Sensor Location through Linear Programming with Triangle Inequality Constraints

Camillo Gentile
National Institute of Standards and Technology
Wireless Communications Technologies Group
Email: camillo.gentile@nist.gov

***Abstract*— Interest in dense sensor networks due to falling price and reduced size has motivated research in sensor location in recent years. While many algorithms can be found in literature, no benchmark exists and most papers fail to compare their results to other competing algorithms. To our knowledge, the algorithm which achieves the best performance in sensor location uses semi-definite relaxation of a quadratic program to solve for sensor location. We propose solving the same program, however without relaxing the constraints, but rather transforming them into *linear* triangle inequality constraints. Our linear program ensures a tighter solution to the problem. We benchmark ours against the competing algorithm, and provide extensive experimentation to substantiate the robustness of our algorithm even in the presence of high levels of noise.**

I. Introduction

The falling price and reduced size of sensors in recent years have fueled the deployability of dense networks to monitor and relay environmental properties such as temperature, moisture, and light [1]. The ability to self-organize and find their locations autonomously and with high accuracy proves particularly useful in military and public safety operations. In dense networks, multilateration can render good location accuracy despite significant errors in range estimates between sensors. This has launched a research area known as sensor location which seeks to process potentially enormous quantities of data collectively to achieve optimal results.

A recent paper on sensor location [2] provides an exhaustive survey of the available techniques for sensor location. The techniques achieving the best performance process the input data in a centralized fashion; however in most, distributed versions are also available. Savvides et al. [3] solve a global non-linear optimization problem through Kalman filtering which yields good results but requires a high number of anchor nodes. The multi-dimensional scaling technique employed by Shang et al. [4] for sensor location achieves good results with few anchor nodes, but requires high connectivity. Savarese [5] also achieves good results by employing a two-phase initiation and refinement approach, and as opposed to the aforementioned papers deals with relatively high levels of noise.

To our knowledge, the two algorithms achieving the best performance in sensor location formulate a program with quadratic constraints to minimize a linear objective function [2], [6]. Since some of the constraints are non-convex, the papers differ primarily in their relaxation approaches to render the problem convex. The solution provided by Biswas et al. has greater applicability and yields better results than the one by Doherty et al. Our paper follows their same approach, maintaining the efficiency of convex optimization, however by applying *linear* triangle inequality constraints as opposed to quadratic ones, the problem is automatically convex. As a result, a tighter solution is guaranted since no relaxation of the constraints is needed.

The paper reads as follows. Section II states the general problem adopted from Biswas [2] that we attempt to solve for sensor location. Replacing the quadratic constraints with triangle inequality constraints in the subsequent section transforms the problem into a linear program. As the linear program does not directly yield the sensor locations from the solution, it necessitates reconstruction of the locations as described in Sections IV and V. An extensive number of challenging tests conditions are reported in Section VI to substantiate the robustness of our algorithm to high levels of noise in comparison to the algorithm proposed by Biswas. The last section provides conclusions and directions for further research.

II. Preliminaries

Consider a network with two types of nodes: n_A anchor nodes (or anchors) with known location and n_S sensor nodes (or sensors) with unknown location, for a total of $n = n_A + n_S$ nodes. For simplicity, let the nodes lie on a plane such that node i has location $\mathbf{x}_i \in \mathcal{R}^2$ indexed through $i,\ i = 1 \dots n_A$ for the anchors and $i = n_A + 1 \dots n$ for the sensors. The set N contains all pairs of (anchor, sensor) nodes and (sensor, sensor) nodes between which a link exists: $(i,j),\ i < j,\ j > n_A,\ ||\mathbf{x}_i - \mathbf{x}_j|| < R$, where $||\cdot||$ denotes the Euclidean distance and the network parameter R is known as the *radio range*. The complement set $\bar{N}$ contains all pairs of nodes between which no links exists: $(i,j),\ i < j,\ j > n_A,\ ||\mathbf{x}_i - \mathbf{x}_j|| \geq R$.

