Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg,
Im Neuenheimer Feld 368, 69120 Heidelberg, Germany.

Reconstructing Sensor Locations from Distance Measurements Using the AIMMS Software Framework

Janick Frasch, Dennis Janka, Robert Kircheis
Advisor: Stefan Körkel

June 15, 2011

1 Introduction

Localization problems in sensor networks stem from a broad range of real-world problems such as civil protection, weather forecasting, and military applications. A large number of comparatively cheap to produce sensors, that can, in most cases for budgetary reasons, not be equipped with positioning systems like GPS, is distributed on a surface to measure a certain quantity over a larger area. In order to map the measurements correctly, one needs to reconstruct the sensors' positions from certain local and relative data, which in our case are distance measurements between sensors that are lying sufficiently close. Additionally a set of sensing units with known global coordinates, which we will refer to as *anchors*, is given. In order to ease the notation, by *sensors* we will only refer to those sensing units whose global positions are yet to be determined.

If we assume to have exact data, the underlying mathematical problem is a feasibility problem with, due to the Euclidean distance measurements, quadratic equality constraints. During the first sections of this report we will stick to this assumption, as well as we will only consider a small 2D localization problem in order to discuss the considered approaches exemplarily. In Sections 6.1-6.3 we will show how to extend these approaches for large-scale, noisy and 3D data in order to be able to cope with more general problems. Note however, that even for this rather simple looking 2D no-noise problem an underlying Phase-1 problem is non-convex and a general-purpose solver early due to local infeasibilities even though, at a global scope, a feasible solution exists.

2 Related Work

Several formulations and relaxations, in particular for 2D problems can be found in the literature. In [Tse07] Tseng proposed a second-order conic programmming (SOCP) re-

laxation, which will be explained and proposed for use in our context in Section 4.1. An even more promising and widespread approach is the relaxation of the sensor network localization problem (SNLP) to a semidefinite program (SDP), see, e.g., [BLWY06] and [DKQW10]. The idea behind this is to convert the non-convex quadratic distance constraints into convex constraints by introducing a relaxation to remove the quadratic term in the formulation. This relaxation has the appealing property of providing an exact solution in the case of unique solvability of the original problem [BLWY06]. These ideas are also extendable to noisy problem classes.

Unfortunately however, the current version of AIMMS (3.11) does not feature language elements to specify SDPs directly. Since reformulations of the semidefiniteness condition as regular (in-) equality constraints typically are quite cumbersome and/or require matrix factorizations (which are not available in AIMMS either), we disregard this approach for our purposes.

As typically for SNLPs global solutions are of major interest, one reasonable yet time consuming possibility would be to use a global NLP solver like Couenne [Bel09]. However again AIMMS does not feature an interface to global NLP solvers yet, so we also withdraw this option. Other options that were neglected include a subdivision of the considered x-y domain into cells using binary assignment variables, due to computational complexity.

3 General Problem Formulation

3.1 General NLP Formulations

We first show two ways of formulating the SNLP as a general nonlinear program (NLP) that can be solved by any general purpose NLP solver, however without taking any specific structures into account.

The easiest way to state the localization problem of n sensors $S := \{1, 2, \ldots, n\}$ in two dimensions while assuming to have exact distance measurements is the following: given m anchor locations $a_k \in \mathbb{R}^2$, $k \in A := \{1, 2, \ldots, m\}$ and some distance measurements $d_{ji}, (j,i) \in N_s \subset S \times S$ and $d_{jk}, (j,k) \in N_a \subset S \times A$, find $s_j \in \mathbb{R}^2$, $j \in S$, the location of n sensors, such that

$$\begin{aligned} \|s_j - s_i\|_2^2 &= d_{ji}^2, \quad \forall (j,i) \in N_s \\ \|s_j - a_k\|_2^2 &= d_{jk}^2, \quad \forall (j,k) \in N_a \end{aligned}$$

Or, expressed in NLP form:

$$\min_{s_j} \quad 0 \tag{1a}$$

$$\text{subject to}$$

$$\|s_j - s_i\|_2^2 = d_{ji}^2, \quad \forall (j,i) \in N_s \tag{1b}$$

$$\|s_j - a_k\|_2^2 = d_{jk}^2, \quad \forall (j,k) \in N_a \tag{1c}$$

Apparently, finding a feasible solution for this NLP would solve the (no-noise) SNLP. A phase-1 like problem, which can be solved to obtain a feasible solution for (1), can also be stated directly by introducing additional variables $q_{ji}, \tilde{q}_{ji}, (j,i) \in N_s$ and $q_{jk}, \tilde{q}_{jk}, (j,k) \in N_a$:

$$\min_{s_j, q_{ji}, \tilde{q}_{ji}, q_{jk}, \tilde{q}_{jk}} \quad \sum_{(j,i)\in N_s} \tilde{q}_{ji} + \sum_{(j,k)\in N_a} \tilde{q}_{jk} \tag{2a}$$

subject to

$$\|s_j - s_i\|_2^2 = d_{ji}^2 + q_{ji}, \quad \forall (j,i) \in N_s \tag{2b}$$

$$\|s_j - a_k\|_2^2 = d_{jk}^2 + q_{jk}, \quad \forall (j,k) \in N_a \tag{2c}$$

$$q_{ji} \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{2d}$$

$$-q_{ji} \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{2e}$$

$$q_{jk} \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{2f}$$

$$-q_{jk} \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{2g}$$

Note that this is a *non-convex quadratically-constrained* program which usually exhibits several local minima. As such, it is treatable in AIMMS 3.11 only by the available general purpose NLP solvers.

