Noname manuscript No.
(will be inserted by the editor)

Solving the Sensor Network Localization Problem using an Heuristic Multistage Approach

Andrea Cassioli

Received: date / Accepted: date

Abstract The Sensor Network Localization Problem (SNLP), arising from many applied fields related with environmental monitoring, has attracted much research during the last years. Solving the SNLP deals with the reconstruction of a geometrical structure from incomplete and possibly noisy pairwise distances between sensors. In this paper we present an heuristic multistage approach in which the solving strategy is tailored on the type of problem instance at hand, formulated as a box–constrained optimization problem. We focus on low-noise SNLP, a scenario common in literature for which we propose a geometric routine (based on trilateration) to find a first guess for the sensor configuration. Local searches and problem dependent decomposition techniques are then applied to refine the sensor localizations. Computational results are presented and compared with different test sets available in literature, showing the proposed strategy to be effective for this family of SNLP instances and quite robust in different settings.

Keywords Sensor network localization · Graph realization · Distance geometry · Decomposition Techniques

1 Introduction

The Sensor Network Localization Problem (SNLP) is a very challenging optimization problem which has attracted much research during the last years. In short, the SNLP can be formulated as follows: solving an SNLP instance is to best localize a set of points (which we assume to be contained in a bounded region), on the basis of a subset of all pair-wise distances (often affected by noise) and the actual position of some of them.

Andrea Cassioli
Dipartimento di Sistemi e Informatica Università degli Studi di Firenze,Italy
Tel.: +39-055-4796464
Fax: +39-055-4796363
E-mail: cassioli@dsi.unifi.it

Thus, the aim is the reconstruction of a geometrical structure from incomplete information. SNLP arises from many applicative fields related with the use of ad hoc wireless sensor networks for environmental monitoring. In this context, networks are composed of a large number of densely deployed sensor nodes devoted to collect environmental data, the latter typically including temperature, humidity, surrounding vibrations and many others, depending on the applications.

These data are useful if they can be somehow referenced with the sensor position, or at least the best estimate available. A straightforward solution could be the use of centralized positioning systems, as the GPS system, but in many applications this is not feasible for economical and technical reasons. Indeed, in practical applications the use of thousands of GPS equipped devices will turn out to be quite expensive also for the need of high quality position measures and long lasting batteries to face a significant energy consumption. Other technical reasons discourage GPS usage: for instance, its satellite based technology is mainly suitable for open-air positioning, but often SNLP is applied to indoor applications (for example security monitoring).

Therefore alternative techniques to estimate node positions are being developed on a different basis: they rely on the measurements of distances between neighboring nodes, mainly based on criterion like time of arrival, time-difference of arrival and received signal strength.

Typical applications in which SNLP arises are animals position monitoring (for research studies or protection program), distributed surveillance system (intrusion detection, battleground protection, etc...), traffic monitoring, and items tracking in big logistic sites.

After a short problem introduction in section (2) and overview of related researches in section (3), we propose a multi-stage approach to solve the problem, focusing on low noise instances, in section (4). The latter is based on a geometric routine , introduced in section (5), devoted to construct a first guess solution and decomposition techniques as refinement, as in section (6). Computational results are summarized in section (7).

2 Problem Formulation

The SNLP model is composed by a set P of n_p points $p_i \in \Omega \subset \mathbb{R}^2$: each p_i can be either a *sensor*, a point whose position is not known, or an *anchor*, meaning its position is known. Thus, P can be partitioned in the *sensor set* S and the *anchor set* A, with cardinality $|S| = n_s$ and $|A| = n_a$.

Given a pair of points $(p_i, p_j) \in P \times P$, (i, j) for short, their Euclidean distance $\|p_i - p_j\|_2$ will be denoted by d_{ij}.

The whole distance set can be represented by an Euclidean distance matrix (EDM in short) $D \in \mathbb{R}^{n_p \times n_p}$. For m points, the set of all possible Euclidean distance matrices, denoted as $\mathbb{EDM}^m$, is a rich algebraic structure: the $\mathbb{EDM}^m$ is a convex cone contained in the subspace of all symmetric hollow matrices, that is closely related with the semi definite matrix cone (for more details see for example [15]). Discarding redundant information (i.e. considering only D_{ij} for $i < j$), we can consider $D \in \mathbb{EDM}^m$ as a finite set of pair-wise distances, whose cardinality is $|D| = n_p \cdot (n_p - 1)/2$.

In practical applications, pair-wise distances are known only if points are closer than a threshold r_r, called *radio range*; thus D is not completely specified, but a subset $\bar{D} \subseteq D$ is available:

$$\forall D_{ij} \leq r_r \rightarrow \exists \bar{D}_{ij} \in \bar{D} : \bar{D}_{ij} = D_{ij}$$

Given the set $\bar{D}$, our aim is to localize the position of each sensor, minimizing the difference between the computed distances and those in $\bar{D}$ and, if possible, that $\forall (i,j) : d_{ij} \notin \bar{D} \Rightarrow d_{ij} \geq r_r$.

SNLP can be meaningfully described using a graph representation: an undirected graph $G\langle P, E\rangle$ can model an SNLP instance, as depicted in the example of figure (1), if $\forall \bar{d_{ij}} \in \bar{D}$ the edge e_{ij} between p_i, p_j exists.

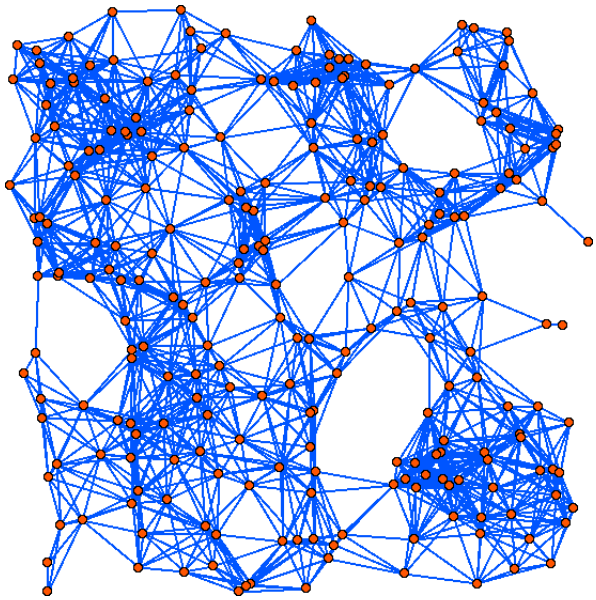


Fig. 1 Example of SNLP graph representation (anchors and sensors have been plotted with the same symbol)

For each point $p \in P$, its neighbor set $\mathcal{N}(p)$ can be defined as:

$$\mathcal{N}(p_i) = \{p_j \in P : \bar{d_{ij}} < r_r\} \tag{1}$$

G connectivity depends on r_r, which is usually tailored to guarantee graph connectivity with high probability (see [7]).

Although SNLP does not require G to be connected, we assume w.l.o.g. each connected component of G to host at least one anchor: indeed, anchors are the user access point to retrieve data, so that if a connected component doesn't host any anchors, user knows no measures.

In literature, many authors simply assume the graph to be connected, as in [37], while others, as in [36], try to define the idea of ill-nodes. The algorithm proposed in [9] is meant to signal which sensors, if any, need to be labeled as outliers.

In practice, distance measures are noisy: *actual distances* (denoted by $\bar{d_{ij}}$) must be distinguished from the *measured distances*, indicated as $\hat{d_{ij}}$. Depending on the application at hand, the kind of noise affecting the measures can be model in different ways, because the underlying statistical distribution depends on the environmental and technological context. For SNLP it is common to consider either an *additive noise*, that is $\hat{d_{ij}} = \bar{d_{ij}} + n_f \cdot \mathbf{N}(0,1)$, or a *multiplicative noise*, that is $\hat{d_{ij}} = \bar{d_{ij}} \cdot \big(1 + n_f \cdot \mathbf{N}(0,1)\big)$, where $\mathbf{N}(\cdot)$ indicates the normal distribution.

Both formulations are parametrized by a so called *noise factor* n_f, a kind of noise intensity measure. We assume in this paper the noise to be multiplicative, which is the most widely used and studied (see for example [5]).