Neighboring nodes i and j measure the link distance $\hat{d}_{ij}$ between them through received-signal-strength or time-of-arrival techniques [7]. Given the locations of the anchor nodes and the measured distances between neighboring nodes in

the network, the general problem considered by both Biswas and Doherty to solve for the locations of the sensors $\mathbf{x}_i, i = n_A + 1 \ldots n$ follows:

$$\begin{array}{ll} \min & \sum_{(i,j)\in N} |\alpha_{ij}| \\ \text{s.t.} & ||\mathbf{x}_i - \mathbf{x}_j|| = d_{ij}, \quad \forall (i,j) \in N \\ & ||\mathbf{x}_i - \mathbf{x}_j|| \geq R, \quad \forall (i,j) \in \bar{N} \end{array} \tag{1}$$

where $d_{ij} = \hat{d}_{ij} + \alpha_{ij}$. The problem minimizes the sum of the absolute *residuals* α_{ij} between the estimated distances d_{ij} and the measured distances $\hat{d}_{ij}$.

The problem as stated above cannot be solved through convex optimization techniques since some of the constraints are non-convex. To overcome this obstacle, Doherty relaxes the problem by removing all the non-convex constraints, reducing it to a convex second-order cone optimization problem. This however limits the estimated locations of the sensor nodes to within the convex hull formed by the anchors. Biswas avoids this limitation not by removing the non-convex contraints, but rather by relaxing the problem to a semi-definite program. The solution to the program yields an expected value and an associated variance for the locations of the sensor nodes in the network.

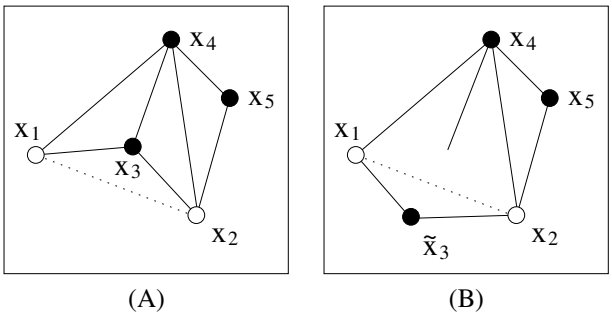

Fig. 1. Location reconstruction

III. Triangle inequality constraints

Rather than relaxing the constraints in (1), we propose applying a different set of geometrical constraints while maintaining the same objective functon. We exploit the triangular structure of the network such that the link distances conform to the triangle inequality constraints. The problem we solve can be stated as follows:

$$\begin{array}{l} \min \sum_{(i,j)\in N} |\alpha_{ij}| \\ \text{s.t.} \\ \left.\begin{array}{l} d_{ij} + d_{jk} \geq d_{ik} \\ d_{ij} + d_{ik} \geq d_{jk} \\ d_{jk} + d_{ik} \geq d_{ij} \end{array}\right\}, \left\{\begin{array}{l} \forall (i,j) \in N \\ \forall (j,k) \in N \\ \forall (i,k) \in N \end{array}\right. \end{array} \tag{2}$$

where $d_{ij} = \hat{d}_{ij} + \alpha_{ij}$. Rewriting the problem in standard form removes the absolute values and introduces bounding constraints:

$$\begin{array}{l} \min \sum_{(i,j)\in N} \alpha^+_{ij} + \alpha^-_{ij} \\ \text{s.t.} \\ \left.\begin{array}{l} d_{ij} + d_{jk} \geq d_{ik} \\ d_{ij} + d_{ik} \geq d_{jk} \\ d_{jk} + d_{ik} \geq d_{ij} \end{array}\right\}, \quad \left\{\begin{array}{l} \forall (i,j) \in N \\ \forall (j,k) \in N \\ \forall (i,k) \in N \end{array}\right. \\ \alpha^+_{ij} \geq 0, \quad \alpha^-_{ij} \geq 0, \qquad \forall (i,j) \in N \end{array} \tag{3}$$

where $d_{ij} = \hat{d}_{ij} + \alpha^+_{ij} - \alpha^-_{ij}$. The solution to the problem above does not directly yield the sensor locations as in (1), but only the values of the residuals of the link distances. Hence the complete algorithm requires an aposteriori *reconstruction* stage described in the following section to furnish the locations of the sensors from the residuals. Note that (3) can be applied to the triangles formed in three-dimensional networks as well. The reconstruction stage for such networks is not explained in this paper for the sake of brevity.