Note that from the problem statement additional constraints

$$\|s_j - s_i\|_2^2 \geq d_{ji}^2, \quad \forall (j,i) \notin N_s$$
$$\|s_j - a_k\|_2^2 \geq d_{jk}^2, \quad \forall (j,k) \notin N_a$$

can be derived, which however did not proof to be beneficial for our problem formulations.

Interestingly, Formulation (2) showed to perform much worse in tests than Formulation (1), even though it eases the inclusion fo additional structure. Seemingly, the internal phase-1 like procedure of commercial NLP solvers make use of more effective procedures trying to obtain feasible solutions, which annot be accessed by other problem formulations.

3.2 Performance Measures for SNLP

After having defined the 2D no-noise SNLP it is imminent that an optimal solution is characterized by fulfilling all given distances equations exactly. This however does not imply uniqueness of a solution. A sufficient condition here is that (a) for each sensor distances to at least three other (linearly independent) sensors or anchors are given and (b) each component in the graph of sensors and given distances as a whole is at least thrice adjacent (in terms of given distances) to each of at least three linearly independent anchors.

Two measures of performance suggest themselves. Firstly, from the algorithmic point of view, it seems reasonable to simply count the number of given distances that are satisfied by an assignment of sensor positions. We will mostly stick to this performance index here. However we propose to also take into consideration the number of sensors

with known correct positions for evaluating the quality of a solution. Whether a sensor is known correct or not could be checked in an analogous fashion to the uniqueness definition above, i.e., by checking whether it is part of a component that fulfills conditions (a) and (b) with respect to the graph induced by the satisfied distances (up to numerical errors).

When considering noisy data, i.e., error-prone distance measurements we have to relax the definition of a satisfied distance further by not only introducing a tolerance to make up for numerical errors, but also by considering a σ-dependent confidence interval around the expectation of the given distance measurements, where σ is the standard deviation.

4 Relaxed Formulations and Approximations

In the context of sensor localization only feasible, i.e., globally optimal solutions are desirable, as already the violation of few distances may allow major changes in the sensors' absolute positions while most of the relative distances are maintained. Therefore good initializations for the NLP variables are necessary to avoid the general-purpose NLP solver being stuck in local minima. To this end we present several relaxation-based and heuristic approaches.

4.1 SOCP Relaxation

Basing on a paper by Tseng [Tse07], we propose to solve a convex feasibility problem first by relaxing Constraints (1b) and (1c) to inequalities. The resulting program reads as

$$\min_{s_j} \quad 0 \tag{3a}$$

$$\text{subject to}$$

$$\|s_j - s_i\|_2^2 \leq d_{ji}^2, \quad \forall (j,i) \in N_s \tag{3b}$$

$$\|s_j - a_k\|_2^2 \leq d_{jk}^2, \quad \forall (j,k) \in N_a \tag{3c}$$

and is hence convex. Clearly (3) is a relaxation of (1). After solving (3) to feasibility, we can use the position assignments s_j of the solution to initialize problem (1) and improve feasibility for the original problem. Note that the sensors will lie in the convex hull of the anchors.

4.2 l_1 / l_∞ Approximation as LP

Another approach to get rid of the quadratic equality constraints in Program (1) is to approximate the Euclidean norm using different norms. In a first attempt we propose to use the l_1 norm. We yield the following optimization problem; note that we should not formulate this problem as a feasibility problem anymore, as in general most distance

constraints will be violated by any feasible solution to the original problem.

$$\min_{s_j,q_{ji},\tilde{q}_{ji},q_{jk},\tilde{q}_{jk}} \quad \sum_{(j,i)\in N_s} \tilde{q}_{ji} + \sum_{(j,k)\in N_a} \tilde{q}_{jk} \tag{4a}$$

subject to

$$\|s_j - s_i\|_1 = d_{ji} + q_{ji}, \quad \forall (j,i) \in N_s \tag{4b}$$
$$\|s_j - a_k\|_1 = d_{jk} + q_{jk}, \quad \forall (j,k) \in N_a \tag{4c}$$
$$|q_{ji}| \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{4d}$$
$$|q_{jk}| \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{4e}$$

For computational ease, Problem (4) can be relaxed to the following linear program (LP):

$$\min_{s_j,q_{ji},\tilde{q}_{ji},q_{jk},\tilde{q}_{jk},\tilde{d}^x_{ji},\tilde{d}^y_{ji},,\tilde{d}^x_{jk},\tilde{d}^y_{jk}} \quad \sum_{(j,i)\in N_s} \tilde{q}_{ji} + \sum_{(j,k)\in N_a} \tilde{q}_{jk} \tag{5a}$$