A Box-Constrained Mathematical Model for SNLP

The use of least–squares based models for the reconstruction of geometric structure, using incomplete and noisy pair–wise distances, have been already proposed for example in [33,34,32] and [21]. The basic idea is to formulate a smooth optimization problem in order to minimize some kind of least-squares error function with respect to the point positions.

An error term is introduced for each $(i, j) \in P \times P, i < j$, depending on whether the pair-wise distance is known or not. The error terms have been chosen as in equation (2). Other error functions have been used for geometric reconstruction problems (as for example [33,34]), but the one in equation (2) has been shown to be very effective (see for instance [21]).

$$e_{ij} = \begin{cases} (i,j) \in \bar{D} \ (d_{ij}^2 - \hat{d}_{ij}^2)^2 \\ \\ (i,j) \notin \bar{D} \ \{\max(r_r^2 - d_{ij}^2, 0)\}^2 \end{cases} \tag{2}$$

While the first kind of term considers measurement errors, the second (referred also as *repulsive term*) penalizes sensors for which the pair-wise distance is not available, but they are closer than r_r.

As introduced in section (2), points are assumed to be deployed in a limited region. The unit square is considered w.l.o.g as feasible set, but any other choice might be possible. Moreover, for each sensor s_i connected to some anchors, its feasible set can be reduced considering the intersection between square boxes (with $2r_r$ width) centered on each connected anchor a_k, which we denote as b_k.

$$\forall s_i \in S, B_i = \begin{cases} \cap_k b_k & \mathscr{N}_a(s_i) \neq \emptyset \\ \\ [0,1]^2 & \mathscr{N}_a(s_i) = \emptyset \vee \cap_k b_k = \emptyset \end{cases} \tag{3}$$

Thus the SNLP can be cast as a box-constrained optimization problem,a as formalized in (4):

$$\begin{array}{ll} \underset{S}{\text{minimize}} & \sum_{(i,j)\in P\times P} e_{ij} \\ s.t. & s_i \in B_i, \forall s_i \in S \end{array} \tag{4}$$

Discarding repulsive contributions, i.e. allowing sensors to be closer than expected, a relaxed error function is considered, as follows in problem (5).

$$\begin{array}{ll} \underset{S}{\text{minimize}} & \sum_{i<j:i,j\in\bar{D}} e_{ij} \\ s.t. & s_i \in B_i, \forall s_i \in S \end{array} \tag{5}$$

The formulation of the SNLP as a box–constrained optimization problem fits well with the typical practical settings: indeed, the lack of information and their noisiness discourage the use of strict constraints on the measures, i.e. repulsive terms might not well describe the problem instance.

3 Literature Overview

The SNLP has been a popular and challenging problem in the last years. Practical applications in which it arises can be found for example in [30, 38] and [22].

SNLP is strictly related with the so-called Euclidean distance matrix completion problem (EDMCP), which can be defined, following [2], as *"...the problem of determining whether or not a given partial matrix can be completed into a Euclidean distance matrix.."*.

Many papers are devoted to the EDMCP, for example [16, 3] and [1]. In [23, 27, 26, 2] and [28] results on EDMCP solution and its relation with semi-definite matrices are shown. A natural extension of EDMCP is to introduce the presence of noise. In order to guarantee sensors to be embedded in a two dimensional space, suitable rank constraints have to be imposed on distance matrix (see [15]). The latter problem is called *embedding problem* and has been widely studied (see for example [1],[3] and [15]) and proved to be a very challenging field.

The most frequently used approach to solve SNLP is based on Semidefinite Programming (SDP) or Second-Order Conic Programming (SOCP) relaxations, the latter being used in the seminal work of Doherty (see [17]).

In [29, 5] and [6], the authors show that if the optimal solution is unique, then the SDP finds it and bounds for the expected error (in the case of additive and multiplicative noise) are presented. Moreover, they present two techniques to improve solution quality: the former uses a regularization terms to avoid sensors to gather together inside the feasible region. A second techniques uses the solution of the SDP model as the starting point for a gradient method.

Further SDP relaxations have been proposed in [40] to overcome the huge dimension of the SDP relaxations, both in variables and constraints. Authors proposed node-based (NSDP) and edge-based (ESDP) SDP models, both aiming to use semi-definiteness constraints only on smaller sub-matrices.

In [39] many theoretical results about SOCP relaxation properties (for examples points will stay in the convex hull of the anchors) and error analysis (between SOCP and SDP) are presented.

Graph connectivity and rigidity foundations have been introduced in [18] and connections with SDP solutions have been exploited for example in [31]. A graph connectivity based approach is presented in [36], the so-called Hop-Terrain algorithm, mainly designed for in-place distributed computing. The latter aspect is investigated also in SDP methods coupled with gradient descent in [4] and in [37] for the MDS approach. Theoretical results about randomized graph rigidity and realization can be found for example in [18] and [7].

An approach related with SDP and SOCP has been followed in [35], where Sum-of-Squares (SOS) relaxations have been used to tackle the SNLP.

One of the state-of-the-art algorithm up to now is the so-called SpaseLoc (see [9, 22]) in which SDP techniques have been coupled with geometric routines in order to tighten relaxations and yield better results.

Further SDP relaxations, based again on a graph representation of the SNLP, have been recently proposed in [24], in which authors use sensor graph connectivity to reduce the SDP problem size. Connections with other methods and many computational results are presented.

Also multi-dimensional scaling (MDS) has been used to recover sensor positions, see for example in [13] and [37]. Linear programming has been used for example in [19]: a simpler model without noise is analyzed and geometric constraints added to tighten the feasible set. A different heuristic approach can be found in [38], in which a discrete model, based on a regular square grid, has been proposed.

4 A Multi-Stage Approach to SNLP

The strategy proposed to solve the SNLP is based on the following observations:

- the *noise factor* affects the distance measurements reliability, but can be usually estimated;
- the *radio range* and the number of *anchors* and *sensors* can be used to estimate the sensor graph connectivity (see [7]);
- the ratio between the number of *anchors* and *sensors* is related with the number of sensors connected to at least one anchor;
- anchor positions are usually assumed to be uniformly distributed over the feasible set, but this is not always the case. In some applications we may face the possibility that anchors are manually placed and hence not covering uniformly the physical space (for example, in surveillance applications, most of them should be located along the boundary).

All the above information are known in advance and SNLP instances might be classified and hence tackled on that basis. In this paper the focus is on low-noise SNLP instance, i.e. distances are affected by a noise with n_f less than 0.1 (yet greater

values might be considered); moreover a reasonable ratio between n_a and n_s is assumed in order to guarantee sensor network connection with high probability (see [7]).

Although such assumptions might seem restrictive, they are common in literature as test sets in numerical experiments (see for example [39,5] or [40]) and more recently in [25] specialized algorithms for the noiseless case has been proposed.

In this setting, measures are still close to actual distances and hence able to well describe the SNLP geometry; moreover a good n_a/n_s ratio implies that most sensors are likely to be close to an anchor and hence there are less degrees of freedom in composing a feasible configuration.

Thus a first guess configuration can be computed by means of geometric routines, based on standard trilateration (see [14]): a sequential positioning scheme is proposed in section 5. This procedure often ends providing a quite good first approximation, that can be used as starting point of a local optimization of problem (5), performed by means of standard box-constrained procedures.

Two different algorithms for box-constrained optimization problems have been tested: L–BFGS–B (see [8]) and a box-constrained non-monotone implementation of the Barzilai-Borwein algorithm (NM–BB) (see [20]), using a filter-based line search.

Experiments shows L-BFGS-B to be more effective than NM-BB for small problems; on the other hand, NM–BB has proved to be very effective especially for high dimension problems, in particular if low accuracy is required.

Therefore, a combination of both local searches has been applied, using first NM–BB with low accuracy and then performing a refinment with L-BFGS-B. In both cases, the local optimization has been performed using the relaxed formulation (5), with a great savings in time.

After the whole sensor set has been optimized, the solution has often only few groups of sensors not well localized. In order to improve the solution quality, but working only on small sensors subsets, a decomposition scheme has been applied using the formulation as in problem (4), i.e. use also repulsive terms (details are given in section 6).

The overall multi–stage algorithm is summarized as follows:

1. Compute a first guess solution by means of geometric prepositioning routine.
2. Locally optimize problem (5) using NM-BB.
3. Refine the solution of problem (5) using L–BFGS–B.
4. Refine the solution applying an ad hoc decomposition technique to problem (4).