The advantage of our approach lies in the linearity of the constraints which ensures the convexity of the problem without relaxing any of the original constraints. In addition, the problem can be solved efficiently through linear programming as opposed to quadratic or semi-definite programming. Note that Whitehouse et al. [8] solve a similar problem to ours in the context of sensor calibration.

IV. Location reconstruction

A. Location propagation

The reconstruction stage yields a unique solution for the locations of the sensor nodes from the estimated distances between neighboring nodes. The stage begins from any two anchor nodes sharing a common neighboring sensor. Take the network in Fig. 1A as an example. The locations of the two anchors $\mathbf{x}_1$ and $\mathbf{x}_2$ are known (and so the distance d_{12} between them) and the distances d_{13} and d_{23} to unknown $\mathbf{x}_3$ are furnished through the solution to (3). Given these four data, [9] provides the set of equations necessary to compute $\mathbf{x}_3$ through the law of cosines. We say that the anchor nodes *propagate* their locations to the unknown sensor node.

Now that $\mathbf{x}_3$ is known, triangle $\triangle_{123}$ becomes a *known triangle*. A known triangle has the following two properties:

1) *node property*: all three of its nodes are known
2) *link property*: all three links exist between its three nodes

Such a triangle serves to propagate the locations of its nodes to unknown sensor nodes connected to it in the exact same manner as the anchors 1 and 2 propagated their known locations to sensor 3. The location of unknown sensor $\mathbf{x}_4$ can be found from its connections to $\triangle_{123}$: the four data required are the now known locations $\mathbf{x}_2$ and $\mathbf{x}_3$ (and so the distance d_{23} between them) and the distances d_{24} and d_{34} to $\mathbf{x}_4$. Alternatively, the location $\mathbf{x}_4$ can be found still through its connections to known triangle $\triangle_{123}$, but exploiting instead the four data: $\mathbf{x}_1$, $\mathbf{x}_3$, d_{14}, and d_{34}. Once $\mathbf{x}_4$ is found through $\triangle_{234}$ (or alternatively $\triangle_{134}$), in the same manner $\mathbf{x}_5$ can be

found through $\mathbf{x}_2$, $\mathbf{x}_4$, d_{25}, and d_{45}. Through propagation in this manner, all the sensor nodes in the network can become known.

B. The voting scheme

Given the four data $\mathbf{x}_1$, $\mathbf{x}_2$, d_{23}, and d_{13}, the set of equations for the law of cosines actually furnishes two candidate locations $\mathbf{x}_3$ and $\tilde{\mathbf{x}}_3$ mirrored about the line between $\mathbf{x}_1$ and $\mathbf{x}_2$. The solution for $\tilde{\mathbf{x}}_3$ appears in Fig. 1B. As shown, this solution is inconsistent with the locations of the other nodes displayed in the network since the distance between $\tilde{\mathbf{x}}_3$ and $\mathbf{x}_4$ is greater than the value d_{34} given through the linear program. The voting scheme described in [10] collectively synthesizes the estimated distances between the sensor nodes in the network to uniquely determine their locations.

If the network contains only two anchor nodes as in Fig. 1, two solutions exist mirrored about the line between the two nodes. Here the mirror solution consistent with $\tilde{\mathbf{x}}_3$ also has $\tilde{\mathbf{x}}_4$ and $\tilde{\mathbf{x}}_5$ mirrored about the line between $\mathbf{x}_1$ and $\mathbf{x}_2$ (i.e. a mirror image of Fig. 1A). Therefore a network requires at least three anchor nodes to yield a unique solution.