subject to

$$s^x_j - s^x_i \leq \tilde{d}^x_{ji}, \quad \forall (j,i) \in N_s \tag{5b}$$
$$s^x_i - s^x_j \leq \tilde{d}^x_{ji}, \quad \forall (j,i) \in N_s \tag{5c}$$
$$s^y_j - s^y_i \leq \tilde{d}^y_{ji}, \quad \forall (j,i) \in N_s \tag{5d}$$
$$s^y_i - s^y_j \leq \tilde{d}^y_{ji}, \quad \forall (j,i) \in N_s \tag{5e}$$
$$s^x_j - a^x_k \leq \tilde{d}^x_{jk}, \quad \forall (j,k) \in N_a \tag{5f}$$
$$a^x_k - s^x_j \leq \tilde{d}^x_{jk}, \quad \forall (j,k) \in N_a \tag{5g}$$
$$s^y_j - a^y_k \leq \tilde{d}^y_{jk}, \quad \forall (j,k) \in N_a \tag{5h}$$
$$a^y_k - s^y_j \leq \tilde{d}^y_{jk}, \quad \forall (j,k) \in N_a \tag{5i}$$
$$\tilde{d}^x_{ji} + \tilde{d}^y_{ji} = d_{ji} + q_{ji}, \quad \forall (j,i) \in N_s \tag{5j}$$
$$\tilde{d}^x_{jk} + \tilde{d}^y_{jk} = d_{jk} + q_{jk}, \quad \forall (j,k) \in N_a \tag{5k}$$
$$q_{ji} \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{5l}$$
$$-q_{ji} \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{5m}$$
$$q_{jk} \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{5n}$$
$$-q_{jk} \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{5o}$$

Inequalities (5b) through (5i) define the distance estimations in x and y direction. Equations (5j) and (5k) gather the slack between distance estimations and targets. Constraints (5l) through (5o) enable the objective function to drive the these slacks to zero in norm.

To yield an even better approximation of the l_2 norm in the original problem a combination of l_1 and l_∞ norm can be used, $\|x-y\|_2 \approx (\sqrt{2}-1)\|x-y\|_1 + (2-\sqrt{2})\|x-y\|_\infty =: \|x-y\|_{1,\infty}$. To motivate this approach, consider the shapes of the unit circles depicted in Figure 1. While $\|\cdot\|_{1,\infty}$ still is an underestimation for $\|x-y\|_2$, the approximation

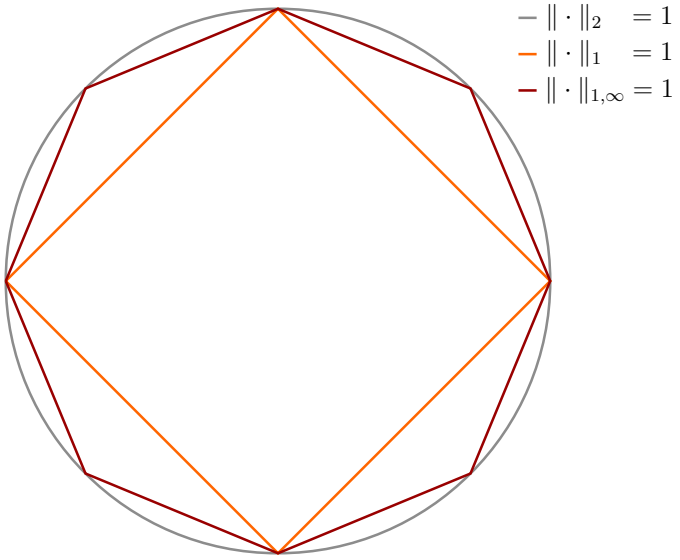

Figure 1: Unit circles of the l_1, l_2 and combined l_1-l_∞ norm

error is drastically reduced compared to using $\|\cdot\|_1$ alone. Defining $\alpha := 2 - \sqrt{2}$ we obtain the following mathematical program as an approximation to (1):

$$\min_{s_j, q_{ji}, \tilde{q}_{ji}, q_{jk}, \tilde{q}_{jk}, \tilde{d}^x_{ji}, \tilde{d}^y_{ji}, \tilde{d}^x_{jk}, \tilde{d}^y_{jk}, \tilde{d}^\infty_{ji}, \tilde{d}^\infty_{jk}} \quad \sum_{(j,i)\in N_s} \tilde{q}_{ji} + \sum_{(j,k)\in N_a} \tilde{q}_{jk} \tag{6a}$$

subject to

$$s^x_j - s^x_i \le \tilde{d}^x_{ji}, \quad \forall (j,i) \in N_s \tag{6b}$$
$$s^x_i - s^x_j \le \tilde{d}^x_{ji}, \quad \forall (j,i) \in N_s \tag{6c}$$
$$s^y_j - s^y_i \le \tilde{d}^y_{ji}, \quad \forall (j,i) \in N_s \tag{6d}$$
$$s^y_i - s^y_j \le \tilde{d}^y_{ji}, \quad \forall (j,i) \in N_s \tag{6e}$$
$$s^x_j - a^x_k \le \tilde{d}^x_{jk}, \quad \forall (j,k) \in N_a \tag{6f}$$
$$a^x_k - s^x_j \le \tilde{d}^x_{jk}, \quad \forall (j,k) \in N_a \tag{6g}$$
$$s^y_j - a^y_k \le \tilde{d}^y_{jk}, \quad \forall (j,k) \in N_a \tag{6h}$$
$$a^y_k - s^y_j \le \tilde{d}^y_{jk}, \quad \forall (j,k) \in N_a \tag{6i}$$
$$\tilde{d}^x_{ji} \le \tilde{d}^\infty_{ji}, \quad \forall (j,i) \in N_s \tag{6j}$$
$$\tilde{d}^y_{ji} \le \tilde{d}^\infty_{ji}, \quad \forall (j,i) \in N_s \tag{6k}$$
$$\tilde{d}^x_{jk} \le \tilde{d}^\infty_{jk}, \quad \forall (j,k) \in N_a \tag{6l}$$
$$\tilde{d}^y_{jk} \le \tilde{d}^\infty_{jk}, \quad \forall (j,k) \in N_a \tag{6m}$$