The proposed multistage strategy is actually stopped as soon as the required accuracy is fulfilled, leading to considerable saving in time.

5 A Sequential Sensor Positioning Procedure

The use of a first phase to generate a good starting configuration for the sensors localization has been proposed in [6,4] and [29]. In these cases SDP relaxations are solved to produce good starting points to initialize a gradient-based local search. In

[9] a more geometric oriented technique is presented in order build up the SNLP solution: the resulting algorithm, named Spaseloc, sequentially solves SDP sub-problems selected by means of sensor graph inspection. Such sub-problems have tunable dimension and use geometric construction to augment sensor connectivity information: geometric routines are used whenever there are no more than two available measurements for the sensors at hand (for more details see [22]).

The routine proposed in this section draws some inspiration from the Spaseloc approach, but it is designed to be parameter free, hence with no tuning needed.

The proposed procedure is an iterative scheme in which sensors are processed sequentially. Let S^k be the set of sensors whose positions have been already decided and temporary fixed at the $k-$th iteration (at the beginining $S^0 \equiv \emptyset$): a sensor $s_i \notin S^k$ is then selected, its position computed and added to S^k (s_i is denoted as *placed*).

Thus, a sensor selection rule and a procedure to *"place"* the selected sensor are needed. Before proceeding, for each sensor s_i the following subsets are introduced:

- $\mathcal{N}_\pi^s(s_i) \subseteq \mathcal{N}(s_i)$ – the set of *placed* neighbor sensors;
- $\mathcal{N}_\pi(s_i) = \mathcal{N}_\pi^s(s_i) \cup \mathcal{N}^a(s_i)$ – the set of placed neighbors;
- p_j^i – the $j-$th point in $\mathcal{N}_\pi(s_i)$ (the superscript is avoided whenever not needed);
- $\chi(c,r)$ – a circle centered in c with radius r (the radius is avoided whenever it can be deduced by the context);

5.1 Sensor selection rule

At each iteration k, a sensor $s_i \notin S^k$ must be selected to be localized. In order to reduce noise influence, whenever possible a sensor for which $\mathcal{N}^a(s_i) \neq \emptyset$ should be selected; moreover it is also reasonable to give prominence to sensors for which $\mathcal{N}_\pi(\cdot)$ is larger, because information about the localization of the already placed sensors can be used.

Thus, at each iteration k, a sensor $s_i \notin S^k$ is selected as:

$$\begin{array}{cc} s_i \in \text{argmax}_S & |\mathcal{N}^a(s_i)| \\ s.t. & |\mathcal{N}_\pi(s_i)| \geq 3 \\ & s_i \notin S^k \end{array}$$

In case no sensors fulfill these properties, the selected sensor is choosen as

$$\begin{array}{cc} s_i \in \text{argmax}_S & |\mathcal{N}_\pi(s_i)| \\ s.t. & s_i \notin S^k \end{array}$$

In case that multiple choices are available, the sensor with smallest index is selected.

We finally remark there are many other ways to arrange a sequential procedure similar to what we have presented in this section (for instance we might apply random based strategy).

5.2 Sensor position selection rule

Let s_i be the chosen sensor to be localized. All points already positioned will be denoted with the letter p, no matter if they are sensors or anchors.

If pair-wise distances were exact, s_i belongs to the intersections of circles $\chi(p_j^i, \hat{d}_{ij})$, that is the solution of the system of equations (6).

$$\|s_i - p_j^i\|^2 = \hat{d}_{ij}^2 \quad \forall j = 1, \ldots, k \tag{6}$$

In case of noisy or missing information (i.e. no more than two available measures), solution of (6) might neither be unique nor exist, as depicted for example in figure (2).

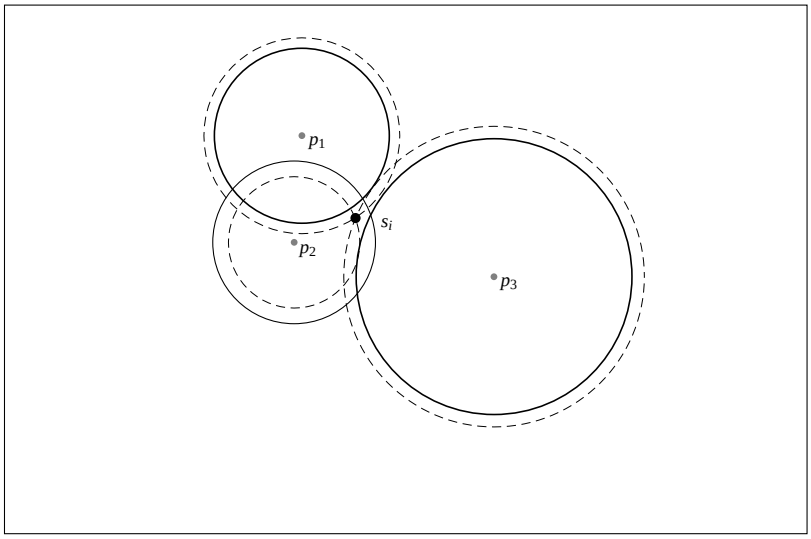


Fig. 2 The effect of noisy pairwise distances on the standard trilateration (dashed lines represent the exact measures, while the solid lines are inexact).

To overcome this problem, we use the concept of *radical center* as a surrogate of the circles intersection when the latter does not exists. Given three circles, their radical center is the intersection of the three pair-wise *radical lines*. The radical line is perpendicular to the segment connecting circle centers and pass through their intersections if they exist; otherwise, it is closer to the bigger circle (more details can be found for example in [14]).

Thus, if circles intersection is unique, then such point is the radical center. In the noisy case, the radical center can be used as an estimate of the point s true position, as in figure (3).

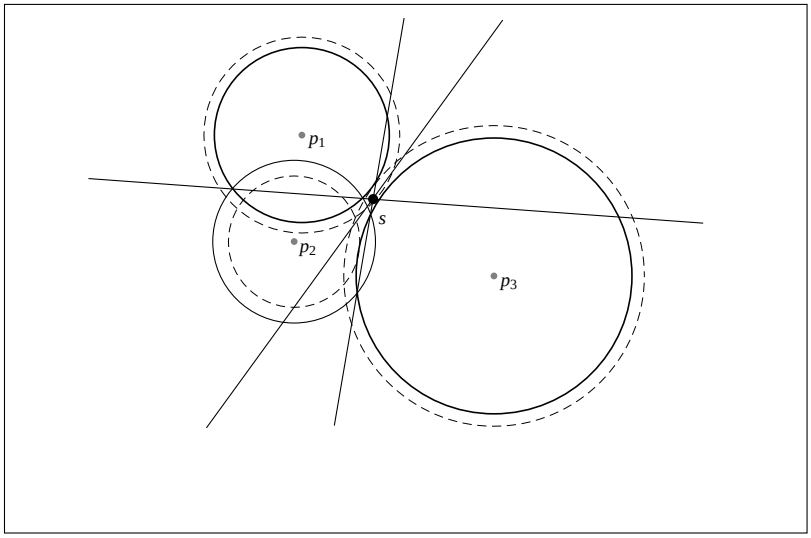


Fig. 3 Radical lines with inexact distances: their intersection is a good putative location for the actual sensor position.

It must be also noticed that:

- the radical line is always defined for each pair of circles;
- the radical center is always defined for each triple of circles;

The radical center computed from noisy distances can sometime be far from (even located outside the feasible region). A simple procedure to improve s_i localization is sketched in algorithm (1). s_i can be moved towards points in $\mathcal{N}_\pi(s_i)$, adding a displacement obtained as the average among weighted vectors $s_i - p_j$, as depicted in figure (4).

input: the sensor s_i
foreach $p_j \in \mathcal{N}_\pi(s_i)$ **do**
  $\hat{d} \leftarrow \|s_i - p_j\|$
  $\beta \leftarrow \sqrt{\frac{\hat{d} - d_{ij}}{\hat{d}}}$
  $v(p_j) \leftarrow \beta(s_i - p_j)$
end
$s_i \leftarrow s_i + \frac{1}{|\mathcal{N}_\pi(s_i)|} \sum_{p_j \in \mathcal{N}_\pi(s_i)} v(p_j)$
project s_i onto the feasible set;

Algorithm 1: A procedure to reduce possible radical center drifting away.