V. Artificial links

This section describes the three cases in which the normal links between neighboring nodes considered thus far call for supplemental non-existent links, or *artificial links*, to enable location reconstruction of the sensor nodes for all network topologies.

A.

Certain topologies of the network halt location propagation to some sensor nodes. In order to isolate these nodes, the network topology alone enables propagating location from all pairs of anchor nodes before actually solving for the link distances in (3). If not all sensor nodes can be found, artificial links are necessary. Take Fig. 2A as an example (normal and artificial links appear as dark and light lines respectively): the locations of sensor nodes $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ are known through other nodes in the network, but $\mathbf{x}_4$ cannot be found since it has no connections to a known triangle. If $\triangle_{123}$ were known, then $\mathbf{x}_4$ could be found through its connections to d_{24} and d_{34} to $\mathbf{x}_2$ and $\mathbf{x}_3$ respectively. Inserting the artificial link between $\mathbf{x}_2$ and $\mathbf{x}_3$ satisfies the link property of a known triangle rendering $\triangle_{123}$ known, and in turn $\mathbf{x}_4$ as well.

The artificial link between $\mathbf{x}_i$ and $\mathbf{x}_j$ generates a set of new triangles in the network, and so introduces a corresponding set of triangle inequality constraints in the linear program (3). Each artificial link also generates an additional bounding constraint in the problem for the positive residual α_{ij}^+, stemming from the given $d_{ij} = \hat{d}_{ij} + \alpha_{ij}^+ \geq R$. If we set $d_{ij} = R$, the positive residual appears in the bounding contraints as $\alpha_{ij}^+ \geq 0$, but not in the objective function since we have no real estimate on its value, and so no motivation to minimize it. Since we do not attempt to minimize α_{ij}^+, aritificial links are less restrained than normal links.

B.

Another case necessitating artificial links is for nodes with only one connection to the network, as shown in Fig. 2B. The locations of sensor nodes $\mathbf{x}_1$ through $\mathbf{x}_4$ are known, but not that of sensor 5 which connects to the network only through sensor 4. Since $\mathbf{x}_4$ is a node common to known triangle $\triangle_{134}$, inserting the artificial link between $\mathbf{x}_5$ and a node of $\triangle_{134}$ other than $\mathbf{x}_4$, such as $\mathbf{x}_1$, makes $\mathbf{x}_5$ known through the four data $\mathbf{x}_1, \mathbf{x}_4, d_{15}$, and d_{45}.

While the single artificial link between $\mathbf{x}_5$ and $\mathbf{x}_1$ suffices to make $\mathbf{x}_5$ known, as mentioned previously since artificial links are individually less restrained than normal links, adding multiple artificial links guarantees a tighter solution to the problem; hence we also add artificial links between $\mathbf{x}_5$ and $\mathbf{x}_2$ and between $\mathbf{x}_5$ and $\mathbf{x}_3$, since $\mathbf{x}_2$ and $\mathbf{x}_3$ are also nodes of known triangle $\triangle_{234}$ common to $\mathbf{x}_4$.

C.

An anchor node in the network can be incorporated in the linear program (3) by including the constraints associated with all triangles between it, another anchor node, and a sensor node neighboring both anchors, and setting the residual of the link distance between the anchor pair to zero since this distance is known.

In certain network layouts, an anchor node cannot be included in the program because no sensor node neighbors both it and another anchor node. Artificial links are necessary in this case. Take Fig. 2C as an example. None of the sensors 3, 4, or 5 neighbor both anchors 1 and 2. In order to incorporate in the program the known link distance d_{12} between the two anchor nodes, artificial link can be added between $\mathbf{x}_2$ and $\mathbf{x}_3$ such that $\triangle_{123}$ appears in the constraints. Alternatively, the artificial link between $\mathbf{x}_1$ and $\mathbf{x}_5$ can be added such that $\triangle_{125}$ appears in the constraints. As in the case for singly-connected sensor nodes, since artificial links are individually less restrained than normal links, adding multiple artificial links guarantees a tighter solution to the problem; hence the artificial links between $\mathbf{x}_1$ and $\mathbf{x}_4$, between $\mathbf{x}_3$ and $\mathbf{x}_5$, and between $\mathbf{x}_2$ and $\mathbf{x}_4$ are also added in the problem to include intermediate triangles $\triangle_{134}$, $\triangle_{345}$, and $\triangle_{245}$ between $\mathbf{x}_1$ and $\mathbf{x}_2$. We found that constraints for triangles such as $\triangle_{124}$ with more than one artificial link in the problem are too loose to improve the solution, and so only increase computation.

