---

<sup>2</sup>The literature on convex relaxations is extensive, however, since this is not the focus of our paper we do not give a thorough review of this approach, but rather highlight few ideas and their limitations.

the anchor sensors (see, for example, [11] and references therein), is given by the following smooth and non-convex problem:

$$\min_{\mathbf{x} \in \mathbb{R}^{n_N}} \left\{ \sum_{(i,j) \in \mathcal{E}_1} \left( \|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2 \right)^2 + \sum_{(i,j) \in \mathcal{E}_2} \left( \|\mathbf{x}_i - \mathbf{a}_j\|^2 - d_{ij}^2 \right)^2 \right\}. \quad (2)$$

While this formulation is smooth, it lacks the statistical interpretation of Problem (1). Additionally, it is suggested in [10], that at least for the Single Source Localization problem, solving formulation (1) produces more accurate estimations when compared with formulation (2). Moreover, it does not change the fact that Problem (2) remains non-convex. Therefore, also in this case convex relaxations, such as SDP relaxations, are often used. However, solving these relaxations come with drawbacks, which have already been discussed above. Therefore, even though many methods were developed to tackle this formulation, we focus our study on the formulation (1) and provide a comparison between methods that tackles this model.

It is worth mentioning that the literature on the WSN localization problem also includes heuristic approaches (see, for example, [24], [16] and [1]). However, these approaches have either weak or no theoretical guarantees on both the convergence of the sequence and quality of the resulting solution.

## 1.2 Related First-Order Methods

In this paper, we aim at solving the original unconstrained non-smooth and non-convex Problem (1) directly, using first-order methods, namely methods that exploit information on values and (sub)gradients of the involved functions. We now focus on three relevant works [27], [30] and [33], which also propose, among other things, first order methods for solving Problem (1). In these works, the first-order method to solve Problem (1) is used to improve the localization accuracy of solutions that were obtained from solving some convex relaxation of Problem (1), see Section 1.1 for a discussion about convex relaxations. The description of this two-stage strategy is postponed to Section 6. At this moment, we focus the discussion on the first-order algorithms as standalone optimization algorithms for solving Problem (1).

In [27], a linear constrained equivalent reformulation of Problem (1) is studied and a fully distributed ADMM method was suggested to solve the non-convex reformulated problem. The theoretical result of this work, which is based on [3], says that the generated sequence convergence to a local minima<sup>3</sup>. While this algorithm performance seems to be promising, it posses some practical challenges: (i) tuning of

---

<sup>3</sup>We have found this result misleading. In [3] the authors provides conditions, in the non-convex setting, for a sequence generated by ADMM to have limit points, and for these limit points to be KKT points, however no guarantees for local minima are established there.

several parameters, (ii) a sub-algorithm to solve the non-convex problems with respect to the non-anchor sensors and (iii) the algorithm may generate iterations which are not well-defined. We will discuss these limitations further in Section 6.

In [30], a reformulation of Problem (1) as a constrained minimization is considered<sup>4</sup>, similar to the reformulation that we suggest below. This reformulation requires the definition of new variables, and immediately suggests an Alternating Minimization based algorithm that they call Non-Convex Sequential Greedy method. The obtained algorithm, as the algorithm of [27], is not always well-defined (as we discuss below). This is a fully distributed algorithm that is shown to generate a converging sequence of function values. The authors also show that any limit point of the generated sequence is a KKT point of the reformulated problem (but no relations to the original problem are provided)

In [33], the authors reformulate the problem as in [32], but instead of employing a convex relaxation, they aim at solving the reformulated non-convex problem using the classical Projected Gradient algorithm. However, the paper does not provide any theoretical guarantees about their algorithm.

### 1.3 Our Contribution

As we wish to solve the original non-smooth formulation given in Problem (1) using first-order methods, we first address the inherent non-smoothness of the problem. We adopt a simple variational representation of the Euclidean norm, which enables us to transform Problem (1) into an equivalent constrained smooth and non-convex minimization problem. This formulation, which is closely related to the formulation used in [30], will serve as a starting point for our study.

Our first contribution is an algorithmic framework for solving the original non-convex Problem (1), that includes the whole spectrum ranging from a fully centralized to a fully distributed algorithm. The concept of Alternating Minimization (AM), which stands at the heart of our framework, exploits the structure of our reformulation, to produce a well-defined and parameter-free algorithm. Moreover, our second contribution is proving theoretical guarantees of this algorithm, which are stronger in comparison to those obtained in the two mentioned works [27] and [30]. More precisely, we prove global convergence of the generated sequence of network location estimates (in the sense that the whole sequence converges), that is, the location estimates of each sensor converges to a unique limit point<sup>5</sup>. In addition, we show that this unique limit point of the network location estimates is a critical point of the function under minimization in Problem (1), where a point is a critical point if its sub-differential contains the zero

---

<sup>4</sup>It should be noted that the authors of [30] study a more general form of Problem (1), which allows for the anchors to have noisy location estimates.

<sup>5</sup>Since we are dealing with a non-convex problem, this point obviously depends on the starting point of the algorithm.

vector. In order to do so, we use a recent proof technique, which was proposed in [4] and [5], and later was unified and simplified in [14] (see also [15] for a recent concise description of this technique). To the best of our knowledge, this is the first work that proves a global convergence result for a first-order algorithm that solves the original non-convex WSN problem.

As mentioned above, our algorithm implementation can range from fully distributed, i.e., each sensor updates its own location estimate through information received over communication, to fully centralized, where the location estimate of the entire network is calculated on one central processor.

Additionally, we show that the fully distributed version of our algorithm, to be developed below, can be seen as a modification of the algorithm presented in [30], which is always well-defined, and enjoys our stronger convergence results.

The paper is organized as follows: the smooth and non-convex equivalent formulation of Problem (1) is introduced in Section 2. In Section 3, we introduce the fully centralized and fully distributed algorithms for tackling Problem (1). We present the Unifying AM Algorithm that captures both centralized and distributed versions, in Section 4, and discuss the capability of this unified approach to also capture partially-distributed and partially-parallelized algorithmic variations. We analyze the convergence of the Unifying AM Algorithm in Section 5. In Section 6, we conduct numerical experiments to evaluate the performance of our algorithms and compare them with existing methods.

## 2 Problem Formulation

In the introduction above, we have already mentioned that since Problem (1) is non-smooth, our first step is to derive an equivalent smooth reformulation of the problem, where by equivalent we mean that a critical point of the reformulated problem is also a critical point of Problem (1), and vice versa.

Notice that the first sum in the objective function of Problem (1) can be written explicitly as

$$\sum_{(i,j) \in \mathcal{E}_1} \left( \|\mathbf{x}_i - \mathbf{x}_j\|^2 - 2d_{ij} \|\mathbf{x}_i - \mathbf{x}_j\| + d_{ij}^2 \right). \quad (3)$$

The non-smoothness of (3) comes from the terms  $\|\mathbf{x}_i - \mathbf{x}_j\|$  (the true distance between sensors  $i$  and  $j$ ) for all  $(i, j) \in \mathcal{E}_1$ . To eliminate this, we first denote by  $\mathcal{B} \equiv B_1(\mathbf{0}_n)$  the unit ball in  $\mathbb{R}^n$  centered at the origin, and by  $\mathcal{B}^M$  the Cartesian product of  $M$  such balls, where  $M$  is the number of edges in the network. Motivated by the recent work [22] that focuses on the Single Source Localization problem, we

use the following fact on the WSN problem, which easily follows from the Cauchy-Schwartz inequality:

$$\|\mathbf{x}_i - \mathbf{x}_j\| = \max \{ \mathbf{u}_{ij}^T (\mathbf{x}_i - \mathbf{x}_j) : \mathbf{u}_{ij} \in \mathcal{B} \},$$

where  $\mathbf{u}_{ij} \in \mathbb{R}^n$  is an auxiliary variable defined for each pair  $(i, j) \in \mathcal{E}_1$ , that achieves the optimal value of  $\|\mathbf{x}_i - \mathbf{x}_j\|$  at  $(\mathbf{x}_i - \mathbf{x}_j) / \|\mathbf{x}_i - \mathbf{x}_j\|$  when  $\mathbf{x}_i - \mathbf{x}_j \neq \mathbf{0}_n$  (otherwise any  $\mathbf{u}_{ij} \in \mathcal{B}$  would be an optimal solution, but in this paper we will always take  $\mathbf{u}_{ij} = \mathbf{0}_n$ ).

Similar manipulations can be done on the second sum in the objective function of Problem (1). Therefore, by omitting the constant terms  $d_{ij}^2$ ,  $(i, j) \in \mathcal{E}$ , we arrive at the following smooth and constrained equivalent reformulation of Problem (1):

$$\min_{\mathbf{x} \in \mathbb{R}^{nN}, \mathbf{u} \in \mathcal{B}^M} \left\{ \sum_{(i,j) \in \mathcal{E}_1} \left( \|\mathbf{x}_i - \mathbf{x}_j\|^2 - 2d_{ij} \mathbf{u}_{ij}^T (\mathbf{x}_i - \mathbf{x}_j) \right) + \sum_{(i,j) \in \mathcal{E}_2} \left( \|\mathbf{x}_i - \mathbf{a}_j\|^2 - 2d_{ij} \mathbf{u}_{ij}^T (\mathbf{x}_i - \mathbf{a}_j) \right) \right\}, \quad (4)$$

where  $\mathbf{u}$  is the vector obtained by concatenating the vectors  $\mathbf{u}_{ij}$  for all  $(i, j) \in \mathcal{E}$  into a single column vector.

For the sake of simplifying the developments to come, we wish to rewrite Problem (4) using matrix notations. To this end, one could notice that the network of the original problem can be viewed as an undirected graph, where the vertices are the sensors, and  $\mathcal{E}$  is the set of edges. For simplicity, we fix some ordering of the set  $\mathcal{E}$  such that all edges in  $\mathcal{E}_1$  proceed the edges in  $\mathcal{E}_2$ . We now define several essential matrices for the formulation of the WSN localization problem:

- $\tilde{\mathbf{Q}} \in \mathbb{R}^{|\mathcal{E}_1| \times N}$  is the arc-node incidence matrix of the subgraph containing only the vertices in  $\mathcal{V}$  and edges in  $\mathcal{E}_1$ , i.e., if  $(i, j) \in \mathcal{E}_1$  is the  $l$ -th edge in the set  $\mathcal{E}_1$ , then  $\tilde{\mathbf{Q}}_{li} = 1$ ,  $\tilde{\mathbf{Q}}_{lj} = -1$  and all other entries in the  $l$ -th row are equal to 0.
- $\tilde{\mathbf{A}} \in \mathbb{R}^{|\mathcal{E}_2| \times N}$  is the indicator matrix of the set of edges  $\mathcal{E}_2$ , i.e.,  $\tilde{\mathbf{A}}_{li} = 1$  if the  $l$ -th edge of  $\mathcal{E}_2$  connects the sensor  $i \in \mathcal{V}$  to an anchor in  $\mathcal{A}$  and all other entries in the  $l$ -th row are equal to 0.
- $\tilde{\mathbf{B}} \in \mathbb{R}^{|\mathcal{E}_2| \times m}$  is such that  $\tilde{\mathbf{B}}_{lj} = -1$  if the  $l$ -th edge of  $\mathcal{E}_2$  connects a certain sensor in  $\mathcal{V}$  to the anchor  $j + N \in \mathcal{A}$ . It should be noted that the matrices  $\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{B}}$  form together an arc-node incidence matrix of the subgraph containing the vertices in  $\mathcal{V} \cup \mathcal{A}$  and edges in  $\mathcal{E}_2$ .
- $\tilde{\mathbf{D}} \in \mathbb{R}^{M \times M}$  is the diagonal matrix with  $d_{ij}$ ,  $(i, j) \in \mathcal{E}$ , as the entries on the diagonal.
- $\tilde{\mathbf{P}} \equiv \tilde{\mathbf{Q}}^T \tilde{\mathbf{Q}} + \tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$  (note that the matrix  $\tilde{\mathbf{Q}}^T \tilde{\mathbf{Q}}$  is the so-called Laplacian matrix of the corresponding subgraph).