Thus, for the sensor selected sensor s_i, a different strategy is used to choose its first guess position, depending on the number of connected anchors and placed sensors. In

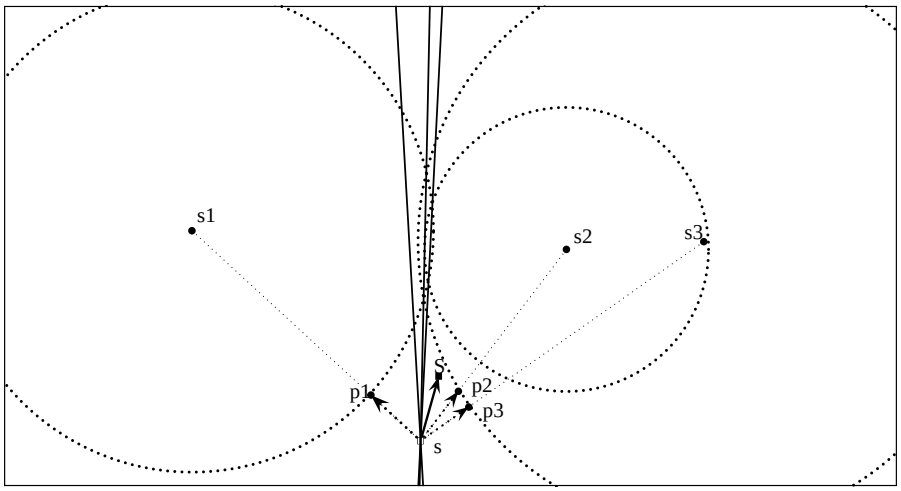


Fig. 4 An example of the execution of Algorithm (1): the result of using radical lines intersection is the point s, which is then corrected by the average of vectors p_i, finding the S point.

each proposed strategy, after a pure geometric localization, a refinement is performed by means of a local optimization, involving only the coordinates of the sensor at hand. The local search uses the optimization subproblem (7), in which only contributions from already localized sensors are considered.

$$\begin{array}{ll} \text{minimize} & \sum_{j \in \mathcal{N}_\pi(s_i)} e_{ij} \\ S & \\ s.t. & s_i \in B_i, \forall s_i \in S \end{array} \tag{7}$$

Solution of subproblem (7) is usually extremely fast and performed using the L-BFGS-B algorithm.

More than two placed neighbors

In this case, having $|\mathcal{N}_\pi(s_i)| = k \geq 3$, we can directly use the radical center as an estimate for the true s_i position. To such purpose we first notice that:

- being the measures noisy, each triple of placed points we choose might give us a different radical center;
- to ease numerical problem, it is convenient to use radical lines with the greatest difference in slope
- if the radical lines are not parallel, then we only need two of them to compute the radical center;

Thus we proceed using the pair of radical lines with the greatest difference in slope. Their intersection is corrected using Algorithm (1) and then a local optimization is started to finally localize the sensor position.

A single placed neighbor

In case $\mathcal{N}_\pi(s_i) = \{p_j\}$, s_i is randomly placed along $\chi(p_j, \hat{d}_{ij}) \cap B_i$, and then locally optimized to refine its position.

A pair with two placed neighbors

If more then one sensor is connected to two placed points, we look for a pair (s_i, s_j) such that:

- $\hat{d}_{ij} = \|s_i - s_j\|_2$ is known;
- $|\mathcal{N}_\pi(s_i)| = \left|\mathcal{N}_\pi(s_j)\right| = 2$;
- $n_{cs} = |\mathcal{N}_\pi(s_i) \cap \mathcal{N}_\pi(s_j)| < 2$;

The common link between s_i, s_j can be useful to localize them reducing errors. Two situations can be distinguished:

- $n_{cs} = 0$ – the situation is the one depicted in figure (5): considering the possible combinations of the circles intersections, we end up with four possible configurations.

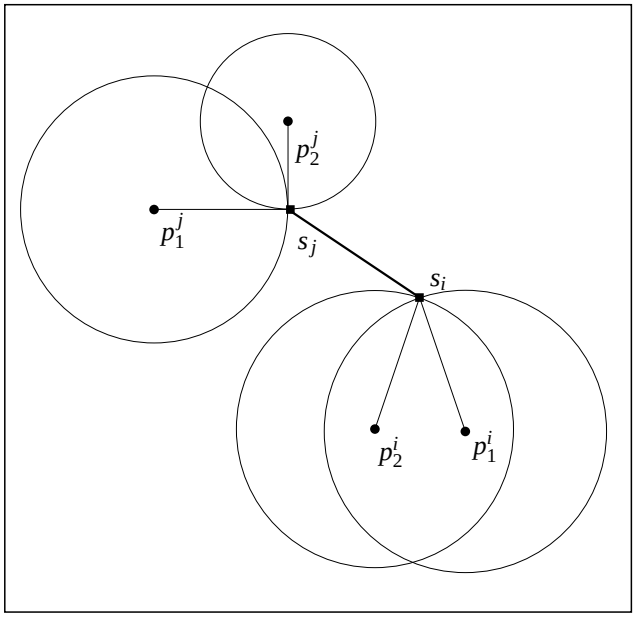


Fig. 5 Example for the case $|\mathcal{N}_\pi(s_i) \cap \mathcal{N}_\pi(s_j)| = 0$

Using problem 4 formulation, restricted to the pair s_i, s_j, the configuration with the best objective function value is chosen. Then a local search is started to refine the solution.

– $n_{cs} = 1$ – in this case $\mathcal{N}_\pi(s_i)$ and $\mathcal{N}_\pi(s_j)$ share a common point, as depicted in figure (6). Mid points of $\chi(p_1^i), \chi(p_2^i)$ and $\chi(p_1^j), \chi(p_2^j)$ are computed and used as starting point for the local optimization refinement step.

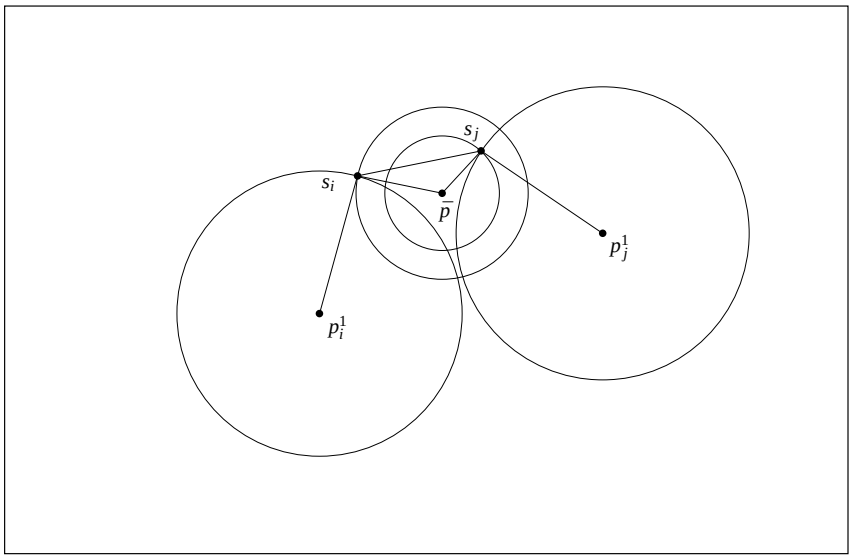


Fig. 6 Example for the case $|\mathcal{N}_\pi(s_i) \cap \mathcal{N}_\pi(s_j)| = 2$

6 Decomposition Techniques for SNLP

SNLP instances are generally composed of thousands of sensors, with a number of variables of the same order: thus, the use of decomposition techniques can be very beneficial, for sequentially solving smaller problems reduce memory requirements and function evaluation can be performed efficiently (see [10]).

Moreover, from our experience, the result of the minimization of problem (5) is usually of good quality, so that most sensors are well located, while only small groups needs to be refined.

In a general decomposition framework, as the one described in Algorithm 2, at any iteration k, some variables are kept fixed to their current values, and the others, the so called *working set* W^k are determined by solving the corresponding sub-problem.