If a node is completely disconnected from the network we have no information about it except that it lies beyond radio range R from all other nodes in the network, hence there is no deterministic manner to compute its location with any meaningful accuracy.

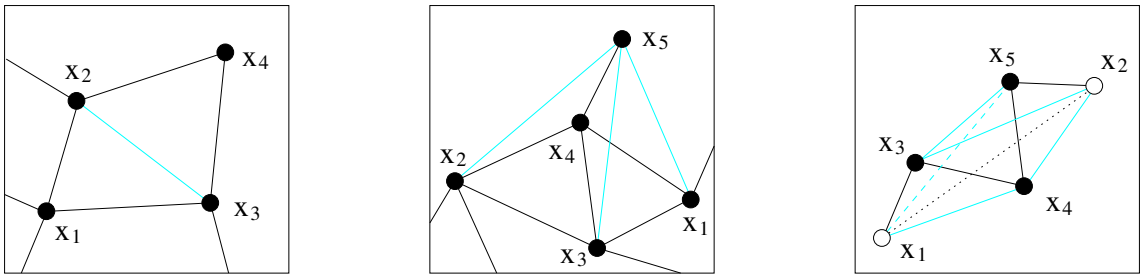

Fig. 2. Artificial Links

VI. Experimental setup and results

In order to quantify the performance of our algorithm in comparison to Biswas, we conduct experiments on a network with the same structure. The network contains 50 sensor nodes uniformly distributed throughout a one by one unit area. The three varying parameters are the number of anchor nodes, the radio range, and the noisy factor of the link distances. As Biswas, the ground-truth link distances $\bar{d}_{ij}$ between neighboring nodes i and j are perturbed with zero-mean unit-variance Gaussian noise $\mathcal{N}(0,1)$ and the varying parameter $noise$. So the algorithm accepts as input the noisy link distances $\hat{d}_{ij} = \bar{d}_{ij} * (1 + \mathcal{N}(0,1) * noise)$.

Figure 3 illustrates an example network with three anchors, $R = 0.25$, and $noise = 0.1$. The anchors and sensors appear as dark and light asterisks respectively, and the normal and artificial links as dark and light lines between neighboring nodes. The network contains 211 normal links for an average node connectivity of 7.9623, and 14 artificial links. Once the linear program has been solved, the algorithm yields the estimated locations of the sensors through the reconstruction stage. The true and estimated locations appear in Figure 4 as dark and light asterisks connected by an error line. The average location error is 0.0597.

The computational complexity of the simplex algorithm typically varies as $\mathcal{O}(\#constraints)$ [11]. The sparsity of our constraint matrix allows for more efficient algorithms than the simplex to solve our linear program, and so its expected complexity should not exceed $\mathcal{O}(\#constraints)$. The upper bound for the number of constraints coincides with a fully-connected network, where the number of triangles is $\binom{n}{3}$, and each one introduces three constraints for a total $\#constraints = \frac{n(n-1)(n-2)}{2}$ and complexity $\mathcal{O}(n^3)$. The observed complexity for up to $R = 0.40$ is typically much less. $MATLAB$ solved the linear program described above with 1692 contraints using an interior-point algorithm in less than one second on a 1GHz Pentium IV processor.