Since each sensor is in  $\mathbb{R}^n$  it will be convenient to define the Kronecker matrix product of  $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{D}}, \tilde{\mathbf{P}}$  and  $\tilde{\mathbf{Q}}$  with the identity matrix  $\mathbf{I}_n$ , which are denoted by  $\mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{P}$  and  $\mathbf{Q}$ , respectively.

Finally, denoting by  $\mathbf{a} \in \mathbb{R}^{nm}$ , the vector derived by concatenating the vectors  $\mathbf{a}_j, j \in \mathcal{A}$ , into a single column vector, we see that Problem (4) can be written equivalently as:

$$\min_{\substack{\mathbf{x} \in \mathbb{R}^{nN} \\ \mathbf{u} \in \mathcal{B}^M}} \left\{ \mathbf{x}^T \mathbf{P} \mathbf{x} - 2 (\mathbf{W} \mathbf{u} + \mathbf{A}^T \mathbf{B} \mathbf{a})^T \mathbf{x} + 2 \mathbf{s}^T \mathbf{u} \right\}, \quad (5)$$

where we define the matrix  $\mathbf{W} = [\mathbf{Q}^T, \mathbf{A}^T] \mathbf{D}$  and the vector  $\mathbf{s} = ([\mathbf{0}_{nm \times n|\mathcal{E}_1|}, \mathbf{B}^T] \mathbf{D})^T \mathbf{a}$ .

We denote the objective function of Problem (5) by  $G(\mathbf{x}, \mathbf{u})$ . We will also use the following function which takes the constraints on the block  $\mathbf{u}$  via its indicator formulation

$$F(\mathbf{x}, \mathbf{u}) := G(\mathbf{x}, \mathbf{u}) + \sum_{(i,j) \in \mathcal{E}} \delta_{\mathcal{B}}(\mathbf{u}_{ij}), \quad (6)$$

where  $\delta_{\mathcal{B}}(\cdot)$  denotes the indicator function of the set  $\mathcal{B}$  (which is defined to be zero in  $\mathcal{B}$  and  $+\infty$  outside). This results in the following unconstrained optimization problem

$$\min \{ F(\mathbf{x}, \mathbf{u}) : \mathbf{x} \in \mathbb{R}^{nN}, \mathbf{u} \in \mathbb{R}^{nM} \}. \quad (7)$$

This formulation will be the starting point of our study. We are interested in designing a simple and parameter-free algorithm for solving Problem (1) via its equivalent reformulation (7), which globally converges to a critical point of the original problem (1). To this end, we will show that our algorithm finds a critical point of  $F(\cdot, \cdot)$ , that is, a pair  $(\mathbf{x}, \mathbf{u})$  which satisfies

$$(\mathbf{0}_{nN}, \mathbf{0}_{nM}) \in \partial F(\mathbf{x}, \mathbf{u}) = \{\nabla_{\mathbf{x}} F(\mathbf{x}, \mathbf{u})\} \times \{\partial_{\mathbf{u}} F(\mathbf{x}, \mathbf{u})\},$$

where  $\partial\varphi$  denotes the classical subdifferential of  $\varphi$ , that can be used here since the function  $\mathbf{u} \rightarrow F(\mathbf{x}, \mathbf{u})$  is convex for any fixed  $\mathbf{x} \in \mathbb{R}^{nN}$ .

In order to simplify the inclusion above we will first need the following additional notation. We denote by  $\mathcal{N}(i)$  the set of sensor  $i$ 's neighbors, i.e.,  $j \in \mathcal{N}(i)$  means that  $(i, j) \in \mathcal{E}$  or<sup>6</sup>  $(j, i) \in \mathcal{E}$ . We also denote  $M_i = |\mathcal{N}(i)|$  and thus  $M = \frac{1}{2} \sum_{i=1}^N M_i$ .

---

<sup>6</sup>Since  $\mathcal{E}$  includes only pairs  $(i, j)$  for which  $i < j$ , we need to consider also the pairs where  $j < i$ .

Therefore, the inclusion above easily translates into the following conditions:

$$\mathbf{0}_n = \sum_{j \in \mathcal{N}(i) \cap \mathcal{V}} (\mathbf{x}_i - \mathbf{x}_j - d_{ij} \mathbf{u}_{ij}) + \sum_{j \in \mathcal{N}(i) \cap \mathcal{A}} (\mathbf{x}_i - \mathbf{a}_j - d_{ij} \mathbf{u}_{ij}), \quad (8)$$

for all  $i \in \{1, 2, \dots, N\}$  and

$$\mathbf{0}_n \in -2d_{ij} (\mathbf{x}_i - \mathbf{x}_j) + \partial \delta_{\mathcal{B}}(\mathbf{u}_{ij}), \quad \forall (i, j) \in \mathcal{E}_1, \quad (9)$$

$$\mathbf{0}_n \in -2d_{ij} (\mathbf{x}_i - \mathbf{a}_j) + \partial \delta_{\mathcal{B}}(\mathbf{u}_{ij}), \quad \forall (i, j) \in \mathcal{E}_2, \quad (10)$$

where we define  $\mathbf{u}_{ji} = -\mathbf{u}_{ij}$  for all  $(i, j) \in \mathcal{E}_1$ . In the Appendix below we prove that a vector  $\mathbf{x}$ , which satisfies these inequalities, must be a critical point of the original problem (1) as recorded in the following result.

**Proposition 1.** *Let  $(\mathbf{x}^*, \mathbf{u}^*)$  be a critical point of Problem (7), then  $\mathbf{x}^*$  is a critical point of the original problem (1).*

In the forthcoming section we develop a simple solution scheme for solving Problem (7), which globally converges to a pair that satisfies the premises of Proposition (1) and therefore finds a critical point of the original WSN problem. To the best of our knowledge, this is the first algorithm that is guaranteed to converge to a critical point of the original non-smooth and non-convex problem (in the papers mentioned in Section 1.2 only *subsequences convergence* to critical/KKT points of the *reformulated problem* is proven).

### 3 Centralized and Distributed Algorithms

The two blocks structure of Problem (7) (in addition to the fact that there is no coupling constraint between these blocks) immediately suggests the application of optimization methods that employ the concept of Alternating Minimization (AM), which is a very useful technique to tackle complex convex and non-convex problems. The main reason for that is the ability to exploit the following nice feature of Problem (7): each sub-problem with respect to one block of variables (while the other block remains fixed) can be easily solved.

In the context of Problem (7), a basic AM based algorithm will have the following form

## A Centralized Alternating Minimization for WSN

**Initialization.**  $\mathbf{u}^0 \in \mathbb{R}^{nM}$

**General Step.** For  $k \in \mathbb{N}$ ,

1. **Sensors update.**

$$\mathbf{x}^{k+1} = \underset{\mathbf{x} \in \mathbb{R}^{nN}}{\operatorname{argmin}} F(\mathbf{x}, \mathbf{u}^k) = \mathbf{P}^{-1} (\mathbf{W}\mathbf{u} + \mathbf{A}^T \mathbf{B}\mathbf{a})^T. \quad (11)$$

2. **Auxiliary update.** For all  $(i, j) \in \mathcal{E}$  we denote

$$\mathbf{v}_{ij}^{k+1} = \begin{cases} \mathbf{x}_i^{k+1} - \mathbf{x}_j^{k+1}, & (i, j) \in \mathcal{E}_1, \\ \mathbf{x}_i^{k+1} - \mathbf{a}_j, & (i, j) \in \mathcal{E}_2. \end{cases}$$

Then

$$\mathbf{u}_{ij}^{k+1} = \begin{cases} \frac{\mathbf{v}_{ij}^{k+1}}{\|\mathbf{v}_{ij}^{k+1}\|}, & \mathbf{v}_{ij}^{k+1} \neq \mathbf{0}_n, \\ \mathbf{0}_n, & \text{otherwise.} \end{cases} \quad (12)$$

$$\mathbf{u}_{ji}^{k+1} = -\mathbf{u}_{ij}^{k+1}$$

The sub-problem in (11) is solved explicitly by solving the linear equation  $\nabla_{\mathbf{x}} F(\mathbf{x}^{k+1}, \mathbf{u}^k) = \mathbf{0}$  (in this respect see also Proposition 2 below). Therefore, the obtained solution (see exact formula above) means that this algorithm is centralized in the sense that the location update rule (11) requires to perform the computation on a single processing unit. The problem of minimizing the function  $F(\mathbf{x}^{k+1}, \mathbf{u})$  with respect to  $\mathbf{u}$  consists of separable minimization problems (see (4)), each of minimizing a linear function over the unit ball, which results with the formula given in (12). Before proceeding, we note that this algorithm generates a well-defined sequence (this is a property that the algorithms mentioned in the Introduction do not necessarily possess, more on that in Section 6).

A closer look on Problem (7) reveals that the AM approach can be used in a more refined way to produce a fully distributed algorithm. More precisely, we would like to allow each sensor to update its own location estimate, and to this end, we will look at each sub-block  $\mathbf{x}_i$ ,  $i \in \{1, 2, \dots, N\}$ , as a separated block of variables and employ the AM approach on the  $N + 1$  blocks:  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  and  $\mathbf{u}$ , as explicitly recorded now.

### A Distributed Alternating Minimization for WSN

**Initialization.**  $\mathbf{x}^0 \in \mathbb{R}^{nN}$  and  $\mathbf{u}^0 \in \mathbb{R}^{nM}$

**General Step.** For  $k \in \mathbb{N}$ ,

1. **Sensors update.** For all  $i \in \{1, 2, \dots, N\}$

$$\mathbf{x}_i^{k+1} = \frac{1}{M_i} \left( \sum_{\substack{j \in \mathcal{N}(i) \cap \mathcal{V} \\ j < i}} \mathbf{x}_j^{k+1} + \sum_{\substack{j \in \mathcal{N}(i) \cap \mathcal{V} \\ j > i}} \mathbf{x}_j^k + \sum_{j \in \mathcal{N}(i)} d_{ij} \mathbf{u}_{ij}^k + \sum_{l \in \mathcal{N}(i) \cap \mathcal{A}} \mathbf{a}_l \right). \quad (13)$$

2. **Auxiliary update.** The same as in (12).

The update rule of the sensors as given in (13) easily follows from writing the optimality condition of minimizing the function  $\mathbf{x}_i \rightarrow F(\mathbf{x}_1^{k+1}, \dots, \mathbf{x}_{i-1}^{k+1}, \mathbf{x}_i, \mathbf{x}_{i+1}^k, \dots, \mathbf{x}_N^k)$  (see (8) in this respect).

Note that while this algorithm is fully distributed, it is serial in updating the blocks  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ . However, parallelization can be achieved in the updating of the auxiliary variables  $\mathbf{u}_{ij}$ ,  $(i, j) \in \mathcal{E}$ . In the upcoming section we discuss how one can further parallelize this algorithm when updating the blocks  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ .

## 4 A Unifying AM Algorithm for WSN

The centralized and distributed algorithms presented above are both particular instances of our Unifying AM Algorithm, which is developed next. The idea is to divide the set of sensors into  $q \in \{1, 2, \dots, N\}$  disjoint clusters  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_q$  such that  $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \dots \cup \mathcal{C}_q = \{1, 2, \dots, N\}$ , thus forming a partition of the set of sensors  $\mathcal{V}$ . It is easy to see that the fully centralized version can be recovered when the number of clusters is  $q = 1$ , while the fully distributed version is recovered when  $q = N$ .

In order to derive an AM based algorithm that exploits the division of the block  $\mathbf{x}$  into clusters we will need to denote sub-vectors with respect to this division. For each cluster  $\mathcal{C}_i$ ,  $1 \leq i \leq q$ , we denote by  $\bar{\mathbf{x}}_i$  the vector which is constructed by concatenating the vectors  $\mathbf{x}_j$  for all  $j \in \mathcal{C}_i$ . Therefore, we can now apply the AM technique to Problem (7), with respect to the  $q+1$  blocks:  $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \dots, \bar{\mathbf{x}}_q$  and  $\mathbf{u}$ . More precisely, the algorithm is recorded now in an abstract form (explicit formulas can be easily derived by writing the corresponding optimality conditions).