Global convergence properties of the generated sequence $\{x^k\}$ depend on the rule for selecting the working set W^k. For non-convex problems with box constraints and one linear equality constraint, a convergent scheme has been proposed in [12].

input: $x^0 \in \mathscr{F}$
for $k = 0, 1, \ldots$ **do**
  choose a working set $W^k \subset \{1, \ldots, n\}$;

$$\begin{aligned} x^{k+1} \in \arg\min \ & f(x) \\ s.t. \ & \\ & l_i \leq x_i \leq u_i \ i \in W^k \\ & x_i = x_i^k \qquad \forall i \notin W^k \end{aligned} \tag{8}$$

end

Algorithm 2: General Decomposition Scheme

In this work we apply a novel working set selection rule, namely the ε-MVD rule proposed in [11], which gives great freedom in the choice of the working set. In brief, at each iteration k, to ensure convergence, at least the variables that mostly violate stationarity conditions are required to be included in W^k.

Denoting with $\mathscr{F}$ the feasible box, for any point $x \in \mathscr{F}$, the reduced gradient $\nabla^{red} f(x)$ is defined as follows

$$\nabla_i^{red} f(x) = \left\{ \begin{array}{ll} \min\{0, \nabla_i f(x)\} & \text{if } x_i = l_i \\ \nabla_i f(x) & \text{if } l_i < x_i < u_i \\ \max\{0, \nabla_i f(x)\} & \text{if } x_i = u_i \end{array} \right\} \quad \forall i = 1, \ldots, n \tag{9}$$

A point $\bar{x} \in \mathscr{F}$ is a stationary point if and only if $\nabla^{red} f(\bar{x}) = 0$.

Based on the violation of the optimality condition (9), we introduce a decomposition algorithm for the case of feasible set defined by box constraints. For any point $x^k \in \mathscr{F}$, we denote by $I_r(x^k)$ the index set such that, $i \in I_r(x^k)$ implies

$$|\nabla_i^{red} f(x)| \geq |\nabla_j^{red} f(x)| \quad \text{for all } j = 1, \ldots, n. \tag{10}$$

We also introduce, for a given scalar $\varepsilon > 0$, the following indices (provided they exist)

$$j^k \in \arg\max_j \{\nabla_j f(x^k) : \ \nabla_j f(x^k) > 0, \ x_j^k \geq l_j + \varepsilon\} \tag{11}$$

$$p^k \in \arg\min_p \{\nabla_p f(x^k) : \ \nabla_p f(x^k) < 0, \ x_p^k \leq u_p - \varepsilon\} \tag{12}$$

and the index sets

$$I_L^k = \{h : x_h^k \leq l_h + \varepsilon \text{ and } \nabla_h^{red} f(x^k) > \nabla_{j^k}^{red} f(x^k)\} \tag{13}$$

$$I_U^k = \{h : x_h^k \geq u_h - \varepsilon \text{ and } \quad \nabla_h^{red} f(x^k) < \nabla_{p^k}^{red} f(x^k)\} \tag{14}$$

Note that we set

- $I_L^k = \{h : x_h^k \leq l_h + \varepsilon \text{ and } \nabla_h^{red} f(x^k) > 0\}$ whenever j^k is not defined;
- $I_U^k = \{h : x_h^k \geq u_h - \varepsilon \text{ and } \nabla_h^{red} f(x^k) < 0\}$ whenever p^k is not defined.

input: $x^0 \in \mathscr{F}$, $\varepsilon > 0$

for $k = 0, 1, \ldots$ **do**
  if $\exists i^k \in I_r(x^k)$ *s.t. of the following conditions holds*

  (a) $l_{i^k} + \varepsilon \leq x^k_{i^k} \leq u_{i^k} - \varepsilon$;
  (b) $x^k_{i^k} \leq l_{i^k} + \varepsilon$ *and* $\nabla_{i^k} f(x^k) < 0$;
  (c) $x^k_{i^k} \geq u_{i^k} - \varepsilon$ *and* $\nabla_{i^k} f(x^k) > 0$
  then
    choose any W^k s.t. $i^k \in W^k$;
  else
    choose any W^k s.t. $\left(I^k_L \cup I^k_U \cup \{j_k\} \cup \{p_k\}\right) \subseteq W^k$;
  find

$$x^{k+1} \in \arg\min \; f(x)$$
$$s.t.$$
$$l_i \leq x_i \leq u_i \;\; i \in W^k$$
$$x_i = x^k_i \qquad \forall i \notin W^k$$

end

Algorithm 3: ε-MVD Algorithm

The rationale underlying the ε-MVD rule is to select the variables by means of an estimate of the active constraints to prevent possible *pathological* cases,mainly due to the fact that the reduced gradient is not continuous, and described in details in [10]
.

If a *pathological* case is detected, we must insert in the working set:

- the two indices j^k, p^k (provided they exist) corresponding to variables which *sufficiently* violate the optimality conditions and in the limit cannot have a *pathological* behavior;
- all the indices corresponding to variables which locally *strongly* violate the optimality conditions but in the limit can have a *pathological* behavior as above, namely the indices of I^k_L and of I^k_U.

The decomposition scheme, as in algorithm (2), equipped with the ε-MVD rule, converges to a stationary point, as we stated in proposition (1).

Proposition 1 *Let* $\{x^k\}$ *be the sequence generated by algorithm (2) with the* ε-MVD *working set selection rule as in algorithm (3). Every limit point of* $\{x^k\}$ *is a stationary point.*

The proof of Proposition (1) can be found in [11]. The ε-MVD rule gives a great degree of freedom in the choice of the working set, because only requires certain variables to be in the working set.

We apply the decomposition scheme introduced so far in order to refine the SNLP solution found minimizing problem (5). First of all, we select the t variables (usually

one or two) that mostly violates the stationarity condition defined in (9). Then we compose W^k in order to fulfill the ε-MVD rule.

The working set is augmented on a problem structure basis:

1. if the x or y coordinate of a sensor has been selected in the W^k, then add also the other:
$$\forall s_i : x_i \in W^k \vee y_i \in W^k \rightarrow W^k = W^k \cup s_i$$
2. since errors often affect connected groups, coordinates of sensors belonging to the same neighbor are added to W^k.
To select them, we use the concept of *hop distance* (see for example [36]): given two nodes in a graph v_1, v_2, their "hop distance" $H(v_1, v_2)$ is defined as smallest number of arcs needed to go from v_1 to v_2. Using the SNLP graph formulation (see section (2), the hop distance directly apply to the sensor neighbour structure. Thus, given a sensor s, the hop distance induces a sequence of sensor set $H_i(s)$ related with the sensor neighbor ($H_1(s) = \mathscr{N}(s)$), as depicted in Figure 7.

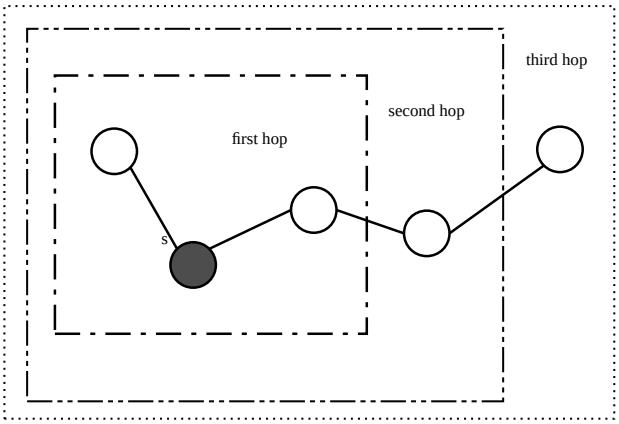


Fig. 7 An example of node neighbor depending on its hop distance from the others.

Then, $\forall s_i \in W_k$, we proceed recursively up to the $l-$th hop (typically two), adding the connected sensors to W^k.

7 Computational Results

To validate the proposed strategy to solve the SNLP, extensive computational tests have been performed using test sets from [39, 9] and [24]. The following settings has been used, if not otherwise specified:

- multiplicative noise is considered;
- results are the average among 100 randomly generated instances;

– disconnected components (i.e. a group of sensors including no anchors) have been removed;
– all tests has been performed on an Pentium 4 standard desktop machine with 1Gb ram and 512 KB cache size, running Linux, using our C++ code compiled with the g++ GNU compiler with machine dependent optimization flags.
– the average and the median of the error measure in uses and the cpu time is reported (against the results published, to ease the reader);

Test set from [39]

The test set used in [39] (very similar to the one in [40]) is composed by two different test suite.

The first test set is composed by nine test cases, using multiplicative noise with different strength, from the noiseless case to 1% of noise.

