
A Distributed Method for Solving Semidefinite Programs Arising from Ad Hoc Wireless Sensor Network Localization

Pratik Biswas¹ and Yinyu Ye²

¹ Electrical Engineering, Stanford University, Stanford, CA 94305, USA.
pbiswas@stanford.edu

² Management Science and Engineering and, by courtesy, Electrical Engineering,
Stanford University, Stanford, CA 94305, USA. yinyu-ye@stanford.edu

Summary. We describe a distributed or decomposed semidefinite programming (SDP) method for solving Euclidean metric localization problems that arise from ad hoc wireless sensor networks. Using the distributed method, we can solve very large scale semidefinite programs which are intractable for the centralized methods. Our distributed or decomposed SDP scheme also seems to be applicable to solving other Euclidean geometry problems where points are locally connected.

1 Introduction

There has been an increase in the use of semidefinite programming (SDP) for solving wide range of Euclidean distance geometry problems, such as data compression, metric-space embedding, ball packing, chain folding etc. One application of the SDP Euclidean distance geometry model lies in ad hoc wireless sensor networks which are constructed for monitoring environmental information (temperature, sound levels, light etc) across an entire physical space. Typical networks of this type consist of a large number of densely deployed sensor nodes which gather local data and communicate with other nodes within a small range. The sensor data from these nodes are relevant only if we know what location they refer to. Therefore knowledge of each sensor position becomes imperative. Generally, the use of a GPS system is a very expensive solution to this requirement.

Indeed, other techniques to estimate node positions are being developed that rely just on the measurements of distances between neighboring nodes [BY04, BHE00, DGP01, GKW02, HB01, HMS01, NN01, SRL02, SHS01, SHS02, SRZ03]. The distance information could be based on criterion like time of arrival, angle of arrival and received signal strength. Depending on the accuracy of these measurements and processor, power and memory constraints at each of the nodes, there is some degree of error in the distance

information. Furthermore, it is assumed that we already know the positions of a few anchor nodes. The problem of finding the positions of all the nodes given a few anchor nodes and partial distance information between the nodes is called the position estimation or localization problem.

In particular, the paper [BY04] describes an SDP relaxation based model for the position estimation problem in sensor networks. The optimization problem is set up so as to minimize the error in the approximate distances between the sensors. Observable traces are developed to measure the quality of the distance data. The basic idea behind the technique is to convert the non-convex quadratic distance constraints into linear constraints by introducing relaxations to remove the quadratic term in the formulation. The performance of this technique is highly satisfactory compared to other techniques. Very few anchor nodes are required to accurately estimate the position of all the unknown sensors in a network. Also the estimation errors are minimal even when the anchor nodes are not suitably placed within the network. More importantly, for each sensor the model generates numerical data to measure the reliability and accuracy of the positions computed from the model, which can be used to detect erroneous or outlier sensors.

Unfortunately, the existing SDP solvers have very poor scalability. They can only handle SDP problems with the dimension and the number of constraints up to few thousands, where in the SDP sensor localization model the number of constraints is in the order of $O(n^2)$, where n is the number of sensors. The difficulty is that each iteration of interior-point algorithm SDP solvers needs to factorize and solve a dense matrix linear system whose dimension is the number of constraints. While we could solve localization problems with 50 sensors in few seconds, we have tried to use several off-the-shell codes to solve localization problems with 200 sensors and often these codes quit either due to memory shortage or having reached the maximum computation time.

In this report we describe an iterative distributed SDP computation scheme to overcome this difficulty. We first partition the anchors into many clusters according to their physical positions, and assign some sensors into these clusters if a sensor has a direct connection to one of the anchors. We then solve semidefinite programs *independently* at each cluster, and fix those sensors' positions which have high accuracy measures according the SDP computation. These positioned sensors become "ghost anchors" and are used to decide the remaining un-positioned sensors. The distributed scheme then repeats.

The distributed scheme is highly scalable and we have solved randomly generated sensor networks of 4,000 sensors in few minutes for a sequential implementation (that is, the cluster SDP problems are solved sequentially on a single processor), while the solution quality remains as good as that of using the centralized method for solving small networks. We remark that our distributed or decomposed computation scheme should be applicable to solving other Euclidean geometry problems where points are locally connected.

2 The Semidefinite Programming Model

We first present a quadratic programming formulation of the position estimation problem, then introduce its semidefinite programming model.

For simplicity, let the sensor points be placed on a plane. Recall that we have m known anchor points $a_k \in \mathcal{R}^2$, $k = 1, \dots, m$, and n unknown sensor points $x_j \in \mathcal{R}^2$, $j = 1, \dots, n$. For every pair of two points, we have a Euclidean distance measure if the two are within a communication distance range R . Therefore, say for $i, j \in N_1$, we are given Euclidean distance data \hat{d}_{ij} between unknown sensors i and j , and for $k, j \in N_2$ we know distance \hat{d}_{kj} between anchor k and sensor j . Note that for the rest of pairs we have only a lower bound R for their pair-wise distances. Therefore, the localization problem can be formulated as an error minimization problem with mixed equalities and inequalities:

$$\begin{aligned} & \text{minimize} \quad \sum_{i,j \in N_1, i < j} \alpha_{ij} + \sum_{k,j \in N_2} \alpha_{kj} \\ & \text{subject to} \quad \|x_i - x_j\|^2 = (\hat{d}_{ij})^2 + \alpha_{ij}, \quad \forall i, j \in N_1, i < j, \\ & \quad \|a_k - x_j\|^2 = (\hat{d}_{kj})^2 + \alpha_{kj}, \quad \text{for } k, j \in N_2, \\ & \quad \|x_i - x_j\|^2 \geq R^2, \quad \text{for the rest } i < j, \\ & \quad \|a_k - x_j\|^2 \geq R^2, \quad \text{for the rest } k, j, \\ & \quad \alpha_{ij} \geq 0, \quad \alpha_{kj} \geq 0. \end{aligned}$$