## A Unifying Alternating Minimization for WSN

**Initialization.**  $\mathbf{x}^0 \in \mathbb{R}^{nN}$  and  $\mathbf{u}^0 \in \mathbb{R}^{nM}$

**General Step.** For  $k \in \mathbb{N}$ ,

1. **Sensors update.** For all  $i \in \{1, 2, \dots, N\}$

$$\begin{aligned}\bar{\mathbf{x}}_1^{k+1} &= \underset{\bar{\mathbf{x}}_1}{\operatorname{argmin}} F\left(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2^k, \dots, \bar{\mathbf{x}}_q^k, \mathbf{u}^k\right), \\ \bar{\mathbf{x}}_2^{k+1} &= \underset{\bar{\mathbf{x}}_2}{\operatorname{argmin}} F\left(\bar{\mathbf{x}}_1^{k+1}, \bar{\mathbf{x}}_2, \dots, \bar{\mathbf{x}}_q^k, \mathbf{u}^k\right), \\ &\vdots \\ \bar{\mathbf{x}}_q^{k+1} &= \underset{\bar{\mathbf{x}}_q}{\operatorname{argmin}} F\left(\bar{\mathbf{x}}_1^{k+1}, \bar{\mathbf{x}}_2^{k+1}, \dots, \bar{\mathbf{x}}_q, \mathbf{u}^k\right).\end{aligned}$$

2. **Auxiliary update.** The same as in (12).

Before analyzing the Unifying AM Algorithm presented above, we would like to provide another advantage of the idea of dividing the sensors into disjoint clusters.

There are many ways in which one may divide the sensors into clusters, each with its own benefits. The most naive approach would be *geographical clustering*, in which several neighboring sensors are collected into one cluster. This division makes the algorithm more centralized, since the updates of each cluster take into account the interrelations between the sensors that comprise it. Alternatively, one may use *colored clustering*, i.e., clusters that consist of non-neighboring sensors. In this clustering approach, each cluster update is equivalent to independently updating each of the sensors that comprise it, similar to their update in the fully distributed algorithm presented above (see step (13)). However, in sharp contrast to the fully distributed update, the independent sensor updates in each colored cluster can be done in parallel rather than serially. Obtaining a colored clustering can be done efficiently via a distributed graph-coloring algorithm [6]. To summarize, colored clustering results in a fully distributed and partially parallel algorithm. We demonstrate the benefits of the colored clustering approach in Section 6, and therefore it deserves a deeper study in a future work.

## 5 Analysis of the Unifying AM Algorithm

### 5.1 Basic Properties

Recalling Problem (5)

$$\min_{\substack{\mathbf{x} \in \mathbb{R}^{nN} \\ \mathbf{u} \in \mathcal{B}^M}} \left\{ \mathbf{x}^T \mathbf{P} \mathbf{x} - 2 (\mathbf{W} \mathbf{u} + \mathbf{A}^T \mathbf{B} \mathbf{a})^T \mathbf{x} + 2 \mathbf{s}^T \mathbf{u} \right\},$$

throughout the rest of the paper we will pose the following mild assumption on the networks that we can handle.

**Assumption 1.** (i) *The graph obtained by the network is connected.*

(ii) *There is at least one anchor sensor, that is,  $m \geq 1$ .*

It should be noted that if Item (i) does not hold, namely, the network has more than one connected component, then it can be divided into disjoint connected sub-networks that can be treated separately, as long as each such sub-network contains an anchor node.

Under this assumption on the network we can obtain the following basic properties of our problem (due to the technical nature of the proof, we postpone it to the Appendix below).

**Proposition 2.** (i) *The matrix  $\mathbf{P}$  is positive definite.*

(ii) *The function  $\mathbf{x} \rightarrow F(\mathbf{x}, \mathbf{u})$  is strongly convex for any fixed  $\mathbf{u}$ .*

(iii) *The function  $F(\cdot, \cdot)$  is coercive, that is,  $F(\mathbf{x}, \mathbf{u}) \rightarrow \infty$  as  $\|(\mathbf{x}, \mathbf{u})\| \rightarrow \infty$ .*

(iv) *Problem (7) attains its optimal solution.*

## 5.2 Global Convergence

Our main goal in this part is to show that the Unifying AM Algorithm presented above globally converges to a critical point of the original Problem (1). As discussed in Section 2, we will first show that our algorithm finds critical points  $(\mathbf{x}^*, \mathbf{u}^*)$  of Problem (7) and then the desired conclusion will follow from Proposition 1. To this end we will rely on a recent proof methodology of first-order methods in the non-convex setting, which was developed first in [4, 5] and was extended and simplified in [14] (see [15] for a recent simple and concise summary of the methodology). The main mathematical tool that stands at the heart of this proof methodology, is the Kurdyka-Łojasiewicz (KL) property [20, 19] (see [13] for an extension to non-smooth functions).

The general result in [14] states that if an algorithm, which is designed to solve a specific optimization problem, generates a gradient-like descent sequence (in terms of [15, Definition 6.1, p. 2147]), then it globally converges to a critical point of the problem. A recent modification, given in [29], shows that if the algorithm generates a gradient-like descent sequence only with respect to a subset of the variables (see [29, Definition 4.2, p. 661]), then global convergence can be shown with respect to that subset. In our case, we are interested in proving global convergence only with respect to the original variable  $\mathbf{x}$ .

Therefore, we focus on showing that the Unifying AM Algorithm for solving Problem (7) generates a sequence  $\{\mathbf{x}^k\}_{k \in \mathbb{N}}$  which is indeed a gradient-like descent sequence for minimizing the objective function  $F$ . Our main result is as follows.

**Theorem 3.** *Let  $\{(\mathbf{x}^k, \mathbf{u}^k)\}_{k \in \mathbb{N}}$  be a sequence generated by the Unifying AM Algorithm. Then, the sequence  $\{\mathbf{x}^k\}_{k \in \mathbb{N}}$  has a finite length, i.e.,  $\sum_{k=1}^{\infty} \|\mathbf{x}^{k+1} - \mathbf{x}^k\| < \infty$ , and it converges to some  $\mathbf{x}^*$ , which is a critical point of Problem (1).*

In order to prove this result, we first present three auxiliary lemmas. For the sake of proving the lemmas, we need to recall that the vector  $\mathbf{x}$  is split into  $q$  blocks according to the clusters  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_q$ , that is,  $\mathbf{x} = (\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \dots, \bar{\mathbf{x}}_q)$ . During the updating process of the Unifying AM Algorithm, we will need the following notations. Fix an iteration  $k \in \mathbb{N}$  and a block index  $1 \leq i \leq q$ ,

$$\mathbf{x}^{k,i} = (\bar{\mathbf{x}}_1^{k+1}, \dots, \bar{\mathbf{x}}_{i-1}^{k+1}, \bar{\mathbf{x}}_i^k, \bar{\mathbf{x}}_{i+1}^{k+1}, \dots, \bar{\mathbf{x}}_q^k).$$

Therefore, we obviously have that  $\mathbf{x}^{k,0} = \mathbf{x}^k$  and  $\mathbf{x}^{k,q} = \mathbf{x}^{k+1}$ . We will also need to specify the objective function  $G$  (defined in (4)) with respect to each block  $\bar{\mathbf{x}}_i$ ,  $1 \leq i \leq q$ , separately. We define

$$F_i^k(\boldsymbol{\xi}) := F(\bar{\mathbf{x}}_1^{k+1}, \dots, \bar{\mathbf{x}}_{i-1}^{k+1}, \boldsymbol{\xi}, \bar{\mathbf{x}}_{i+1}^k, \dots, \bar{\mathbf{x}}_q^k, \mathbf{u}^k).$$

It is easy to see that  $F_i^k(\boldsymbol{\xi})$  is a quadratic function in  $\boldsymbol{\xi}$ , which is strongly convex (similar proof as in Proposition 2(ii)). Lastly, for convenience of the proofs, and to correspond with the scheme in [29], we denote from now on  $\mathbf{w}^{k+1} = \mathbf{u}^k$ , for all  $k \in \mathbb{N}$ .

We begin with the first lemma that show a sufficient decrease property of the sequence of function values.

**Lemma 4.** *Let  $\{(\mathbf{x}^k, \mathbf{u}^k)\}_{k \in \mathbb{N}}$  be a sequence generated by the Unifying AM Algorithm. Then, there exists  $\rho_1 > 0$  such that for all  $k \in \mathbb{N}$ , we have*

$$\rho_1 \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 \leq F(\mathbf{x}^k, \mathbf{w}^k) - F(\mathbf{x}^{k+1}, \mathbf{w}^{k+1}).$$

*Proof.* Fix  $k \in \mathbb{N}$ . Since each  $F_i^k(\boldsymbol{\xi})$ ,  $1 \leq i \leq q$ , is  $\sigma_i$ -strongly convex (for some  $\sigma_i > 0$ ), we obtain from its definition that

$$F_i^k(\bar{\mathbf{x}}_i^k) \geq F_i^k(\bar{\mathbf{x}}_i^{k+1}) + \left\langle \nabla F_i^k(\bar{\mathbf{x}}_i^{k+1}), \bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_i^{k+1} \right\rangle + \frac{\sigma_i}{2} \|\bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_i^{k+1}\|^2 = F_i^k(\bar{\mathbf{x}}_i^{k+1}) + \frac{\sigma_i}{2} \|\bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_i^{k+1}\|^2,$$

where the last equality follows from the fact that  $\nabla F_i^k(\bar{\mathbf{x}}_i^{k+1}) = \mathbf{0}$ , which follows from the optimality of  $\bar{\mathbf{x}}_i^{k+1}$  with respect to  $F_i^k$  according to the update rule of the Unifying AM Algorithm. In addition, using our compact notations, we obviously have that  $F_i^k(\bar{\mathbf{x}}_i^k) = F(\mathbf{x}^{k,i-1}, \mathbf{u}^k)$  and  $F_i^k(\bar{\mathbf{x}}_i^{k+1}) = F(\mathbf{x}^{k,i}, \mathbf{u}^k)$ . Therefore, for all  $1 \leq i \leq q$  we have

$$F(\mathbf{x}^{k,i-1}, \mathbf{u}^k) \geq F(\mathbf{x}^{k,i}, \mathbf{u}^k) + \frac{\sigma_i}{2} \|\bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_i^{k+1}\|^2.$$

Summing this inequality for all  $1 \leq i \leq q$ , yields

$$F(\mathbf{x}^k, \mathbf{u}^k) = F(\mathbf{x}^{k,0}, \mathbf{u}^k) \geq F(\mathbf{x}^{k,q}, \mathbf{u}^k) + \sum_{i=1}^q \frac{\sigma_i}{2} \|\bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_i^{k+1}\|^2 \geq F(\mathbf{x}^{k+1}, \mathbf{u}^k) + \frac{\sigma}{2} \|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2,$$

where  $\sigma = \min\{\sigma_1, \sigma_2, \dots, \sigma_q\}$ . In addition, from the updating rule of the  $\mathbf{u}$ -block we have that

$$F(\mathbf{x}^k, \mathbf{u}^{k-1}) \geq F(\mathbf{x}^k, \mathbf{u}^k).$$

Combining the last two inequalities and using the fact that  $\mathbf{w}^{k+1} = \mathbf{u}^k$ ,  $k \in \mathbb{N}$ , yields the desired result.  $\square$

An immediate consequence of the first lemma is that the Unifying AM Algorithm generates a bounded sequence.

**Lemma 5.** *Let  $\{(\mathbf{x}^k, \mathbf{u}^k)\}_{k \in \mathbb{N}}$  be a sequence generated by the Unifying AM Algorithm. Then, the sequence is bounded.*

*Proof.* From Lemma 4 that the sequence  $\{F(\mathbf{x}^k, \mathbf{w}^k)\}_{k \geq 1}$  decreases and therefore the sequence  $\{(\mathbf{x}^k, \mathbf{w}^k)\}_{k \geq 1}$  belongs to the level set of the function  $F$  at the level  $F(\mathbf{x}^1, \mathbf{w}^1)$ . Using Proposition 2(iii), we know that  $F$  is coercive and thus has bounded level sets [7, Proposition 11.12, p. 158], which completes the proof.  $\square$

The last lemma shows that at each iteration of the Unifying AM Algorithm, there exists a subgradient of the objective function  $F$  that is bounded by the norm of the difference between two corresponding iterates of the  $\mathbf{x}$  block.