Results are depicted in table 1, in which we listed the average value and the median both for the execution time and the error, measured as $e_\infty = \max_i \|s_i - \hat{s}_i\|$.

				average		median		results in [39]	
n_s	n_a	n_f	r_r	e_∞	t	e_∞	t	e_∞	t
900	100	0	0.06	0.101778238	0.9241	0.03867	0.705	0.11	0.2
900	100	0.001	0.06	0.14347075	0.7915	0.03878	0.58	0.17	0.4
900	100	0.01	0.06	0.096472033	0.926	0.03591	0.815	0.17	1.6
1800	200	0	0.06	0.0118778481	1.5009	0.00971	1.33	0.77	0.8
1800	200	0.001	0.06	0.012569626213	1.862	0.00859	1.72	0.09	1.8
1800	200	0.01	0.06	0.012587101758	1.4454	0.00983	1.285	0.094	2.9
3600	400	0	0.035	0.0272958409	3.8693	0.00710	3.21	0.09	1.9
3600	400	0.001	0.035	0.0274628087	3.6818	0.00726	3.185	0.09	5.1
3600	400	0.01	0.035	0.0375357846	4.0535	0.00759	3.485	0.09	6.1

Table 1 Result for the first test suite as in [39].

Data in table 1 show how very good performance can be achieved both in terms of accuracy and speed. It has been observed that, during the test execution, it is common for the proposed strategy to terminate just after the geometric positioning. Hence, for low noise, measured distances represent enough information to efficiently solve the problem.

An example of our algorithm execution, for one instance of problem type 1, is reported in figure (8): the former depicts the output of the geometric routine used to build up a first guess, while the latter is the output of the final stage, i.e. the decomposition technique (intermediate step plots are skipped being similar to the latter).

To investigate the proposed algorithm robustness, a second test set from [39] (which is very similar to the one in [5]) has been performed. Four anchors are kept fixed at $\{(0.05;0.05),(0.05;0.95),(0.95;0.05),(0.95;0.95)\}$, while 60 sensors are generated uniformly over the unitary square. The radio range is 0.3; noise is still multi-

plicative, but its source comes from different distributions and with different strength, as summarized in table 2.

test	n_f	noise type
N1	0.1	$\mathbf{N}(0,1)$
N2	0.2	$\mathbf{N}(0,1)$
U1	0.1	$\mathbf{U}\left(-\sqrt{3},\sqrt{3}\right)$
U2	0.2	$\mathbf{U}\left(-\sqrt{3},\sqrt{3}\right)$
AN1	0.1	$\texttt{if } x \sim \mathscr{U}(0,1) < 0.5 \texttt{ then } \mathbf{N}(1,1) \texttt{ else } \mathbf{N}(-1,1)$
AN2	0.2	$\texttt{if } x \sim \mathscr{U}(0,1) < 0.5 \texttt{ then } \mathbf{N}(1,1) \texttt{ else } \mathbf{N}(-1,1)$

Table 2 Noise distributions in the second test suite from [39]

The normal uniform and normal distribution are denoted as **U** and **N**, respectively. The error measure is $e = \sum_{s_i \in S} \|s_i - \hat{s}_i\|^2$.

	results in [39]			
test	e	e_{avg}	e_{med}	e_{var}
N1	0.24	0.781	0.269	1.888
N2	0.48	0.410	0.139	0.315
U1	0.41	0.838	0.462	1.911
U2	0.29	0.423	0.193	0.285
AN1	0.63	0.671	0.258	0.652
AN2	0.71	0.321	0.309	0.020

Table 3 Results for second test set from [39].

As expected, the algorithm is less effective, since higher noise leads to less measures reliability and hence the geometric preprocessing routine might perform poorly. Looking at results in Table (3), as the median is quite lower than the average value, performances decay might be due to in few specific instances.

Test set from [24]

A more extensive test set has been proposed recently in [24]. The error measure used is the root mean square deviation, that is:

$$rmsd = \sqrt{\frac{1}{n_s} \sum_{s \in S} \|s_i - \hat{s}_i\|^2}$$

Results in [24] are the average among five tests, executed on a PowerPC 1.3Ghz with 2Gbyte ram.

				average		median		results in [24]	
n_s	n_a	n_f	r_r	*rmsd*	*t*	*rmsd*	*t*	*rmsd*	*t*
500	50	0	0.1	0.026359772927	0.2247	0.02429	0.2	0.014	10
500	50	0	0.2	0.001063622442	0.3042	2.8E09	0.22	1.9E10	6.5
500	50	0	0.3	5.4049878E10	0.4447	4.7E10	0.42	8.5E10	5.9
1000	100	0	0.1	0.003168405634	0.744	1.2E14	0.655	2E08	13.9
1000	100	0	0.2	1.02248214E09	1.241	9.8E10	1.09	8.9E11	10.3
1000	100	0	0.3	0.000197521499	1.6425	1.7E10	1.47	4.1E11	10.2
1000	100	0.1	0.1	0.0074270632	1.2378	0.00503	1.19	0.014	28.2
1000	100	0.1	0.2	0.0075539737	4.1639	0.00753	4.23	0.014	15.7
1000	100	0.1	0.3	0.0090655208	7.8687	0.00906	7.3	0.024	15.7
2000	100	0	0.1	0.002329811787	3.5113	1.3E08	3.18	1.2E07	30.7
2000	100	0	0.2	7.7111866E10	8.0516	6.9E10	7.65	3E11	24.2
2000	100	0	0.3	1.78037548E10	6.7105	1.4E10	6.52	3.2E11	24.1
2000	100	0.1	0.1	0.0063913485	3.6911	0.00533	3.405	0.012	41.6
2000	100	0.1	0.2	0.0081932296	21.1493	0.00819	22.55	0.014	38.2
2000	100	0.1	0.3	0.0091991885	41.5732	0.00915	38.395	0.022	40.5
4000	100	0	0.06	0.007077709287	7.5416	0.00518	7.01	1.4E07	204.1
4000	100	0	0.08	0.001806210002	11.0869	2.1E08	9.345	1E07	113.1
4000	100	0	0.1	0.000341724663	16.345	7.9E09	14.65	8.2E08	84.5
4000	100	0.1	0.06	0.0093600173	6.9185	0.00776	6.66	0.01	240.4
4000	100	0.1	0.08	0.0062856123	13.5703	0.00539	12.375	0.01	150.5
4000	100	0.1	0.1	0.0068843061	21.3916	0.00634	20.325	0.01	123.9

Table 4 Results of the test set from [24].

Results show our approach as a competitive strategy in most of the tests. It should be remarked that radio ranges used in these tests are quite large, compared to other test suites and connectivity requirements as in [7].

Test set from [9]

Part of the extensive test suite as in [9] has been executed. It contains a broad variety of problem instances and can be useful to test our method robustness. Tests use as error measure the following:

$$e = \frac{1}{n_s} \sum_{i=1}^{n_s} \|s_i - \hat{s}_i\|$$

Results have been compared with the best shown in [9], both in terms of cpu time and accuracy, that is the ones achieved by the I0 algorithm variant. In [9], results are the average among 10 tests, executed on a 2.4Ghz CPU with 1Gbyte ram. A first test set considers noiseless SNLP instances with different number of sensors and anchors,

such that $n_p = n_a^2$. The radio range has been chosen in order to ensure connectivity with high probability. Results are listed in table 5.

		average		median		results in [9]	
n_p	n_a	t	e	t	e	t	e
49	7	0.0032	2.5E05	0	4.2E15	0.18	4.6E08
100	10	0.0112	9.2E05	0.01	6.9E14	0.35	1.5E07
225	15	0.0619	0.00583	0.04	3.9E12	0.82	4.5E07
529	23	0.2863	0.00488	0.27	2.2E11	2.02	2.2E06
1089	33	1.7187	0.00794	1.505	1.4E06	4.48	0.00012
2025	45	3.7355	0.01030	3.25	6.1E06	8.85	0.00015
3969	63	11.8944	0.01272	10.575	4.5E06	18.79	0.00012
5041	71	15.7727	6.6E05	13.955	2.3E10	27.19	0.00015
6084	78	21.4673	0.00717	19.37	3.0E10	33.66	0.00017
7056	84	35.0069	0.01348	30.93	3.0E10	40.59	5.2E05
8100	90	47.2375	0.01553	42.485	6.2E10	47.87	0.00027
9025	95	58.328	0.01254	54.695	3.4E07	54.41	0.00021
10000	100	66.7781	0.01714	64.335	8.5E07	59.33	0.00020

Table 5 Noiseless SNLP instances.