Let $X = [x_1 \ x_2 \ \dots \ x_n]$ be the $2 \times n$ matrix that needs to be determined. Then

$$\begin{aligned} \|x_i - x_j\|^2 &= e_{ij}^T X^T X e_{ij}, \\ \|a_i - x_j\|^2 &= (a_i; e_j)^T [I \ X]^T [I \ X] (a_i; e_j), \end{aligned}$$

where e_{ij} is the vector with 1 at the i th position, -1 at the j th position and zero everywhere else; and e_j is the vector of all zero except -1 at the j th position. Let $Y = X^T X$. Then the problem can be rewritten as:

$$\begin{aligned} & \text{minimize} \quad \sum_{i,j \in N_1, i < j} \alpha_{ij} + \sum_{k,j \in N_2} \alpha_{kj} \\ & \text{subject to} \quad e_{ij}^T Y e_{ij} = (\hat{d}_{ij})^2 + \alpha_{ij}, \quad \forall i < j \in N_1, \\ & \quad (a_k; e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; e_j) = (\hat{d}_{kj})^2 + \alpha_{kj}, \quad \forall k, j \in N_2, \\ & \quad e_{ij}^T Y e_{ij} \geq R^2, \quad \forall i < j \notin N_1, \\ & \quad (a_k; e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; e_j) \geq R^2, \quad \forall k, j \notin N_2, \\ & \quad Y = X^T X, \quad \alpha_{ij} \geq 0, \quad \alpha_{kj} \geq 0. \end{aligned} \tag{1}$$

Unfortunately, the above problem is not a convex optimization problem. Doherty et al. [DGP01] ignore the non-convex inequality constraints but keep the convex ones, resulting in a convex second-order cone optimization problem. A drawback of their technique is that all position estimations will lie in the convex hull of the known points. Others have essentially used various types of

nonlinear equation and optimization solvers to solve similar quadratic models, where final solutions are highly dependent on initial solutions and search directions.

Our method is to relax problem (1) to a semidefinite program:

$$\begin{aligned}
& \text{minimize} \quad \sum_{i,j \in N_1, i < j} \alpha_{ij} + \sum_{k,j \in N_2} \alpha_{kj} \\
& \text{subject to} \quad e_{ij}^T Y e_{ij} = (\hat{d}_{ij})^2 + \alpha_{ij}, \quad \forall i < j \in N_1, \\
& \quad (a_k; e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; e_j) = (\hat{d}_{kj})^2 + \alpha_{kj}, \quad \forall k, j \in N_2, \\
& \quad e_{ij}^T Y e_{ij} \geq R^2, \quad \forall i < j \notin N_1, \\
& \quad (a_k; e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; e_j) \geq R^2, \quad \forall k, j \notin N_2, \\
& \quad Y \succeq X^T X, \quad \alpha_{ij} \geq 0, \quad \alpha_{kj} \geq 0.
\end{aligned} \tag{2}$$

The last matrix inequality is equivalent to (Boyd et al. [BEF94])

$$\begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} \succeq 0.$$

Thus, the problem can be written as a standard SDP problem:

$$\begin{aligned}
& \text{minimize} \quad \sum_{i,j \in N_1, i < j} \alpha_{ij} + \sum_{k,j \in N_2} \alpha_{kj} \\
& \text{subject to} \quad (1; 0; \mathbf{0})^T Z (1; 0; \mathbf{0}) = 1 \\
& \quad (0; 1; \mathbf{0})^T Z (0; 1; \mathbf{0}) = 1 \\
& \quad (1; 1; \mathbf{0})^T Z (1; 1; \mathbf{0}) = 2 \\
& \quad (\mathbf{0}; e_{ij})^T Z (\mathbf{0}; e_{ij}) = (\hat{d}_{ij})^2 + \alpha_{ij}, \quad \forall i < j \in N_1, \\
& \quad (a_k; e_j)^T Z (a_k; e_j) = (\hat{d}_{kj})^2 + \alpha_{kj}, \quad \forall k, j \in N_2, \\
& \quad (\mathbf{0}; e_{ij})^T Z (\mathbf{0}; e_{ij}) \geq R^2, \quad \forall i < j \notin N_1, \\
& \quad (a_k; e_j)^T Z (a_k; e_j) \geq R^2, \quad \forall k, j \notin N_2, \\
& \quad Z \succeq 0, \quad \alpha_{ij} \geq 0, \quad \alpha_{kj} \geq 0.
\end{aligned} \tag{3}$$

The matrix of Z has $2n + n(n+1)/2$ unknown variables. If N_1 is sufficiently large and all distance measures are perfect, then there is an unique optimal \bar{Z} solution, with zero objective value, for (3). Moreover, in

$$\bar{Z} = \begin{pmatrix} I & \bar{X} \\ \bar{X}^T & \bar{Y} \end{pmatrix}, \tag{4}$$

we must have $\bar{Y} = (\bar{X})^T \bar{X}$ and \bar{X} equal true positions of the unknown sensors. That is, the SDP relaxation solves the original problem exactly. More precisely, we have the following theorem.