**Lemma 6.** *Let  $\{(\mathbf{x}^k, \mathbf{u}^k)\}_{k \in \mathbb{N}}$  be a sequence generated by the Unifying AM Algorithm. Then, there exist a scalar  $\rho_2 > 0$  and a vector  $\mathbf{y}^{k+1} \in \partial F(\mathbf{x}^{k+1}, \mathbf{w}^{k+1})$  for all  $k \in \mathbb{N}$ , such that*

$$\|\mathbf{y}^{k+1}\| \leq \rho_2 \|\mathbf{x}^{k+1} - \mathbf{x}^k\|.$$

*Proof.* Let  $k \in \mathbb{N}$ . Since  $\mathbf{w}^{k+1} = \mathbf{u}^k$ , by the definition of  $F(\cdot, \cdot)$  (see (6)) we get

$$\partial F(\mathbf{x}^{k+1}, \mathbf{u}^k) = \nabla G(\mathbf{x}^{k+1}, \mathbf{u}^k) + \left( \mathbf{0}_{nN}, \partial \delta_{\mathcal{B}^M}(\mathbf{u}^k) \right).$$

Therefore, by using the optimality condition of the updating rule for the  $\mathbf{u}$ -block, we have that

$$\mathbf{0}_{nM} \in \nabla_{\mathbf{u}} G(\mathbf{x}^k, \mathbf{u}^k) + \partial \delta_{\mathcal{B}^M}(\mathbf{u}^k).$$

Setting,

$$\mathbf{y}^{k+1} := \nabla G(\mathbf{x}^{k+1}, \mathbf{u}^k) - \left( \mathbf{0}_{nN}, \nabla_{\mathbf{u}} G(\mathbf{x}^k, \mathbf{u}^k) \right),$$

we have that  $\mathbf{y}^{k+1} \in \partial F(\mathbf{x}^{k+1}, \mathbf{u}^k)$ . For simplicity, we define  $\mathbf{y}^{k+1} := (\mathbf{y}_{\mathbf{x}}^{k+1}, \mathbf{y}_{\mathbf{u}}^{k+1})$ . In addition, from the optimality condition of the updating rule of each  $\bar{\mathbf{x}}_i$ -block,  $1 \leq i \leq q$ , we have that  $\nabla_{\bar{\mathbf{x}}_i} G(\mathbf{x}^{k,i}, \mathbf{u}^k) = \nabla_{\bar{\mathbf{x}}_i} F(\mathbf{x}^{k,i}, \mathbf{u}^k) = \nabla F_i^k(\bar{\mathbf{x}}_i^{k+1}) = \mathbf{0}$ . Therefore,

$$\|\mathbf{y}_{\mathbf{x}}^{k+1}\| = \|\nabla_{\mathbf{x}} G(\mathbf{x}^{k+1}, \mathbf{u}^k)\| \leq \sum_{i=1}^q \|\nabla_{\bar{\mathbf{x}}_i} G(\mathbf{x}^{k+1}, \mathbf{u}^k)\| = \sum_{i=1}^q \|\nabla_{\bar{\mathbf{x}}_i} G(\mathbf{x}^{k+1}, \mathbf{u}^k) - \nabla_{\bar{\mathbf{x}}_i} G(\mathbf{x}^{k,i}, \mathbf{u}^k)\|.$$

Since  $G(\cdot, \mathbf{u}^k)$  (see (5)) is a quadratic function, there exist positive constants  $\alpha_i$ ,  $1 \leq i \leq q$ , such that

$$\|\nabla_{\bar{\mathbf{x}}_i} G(\mathbf{x}^{k+1}, \mathbf{u}^k) - \nabla_{\bar{\mathbf{x}}_i} G(\mathbf{x}^{k,i}, \mathbf{u}^k)\| \leq \alpha_i \|\mathbf{x}^{k+1} - \mathbf{x}^{k,i}\| \leq \alpha_i \|\mathbf{x}^{k+1} - \mathbf{x}^k\|,$$

where the last inequality follows from the definition of  $\mathbf{x}^{k,i}$ . Similarly, since  $\mathbf{u} \rightarrow G(\mathbf{x}, \cdot)$  is linear (see (5)), there exists a positive parameter  $\beta > 0$  for which

$$\|\mathbf{y}_{\mathbf{u}}^{k+1}\| = \|\nabla_{\mathbf{u}} G(\mathbf{x}^{k+1}, \mathbf{u}^k) - \nabla_{\mathbf{u}} G(\mathbf{x}^k, \mathbf{u}^k)\| \leq \beta \|\mathbf{x}^{k+1} - \mathbf{x}^k\|,$$

Combining these bounds and setting  $\rho_2 = \sum_{i=1}^q \alpha_i + \beta > 0$ , results in the following

$$\|\mathbf{y}^{k+1}\| \leq \|\mathbf{y}_{\bar{\mathbf{x}}}^{k+1}\| + \|\mathbf{y}_{\mathbf{u}}^{k+1}\| \leq \rho_2 \|\mathbf{x}^{k+1} - \mathbf{x}^k\|,$$

which completes the proof.  $\square$

Now, we are ready to provide the proof of our main result.

*Proof of Theorem 3.* The proof is based on [29, Theorem 4.3, p. 662], which involves two requirements:

Table 1: Computation and communication cost per iteration

Method	Sensor $i$						Total Computational Cost	
	Computational Operations	Storage Size	In Msg. #	Out Msg. size	Out Msg. #	Out Msg. size	Sequential	Parallelized
AM-FC	-	-	-	-	-	-	$O(nN^3 + nM)$	-
AM-FD	$O(nM_i)$	$n$	$M_i$	$n$	1	$n$	$O(nM)$	-
AM-CC	$O(nM_i)$	$n$	$M_i$	$n$	1	$n$	$O(nM)$	$O\left(\sum_{j=1}^q \max_{i \in \mathcal{C}_j} nM_i\right)$
SF	$O(nM_i)$	$n$	$M_i$	$n$	1	$n$	$O(nM)$	$O(\max_{i \in \mathcal{V}} nM_i)$
AG	$O(nM_i)$	$n$	$M_i$	$n$	1	$n$	$O(nM)$	$O(\max_{i \in \mathcal{V}} nM_i)$
ADMM-H	$O(nM_i + n^3T_i)^a$	$nM_i$	$M_i$	$2n$	$M_i$	$2n$	$O(nM + n^3T)^b$	$O(\max_{i \in \mathcal{V} \cup \mathcal{A}} nM_i + \max_{i \in \mathcal{V}} n^3T_i)$

<sup>a</sup> Notation  $T_i$  refers to the number of iteration required by the non-convex Newton algorithm to converge.

<sup>b</sup>  $T = \sum_{i \in \mathcal{V}} T_i$ .

(i) showing that  $\{(\mathbf{x}^k, \mathbf{w}^k)\}_{k \in \mathbb{N}}$  is a bounded gradient-like descent sequence for minimizing  $F$ , and (ii) proving that  $F$  is a semi-algebraic function. Since the function  $G$  is a quadratic polynomial function (see (4)) and the ball  $\mathcal{B}$  is a semi-algebraic set, it follows immediately that  $F$  is semi-algebraic. In order to prove the first requirement and in view of Lemmas 4, 5 and 6, it is left to show that (cf. [29, Definition 4.2, p. 661])

$$\limsup_{n_k \rightarrow \infty} F(\mathbf{x}^{n_k}, \mathbf{w}^{n_k}) \leq F(\mathbf{x}^*, \mathbf{w}^*),$$

where  $(\mathbf{x}^*, \mathbf{w}^*)$  is a limit point of the subsequence  $\{(\mathbf{x}^{n_k}, \mathbf{w}^{n_k})\}_{k \in \mathbb{N}}$ . Indeed, using the fact that  $F(\mathbf{x}^k, \mathbf{w}^k) = G(\mathbf{x}^k, \mathbf{w}^k)$ , for all  $k \in \mathbb{N}$ , and by the continuity of  $G$  we obtain that

$$\limsup_{n_k \rightarrow \infty} F(\mathbf{x}^{n_k}, \mathbf{w}^{n_k}) = \limsup_{n_k \rightarrow \infty} G(\mathbf{x}^{n_k}, \mathbf{w}^{n_k}) = \lim_{n_k \rightarrow \infty} G(\mathbf{x}^{n_k}, \mathbf{w}^{n_k}) = G(\mathbf{x}^*, \mathbf{w}^*) \leq F(\mathbf{x}^*, \mathbf{w}^*),$$

where the last inequality follows since we always have that  $G(\mathbf{x}, \mathbf{u}) \leq F(\mathbf{x}, \mathbf{u})$  for all  $\mathbf{x} \in \mathbb{R}^{nN}$  and  $\mathbf{u} \in \mathbb{R}^{nM}$ . This completes the requirements and therefore we obtain from [29, Theorem 4.3, p. 662] that  $\{\mathbf{x}^k\}_{k \in \mathbb{N}}$  converges to some  $\mathbf{x}^*$ , and, for any limit point  $\mathbf{w}^*$  of  $\{\mathbf{w}^k\}_{k \in \mathbb{N}}$ ,  $(\mathbf{x}^*, \mathbf{w}^*)$  is a critical point of  $F$ .

Since  $\mathbf{w}^{k+1} = \mathbf{u}^k$ ,  $k \in \mathbb{N}$ , we obtain that the same statement is true for any limit point  $\mathbf{u}^*$  of  $\{\mathbf{u}^k\}_{k \in \mathbb{N}}$ , which proves that  $(\mathbf{x}^*, \mathbf{u}^*)$  is a critical point of  $F$ . The result now follows from Proposition 1.  $\square$

## 6 Numerical Experiments

In order to provide a complete insight on the relation of our range of algorithms with the methods already available in the literature, the following algorithms are compared:

1. AM-U- $q$ , our Unifying AM Algorithm with  $q$  clusters presented above in Section 4. This is a semi-distributed version, where we randomly generated  $q$  geographical clusters. The geographical clusters were created similarly to well-known LEACH algorithm described in [18], where  $q$  sensors

are randomly chosen as cluster-heads and sensors are associated to the sensor with the closest cluster-head. However, in order to use only available information, the distance between the sensors and the cluster-heads are measured by the minimal number of edges between them, and in the case of a draw the sensor is associated with the cluster-head with minimal measured distance.

2. AM-FC, our Fully Centralized Alternating Minimization presented above in Section 3, which is equivalent to the AM-U- $q$  algorithm with  $q = 1$ .
3. AM-FD, our Fully Distributed Alternating Minimization presented above in Section 3, which is equivalent to the AM-U- $q$  algorithm with  $q = N$ .
4. AM-CC, the Colored Clustered Alternating Minimization, in which clusters are built from unconnected sensors using a graph coloring algorithm taken from [6, Section 3.1], such that the updates of each node in the clusters can be parallelized, and the algorithm is fully distributed.
5. SF, namely the Nesterov’s Accelerated Projected Gradient method of [32] which solves a convex relaxation version of Problem (1) implemented in a distributed fashion.
6. EML, an SDP relaxation of Problem (1) presented in [31].
7. ADMM-H, hybrid ADMM algorithm suggested by [27] which is a distributed method with an active transition from the convex relaxation suggested by [32] to the non-convex model of Problem (1). The ADMM-H requires several parameters, including a regularization parameter, and parameters that control the shift from the convex to the non-convex model. The parameters used in our experiments are given in Table 2.

We define an iteration of each of these methods, as the period in which all sensor locations are updated. Specifically, for the fully distributed versions, each iteration consists of each sensor receiving updates messages from all its neighbors, updating its location estimation, and sending update messages to all its neighbors.