$$\alpha \cdot \tilde{d}^{\infty}_{ji} + (1-\alpha) \cdot (\tilde{d}^{x}_{ji} + \tilde{d}^{y}_{ji}) = d_{ji} + q_{ji}, \quad \forall (j,i) \in N_s \tag{6n}$$

$$\alpha \cdot \tilde{d}^{\infty}_{jk} + (1-\alpha) \cdot (\tilde{d}^{x}_{jk} + \tilde{d}^{y}_{jk}) = d_{jk} + q_{jk}, \quad \forall (j,k) \in N_a \tag{6o}$$

$$q_{ji} \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{6p}$$

$$-q_{ji} \leq \tilde{q}_{ji}, \quad \forall (j,i) \in N_s \tag{6q}$$

$$q_{jk} \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{6r}$$

$$-q_{jk} \leq \tilde{q}_{jk}, \quad \forall (j,k) \in N_a \tag{6s}$$

In addition to the constraints of LP (5), inequalities defining the l_∞ distance measurements are introduced by Constraints (6j) through (6m). Equalities (6n) and (6o) are updated accordingly to incorporate the l_∞ distance measurements in the definition of the slack variables $q_{\cdot,\cdot}$.

4.3 l_1 Approximation as MILP

The drawback in computing only LP-relaxations to the l_1 and $l_{1,\infty}$ norm approximations is the inexact estimation of the actually realized distances by (5b) through (5i). If $d_{\cdot,\cdot}$ are nonzero, at most one constraint of each pair (5b,5c), etc. can be active in any feasible solution for (5) or (6). However, in order to have exact estimates of the realized distances in x and y direction it would be desirable to have exactly one active distance, a requirement that cannot be formulated by means of linear programming. It even cannot be modeled by soft constraints, as punishing slack in Constraints (5b) through (5i) drives all sensors positions to one location, which counteracts the original objective of satisfying the given distances as tightly as possible.

One way to overcome this drawback is to formulate the activeness condition of exactly one of the mentioned constrains explicitly by means of mixed-integer linear programming (MILP). Therefor Constraints (5b) and (5c) are complemented by

$$s^x_j - s^x_i \geq \tilde{d}^x_{ji} + b^{(1)}_{ji} \cdot M, \quad \forall (j,i) \in N_s \tag{7a}$$

$$s^x_i - s^x_j \geq \tilde{d}^x_{ji} + b^{(2)}_{ji} \cdot M, \quad \forall (j,i) \in N_s \tag{7b}$$

$$b^{(1)}_{ji} + b^{(2)}_{ji} = 1, \quad \forall (j,i) \in N_s \tag{7c}$$

$$b^{(1)}_{ji}, b^{(2)}_{ji} \in \mathbb{B}, \quad \forall (j,i) \in N_s \tag{7d}$$

where M is a sufficiently large so called big-M constant. For all other pairs in Constraints (5b) through (5i), the complementing constraints are built analogously.

A major drawback of this method that has to be mentioned is the computational complexity of mixed-integer linear programming, which precludes the application of this approach to large-scale problems.

5 Fixing Certain Sensors

We developed a scheme that successively fixes certain sensors and regards them as pseudo-anchors. Afterwards, each of the emerging *reduced* SNLPs is solved with one

of the methods presented in this work.

5.1 Outline of the Algorithm

The idea of the algorithm is the following: If a sensor $\bar{s}$ is adjacent to two anchors, there exist exactly two possible locations for $\bar{s}$, given by two circle intersection points. If we fix $\bar{s}$ to one of them, further sensors can be fixed by trilateration, namely those that are adjacent to two anchors and to $\bar{s}$. We refer to them as *children* of $\bar{s}$ and to $\bar{s}$ as their *father*. Now we end up with two reduced SNLPs where $\bar{s}$ and its children are regarded as anchors. We can solve it now by one of the methods presented above.

This procedure can be iterated: We can successively take sensors that are adjacent to two anchors, fix them to one of the two circle intersection points, and fix all their children by trilateration. Of course, this leads to 2^n reduced SNLPs, where n is the number of fixed fathers. To limit the number of reduced SNLPs, one can define the maximum number of father nodes to be fixed. The algorithm then tries to fix those sensors with the largest number of children. Note that this is not always possible as some sensors may only become fathers if some sensors (with a possibly smaller number of children!) become fathers as well.

The algorithm is stated formally in two parts. We begin with a first sweep (see Algorithm 1) to determine the father/children-structure of the given problem. In the second run, Algorithm 2, we enumerate all possible choices of circle intersection points and solve the corresponding reduced SNLPs. To simplify the notation, we will denote all elements of the extended anchor sets $\bar{A}$ by $\bar{a}_i$. In this context, these can either be anchors or (temporarily) fixed sensors.

5.2 Properties

An obvious advantage of the algorithm is that it can be parallelized in a straightforward manner: For each choice of intersection points, the reduced SNLPs can be solved independently. A disadvantage is that the algorithm is quite prone to roundoff errors due to the computation of the intersection points which involves computing square roots, hence special care has to be taken when setting computation tolerances.

5.3 Some modifications

By making some easy modifications, one can avoid solving all 2^n reduced SNLPs: First of all, one can judge a solution independently from other solutions depending on the performance measure. In this case, one can terminate the algorithm as soon as an optimal solution with respect to this measure is found. If, for example the number of realized distances is used to measure the quality of a solution, one would terminate the algorithm as soon as a solution is found which realizes all given distances. Secondly, one can check before the solution of the reduced SNLP if all given distances between the fixed sensors are realized, as certain choices of intersection points may lead to contradictions. If this is the case, one can skip the solution of the reduced SNLP for this particular choice of intersection points and prune this branch.