Theorem 1. *Let all distance measures \hat{d}_{ij} and \hat{d}_{kj} be perfect. Then, the minimal value of (3) is zero. Moreover, let N_1 and N_2 be the set of all unknown sensors and anchors, the number of anchors equal 3, and the left-hand matrix of linear equations of (3) has full rank. Then we must have*

$$\bar{Y} = (\bar{X})^T \bar{X}$$

in the optimal solution \bar{Z} of (3), i.e., \bar{Z} has rank 2, and \bar{X} represents the true positions of all unknown sensors.

Proof. Let X^* be the true position matrix of the n unknown points, and

$$Z^* = \begin{pmatrix} I & X^* \\ (X^*)^T & (X^*)^T X^* \end{pmatrix}.$$

Then Z^* is a feasible solution for (3) with all $\alpha_{ij} = 0$ and $\alpha_{jk} = 0$. But the objective value of (3) is greater than or equal to zero, so the minimal value of (3) must be zero.

If N_1 and N_2 contain all unknown sensors and anchors, that is, we have perfect distance relations between all sets of points, we have

$$|N_1| = n(n-1)/2 \quad \text{and} \quad |N_2| = 3n$$

which give $3n + n(n-1)/2$ of linear equations in (3). This number equals $2n + n(n+1)/2$, the total number of variable in matrix Z . Furthermore, if the left-hand constraint matrix of these linear equations is full rank, Z would be uniquely determined and it must be Z^* , so that $\bar{X} = X^*$ and $\bar{Y} = (X^*)^T X^* = \bar{X}^T \bar{X}$. \square

As discussed in [BY04], the condition to have a full rank constraint matrix is that the three anchors are not on a same line.

Generally, for imperfect information cases, we have $\bar{Y} - \bar{X}^T \bar{X} \succeq 0$. This inequality constitutes error analyses of the position estimation. For example,

$$\text{Trace}(\bar{Y} - \bar{X}^T \bar{X}) = \sum_{j=1}^n (\bar{Y}_{jj} - \|\bar{x}_j\|^2),$$

the total trace of the difference matrix, measures the efficiency and quality of distance data d_{ij} and d_{kj} . In particular, individual trace

$$\bar{Y}_{jj} - \|\bar{x}_j\|^2, \tag{5}$$

helps us to evaluate the position estimation, \bar{x}_j , for sensor j . The smaller the trace, the higher accuracy of the estimation.

3 A Distributed SDP Method

A round of the distributed computation method is straightforward and intuitive:

1. Partition the anchors into a number of clusters according to their geographical positions. In our implementation, we partition the entire sensor area into a number of equal-sized squares and those anchors in a same square form a regional cluster.

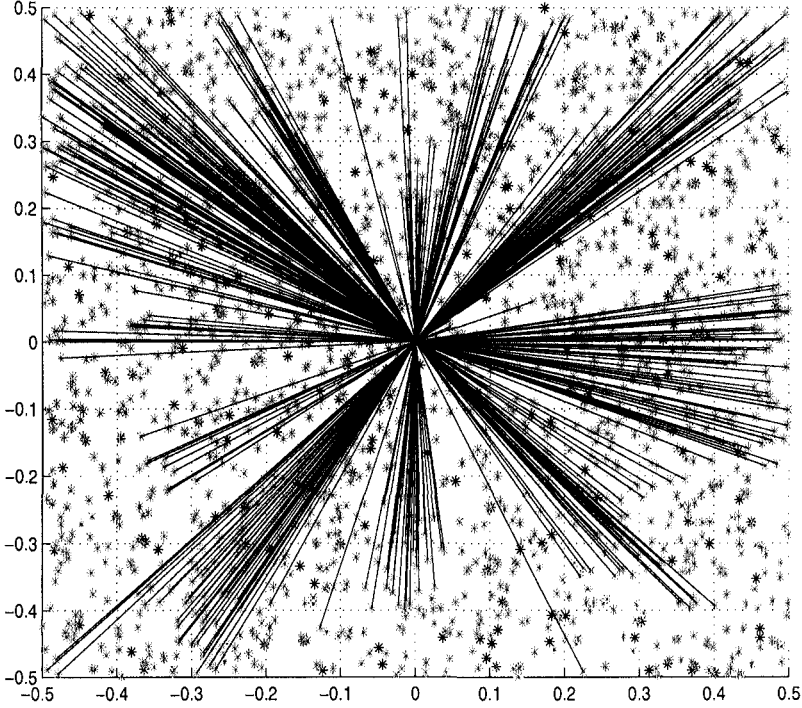


Fig. 1. First round position estimations for the 2,000 sensor network, noisy-factor=0, radio-range=.06, and the number of clusters=49.

2. Each (unpositioned) sensor sees if it has a direct connection to an anchor (within the communication range to an anchor). If it does, it becomes an unknown sensor point in the cluster to which the anchor belongs. Note that a sensor may be assigned into multiple clusters and some sensors are not assigned into any cluster.
3. For each cluster of anchors and unknown sensors, formulate the error minimization problem for that cluster, and solve the resulting SDP model if the number of anchors is more than 2. Typically, each cluster has less than 100 sensors and the model can be solved efficiently.
4. After solving each SDP model, check the individual trace (5) for each unknown sensor in the model. If it is below a predetermined small tolerance, label the sensor as *positioned* and its estimation \bar{x}_j becomes an “anchor”. If a sensor is assigned in multiple clusters, we choose the \bar{x}_j that has the smallest individual trace. This is done so as to choose the best estimation of the particular sensor from the estimations provided by solving the different clusters.
5. Consider positioned sensors as anchors and return to Step 1 to start the next round of estimation.