We begin by discussing the computational complexity and communication requirements of each method. Table 1 summarizes the computational cost, storage requirements, and communication cost (ingoing and outgoing messages) per iteration for each sensor  $i$ , as well as the computational cost per iteration of the parallelized and sequential implementation of each algorithm. In the methods compared we also include Nesterov’s Accelerated Gradient (AG) Method [23] which is used for initialization of some methods (see details below). For methods which are not parallelizable, such as AM-FC and AM-FD, we

only give the sequential computational cost. For the centralized method AM-FC, the total sequential computation cost is derived by noting that step (11) requires calculating the inverse of an  $N \times N$  matrix in  $\mathbb{R}^n$  during initialization, while in step (12) we calculate the Euclidean norm of  $M$  vectors in  $\mathbb{R}^n$ . As for the distributed methods AM-FD, AM-CC, SF and AG, each iteration of the algorithm calculates  $n$  inner products of two vectors of size  $M_i$  (in addition to step (12)). The total parallelized computational cost of AM-CC is determined by the cluster with greatest computation cost, since the method is parallelized within each cluster and sequential among the clusters. However, since SF and AG are fully parallelizable among the sensors, their total parallelized computation cost is determined by the greatest computation cost among the sensors. We note that in each iteration of ADMM-H, each non-anchor  $i$  runs a non-convex Newton's method which may not converge, and the number of iterations it uses, which is denoted in the table by  $T_i$ , is unknown. Furthermore, when using the ADMM-H method, each sensor maintains a vector with a size which is proportional to the number of its neighbors, and at each iteration it sends *different* messages to each neighbor, each consisting of a vector in  $\mathbb{R}^n$ . In contrast, all other distributional methods maintain only one vector in  $\mathbb{R}^n$ , and at each iteration each sensor broadcasts its own estimated location (a vector in  $\mathbb{R}^n$ ) to all its neighbors, resulting in lower energy consumption and communication time.

We now investigate the empirical performance of our proposed method for several networks. We use two benchmark networks available in the Stanford's Computational Optimization Laboratory web site [35]. The first network consists of  $K = 500$  sensors with  $m = 10$  anchor sensors, and the second network consists of  $K = 1000$  sensors with  $m = 20$  anchors. In addition, for the latter set of parameters we generated a random network for  $K = 1000$  in which all sensor locations (anchor and non-anchor) are randomly generated in the two dimensional box  $[-0.5, 0.5]^2$ . Similarly, we generated four additional random networks with  $K = 2000$ ,  $K = 3000$ ,  $K = 5000$  and  $K = 10000$ , and for all four networks 2% of the sensors are anchors. We also used the random  $K = 1000$  network to create six more networks, which differ in the amount of anchors and in their average node degree. For all networks, the measurement noise for each of the distance measurements between the sensors is a Gaussian random variable with zero mean and a standard deviation of  $\sigma$ . Similarly to the two benchmark networks, we took  $\sigma$  to be 7% of the radio communication radius  $r$ .

For each network, a random initial point  $\mathbf{x}^0$  from a uniform distribution  $\text{Unif}(-0.01, 0.01)^{nN}$  is taken, and is used for all the compared methods. For the AM based-methods we always initialized  $\mathbf{u}^0 = \mathbf{0}_{nM}$ . Note that the ADMM-H uses a two-stage approach in the sense that it first solves a relaxation of Problem (1), as explained in Section 1, to obtain a more favorable starting point for solving the non-convex formulation, by using a "smooth transition" from one stage to the other (for exact details see

Table 2: Network and Method Parameters

K	Network Parameters				Method Parameters			
	m	r	$\sigma$	Average $M_i$	$\epsilon_c$	$\zeta_c$	$\tau_c$	AG $d_{\max}$
Benchmark								
500	10	0.3	0.02	14.15	0.04	0.2	0.015	70
1000	20	0.1	0.007	11.01	0.003	0.05	0.002	50
Random								
Changing $K$								
1000	20	0.061	0.00427	11.09	0.003	0.05	0.002	25
2000	40	0.043	0.00301	11.22				23
3000	60	0.035	0.00245	11.07				22
5000	100	0.029	0.00203	12.79				27
10000	200	0.025	0.00172	18.43				37
Changing $m$								
985	5	0.061	0.00427	10.90	0.003	0.05	0.002	25
990	10	0.061	0.00427	10.97				25
1010	30	0.061	0.00427	11.19				25
Changing $M_i$								
1000	20	0.049	0.00340	7.21	0.003	0.05	0.002	18
1000	20	0.057	0.00398	9.71				24
1000	20	0.067	0.00466	13.03				27

[27]). Such a two-stage approach enables reaching the vicinity of better solutions. We also utilized such an approach when applying our AM-U- $q$ , AM-FD, and AM-CC algorithms. Specifically, before starting to run the algorithm we first run a fixed number of iterations (in our experiments we did 100) of the Accelerated Gradient (AG) method [23] on the following optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^{nN}} F(\mathbf{x}, \mathbf{u}^0),$$

where  $\mathbf{u}^0$  is given. The AG method is applicable due to the fact that, given  $\mathbf{u}^0$ , the objective function is a strongly convex quadratic function of  $\mathbf{x}$ , which has a Lipschitz continuous gradient with a parameter that can be bounded by  $L = 2(2d_{\max} + m)$ , where  $d_{\max}$  is the maximal degree of any non-anchor sensor in the network with respect to other non-anchors (see Table 2). We add the suffix “-AG100” to the methods’ names to denote this initialization. Note that running the AG method on this problem can be done in a fully distributed and fully parallelized fashion, with the same communication cost per iteration as AM-FD and AM-CC (for more details see Table 1 and Section 6.3). All methods were ran for  $10^3$  iterations, in order to have the same number of communication rounds. The parameters of the networks, as well as the specific parameters used for each method, are summarized in Table 2.

<sup>6</sup>We note that in the benchmark instances, the communication radius is as reported in [35]. However, in these instances, not all sensor pairs with distance lower than the communication radius generate edges. In fact, for all sensors the number of

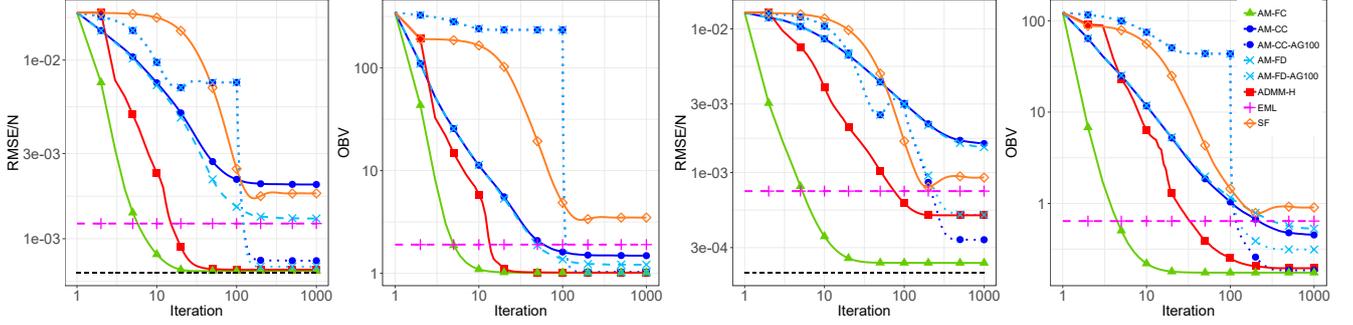


Figure 1: Comparison with convex relaxation on the benchmark networks with  $K = 500$  (two left graphs) and  $K = 1000$  (two right graphs). The root of the CRLB divided by  $N$  is given by the dashed black line.

We now discuss the criteria for comparing the methods. For each instance we generated  $R = 50$  random realizations of the measured distances, and run each method on each of these inputs. We then compared the methods using several criteria - the average objective value (OBV) of Problem (1), the root average mean squared error (RMSE), and the estimated bias. The OBV will serve as an indication of how good are the methods in solving Problem (1), whereas the RMSE, given by

$$\text{RMSE} = \sqrt{\frac{R}{\sum_{l=1}^R \sum_{i \in \mathcal{V}} \|\mathbf{x}_i^l - \mathbf{x}_i^{\text{real}}\|^2}},$$

where  $\mathbf{x}_i^{\text{real}}$  is the real location of sensor  $i$ , will provide the true average approximation error of the tested method, and  $\mathbf{x}_i^l$  is the estimated location for realization  $l \in \{1, 2, \dots, R\}$ . Although the tested method may be biased we are going to compare the RMSE with the root of the Cramer-Rao lower bound (CRLB) (see [26] and [25]). For completeness, we will also provide an estimation of the bias of each method, given by:

$$\widehat{\text{bias}} = \sum_{l=1}^R \frac{\mathbf{x}_i^l - \mathbf{x}_i^{\text{real}}}{R}.$$

We also compare the methods' running times, where the running times of parallelizable steps in each iteration were computed as the maximal computation time over all parallelized components.

All experiments were ran on an Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz with a total of 300GB RAM and 72 threads, using MATLAB 2019a, where each method was allowed to run on only one thread and up to 16GB of RAM.

---

neighbors was truncated, and anchor sensors have neighbors which are further away than the communication radius. The values of  $\sigma$  for these instances are taken from [27].

## 6.1 Solving the Original Formulation vs. Relaxation

We begin by presenting a comparison between the centralized method AM-FC, and the distributed methods AM-FD and AM-CC, for solving Problem (1), with the SF method of [32] and EML of [31], which both solve different relaxations of Problem (1). We note that the results for the EML method are reported as the final value obtained by the centralized algorithm, and therefore the iterations count has no significance. Indeed, while the EML has a distributed version, which is also described in [31], when attempting to run this distributed version on the benchmark networks, each iteration took approximately a second for each sensor, making this method not practical for sequential experiments. The results of running  $10^3$  iterations of these methods, starting from a common random point  $\mathbf{x}^0 \in U(-0.01, 0.01)^{2N}$  on the benchmark networks with  $K = 500$  and  $K = 1000$ , are presented in Figure 1. The summary of the results after  $10^3$  iterations is also available in Table 3. For  $K = 500$  and  $K = 1000$  the AM-CC used 127 clusters and 32 clusters, respectively, where the clusters for  $K = 1000$  are depicted in Figure 2.

We note that the AM-FC has lower RMSE than both SF and EML, illustrating that the non-relaxed problem leads to a better location estimation than the global optimal solution of the relaxed problem. In contrast, the distributed methods without initialization AM-FD and AM-CC have worse location estimation in terms of RMSE than the EML and sometimes SF method (although both the EML and SF methods have worse average function value). However, when initialized by the AG method (where the AG method is used in the first 100 out of the 1000 iterations) both AM-FD-100AG and AM-CC-100AG have superior RMSE and function values to EML and SF, demonstrating the benefit of using the AG method as an initialization procedure before solving the non-convex model. Moreover, the benefit of using

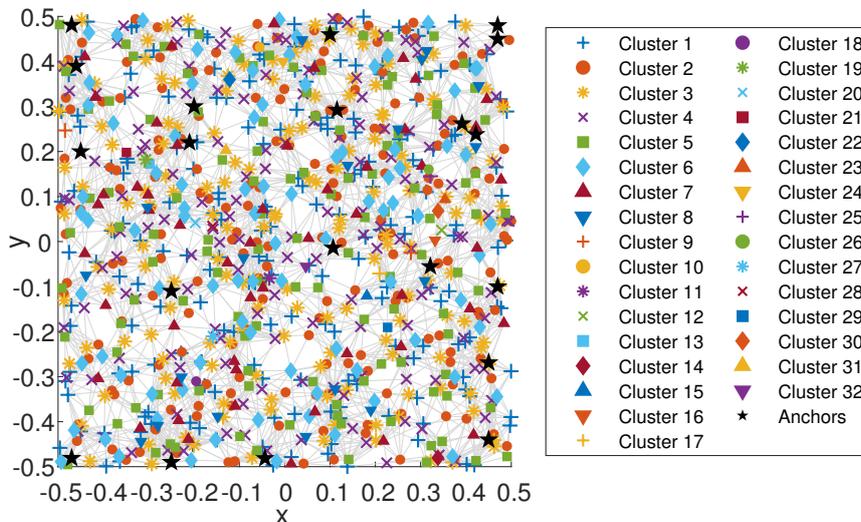


Figure 2: The colored clusters used by the AM-CC algorithm for the benchmark instance with  $K = 1000$ . There are 32 clusters, each with non-neighboring sensors, with sizes ranging from 1 to 200.