A fixed $n_p = 3969$, a sort of middle problem size, is then considered, performing tests in which only one problem parametr change:

n_f – in table 6, n_f values from 0.1 to 0.5 are tested with a fixed $n_a = 63$ and $r_r = 0.0334$;

	average		median		results in [9]	
n_f	t	e	t	e	t	e
0.01	9.5234	0.00680	8.83	0.00076	20.38	0.00096
0.05	9.83	0.01280	9.37	0.00236	21.56	0.00315
0.1	10.8775	0.00707	10.3	0.00402	21.46	0.00687
0.2	10.976	0.01296	10.365	0.00669	21.55	0.0155
0.3	11.2225	0.01927	10.84	0.00798	21.09	0.0151
0.4	10.9147	0.01493	10.485	0.00917	21.3	0.0198
0.5	11.3258	0.01658	10.66	0.00997	22.07	0.0305

Table 6 SNLP tests from [9] varying the noise strength.

n_a – in table 7, results for noiseless problem instances with different number of anchors (from 5 to 60), with a fix $r_r = 0.0334$;

r_r – noiseless instances are considered varying the radio range from 0.0304 to 0.0334, with 63 anchors (results are listed in table 8);

	average		median		results in [9]	
n_a	t	e	t	e	t	e
40					19.3	0.00105
50	11.312	0.03028	9.97	5.0E06	19.38	0.00111
100	8.2147	0.02113	7.155	1.5E06	19.16	0.00088
150	6.8478	0.00832	5.2E07	6.16	18.86	0.00027
200	5.9082	0.01624	5.34	1.2E08	18.77	4.9E05
250	5.5249	0.00887	4.91	2.4E07	18.55	1.7E05
300	5.20010	0.00950	4.91	2.4E07	18.2	1.5E05
350	4.4121	0.01079	3.81	1.7E11	18.13	7.5E06
400	4.2368	0.01381	3.505	1.3E11	18.16	6.4E06

Table 7 SNLP tests from [9] varying the number of anchors.

	average		median		results in [9]	
rr	t	err	t	err	t	err
0.0304	15.2339	0.09802	13.75	0.00391	18.03	0.00244
0.0306	14.5301	0.0598	12.69	0.00198	18.13	0.00112
0.0308	13.86	0.05968	12.715	0.00124	18.64	0.00246
0.031	12.6298	0.05147	11.475	0.00067	18.39	0.00109
0.0312	13.0691	0.05330	11.515	0.00060	18.3	0.00248
0.0314	12.0979	0.05213	11.29	0.0006	18.9	0.00055
0.0316	11.61	0.04365	10.76	0.00019	18.95	0.00048
0.0318	11.2859	0.03449	10.695	5.8E05	19.06	0.00030
0.032	11.4092	0.03493	10.155	4.5E05	18.91	0.00042
0.0322	10.7439	0.02884	9.845	2.5E05	18.89	0.00028
0.0324	10.3411	0.03509	9.17	7.8E06	18.91	0.00052
0.0326	9.5634	0.01412	8.62	5.4E06	18.96	0.00041
0.0328	9.8509	0.02344	9.25	4.2E06	18.87	0.00023
0.033	9.3983	0.01272	8.11	4.5E06	18.83	0.00021
0.0332	9.2544	0.02110	8.31	3.5E06	19.37	0.00062
0.0334	9.2789	0.03469	8.34	4.0E06	18.79	0.00012

Table 8 Noiseless SNLP instances varying the radio range

Tests results illustrated in table 5 to 8 confirm the proposed approach to be effective, also if the problem instances are not necessarly low-noise ones. Indeed, looking at table 7, also if n_f increases, good results can be achieved.

8 Conclusion

In this paper an heuristic multistage approach to solve the Sensor Network Localization Problem is proposed. SNLP has been formulated as a box-constrained optimization problem, where a least-squares like function is minimized to fit measured

data. From the problem instance information, mainly the amount of noise affecting the measurements and the ratio between anchors and sensors, an efficient strategy can be deployed, involving both standard optimization algorithms and ad hoc procedures. The focus has been on the case of low–noise SNLP, presenting a sequential geometric procure, mainly based on the the use of the radical center as an estimate of the sensor position, in order to find a first guess for the SNLP solution. Such a procedure turns out to produce, for low noise instances, an accurate initial guess for the sensors deployment, leading to a great saving in time during the overall optimization. Solution refinements have been performed by means of decomposition techniques specifically designed for the SNLP, so that the working set selection rule exploits the sensor network structure.

Computational tests have shown our strategy to be effective for the kind of instances it has been designed for. However, tests against different kind of problem instances showed a certain degree of robustness also when problem parameters are chosen out of the required range. For this reason, we consider our approach a feasible way to solve SNLP in practical applications.

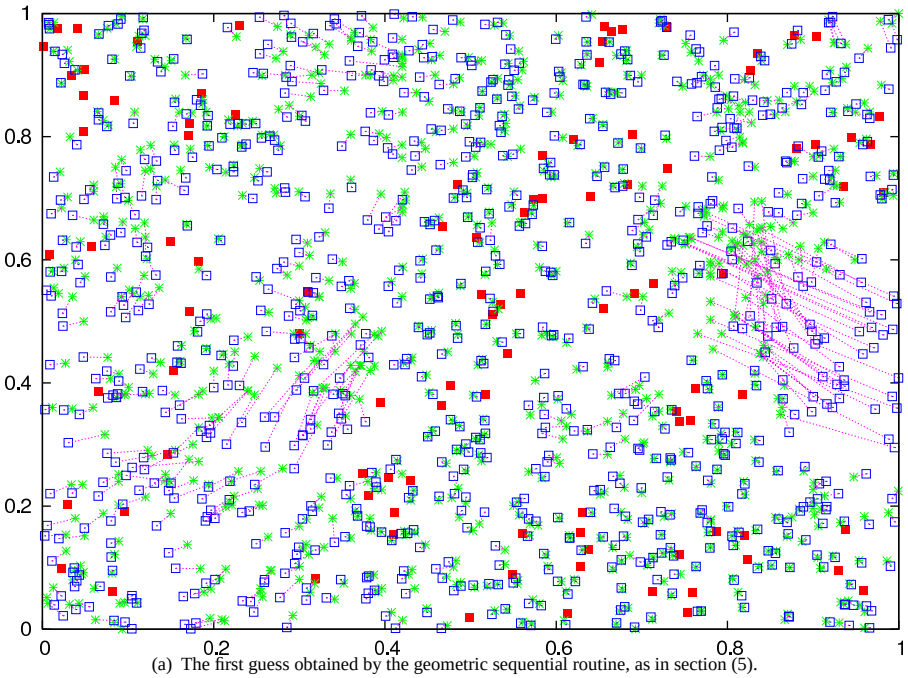
References

1. Alfakih, A.Y.: Graph rigidity via euclidean distance matrices. Linear Algebra and its Applications **310**, 149–165 (2000)
2. Alfakih, A.Y.: On the uniqueness of euclidean distance matrix completions. Linear Algebra and its Applications **370**, 1–14 (2003)
3. Alfakih, A.Y., Wolkowicz, H.: Euclidean distance matrices and the molecular conformation problem. Tech. Rep. CORR 2002-17, Department of combinatorics and optimization University of Waterloo Waterloo, Ontario Canada, Ontario Canada N2L 3G1 (2002)
4. Biswas, P., Liang, T.C., Wang, T.C., Ye, Y.: Semidefinite programming based algorithms for sensor network localization. ACM Transactions on Sensor Networks **2**(2), 188–220 (2006)
5. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Proceedings of the third international symposium on Information processing in sensor networks, Information Processing In Sensor Networks. ACM Press, New York, NY, USA (2004)
6. Biswas, P., Ye, Y., Wang, T.C., Liang, T.C.: Semidefinite programming approaches for sensor network localization with noisy distance measurements. IEEE transactions on automation science and engineering **3**(4), 360–371 (2006)
7. Boyd, S.P., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE/ACM Trans. Netw. **14**(SI), 2508–2530 (2006). DOI 10.1109/tit.2006.874516
8. Byrd, R.H., Nocedal, J., Lu, P., Zhu, C.: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. Tech. rep., Northwestern University, Department of Electrical Engineering and Computer Science (1995)
9. Carter, M.W., Jin, H.H., Saunders, M.A., Ye, Y.: Spaseloc: An adaptive subproblem algorithm for scalable wireless sensor network localization. SIAM Journal on Optimization **17**(4), 1102–1128 (2006)
10. Cassioli, A.: Global optimization of highly multimodal problems. Ph.D. thesis, DSI - Universita' degli Studi di Firenze (2009)
11. Cassioli, A., Sciandrone, M.: A convergent decomposition method for box-constrained optimization problems. Optimization Letters (2009). To appear
12. Chang, C., Hsu, C., Lin, C.: The analysis of decomposition methods for support vector machines. IEEE Transactions on Neural Networks **11**(4), 1003–1008 (2000)
13. Costa, J., Patwari, N., Hero III, A.O.: Distributed multidimensional scaling with adaptive weighting for node localization in sensor networks. ACM Trans. on Sensor Networks (2005)
14. Coxeter, H.S.M., Greitzer, S.L.: Geometry revisited. The Mathematical Association of America (1967)
15. Dattorro, J.: Convex Optimization & Euclidean Distance Geometry. Meboo Publishing USA (2005)