$\bar{A} = \{a_1, \ldots, a_m\}$;
$\bar{S} = \{s_1, \ldots, s_n\}$;
repeat
  $\bar{A}' = \{\}$;
  for $i = 1$ **to** $|\bar{A}|$ **do**
    for $j = i$ **to** $|\bar{A}|$ **do**
      $E = \{s \in \bar{S} \mid s$ adjacent to $\bar{a}_i$ and $\bar{a}_j\}$;
      $\texttt{father}(s_k) = s_k, \quad k = 1, \ldots, |E|$;
      for $k = 1$ **to** $|E|$ **do**
        if $\texttt{father}(s_k) = s_k$ **then**
          Set $\texttt{father}(s_l) = s_k$ for all s_l adjacent to s_k
        end
      end
      $\bar{A}' = \bar{A}' \cup E$;
      $\bar{S} = \bar{S} \setminus E$;
    end
  end
  $\bar{A} = \bar{A} \cup \bar{A}'$;
until $\bar{A}' = \{\}$;

Algorithm 1: Determining the father/children structure of the problem

6 Extensions for More General Cases

So far all algorithmic ideas have been discussed exemplarily for the small 2D no noise benchmark problem of the 2011 MOPTA competition. In the following we discuss how these ideas can be transferred to be used for other, more general problem classes from this competition.

6.1 Extensions for Treating Large Problems

Going from small scale to large scale problems, the problem formulations do not need to be modified. However certain methods that are expensive in terms of computation time and/or memory consumption might not be reasonably applicable anymore. This particularly holds for the MILP formulation of the l_1 approximation.

One approach to overcome such problems is to split the large-scale problem into several smaller ones and compute solutions to the local problems for variable initialization before computing a few iterations of the full NLP problem. To split the large-scale problem we propose the following procedure: First a minimum distance spanning forest is computed in the full graph induced by the set of all anchors, using a Kruskal-like greedy algorithm that stops when only a certain number k_{MSF} components is left. Next, the closest anchor to each sensor (in terms of hops) is computed. Each component of the spanning tree together with those sensors whose closest anchor is in that component induce a subgraph of the large-scale problem. On these smaller problems solutions principally can be computed by any of the algorithms and procedures explained earlier. However,

Choose q "branching sensors" $\bar{s}_1, \ldots, \bar{s}_q$ such that
$\sum_{i=1}^{q} |\{s \in \bar{A} \mid \texttt{father}(s) = \bar{s}_i\}| \to \max$ and dependencies fulfilled;
$B = \{\bar{s}_1, \ldots, \bar{s}_q\}$;
`bestSolution` $= 0$;
repeat
  Pick choice of intersection points `choice`$(\bar{s}_k)$ $\forall \bar{s}_k \in B$;
  $\bar{A} = \{a_1, \ldots, a_m\} \cup B$;
  $\bar{S} = \{s_1, \ldots, s_n\} \setminus B$;
  for $k = 1$ **to** q **do**
    Recover elements $\bar{a}_i$ and $\bar{a}_j$ that lead to fixation of $\bar{s}_k$ from algorithm 1;
    $E = \{s \in \bar{S} \mid s$ adjacent to $\bar{a}_i$ and $\bar{a}_j\}$;
    Compute intersection point `choice`$(\bar{s}_k)$ and fix coordinates of $\bar{s}_k$ to this location;
    $E = E \setminus \{\bar{s}_k\}$;
    for $l = 1$ **to** $|E|$ **do**
      Compute position of sensor s_l by trilateration from positions of $\bar{a}_i$, $\bar{a}_j$ and $\bar{s}_k$ and fix coordinates of s_l to this location;
      $\bar{A} = \bar{A} \cup \{s_l\}$;
      $\bar{S} = \bar{S} \setminus \{s_l\}$;
    end
  end
  Solve reduced SNLP with anchor set $\bar{A}$ and sensor set $\bar{S}$ using one or several of the techniques presented above, obtain performance measure `currSolution` and corresponding positions s for all sensors;
  if `currSolution` " $>$ " `bestSolution` **then**
    `bestSolution` $=$ `currSolution`;
    `bestPos` $= s$
  end
until all `choices` enumerated ;
Terminate with `bestSolution`;

Algorithm 2: Enumerating and solving reduced SNLPs

in the current status of implementation, only the solution of generic NLP problems on the subgraphs is supported. In the GUI this method can be accessed by the button 'Small NLP'.

6.2 Extension to Noisy Data

Considering noisy instead of exact distance measurements, we cannot demand exact adherence of distances anymore, but rather aim to find a solution that keeps the deviation between the true, but unaccessible distances and the distances achieved by the solution small. To that end we compute means and standard deviations from given distance measurements. Due to lack of further information we assume the errors in the distance measurements to be normally distributed around the true distance. If two distance measurements are given between a pair of objects, we compute the expectation as the arithmetic mean and the standard deviation as the difference between either measurement and the mean (maximum-likelihood estimators). If only one distance measurement is available we choose this as the mean and assume a certain value for the standard deviation larger than all computed standard deviations. If no distance measures are given between a pair of sensors, no constraints are induced.