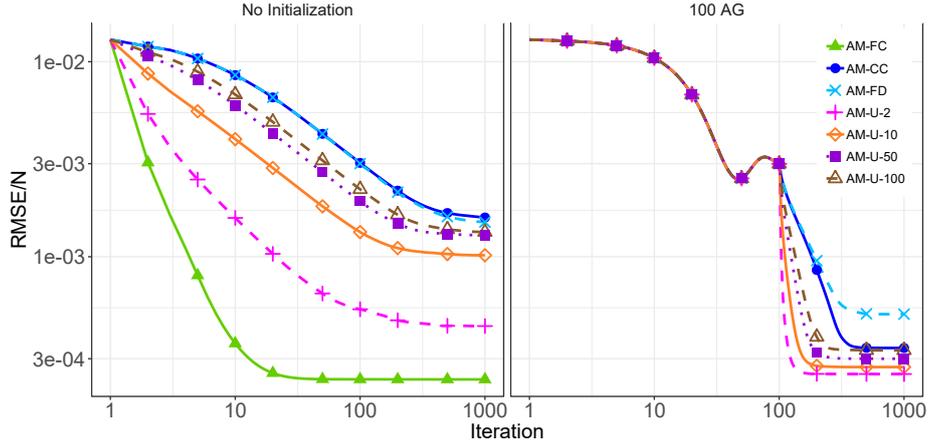


Figure 3: RMSE Comparison for AM-U- $q$  with various number of clusters on benchmark example with  $K = 1000$ . The left graph shows the comparison without AG initialization and the right graph shows the comparisons with AG initialization.

parallelization through clustering, as done in AM-CC and AM-CC-100AG, is evident in the short running times of these methods, which require less than a second to run. This is in contrast to the distributed but non-parallelizable AM-FD, which takes almost as much time as the centralized AM-FC, despite the problem’s size. We also see that for all methods that solve the non-convex Problem (1), there is a high correlation between OBV and RMSE, whereas both EML and SF have a higher OBV than AM-CC and AM-FD while having a lower value of RMSE, suggesting that Problem (1) is a good predictor of the RMSE performances for low enough objective values.

## 6.2 The Effect of Clustering

We next explore the effect of geographical clustering on the performance of our method, with and without AG initialization. We note that AG initialization is not shown for the AM-FC algorithm since no improvement was achieved by adding this initialization.

The RMSE results for  $q = 2, 10, 50, 100$  clusters are presented in Figure 3, and in order to illustrate the output of the geographical clustering, the clusters produced for  $q = 2$  and  $q = 10$  are presented in Figure 4.

As one might expect, when the number of clusters decreases the algorithm becomes more centralized and produces lower RMSE values. Moreover, initializing the methods via the AG method improved the results significantly, reducing the RMSE of AM-U-2 and AM-U-10 close to that of the AM-FC. Thus, we can conclude, that if we can not run the fully centralized version AM-FC due to computational or storage restriction, there is still a benefit of using the AM-U method with some level of clustering, since together with an AG initialization, this could lead to a lower RMSE value. We note that, as shown in the

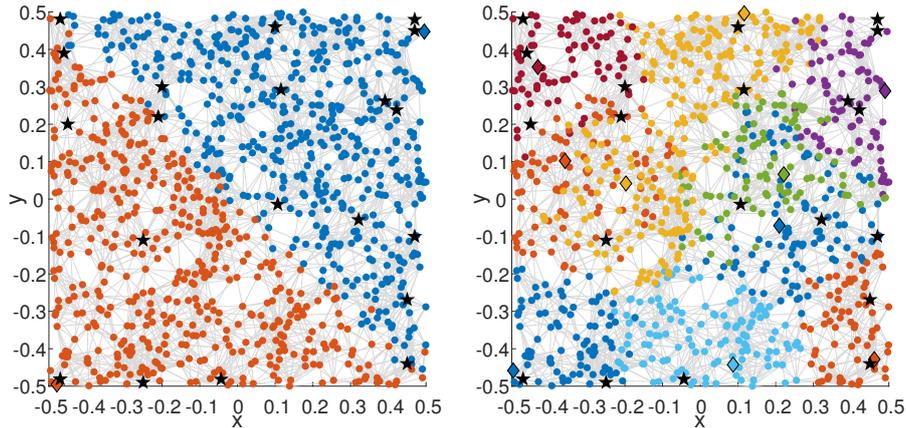


Figure 4: Results of geographical clustering for the benchmark example with  $K = 1000$ , with  $q = 2$  clusters (right) and  $q = 10$  clusters (left). The sensors which belong to each cluster are represented as dots with the same color and its cluster head is denoted by a diamond with the same color, whereas anchors are denoted by stars.

previous section, we observed a high correspondence between the OBV and RMSE values for different clustering techniques, where a lower OBV implied a lower RMSE value.

### 6.3 Comparison to ADMM-H

We now compare the performance of the AM-U method to this of the ADMM-H method of [27]. As we saw in Figure 1 and Table 3, ADMM-H has superior performance to that of AM-CC-100AG for the

Table 3: Method comparison on benchmark networks

Method	RMSE	$\ \widehat{\text{bias}}\ $	OBV	Avg. run time (seconds)	
	Avg. (stdv)		Avg. (stdv)	Parallelized	Sequential
Benchmark $K = 500$					
AM-FC	<b>3.24e-01</b> (1.28e-02)	0.049	1.02e+00 (2.75e-02)	-	4.28
EML	5.96e-01 (2.17e-02)	0.492	1.90e+00 (7.05e-02)	-	25.28
AM-CC	9.86e-01 (5.88e-02)	0.859	1.49e+00 (1.05e-01)	0.90	4.27
AM-CC-AG100	3.69e-01 (3.54e-02)	0.132	1.03e+00 (2.91e-02)	0.87	3.97
AM-FD	6.36e-01 (1.51e-01)	0.330	1.21e+00 (1.51e-01)	-	3.70
AM-FD-AG100	3.42e-01 (2.60e-02)	0.071	1.02e+00 (2.81e-02)	-	3.43
SF	8.80e-01 (2.39e-02)	0.798	3.49e+00 (1.16e-01)	0.24	63.09
ADMM-H	<b>3.29e-01</b> (1.36e-02)	0.492	<b>1.01e+00</b> (2.75e-02)	1.40	452.6
Benchmark $K = 1000$					
AM-FC	<b>2.30e-01</b> (1.26e-02)	0.105	<b>1.75e-01</b> (4.45e-03)	-	8.99
EML	7.28e-01 (1.89e-02)	0.697	6.41e-01 (2.68e-02)	-	59.43
AM-CC	1.56e+00 (1.78e-01)	1.429	4.52e-01 (3.96e-02)	0.30	7.31
AM-CC-AG100	<b>3.34e-01</b> (3.05e-02)	0.218	<b>1.86e-01</b> (6.96e-03)	0.24	6.31
AM-FD	1.48e+00 (2.18e-01)	1.259	5.29e-01 (5.85e-02)	-	8.19
AM-FD-AG100	4.98e-01 (5.95e-02)	0.415	3.13e-01 (1.38e-02)	-	7.11
SF	9.05e-01 (2.12e-02)	0.875	9.04e-01 (3.07e-02)	0.23	107.07
ADMM-H	4.94e-01 (1.80e-01)	0.308	1.95e-01 (1.71e-02)	1.38	875.95

benchmark with  $K = 500$  but inferior for the benchmark with  $K = 1000$ . Moreover, the parallelized time of ADMM-H is five times larger than that of AM-CC-100AG.

We now investigate the performance of our leading centralized and distributed methods, AM-FC and AM-CC-100AG, vs. that of ADMM-H for various settings, which are specified in Table 2.

We start with investigating the performance for random networks of sizes  $K = 1000, 2000, 3000, 5000, 10000$ , where the anchor number is 2% of network size,  $r$  is chosen to be the minimal radius which ensures that there exists a CRLB (Fisher Information matrix is invertible) and resulting in an average  $M_i$  closest to 11, and  $\sigma$  is chosen to be 7% of  $r$ . Note that high sequential running times of ADMM-H did not allow to run it on the larger networks. We present the results of this sensitivity analysis in Figure 5 and Table 4. We observe that the AM-CC-100AG achieves lower RMSE values than ADMM-H after about 100 iterations, and continues to improve the RMSE values with more iterations while the ADMM-H already converged. Moreover, AM-FC achieves the lowest RMSE followed by AM-CC-100AG, except for the case for  $K = 10000$ . Indeed, for the case of  $K = 10000$ , AM-CC-100AG achieves the lowest RMSE although its objective value is higher than that of AM-FC, implying that in this case the objective function is not necessarily a good predictor of the RMSE value, which might be due to a high variance in the connectivity of the sensors due to the network's size.

Next we investigate the impact of the number of anchors  $m$ , the average degree  $M_i$  of the non-anchor nodes in the network (or equivalently the radius  $r$ ), and the initialization of  $\mathbf{x}^0$  on the results. For all these experiments we take as a base-line the network with  $K = 1000$ , and change one parameter at each time. The results are given in Figure 6 and Table 4.

When changing the number of anchors, for  $m = 5$  the AM-FC obtains the worst RMSE although it obtains the best function value. We deduce that in this case, Problem (1) is not a good predictor of the estimation ability of a method. However, as  $m$  increases, the performance of all methods improve, where AM-FC again provides the lowest RMSE eventually equal to the CRLB value, followed by AM-CC-100AG. When changing the degree of the network, we choose a minimal degree by choosing the minimal  $R$  which ensures that the graph is connected and a maximal  $R$  which results in an average  $M_i$  of 13. All methods improve as the degree gets larger. However, as expected, the influence is more significant for the distributed methods. Regardless, AM-CC-100AG is superior to ADMM-H for all tested degree values. Finally, we see that different initialization of the starting point has very limited impact on AM-FC and AM-CC after 1000 iterations, while the ADMM-H algorithm improves as the variance of the initialization increases, making it less robust to different initialization procedures.