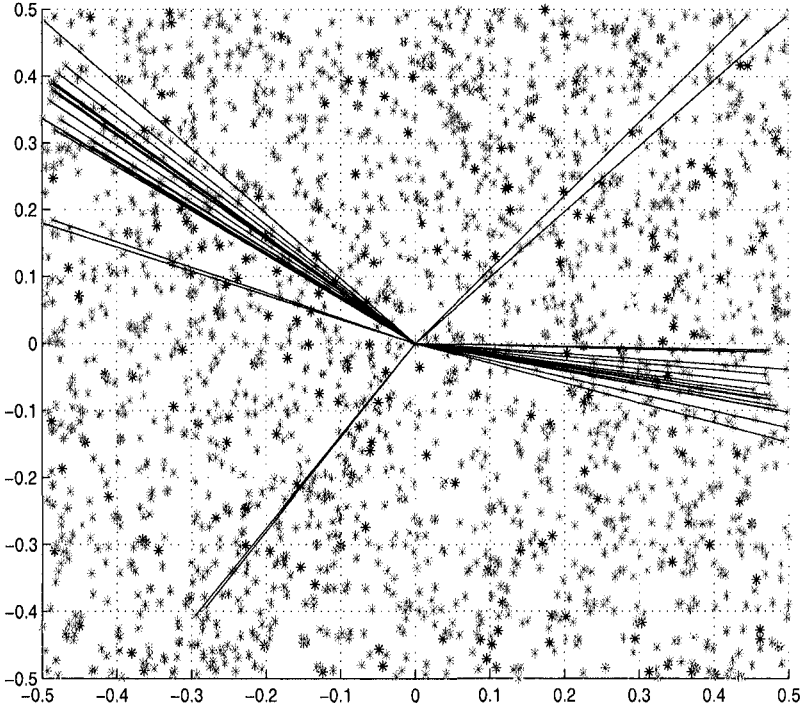


Fig. 2. Second round position estimations for the 2,000 sensor network, noisy-factor=0, radio-range=.06, and the number of clusters=49.

Note that the solution of the SDP problem in each cluster can be carried out at the cluster level so that the computation is highly distributive. The only information that needs to be passed among the neighboring clusters is which of the unknown sensors become positioned after a round of SDP solutions.

In solving the SDP model for each cluster, even if the number of sensors is below 100, the total number of constraints could be in the range of thousands. However, many of those "bounding away" constraints, i.e., the constraints between two remote points, are inactive or redundant at the optimal solution. Therefore, we adapt an iterative active constraint generation method. First, we solve the problem including only partial equality constraints and completely ignoring the bounding-away inequality constraints to obtain a solution. Secondly we verify the equality and inequality constraints and add those violated at the current solution into the model, and then resolve it with a "warm-start" solution. We can repeat this process until all of the constraints are satisfied. Typically, only about $O(n+m)$ constraints are active at the final solution so that the total number of constraints in the model can be controlled at $O(n+m)$.

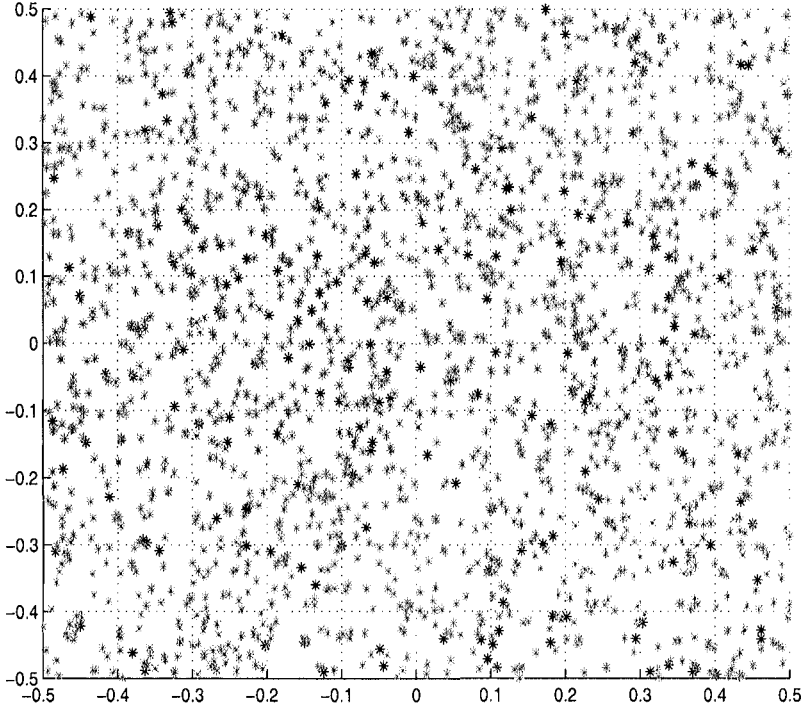


Fig. 3. Third and final round position estimations for the 2,000 sensor network, noisy-factor=0, radio-range=.06, and the number of clusters=49.