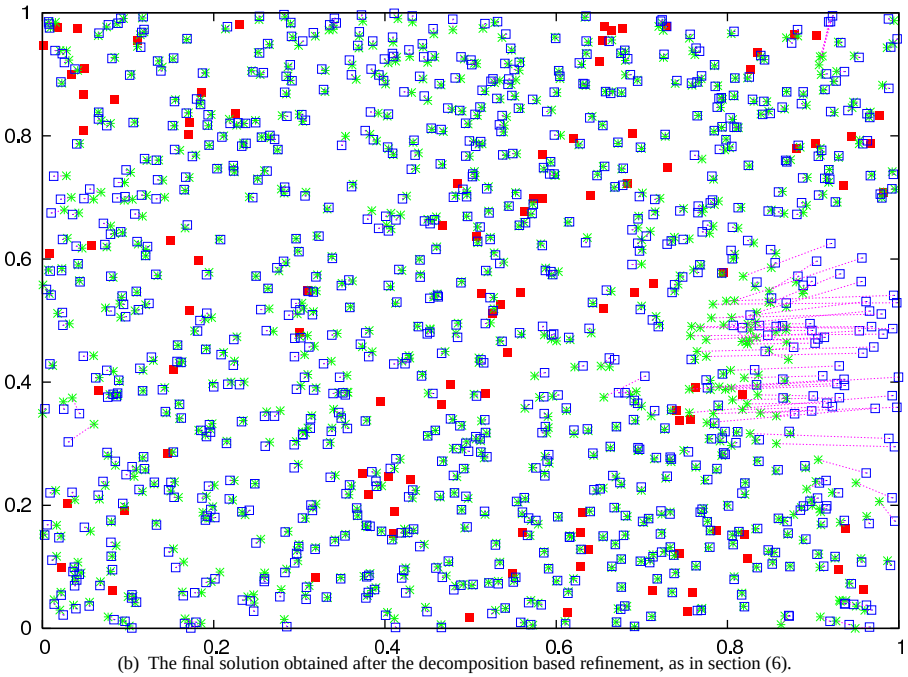
16. Ding, Y., Krislock, N., Qian, J., Wolkowicz, H.: Sensor network localization, euclidean distance matrix completions, and graph realization. Tech. rep., Optimization On Line (2006)
17. Doherty, L., Pister, K.S.J., El Ghaoui, L.: Convex position estimation in wireless sensor networks. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, *INFOCOM*, vol. 3 (2001)
18. Eren, T., Goldenberg, O.K., Whiteley, W., Yang, Y.R., Morse, A.S., Anderson, B.D.O., Belhumeur, P.N.: Rigidity, computation, and randomization in network localization. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies **4**, 2673–2684 vol.4 (2004)
19. Gentile, C.: Distributed sensor location through linear programming with triangle inequality constraints. Tech. rep., National Institute of Standards and Technology (2006)
20. Grippo, L., Sciandrone, M.: Nonmonotone globalization techniques for the barzilai-borwein gradient method. Computational Optimization and Applications **23**, 143–169 (2002)
21. Grosso, A., Locatelli, M., Schoen, F.: Solving molecular distance geometry problems by global optimization algorithms. Computational Optimization and Applications (2007). DOI 10.1007/s10589-007-9127-8
22. Jin, H.H.: Scalable sensor location algorithms for wireless sensor networks. Ph.D. thesis, University of Toronto (2005)
23. Johnson, C.R., Tarazaga, P.: Connections between the real positive semidefinite and distance matrix completion problems. Linear Algebra and its Applications **223/224**, 375–391 (1995)
24. Kim, S., Kojima, M., Waki, H.: Exploiting Sparsity in SDP Relaxation for Sensor Network Localization. SIAM Journal on Optimization **20**(1), 192 (2009). DOI 10.1137/080713380. URL `http://link.aip.org/link/?SJE/20/192/1`
25. Krislock, N., Wolkowicz, H.: Explicit sensor network localization using semidefinite representations and clique reductions. Research Report CORR 2009-04, University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario N2L 3G1, Canada (2009)
26. Laurent, M.: A connection between positive semidefinite and euclidean distance matrix completion problems. Linear Algebra and its applications **273**, 9–22 (1998)
27. Laurent, M.: Matrix completion problems. The Encyclopedia of Optimization **3**, 221–229 (2001)
28. Laurent, M.: Polynomial instances of the positive semidefinite and euclidean distance matrix completion problems. SIAM JOURNAL ON MATRIX ANALYSIS AND APPLICATIONS **22**(3), 874–894 (2001)
29. Liang, T.C., Wang, T.C., Ye, Y.: A gradient search method to round the semidenite programming relaxation solution for ad hoc wireless sensor network localization. Tech. Rep. SOL 2004-2, Department of Management Science and Engineering – Stanford University, Stanford, California 94305-4026 (2004)
30. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pp. 88–97. ACM, New York, NY, USA (2002). DOI 10.1145/570738.570751
31. Man-Cho So, A., Ye, Y.: Theory of semidefinite programming for sensor network localization. Mathematical Programming **108**(2), 367–384 (2007). DOI doi10.1007/s10107-006-0040-1
32. Morè, J.J., Wu, Z.: Smoothing techniques for macromolecular global optimization. In: G.D. Pillo, F. Gianessi (eds.) Nonlinear Optimization and Applications, pp. 297–312. Plenum Press (1996)
33. Morè, J.J., Wu., Z.: Global continuation for distance geometry problems. SIAM J. Optim. **7**, 814–836 (1997)
34. Morè, J.J., Wu, Z.: Issues in large scale global molecular optimization. In: L.T. Biegler, T.F. Coleman, A.R. Conn, F.N. Santosa (eds.) Large Scale Optimization with Applications: Part III: Molecular Structure and Optimization, pp. 99–121. Springer, New York (1997)
35. Nie, J.: Sum of squares method for sensor network localization. Computational Optimization and Applications (2007)
36. Savarese, C., Rabaey, J.M., Langendoen, K.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: ATEC '02: Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference, pp. 317–327. USENIX Association, Berkeley, CA, USA (2002)
37. Shang, Y., Ruml, W.: Improved mds-based localization. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies **4**, 2640–2651 vol.4 (2004). DOI 10.1109/infcom.2004.1354683
38. Simic, S., Sastry, S.: Distributed localization in wireless ad hoc networks. Memorandum UCB/ERL M02/26, UC Berkeley (2002)

39. Tseng, P.: Second-order cone programming relaxation of sensor network localization. Siam J. Optimization **18**(1), 156–185 (2007)
40. Wang, Z., Zheng, S., Boyd, S.P., Ye, Y.: Further relaxations of the sdp approach to sensor network localization. Tech. rep., Department of Electrical Engineering, Stanford University (2006)

Fig. 8 An example of the execution of the proposed algorithm for an instance of problem type 1. Empty squares are the actual sensor position (anchors use filled squares), stars represent the computed positions and the dashed lines connect them.



(a) The first guess obtained by the geometric sequential routine, as in section (5).



(b) The final solution obtained after the decomposition based refinement, as in section (6).