Table 4: Method comparison on random networks

Method	RMSE	$\ \widehat{\text{bias}}\ $	OBV	Avg. run time (seconds)	
	Avg. (stdv)		Avg. (stdv)	Parallelized	Sequential
Random $K = 1000$ , $m = 20$ , $Deg = 11.09$ , $\sqrt{CRLB} = 8.03e-01$					
AM-FC	<b>8.21e-01</b> (5.54e-02)	0.761	<b>7.09e-02</b> (2.34e-03)	-	9.28
AM-CC	4.70e+00 (1.04e-01)	4.652	2.26e-01 (1.42e-02)	0.19	6.97
AM-CC-AG100	<b>2.95e+00</b> (8.21e-02)	2.890	<b>1.25e-01</b> (4.74e-03)	0.17	6.26
ADMM	4.99e+00 (4.15e-02)	4.964	1.53e-01 (5.58e-03)	1.52	876.18
Random $K = 2000$					
AM-FC	<b>7.58e-01</b> (4.72e-02)	0.703	<b>7.53e-02</b> (2.67e-03)	-	29.77
AM-CC	3.31e+00 (5.88e-02)	3.243	2.52e-01 (1.09e-02)	0.26	19.15
AM-CC-AG100	<b>1.45e+00</b> (6.28e-02)	1.379	<b>1.07e-01</b> (3.58e-03)	0.23	17.09
ADMM	3.42e+00 (2.32e-02)	3.387	1.70e-01 (5.72e-03)	1.67	1959.50
Random $K = 3000$					
AM-FC	<b>5.13e-01</b> (2.81e-02)	0.461	<b>7.00e-02</b> (1.46e-03)	-	59.95
AM-CC	3.16e+00 (4.13e-02)	3.120	2.31e-01 (1.10e-02)	0.30	32.25
AM-CC-AG100	<b>1.29e+00</b> (4.36e-02)	1.221	<b>9.16e-02</b> (4.36e-03)	0.27	29.09
ADMM	3.26e+00 (2.25e-02)	3.239	1.45e-01 (3.41e-03)	1.94	2909.90
Random $K = 5000$					
AM-FC	<b>2.39e-01</b> (1.59e-02)	0.169	<b>9.23e-02</b> (9.26e-04)	-	156.30
AM-CC	3.28e+00 (4.58e-02)	3.230	3.13e-01 (1.08e-02)	0.44	69.46
AM-CC-AG100	<b>1.30e+00</b> (4.12e-02)	1.271	<b>1.13e-01</b> (2.31e-03)	0.40	62.69
Random $K = 10000$					
AM-FC	1.02e+00 (1.27e-02)	1.009	<b>2.34e-01</b> (1.64e-03)	-	819.25
AM-CC	2.60e+00 (3.61e-02)	2.556	5.17e-01 (1.85e-02)	0.86	247.93
AM-CC-AG100	<b>6.72e-01</b> (4.20e-02)	0.624	<b>2.53e-01</b> (3.76e-03)	0.78	223.1
Random $K = 985$ , $m = 5$ , $Deg = 10.9$ , $\sqrt{CRLB} = 8.19e-01$					
AM-FC	1.06e+01 (3.70e-01)	10.583	<b>1.02e-01</b> (3.93e-03)	-	8.88
AM-CC	9.93e+00 (1.09e-01)	9.884	2.49e-01 (8.93e-03)	0.19	6.81
AM-CC-AG100	<b>9.08e+00</b> (1.10e-01)	9.040	<b>1.78e-01</b> (8.12e-03)	0.17	6.13
ADMM-H	1.01e+01 (3.46e-02)	10.077	2.13e-01 (5.86e-03)	1.61	939.04
Random $K = 990$ , $m = 10$ , $Deg = 10.97$ , $\sqrt{CRLB} = 8.07e-01$					
AM-FC	<b>2.04e+00</b> (1.26e-01)	1.992	<b>7.27e-02</b> (2.52e-03)	-	9.38
AM-CC	6.68e+00 (1.21e-01)	6.633	2.49e-01 (1.40e-02)	0.21	7.19
AM-CC-AG100	<b>4.92e+00</b> (1.07e-01)	4.857	<b>1.66e-01</b> (6.24e-03)	0.19	6.41
ADMM	7.33e+00 (3.81e-02)	7.310	1.77e-01 (7.17e-03)	1.70	964.16
Random $K = 1010$ , $m = 30$ , $Deg = 11.19$ , $\sqrt{CRLB} = 7.99e-01$					
AM-FC	<b>8.08e-01</b> (8.38e-02)	0.743	<b>7.60e-02</b> (2.84e-03)	-	9.67
AM-CC	3.49e+00 (6.62e-02)	3.425	2.15e-01 (2.06e-02)	0.22	7.25
AM-CC-AG100	<b>1.57e+00</b> (8.89e-02)	1.47	<b>1.04e-01</b> (7.13e-03)	0.19	6.49
ADMM	3.42e+00 (4.02e-02)	3.396	1.19e-01 (7.41e-03)	1.69	958.21
Random $K = 1000$ , $m = 20$ , $Deg = 7.2$					
AM-FC	<b>2.03e+00</b> (1.37e-01)	1.918	<b>2.39e-02</b> (9.87e-04)	-	5.90
AM-CC	5.44e+00 (5.29e-02)	5.405	5.84e-02 (2.70e-03)	0.11	4.78
AM-CC-AG100	<b>4.83e+00</b> (4.57e-02)	4.792	<b>4.25e-02</b> (2.46e-03)	0.10	4.30
ADMM	6.00e+00 (2.78e-02)	5.984	4.68e-02 (1.82e-03)	1.28	811.11
Random $K = 1000$ , $m = 20$ , $Deg = 9.71$ , $\sqrt{CRLB} = 8.07e - 01$					
AM-FC	<b>1.00e+00</b> (7.22e-02)	0.935	<b>5.02e-02</b> (1.75e-03)	-	8.57
AM-CC	4.86e+00 (7.74e-02)	4.810	1.61e-01 (8.82e-03)	0.21	6.77
AM-CC-AG100	<b>3.56e+00</b> (8.04e-02)	3.513	<b>9.36e-02</b> (4.33e-03)	0.18	6.07
ADMM	5.31e+00 (3.20e-02)	5.286	1.05e-01 (4.03e-03)	1.74	961.34
Random $K = 1000$ , $m = 20$ , $Deg = 13.03$ , $\sqrt{CRLB} = 1.81e - 01$					
AM-FC	<b>6.99e-01</b> (8.09e-02)	0.637	<b>1.08e-01</b> (3.90e-03)	-	10.83
AM-CC	4.65e+00 (1.06e-01)	4.601	3.66e-01 (1.82e-02)	0.23	8.13
AM-CC-AG100	<b>2.69e+00</b> (1.83e-01)	2.608	<b>1.90e-01</b> (1.20e-02)	0.21	7.34
ADMM-H	4.71e+00 (5.06e-02)	4.67826	2.33e-01 (9.35e-03)	2.42	1027.10

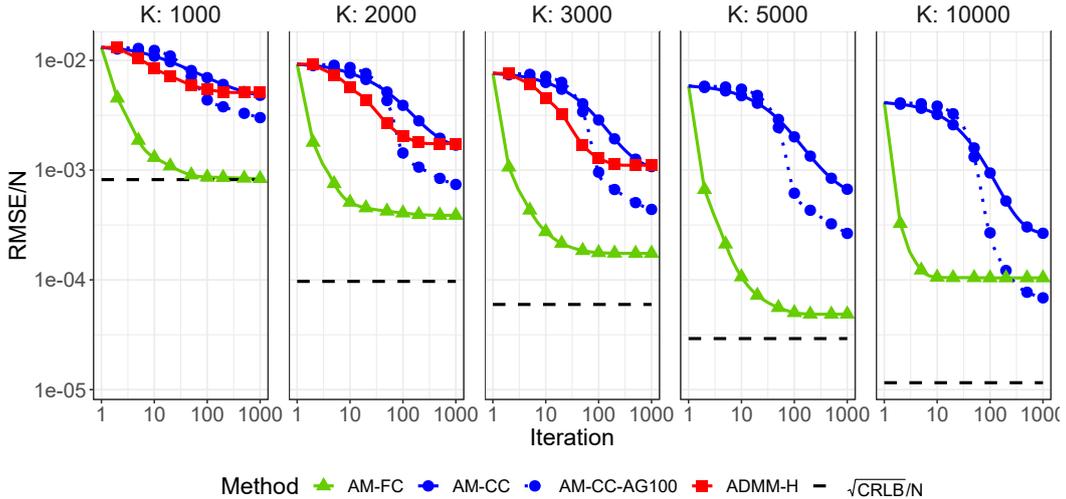


Figure 5: RMSE of AM centralized and distributed methods vs. ADMM-H for various network sizes.

In general, when the objective function of Problem (1) is a good predictor of the performance, the centralized method AM-FC obtains the lowest RMSE and converges after the fewest iterations. Moreover, the distributed method AM-CC-100AG generally has lower RMSE values than ADMM-H after 100 iterations, although it takes longer to converge. We attribute this to the fact that ADMM-H has many parameters which need to be tuned well for each setting, and are therefore less robust to slight changes, including the choice of starting point. This is a further advantage of the AM-U method, which is parameter-free, and the AG initialization that relies only on parameters given by the network structure and can be easily approximated. Furthermore, the AM-CC method is at least five times faster than ADMM-H.

## 7 Summary

In this paper we suggest a first-order framework based on the Alternating Minimization technique to solve the non-convex and non-smooth formulation of the WSN localization problem. This framework, provides a range of implementation, from fully distributable to fully centralized, while also allowing for partial parallelization. We prove that the algorithms generated by this general framework globally converge to critical points of the non-convex and non-smooth problem. We show in our numerical experiments that the fully centralized version is both scalable (up to 10000 sensors), and provides near optimal solution for various instances, while the distributed versions with proper initialization are still superior to existing methods. We also would like to emphasize that as far as we know, there is no other centralized method that efficiently solves the WSNL problem for such large networks.

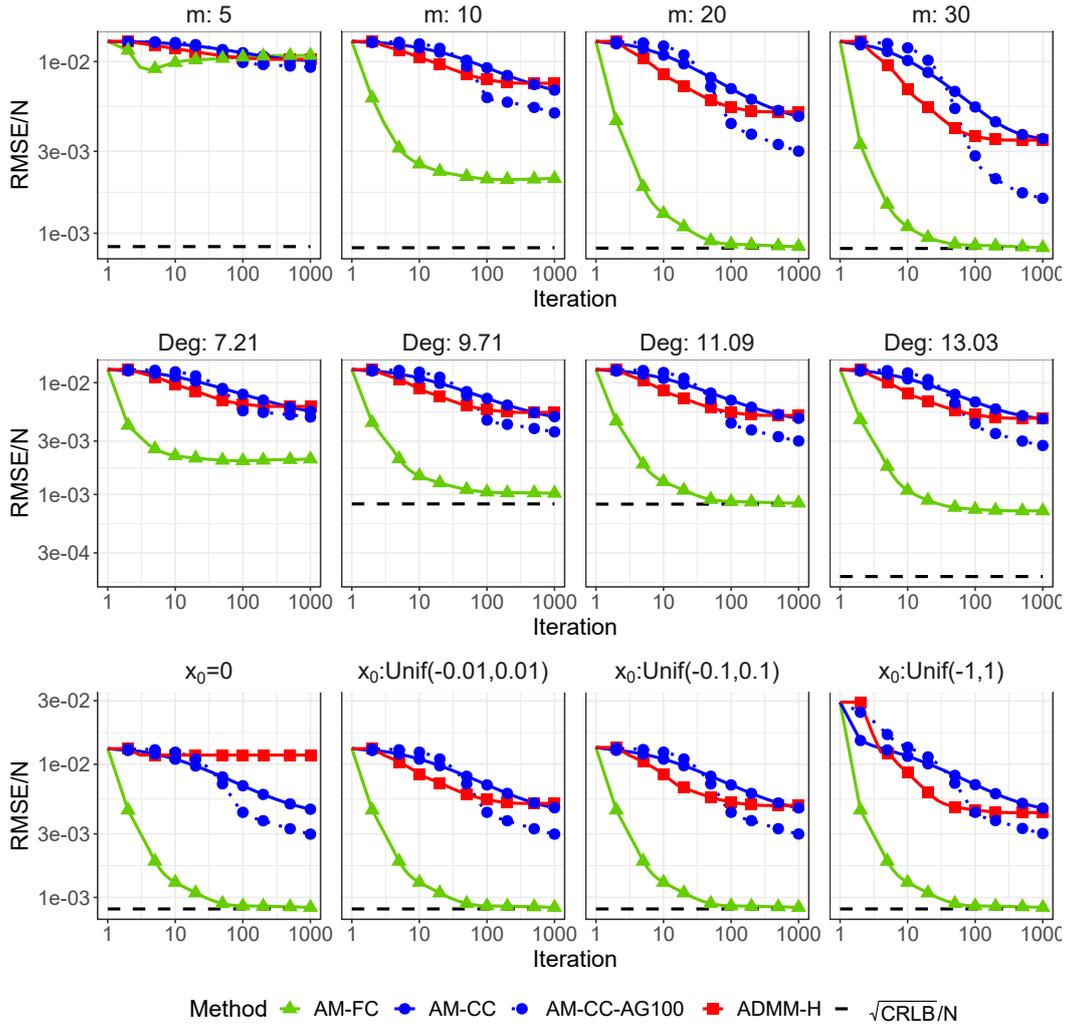


Figure 6: RMSE of AM centralized and distributed methods vs. ADMM-H on a network with  $N = 980$ , for various number of anchors  $m$  (first row), various node degree  $M_i$  (second row), and various initialization of  $\mathbf{x}_0$  (third row).

## 8 Acknowledgments

We would also like to thank Nicola Piovesan for graciously providing the code for the ADMM-H algorithm from [27], and Andrea Simonetto and Geert Leus for graciously providing the code for the EML algorithm from [31]. Last but not least we would like to thank Amir Beck for introducing us to the WSN problem.

## Appendix A Proofs of Proposition 1 and Proposition 2

In this section we prove the two promised results: Proposition 1 and Proposition 2.