Biswas reports the results of a single trial network for the six test conditions described in [2]. The quantative measure for each test condition is the average location error over the sensor nodes

$$\frac{1}{n_S} \sum_{i=n_A+1}^{n} ||\bar{\mathbf{x}}_i - \mathbf{x}_i||, \tag{4}$$

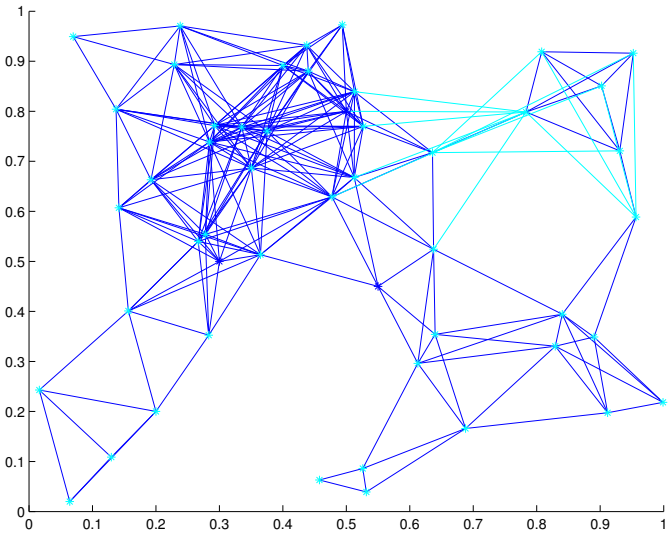

Fig. 3. Link distance estimation

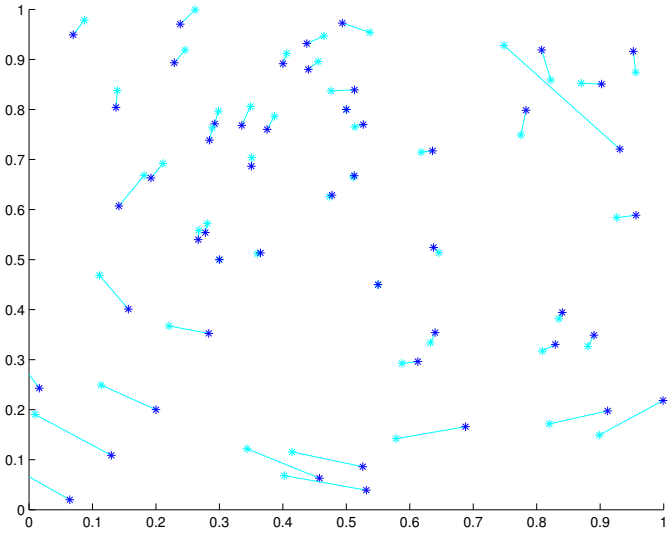

Fig. 4. Location reconstruction

noise	R=0.20			R=0.25			R=0.30		
	3	5	7	3	5	7	3	5	7
0.0	0.0427 **0.0800**	0.0419	0.0408	0.0067 **0.0076**	0.0058	0.0058	$< 1e^{-6}$ $\mathbf{1.8e^{-4}}$	$< 1e^{-6}$	$< 1e^{-6}$
0.1	0.0754	0.0644	0.0638	0.0526	0.0436	0.0270	0.0447	0.0362	0.0245 **0.0640**
0.2	0.0846	0.0801	0.0676	0.0764	0.0649	0.0493	0.0570	0.0566	0.0458
0.3	0.1063	0.0877	0.0873	0.0954	0.0825	0.0772	0.0767	0.0736	0.0459

TABLE I

NUMERICAL RESULTS FOR EXPERIMENTS

where $\bar{\mathbf{x}}_i$ and $\mathbf{x}_i$ denote the ground-truth and estimated locations.

Our paper includes a more extensive superset of their test conditions, spanning a much higher range of noise, for a total of 38 tests. In addition, for each test we carry out ten trials of randomly distributed sensor networks rather than one, equaling 380 trials. The result for each test condition is reported as the average over the ten trials. Table I contains the results for 36 tests as the cross product of $\#anchor = \{3, 5, 7\}$, $R = \{0.20, 0.25, 0.30\}$, and $noise = \{0.0, 0.1, 0.2, 0.3\}$. The average connectivity of the networks for three anchors is 5.4372 for $R = 0.20$, 7.7238 for $R = 0.25$, and 10.2477 for $R = 0.30$. For each slot in the table, our results are reported above, and if available, the corresponding results in [2] are shown below in boldface in the same slot.