Two SDP codes are used in our method, the primal-dual homogeneous and self-dual interior-point algorithm SeDuMi of [Stu01] and the dual interior-point algorithm DSDP2.0 of [BYZ03]. Typically, DSDP2.0 is 2 to 3 times faster than SeDuMi, but SeDuMi is often more accurate and robust. DSDP is faster due to the fact that the sparse data structure of the problem is more suitable for DSDP.

4 Computational Results

Simulations were performed on networks of 2,000 to 4,000 sensor points which are randomly generated in a square region of $[-.5 \ .5] \times [-.5 \ .5]$ using $rand(2, n) - .5$ in MATLAB. The distance between two points is calculated as follows: If the distance is less than a given *radiorange* between $[0, 1]$, a random error was added to it

$$\hat{d}_{ij} = d_{ij} \cdot (1 + randn(1) \cdot noisyfactor),$$

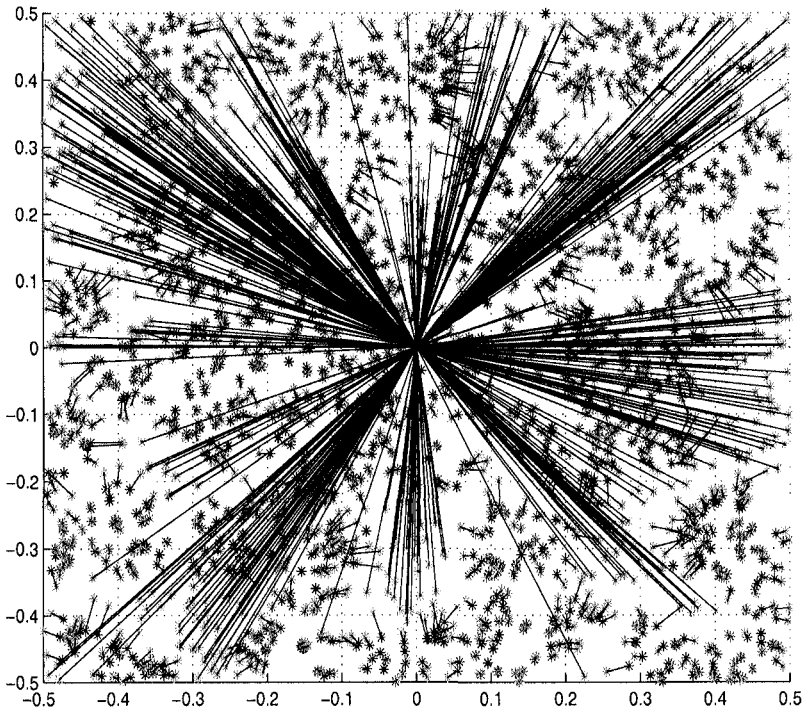


Fig. 4. First round position estimations for the 2,000 sensor network, noisy-factor=0.05, radio-range=.06, and the number of clusters=49.

where *noisyfactor* is a given number between $[0, 1]$. If the distance is beyond the given *radiorange*, no distance information is known to the algorithm except that it is greater than *radiorange*. We generally select the first 10% of the points as anchors, that is, anchors are also uniformly distributed in the same random manner. The tolerance for labeling a sensor as *positioned* is set at $0.01 \cdot (1 + \text{noisyfactor}) \cdot \text{radiorange}$.

Also the original true and the estimated sensor positions are plotted. The blue points refer to the positions of the anchors, green points to the true locations of the unknown sensors and red points to their estimated positions from the computation. The error offset between the true and estimated positions for an individual sensor is depicted by a solid blue line in each figure.

The first simulation is carried out for solving a network localization with 2,000 sensors, where the iterative distributed SDP method terminates in three rounds, see Figures 1, 2 and 3. When a sensor is not positioned, its estimation is typically at the origin. In this simulation, the entire sensor region is partitioned into 7×7 equal-sized squares, that is, 49 clusters, and the radio range is set at .06. The total solution time for the three round computation

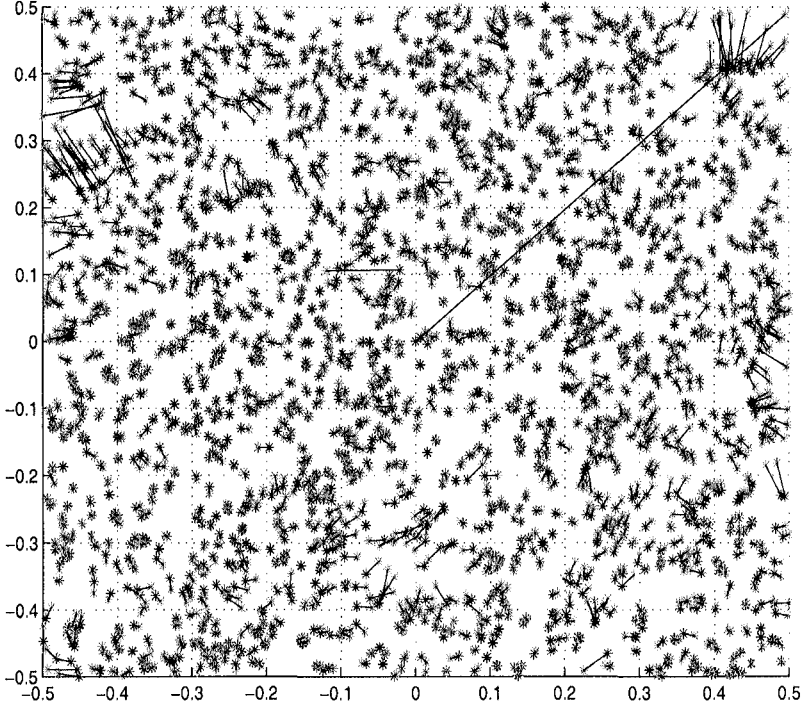


Fig. 5. Sixth round position estimations for the 2,000 sensor network, noisy-factor=0.05, radio-range=.06, and the number of clusters=49.

on a single Pentium 1.2 GHz and 500 MB PC, excluding the computation of \hat{d}_{ij} , is about two minutes.