*Proof of Proposition 1.* Using the conditions obtained in (8), (9) and (10), we obtain that  $\mathbf{x}_i^* - \mathbf{x}_j^* \in \partial\delta_{\mathcal{B}}(\mathbf{u}_{ij}^*)$  for all  $(i, j) \in \mathcal{E}_1$  and  $\mathbf{x}_i^* - \mathbf{a}_j \in \partial\delta_{\mathcal{B}}(\mathbf{u}_{ij}^*)$  for all  $(i, j) \in \mathcal{E}_2$ . Since  $\delta_{\mathcal{B}}(\cdot)$  is a proper, lower semicontinuous and convex function it follows from [8, Theorem 4.20, p. 104] that  $\mathbf{u}_{ij}^* \in \partial\delta_{\mathcal{B}}^*(\mathbf{x}_i^* - \mathbf{x}_j^*)$

for all  $(i, j) \in \mathcal{E}_1$  and  $\mathbf{u}_{ij}^* \in \partial \delta_{\mathcal{B}}^*(\mathbf{x}_i^* - \mathbf{a}_j)$  for all  $(i, j) \in \mathcal{E}_2$ , where  $\delta_{\mathcal{B}}^*$  is the Fenchel conjugate of  $\delta_{\mathcal{B}}$ . From [8, Example 2.31, p. 28] it also follows that  $\delta_{\mathcal{B}}^*(\cdot) = \|\cdot\|$ . Hence  $\mathbf{u}_{ij}^* \in \partial \|\cdot\|(\mathbf{x}_i^* - \mathbf{x}_j^*)$ ,  $(i, j) \in \mathcal{E}_1$ , and that  $\mathbf{u}_{ij}^* \in \partial \|\cdot\|(\mathbf{x}_i^* - \mathbf{a}_j)$ ,  $(i, j) \in \mathcal{E}_2$ . The result now follows by using these facts in (8).

*Proof of Proposition 2.* We start by assuming, without the loss of generality, that the sub-graph built from the sensors with edges  $\mathcal{E}_1$  is connected. While this is not necessarily the case, all the arguments given below can be applied to each connected component, and from the fact that the entire network is connected, according to Assumption 1(ii), we can derive the same result.

Now, in order to prove item (i), we first show that  $\ker \tilde{\mathbf{Q}} = \text{span}\{\mathbf{1}_N\}$ . Indeed, let  $\mathbf{v} \in \ker \tilde{\mathbf{Q}}$ . Each row of  $\tilde{\mathbf{Q}}$  corresponds with some  $(i, j) \in \mathcal{E}_1$ , and includes only two non zero entries, which are 1 at the  $i$ -th entry and  $-1$  at the  $j$ -th entry. Thus, we obtain that  $v_i = v_j$  for all  $(i, j) \in \mathcal{E}_1$ , and since the sub-graph is connected we obtain that  $\mathbf{v} \in \text{span}\{\mathbf{1}_N\}$ . The converse inclusion trivially follows from the structure of  $\tilde{\mathbf{Q}}$ .

Now we will show that  $\ker \tilde{\mathbf{Q}} \cap \ker \tilde{\mathbf{A}} = \{\mathbf{0}_N\}$ . Let  $\mathbf{v} \in \ker \tilde{\mathbf{Q}} \cap \ker \tilde{\mathbf{A}}$ . Since the graph is connected (see Assumption 1(i)), and since we have at least one anchor (see Assumption 1(ii)), there exists a sensor  $i \in \mathcal{V}$  such that  $i$  is connected to an anchor  $j \in \mathcal{A}$ . Thus, by construction, the row associated with edge  $(i, j) \in \mathcal{E}_2$  in the matrix  $\tilde{\mathbf{A}}$  equals to the unit vector  $\mathbf{e}_i^T$ . Therefore, since  $\mathbf{v} \in \ker \tilde{\mathbf{A}}$  we obtain that  $v_i = 0$ , and since  $\mathbf{v} \in \ker \tilde{\mathbf{Q}} = \text{span}\{\mathbf{1}_N\}$ , we must have that  $\mathbf{v} = \mathbf{0}_N$ , which completes the proof.

We now show that  $\mathbf{P}$  is positive definite. Take  $\mathbf{v} \in \mathbb{R}^N$ , then

$$\mathbf{v}^T \tilde{\mathbf{P}} \mathbf{v} = \mathbf{v}^T \left( \tilde{\mathbf{Q}}^T \tilde{\mathbf{Q}} + \tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \right) \mathbf{v} = \mathbf{v}^T \begin{pmatrix} \tilde{\mathbf{Q}}^T \\ \tilde{\mathbf{A}}^T \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{Q}} \\ \tilde{\mathbf{A}} \end{pmatrix} \mathbf{v} = \left\| \begin{pmatrix} \tilde{\mathbf{Q}} \\ \tilde{\mathbf{A}} \end{pmatrix} \mathbf{v} \right\|^2.$$

Since  $\tilde{\mathbf{P}}$  is obviously a positive semi-definite matrix and  $\mathbf{v}^T \tilde{\mathbf{P}} \mathbf{v} = 0$  if and only if  $\mathbf{v} \in \ker \tilde{\mathbf{Q}} \cap \ker \tilde{\mathbf{A}} = \{\mathbf{0}_N\}$ ,  $\tilde{\mathbf{P}}$  is positive definite. By [17, Property IX, p. 27]), the eigenvalues of  $\mathbf{P}$  and  $\tilde{\mathbf{P}}$  are the same, and so  $\mathbf{P}$  is also positive definite.

Item (ii) now follows immediately from item (i) since  $\mathbf{x} \rightarrow G(\mathbf{x}, \mathbf{u})$  is a quadratic function (see (4)) for any fixed  $\mathbf{u}$ .

From [9, Lemma 2.42, p.32] it follows that  $G$  is coercive and since  $F \geq G$ , the result of Item (iii) follows. Lastly, Item (iv) follows from [8, Theorem 2.14, p. 20].

## References

- [1] A. Agarwal et al. “Sensor network localization for moving sensors”. In: *2012 IEEE 12th International Conference on Data Mining Workshops*. IEEE. 2012, pp. 202–209.
- [2] H. M. Ammari. *The Art of Wireless Sensor Networks*. Vol. 1. Springer-Verlag, Berlin, 2014, pp. xvii+830.
- [3] R. Andreani et al. “On augmented Lagrangian methods with general lower-level constraints”. In: *SIAM J. Optim.* 18.4 (2007), pp. 1286–1309.
- [4] H. Attouch and J. Bolte. “On the convergence of the proximal algorithm for nonsmooth functions involving analytic features”. In: *Math. Program.* 116.1-2, Ser. B (2009), pp. 5–16.
- [5] H. Attouch, J. Bolte, and B. F. Svaiter. “Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods”. In: *Math. Program.* 137.1-2, Ser. A (2013), pp. 91–129.
- [6] L. Barenboim and M. Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013, p. 171.
- [7] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Second. CMS Books in Mathematics. Springer, 2017, pp. xix+619.
- [8] A. Beck. *First-Order Methods in Optimization*. Vol. 25. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2017, pp. xii+475.
- [9] A. Beck. *Introduction to nonlinear optimization*. Vol. 19. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2014, pp. xii+282.
- [10] A. Beck, P. Stoica, and J. Li. “Exact and approximate solutions of source localization problems”. In: *IEEE Trans. Signal Process.* 56.5 (2008), pp. 1770–1778.
- [11] P. Biswas et al. “Semidefinite programming approaches for sensor network localization with noisy distance measurements”. In: *IEEE T. Autom. Sci. Eng.* 3.4 (2006), pp. 360–371.
- [12] P. Biswas et al. “Semidefinite programming based algorithms for sensor network localization”. In: *ACM T. Sensor Network* 2.2 (2006), pp. 188–220.
- [13] J. Bolte, A. Daniilidis, and A. Lewis. “The Lojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems”. In: *SIAM J. Optim.* 17.4 (2006), pp. 1205–1223.

- [14] J. Bolte, S. Sabach, and M. Teboulle. “Proximal alternating linearized minimization for nonconvex and nonsmooth problems”. In: *Math. Program.* 146.1-2, Ser. A (2014), pp. 459–494.
- [15] J. Bolte et al. “First order methods beyond convexity and Lipschitz gradient continuity with applications to quadratic inverse problems”. In: *SIAM J. Optim.* 28.3 (2018), pp. 2131–2151.
- [16] S. Goyal and M. S. Patterh. “Modified bat algorithm for localization of wireless sensor network”. In: *Wireless Pers. Commun.* 86.2 (2016), pp. 657–670.
- [17] A. Graham. *Kronecker products and matrix calculus with applications*. Ellis Horwood Limited, 1981.
- [18] W. B. Heinzelman, A. P. Chandrakasan, H. Balakrishnan, et al. “An application-specific protocol architecture for wireless microsensor networks”. In: *IEEE Transactions on wireless communications* 1.4 (2002), pp. 660–670.
- [19] K. Kurdyka. “On gradients of functions definable in o-minimal structures”. In: *Ann. Inst. Fourier (Grenoble)* 48.3 (1998), pp. 769–783.
- [20] S. Łojasiewicz. “Une propriété topologique des sous-ensembles analytiques réels”. In: *Les Équations aux Dérivées Partielles (Paris, 1962)*. Éditions du Centre National de la Recherche Scientifique, Paris, 1963, pp. 87–89.
- [21] K. W. Lui et al. “Semi-definite programming approach to sensor network node localization with anchor position uncertainty”. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2009, pp. 2245–2248.
- [22] D. R. Luke et al. “A simple globally convergent algorithm for the nonsmooth nonconvex single source localization problem”. In: *J. Global Optim.* 69.4 (2017), pp. 889–909.
- [23] Y. E. Nesterov. “A method for solving the convex programming problem with convergence rate  $O(1/k^2)$ ”. In: *Dokl. Akad. Nauk SSSR* 269.3 (1983), pp. 543–547.
- [24] E. Niewiadomska-Szynkiewicz and M. Marks. “Optimization schemes for wireless sensor network localization”. In: *Appl. Math. Comput. Sci.* 19.2 (2009), pp. 291–302.
- [25] N. Patwari et al. “Locating the nodes: cooperative localization in wireless sensor networks”. In: *IEEE Signal processing magazine* 22.4 (2005), pp. 54–69.
- [26] N. Patwari et al. “Relative location estimation in wireless sensor networks”. In: *IEEE Transactions on signal processing* 51.8 (2003), pp. 2137–2148.
- [27] N. Piovesan and T. Erseghe. “Cooperative localization in WSNs: a hybrid convex/nonconvex solution”. In: *IEEE Trans. Signal Inform. Process. Netw.* 4.1 (2018), pp. 162–172.

- [28] W. H. Press et al. *Numerical Recipes: The Art of Scientific Computing*. Third. Cambridge University Press, Cambridge, 2007, pp. xxii+1235.
- [29] S. Sabach, M. Teboulle, and S. Voldman. “A smoothing alternating minimization-based algorithm for clustering with sum-min of Euclidean norms”. In: *Pure Appl. Funct. Anal.* 3.4 (2018), pp. 653–679.
- [30] Q. Shi et al. “Distributed wireless sensor network localization via sequential greedy optimization algorithm”. In: *IEEE Trans. Signal Process.* 58.6 (2010), pp. 3328–3340.
- [31] A. Simonetto and G. Leus. “Distributed maximum likelihood sensor network localization”. In: *IEEE Trans. Signal Process.* 62.6 (2014), pp. 1424–1437.
- [32] C. Soares, J. Xavier, and J. Gomes. “Simple and fast convex relaxation method for cooperative localization in sensor networks using range measurements”. In: *IEEE Trans. Signal Process.* 63.17 (2015), pp. 4532–4543.
- [33] C. Soares, J. Xavier, and J. Gomes. “Distributed, simple and stable network localization”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2014, pp. 764–768.
- [34] Z. Wang et al. “Further relaxations of the semidefinite programming approach to sensor network localization”. In: *SIAM J. Optim.* 19.2 (2008), pp. 655–673.
- [35] Y. Ye. “Computational Optimization Laboratory. Stanford University”. In: (). URL: <http://www.stanford.edu/~yye/Col.html>.