For perfect range measurements with three anchor nodes, our algorithm performs only 12% better for $R = 0.25$, but furnishes an error nearly half the size as Biswas for $R = 0.20$, and an error on the order of 100 times smaller for $R = 0.30$. For seven anchor nodes, $R = 0.30$, and $noise = 0.1$, the error for Biswas is 161% greater than ours. In fact, in the same column their error 0.0640 for $noise = 0.1$ is still 39% greater than our error 0.0459 for $noise = 0.3$; this shows that our algorithm is much more robust to noise. The fifth test condition not appearing in the table included in [2] is for seven anchors, $R = 0.30$, and $noise = 0.05$, yielding error 0.0162 for the proposed algorithm and an error 0.05400 for Biswas, hence 234% greater. The last competing test condition is for seven anchors, $R = 0.40$, and $noise = 0.1$, yielding error 0.0114 for the proposed algorithm and an error 0.0500 for Biswas, hence 338% greater.

VII. CONCLUSIONS AND FURTHER WORK

This paper describes a linear program to solve for location in sensor networks. Drawing on previous approaches employing complex optimization, our approach provides a tighter solution to the problem than its competitors by applying triangle inequality constraints. In order to substantiate its performance, we run an extensive set of experiments in comparison with the published results for the best competing algorithm. Our algorithm outperforms the competing algorithm and proves robust even in the presence of high levels of noise in the measured link distances.

While the algorithm explained in this paper is centralized, it can be easily rendered distributed along the same lines as the competing algorithm. The implementation for the distributed version is currently underway in order to efficiently process networks with several thousands of nodes. In this paper, we also touch on how our proposed algorithm applies to three-dimensional networks as well.

REFERENCES

[1] P.F. Gorder, "Sizing up smart dust," *IEEE Journal on Computing in Sciences and Engineering*, vol. 5, no. 6, pp. 6-9, Nov. 2003.

[2] P. Biswas and Y. Ye, "Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization," *IEEE Conf. on Information Processing in Sensor Networks* pp. 46-54, April 2004,.

[3] A. Savvides, H. Park, and M.B. Srivastava, "The Bits and Flops of the N-hop Multilateration Primitive For Node Localization Problems," *ACM Conf. on Wireless Sensor Networks and Applications*, pp. 112-121, Sept. 2002.

[4] Y. Shang, W. Rumi, Y. Zhang, and M.P.J. Fromherz, "Localization from Mere Connectivity," *ACM Conf. on Mobile Ad Hoc Networking and Computing*, pp. 201-212, June 2003.

[5] C. Savarese, J.M. Rabaey, and J. Beutel, "Location in Distributed Ad-Hoc Wireless Networks," *IEEE Conf. on Acoustics, Speech, and Signal Processing*, pp. 2037-2040, May 2001.

[6] L. Doherty, K.S.J. Pister, and L. El Ghaoui, "Convex Position Estimation in Wireless Sensor Networks," *IEEE Conf. on Information Theory and Communications*, pp. 1655-1663, April 2001.

[7] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *IEEE Computer*, vol. 34, no. 8, pp. 57-66, Aug. 2001.

[8] K. Whitehouse and D. Culler, "Calibration as Parameter Estimation in Sensor Networks," *ACM Conf. on Wireless Sensor Networks and Applications*, pp. 59-67, Sept. 2002.

[9] S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-free positioning in mobile Ad-Hoc networks," *IEEE Hawaii Conf. on System Sciences*, pp. 255-264, Jan. 2001.

[10] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS)," *IEEE Conf. on Global Communications*, pp. 2926-2931, Nov. 2001.

[11] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali, "Linear Programming and Network Flows," *John Wiley & Sons, Inc.*, pp. 375-379, 1990.