As can be seen from the Figures 1, 2 and 3, it is usually the outlying sensors at the boundary or the sensors which do not have many anchors within the radio range that are not estimated in the initial stages of the method. Gradually, as the number of well estimated sensors or 'ghost' anchors grows, more and more of these points are estimated.

The second simulation is for solving the same network of 2,000 sensors and 49 clusters, but the distance d_{ij} is perturbed by a random noise either plus or minus, that is,

$$\hat{d}_{ij} = d_{ij} \cdot (1 + \text{randn}(1) * 0.05),$$

where $\text{randn}(1)$ is a standard normal random number. The iterative distributed SDP method terminates in thirteen rounds, see Figures 4, 5 and 6 for rounds 1, 6 and 13.

It is expected that the noisy cases will take more iterations since the number of 'ghost' anchors added at each iteration will be lower due to higher

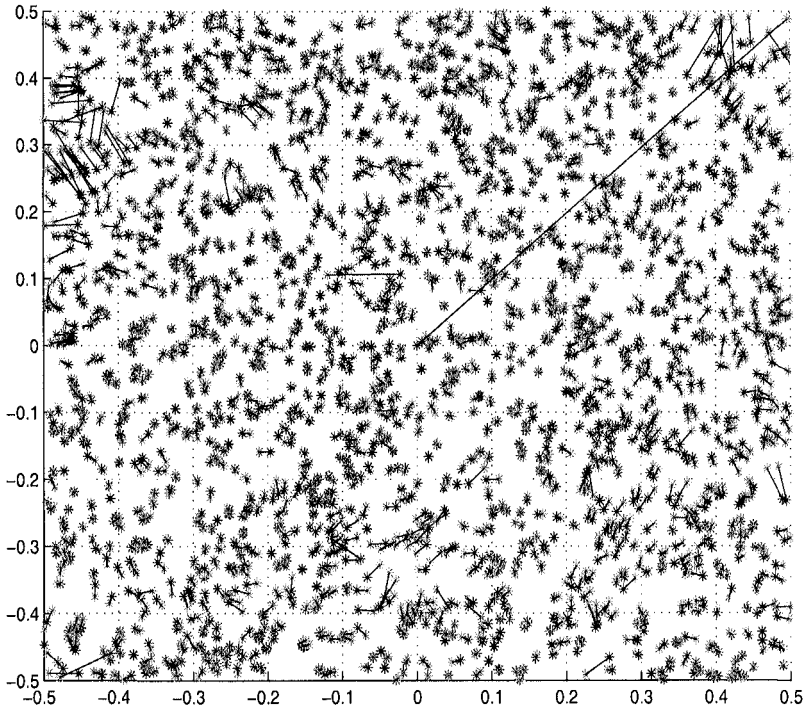


Fig. 6. Thirteenth and final round position estimations for the 2,000 sensor network, noisy-factor=0.05, radio-range=.06, and the number of clusters=49.

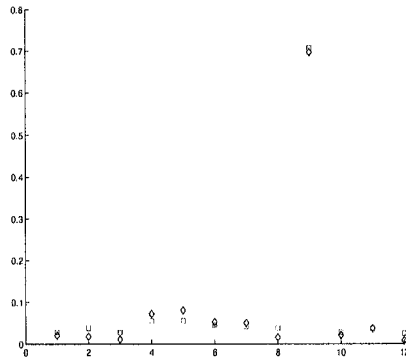


Fig. 7. Diamond: the offset distance between estimated and true positions, Square: the square root of individual trace (5) for the 2,000 sensor network.

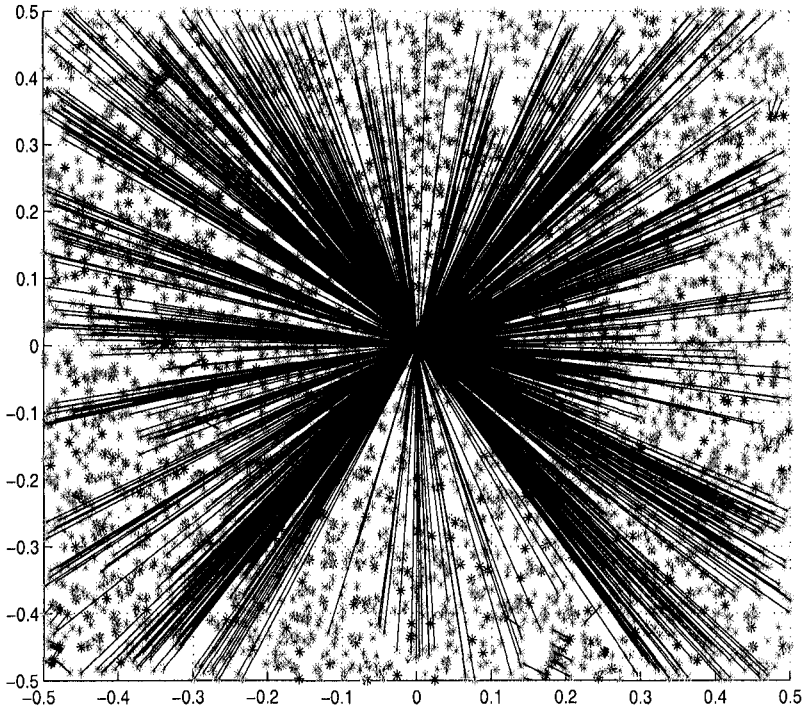


Fig. 8. First round position estimations in the 4,000 sensor network, noisy-factor=0, radio-range=.035, and the number of clusters=100.

errors in estimation. The final rounds mainly refine position estimations for a small number of sensors and each round runs in a few seconds.