Following an idea presented in [BLWY06] we replace equality constraints (1b) and (1c) by confidence constraints

$$\underline{d}_{ji}^2 \leq \|s_j - s_i\|_2^2 \leq \overline{d}_{ji}^2, \quad \forall (j,i) \in N_s \tag{8a}$$

$$\underline{d}_{jk}^2 \leq \|s_j - a_k\|_2^2 \leq \overline{d}_{jk}^2, \quad \forall (j,k) \in N_a \tag{8b}$$

in the general formulation. This allows to retain the problem nature of a feasibility problem, which proofed to yield better solutions than the more intuitive formulation by introducing slack variables identically to Formulation (2).

Basing on the assumption of having normally distributed measurement errors we define $\underline{d}_{ji} := \hat{d}_{ji} - (c_{sol}+1)\cdot\sigma_{ji}$ and $\overline{d}_{ji} := \hat{d}_{ji} + (c_{sol}+1)\cdot\sigma_{ji}$, where $\hat{d}_{ji}$ is the expectation and σ_{ji} the standard deviation of the distance measurements. c_{sol} is a positive constant to be defined by the user. The definition for the sensor-anchor distances works analogously.

6.3 Extension to 3D Data

When considering 3D instead of 2D problems, the only major difference is that distance measures between two sensor/anchor objects need to additionally incorporate the difference in their z coordinate, which is parametric in the objects x and y coordinate. Therefore no additional variables or constraints are introduced in the general NLP and SOCP formulations (though some additional constraints in the l_1 and $l_{1,\infty}$ LP problems are needed). There are however problems with our enumeration heuristic: The intersection of two-spheres in 3D is a circle. If we intersect this circle with the manifold defining the surface profile it might be the case that there are more than two intersection points. These intersections points have to be computed by solving a nonlinear system of equations which in particular involves the parametrization of the manifold. The solution of

this system in the general case is more tedious than the solution if the manifold is flat (as in the 2D case).

The relaxations described in Section 4 can of course be applied to the 3D problems by simply disregarding the third coordinate and perform all the computations with the 2D projections of the 3D problem. We obtain 3D values just by lifting the computed 2D positions on the parametrized manifold. In fact, we obtained good results with this technique in our numerical experiments, see Section 8.

7 AIMMS Model Usage and User Interaction

On opening the submitted AIMMS project file the first user interaction page allows the user to choose the desired problem class. A click will redirect the user to a graphical interface corresponding to the chosen problem class (cf. Figure 2). The structure of these GUI pages is essentially as follows:

The top menu allows the user to reload all variables and parameters of the current problem class or to switch to a different problem class.

In the center of a page, a 2D graph plot shows the (projected) current sensor and anchor positions. Here, anchors are black circles and sensors are blue squares. If distance measurements are given between two objects an edge is drawn. It is colored green or red depending on whether the distance is satisfied by the current solution or not (up to a user defined equality tolerance to be introduced below). To the right of the 2D graph plot, the x and y coordinates of all sensor positions of the current solution are listed.

To the left of the 2D graph plot, the user can choose from a variety of algorithmic options to solve their problem instance. Depending on the chosen problem class several solving routines are available. The corresponding solution procedures are explained in Sections 3 through 5. The l_1, l_1/l_∞, l_1-MILP and SOCP procedures as well as the small NLP approach solve slightly modified problems derived from the desired one and hence are suggested to only be used to generate good initializations before using the NLP procedure to finalize the solution completion. Alternatively the n-best enumeration method explained in Section 5 can be used as initialization procedure where possible. The parameter n sets the number of fathers that will be fixed. The heuristic tries to choose good fathers with many children in order to reduce the variables in the reduced SNLPs as much as possible. Furthermore, the user can choose which method should be used to generate initial values for the reduced NLP. Similar the full SNLP case, we provide the following options: SOCP relaxation, l_1- and l_1/l_∞-relaxation. Additionally, there is the possibility to initialize all free sensor positions to zero or randomly on a circle around a neighbouring anchor. Note that in the worst case 2^n reduced SNLPs have to be solved.

Below the menu buttons for solution computation a tolerance or, in case of noise, a confidence level parameter can be set. Depending on this choice, the number of realized and not realized distances is shown below. The confidence level parameter c_{eq} defines a two-sided interval of distances $d \in [\hat{d}_{\cdot,\cdot} - c_{eq} \cdot \sigma_{\cdot,\cdot}, \hat{d}_{\cdot,\cdot} + c_{eq} \cdot \sigma_{\cdot,\cdot}]$ around the mean of the measured distances $\hat{d}_{\cdot,\cdot}$ (where $\sigma_{\cdot,\cdot}$ is the corresponding standard deviation) which