Note that the estimation from the distributed method possesses good quality and there is no propagation of noisy errors. One sensor on the very up-right corner is unconnected to the network in this noisy data so that it is un-positioned at the final solution. Its individual trace also indicates this fact in the simulation, see Figure 7 for the correlation between individual error offset (blue diamond) and the square-root of trace (red square) for a few sensors whose trace is higher than $0.01 \cdot (1 + \text{noisyfactor}) \cdot \text{radiatorange}$ after the final round.

The third simulation solves a network localization with 4,000 sensors, where the iterative distributed SDP method terminates in five rounds, see Figures 8, 9, and 10 for rounds 1, 3 and 5. In this simulation, the entire sensor region is partitioned into 10×10 equal-sized squares, that is, 100 clusters, and the radio range is set at .035. The total solution time for the five round computation on the single Pentium 1.2 GHz and 500 MB PC, excluding computing \hat{d}_{ij} , is about four minutes.

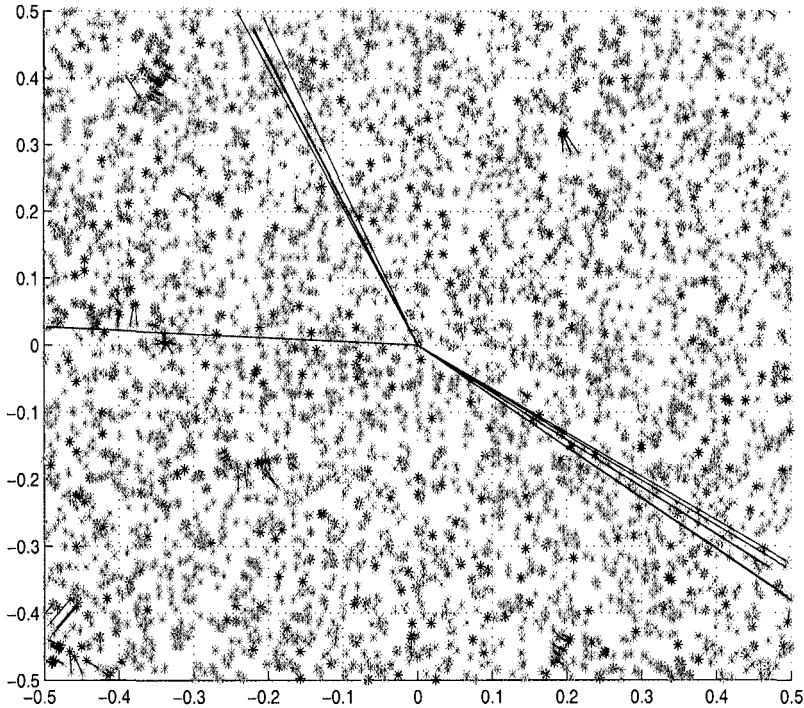


Fig. 9. Third round position estimations in the 4,000 sensor network, noisy-factor=0, radio-range=.035, and the number of clusters=100.

It is interesting to note that the erroneous points are concentrated within particular regions. This clearly indicates that the clustering approach prevents the propagation of errors to other clusters. Again, see Figure 11 for the correlation between individual error offset (blue diamond) and the square-root of trace (red square) for a few sensors whose trace is higher than $0.008 \cdot (1 + \text{noisyfactor}) \cdot \text{rاديorange}$ after the final round of the third simulation.

5 Work in Progress

The current clustering approach assumes that the anchor nodes are more or less uniformly distributed over the entire space. So by dividing the entire space into smaller sized square clusters, the number of anchors in each cluster is also more or less the same.

However this may or may not be the case in a real scenario. A better approach would be to create clusters more intelligently based on local connectivity information. Keeping this in mind, we try and find for each sensor

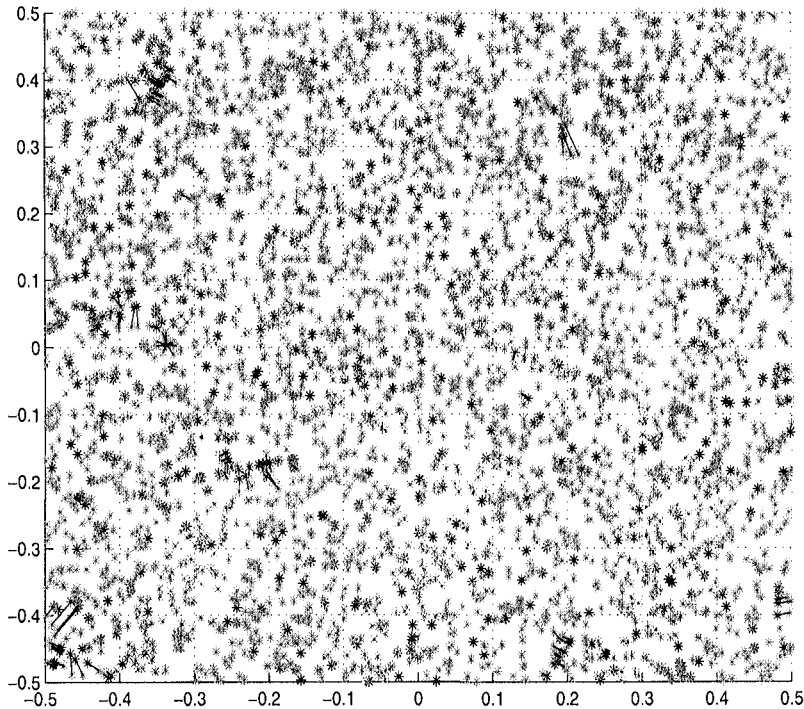


Fig. 10. Fifth and final round position estimations in the 4,000 sensor network, noisy-factor=0, radio-range=.035, and the number of clusters=100.

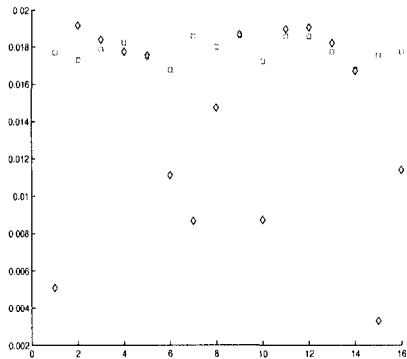


Fig. 11. Diamond: the offset distance between estimated and true positions, Square: the square root of individual trace (5) for the 4,000 sensor network.

its immediate neighborhood, that is, points within radio range of it. It can be said that such points are within one hop of each other. Higher degrees of connectivity between different points can also be evaluated by calculating the minimum number of hops between the 2 points. Using the hop information, we propose to construct clusters which are not necessarily of any particular geometric configuration but are defined by its connectivity with neighborhood points. Such clusters would yield much more efficient SDP models and faster and more accurate estimations.

6 Concluding Remarks

The distributed SDP approach solves with great accuracy and speed very large estimation problems which would otherwise be extremely time consuming in a centralized approach. Also due to smaller independent clusters, the noise or error propagation is quite limited as opposed to centralized algorithms.

In fact, the trace error (5) provides us with a very reliable measure of how accurate the estimation is and is used to discard estimations which may be very inaccurate as well as determining good estimations which may be used in future estimations.

This distributed algorithm is particularly relevant in the ad hoc network scenario where so much emphasis is given to decentralized computation schemes.

References

- [BYZ03] S. J. Benson, Y. Ye and X. Zhang. DSDP, <http://www-unix.mcs.anl.gov/benson/> or <http://www.stanford.edu/yye/Col.html>, 1998-2003.
- [BY98] D. Bertsimas and Y. Ye. Semidefinite relaxations, multivariate normal distributions, and order statistics. *Handbook of Combinatorial Optimization (Vol. 3)*, D.-Z. Du and P.M. Pardalos (Eds.) pp. 1-19, (1998 Kluwer Academic Publishers).
- [BY04] P. Biswas and Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. *Proc. IPSN04* (2004).
- [BEF94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. Linear Matrix Inequalities in System and Control Theory. *SIAM*, 1994.
- [BHE00] N. Bulusu, J. Heidemann, D. Estrin. GPS-less low cost outdoor localization for very small devices. *TR 00-729, Computer Science, University of Southern California*, April, 2000.
- [DGP01] L. Doherty, L. E. Ghaoui, and K. Pister. Convex position estimation in wireless sensor networks. *Proc. Infocom 2001*, Anchorage, AK, April 2001.
- [GKW02] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks. *UCLA/CSD-TR-02-0013, Computer Science, UCLA*, 2002.

- [HB01] J. Hightower and G. Boriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8) (2001) 57-66.
- [HMS01] A. Howard, M. J. Mataric, and G. S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS01)* (2001) 1055-1060.
- [NN01] D. Niculescu and B. Nath. Ad-hoc positioning system. In *IEEE GlobeCom*, Nov. 2001.
- [SRL02] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithm for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conf., Monterey, CA*, June 2002.
- [SHS01] A. Savvides, C. C. Han, and M. Srivastava. Dynamic fine-grained localization in ad hoc networks of sensors. In *ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MOBICON)*, July 2001.
- [SHS02] A. Savvides, H. Park, and M. Srivastava. The bits and flops of the n -hop multilateration primitive for node localization problems. In *1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, 112-121, Atlanta, 2002.
- [SRZ03] Y. Shang, W. Ruml, Y. Zhang and M. Fromherz. Localization From Mere Connectivity, *MobiHoc'03, Annapolis, Maryland*. June 2003,
- [Stu01] J. F. Sturm. Let SeDuMi seduce you, <http://fewcal.kub.nl/sturm/software/sedumi.html>, October 2001.
- [XY97] G.L. Xue and Y. Ye. An efficient algorithm for minimizing a sum of Euclidean norms with applications, *SIAM Journal on Optimization* 7 (1997) 1017-1036.

Multiscale Optimization Methods and Applications

Hager, W.W.; Huang, S.-J.; Pardalos, P.; Prokopyev, O.A.
(Eds.)

2006, XVII, 407 p., Hardcover

ISBN: 978-0-387-29549-7