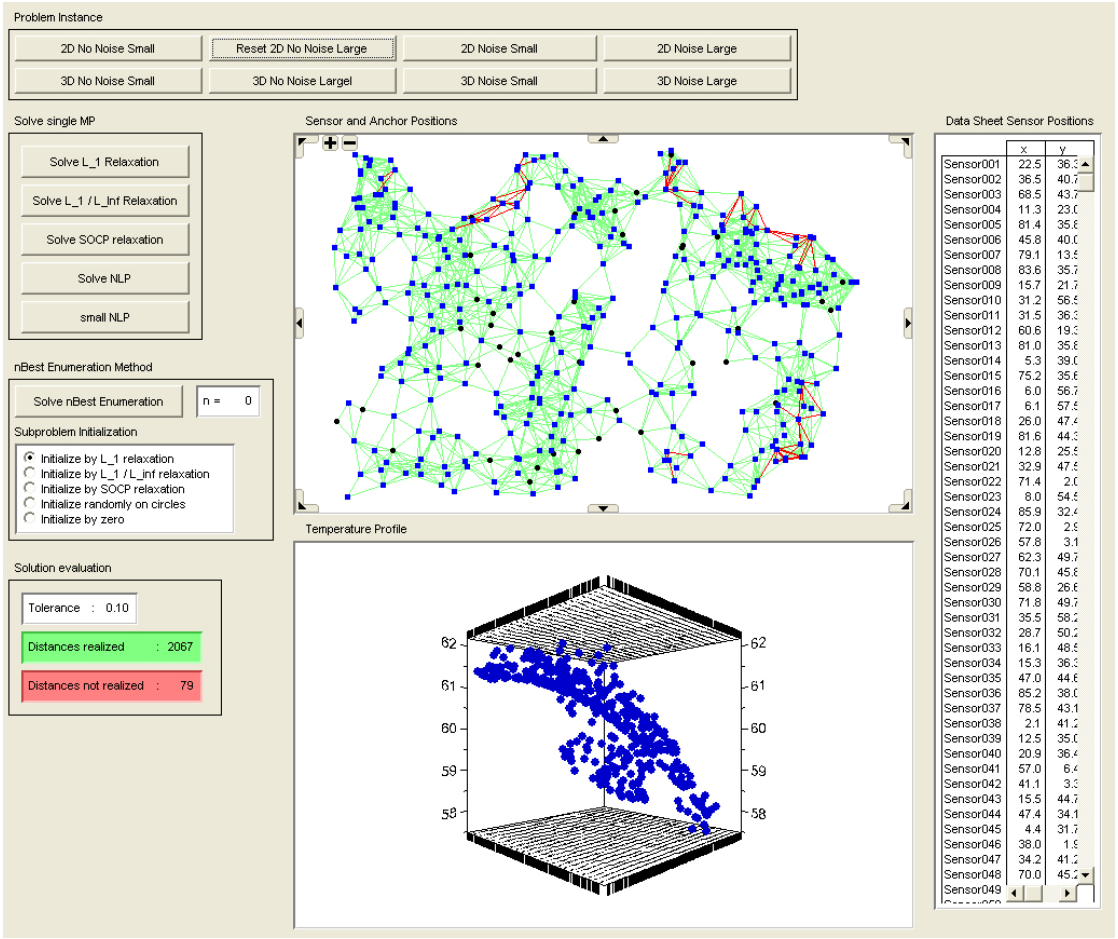

Figure 2: Screenshot of the graphical AIMMS user interface.

are considered to be satisfied. Additionally, in the case of noise, the standard deviation (SD) relaxation parameter c_{sol} introduced in Section 6.2 can be set.

In the lower part of the page, the distribution of the temperatures is drawn. If a 3D problem class is chosen, the surface profile of the corresponding problem instance is additionally drawn left of the temperature chart.

In order to being able to read the data given for the MOPTA 2011 competition into AIMMS a Perl file called 'convert2aimms.pl' is enclosed to our submission, which converts the given files into an AIMMS-conform format. For a given file '3D_NoNoise_Small_Distances.txt' for example, a new file '3D_NoNoise_Small_Distances_aimms.txt' is generated. Note that currently the file names for the external data sources unfortunately are hard coded in AIMMS in the procedure 'MainInitialization'.

8 Results

All solutions described in this section are available as *cases* within our AIMMS submission. Within the GUI, a graphical and textual representation of the results is provided. If not stated otherwise, we used MOSEK to solve LPs and QCPs and SNOPT to solve NLPs.

8.1 No Noise 2D

8.1.1 Small Problem

We could solve the small instance of the 2D problem without noise to optimality, i.e. all squared distances between the computed sensor positions matched the given squared distances up to a tolerance of 10^{-1}. We employed our enumeration heuristic with $n = 2$ and used the SOCP relaxation to generate initial values for the reduced NLPs. We used MOSEK as QCP solver with the default settings for the solution of the SOCP relaxation and SNOPT for the solution of the NLP. The feasibility tolerance of SNOPT should be reduced to 10^{-1} because of the fact that the distances as well as all position values have roundoff errors of the order of 10^{-3}. Overall, our enumeration heuristic works pretty well with this problem instance. By fixing only two sensors, the number of variable sensors could be reduced by 15, which is about one third of the overall number of sensors. The initialization of the reduced NLPs by the l_1- or the l_1/l_∞-relaxation yielded results of similar quality.

8.1.2 Large Problem

In the large problem instance, in our best solution 2070 distances could be realized, with 76 not realized (see Figure 3). The solution was obtained by solving the l_1/l_∞-approximation to generate initial values for the NLP and solving the NLP afterwards. The solver parameters were the same as in the small case. One can see that large areas are in fact identified correctly, the wrongly positioned sensors lie on the border or outside the convex hull of the anchors. The enumeration was not as good as in the small instance,

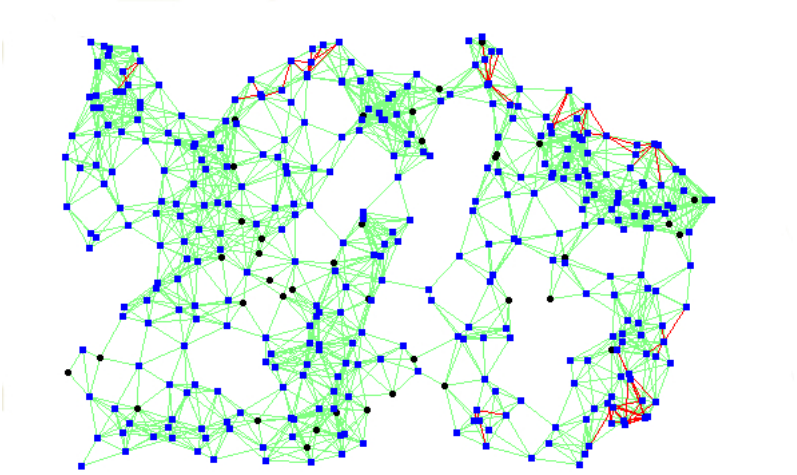

Figure 3: Best solution found for large 2D no noise problem. 2070 distances realized (green), 76 distances not realized (red). Sensors are drawn in blue, anchors in black.

probably due to the roundoff errors mentioned earlier. Furthermore, one would need to employ better heuristics to choose the branching sensors, e.g., try to locate them near anchors in the problematic area, in order to make this heuristic more successful.

8.2 Noise 2D

8.2.1 Small Problem

The best solution obtained was 264 distances realized within the 3σ-confidence interval and 20 distances where the difference was higher than 3 times the estimated standard deviation. The SOCP relaxation was used to generate initial values for the NLP, which is solved with a relaxation parameter $c_{sol} = 1.0$. In contrast to the no noise 2D large problem instance, the wrongly (within the scope of our error model!) identified sensors tend to lie more evenly spread in the interior of the convex hull of the anchors.

8.2.2 Large Problem

The best solution obtained was 1848 realized distances within the 3σ-confidence interval and 291 distances not realized within this confidence level. Again, the standard deviation relaxation parameter c_{sol} was set to 1.0. We used the SOCP relaxation to generate initial values for the NLP. Generally, the solution quality of the noisy problems is harder to assess than those of the problems without noise, especially because the statistical error model is not known.

8.3 No Noise 3D

8.3.1 Small Problem

As in the 2D case, the small no noise problem instance in 3D could be solved pretty well by our implementation. The best solution we found were 236 correctly realized distances and 4 not realized. Due to termination problems with MOSEK we chose CONOPT with the default parameters to solve the QCP resulting from projection of the SOCP relaxation to 2 dimensions. The resulting NLP was again solved by SNOPT.

8.3.2 Large Problem

Also the large problem without noise instance performed similarly than its 2D counterpart. The best solution was 1982 distances realized and 98 not realized. Again, CONOPT was employed to solve the (2D projected) SOCP relaxation, afterwards SNOPT for the NLP. Keep in mind that the enumeration technique is not yet available to yield meaningful results in the 3D case.

8.4 Noise 3D

8.4.1 Small Problem

In our best solution, we could realize 171 distances within the 3σ-confidence interval, with 38 distances not realized correctly. In our computations, the standard deviation parameter was again set to 1.0. Again we first solved the QCP for the SOCP relaxation and the feasibility NLP afterwards.

8.4.2 Large Problem

In this most general formulation, the solution quality of our methods could not reach the level of the other testcases. Only 1447 distances were realized within the 3σ-confidence level, 546 were not. Again, we chose the 2D projection of the SOCP relaxation with CONOPT as initialization for the NLP.

9 Discussion and Outlook

We developed a toolkit to treat sensor network localization problems in 2D and 3D, both with noisy and exact data. The problems without noise could be solved very well by our methods. The best solutions in these cases were obtained by using several techniques: An SOCP relaxation, a l_1/l_∞-relaxation, and an enumeration scheme to reduce the size and complexity of the problem.

In the noisy case, there are still some open questions how to treat the measurement errors adequately. Our formulation makes the most general assumptions on the statistical model of the measurement errors, with satisfactory results in the 2D case.

The 3D noisy case currently suffers from the fact that on the one hand many ideas were not yet implemented for noisy data and on the other hand they were not specifically

adapted for 3D data. The relaxation techniques should certainly take into account the structure of the given manifold instead of solving a 2D projection only. Further directions of research should also include a generalization of the enumeration scheme to 3D as well as to noisy data. The method could be expanded by considering further structures that can be fixed in this context, think, for example, a K_3 which is adjacent to three fixed components.

A large chapter which we have not considered so far is the integration of the temperature measurements to generate further information that can be helpful in the solution process, e.g., to generate further constraints, plausibility checks, etc.

As for the user interface it would be desirable to have more tools to visualize the quality of a solution, for example identify and visualize components of a solution that are fixed. Also, possibilities to assess the uncertainty of a solution – including the corresponding temperature distribution – dependent on different statistical models to be specified by the user can be thought of.

References

[Bel09] P. Belotti. Couenne: a user's manual. Technical report, Lehigh University, 2009.

[BLWY06] Pratik Biswas, Tzu-Chen Lian, Ta-Chung Wang, and Yinyu Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sen. Netw.*, 2:188–220, May 2006.

[DKQW10] Yichuan Ding, Nathan Krislock, Jiawei Qian, and Henry Wolkowicz. Sensor network localization, euclidean distance matrix completions, and graph realization. *Optimization and Engineering*, 11:45–66, 2010. 10.1007/s11081-008-9072-0.

[Tse07] P. Tseng. Second-cone programming relaxation of sensor network localization. *SIAM Journal of Optimization*, 18(2):156–185, 2007.