Toward the Universal Rigidity of General Frameworks

Abdo Y. Alfakih [*] Nicole Taheri [†] and Yinyu Ye[‡]

September 4, 2010

Abstract

We prove that the $(d+1)$-lateration graph with $n(\geq d+1)$ points, when points are in *general position* in $\mathbb{R}^d$, not only is universally rigid but also admits a rank $(n-d-1)$ positive semi-definite stress matrix. We also prove that a similar result holds as in the case of sensor network localization when the graph has $m(\geq d+1)$ anchors.

1 Introduction

One of the most studied problems in distance geometry is the *Graph Realization problem*, in which one is given a graph $G = (V, E)$ and a set of non–negative weights $\{d_{ij} : (i, j) \in E\}$ on its edges, and the goal is to compute a realization of G in the Euclidean space $\mathbb{R}^d$ for a given dimension d, i.e. to place the vertices of G in $\mathbb{R}^d$ such that the Euclidean distance between every pair of adjacent vertices $(i, j) \in E$ equals to the prescribed weight d_{ij}, that is,

$$\|x_i - x_j\| = d_{ij},\ \forall (i, j) \in E,$$

where $x_i \in \mathbb{R}^d$ and $x_j \in \mathbb{R}^d$ are coordinate positions of vertices i and j, respectively; see, e.g., [13, 14, 2, 3, 15, 10, 9, 17, 18, 4, 12, 19]. Here the norm of a vector x, denoted by $\|x\|$, is the 2-norm throughout this paper, and $\mathbf{0}$ denotes a matrix of all zeros whose dimensions will be clear from the context.

This problem and its variants arise from applications in various areas, such as molecular conformation, dimensionality reduction, Euclidean ball packing, and more recently, wireless sensor network localization [5, 7, 6, 17, 8, 18, 16].

Let $P = (p_1, p_2, \cdots, p_n) \in \mathbb{R}^{d\times n}$, where $p_i \in \mathbb{R}^d$, be a given position matrix of n points in the Euclidean space $\mathbb{R}^d$. We assume that p_js are in general positions, meaning that no $(d+1)$

[*]Department of Mathematics and Statistics, University of Windsor, Windsor, Ontario N9B 3P4, Canada. Research supported by the Natural Sciences and Engineering Research Council of Canada. E-mail: alfakih@uwindsor.ca

[†]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305. Research supported in part by DOE Grant de-sc0002009. E–mail: ntaheri@stanford.edu

[‡]Department of Management Science and Engineering, Stanford University, Stanford, CA 94305. Research supported in part by NSF Grant GOALI 0800151 and DOE Grant de-sc0002009. E–mail: yinyu-ye@stanford.edu

points are contained in a lower-dimensional space $\mathbb{R}^{d-1}$. In other words, any $(d+1)$ points, say 1 to $(d+1)$, are affinely independent, i.e., the only solution of the following system of equations:

$$\sum_{i=1}^{d+1} x_i p_i = \mathbf{0} \quad \text{and} \quad \sum_{i=1}^{d+1} x_i = 0,$$

is the trivial solution $x_i = 0$ for all $i = 1, \ldots, d+1$. There is also a stronger notion: generic positions, meaning that the coordinates of $p_1, p_2, \ldots, p_n$ are algebraically independent over the integers. i.e., there does not exist a non-zero polynomial f with integer coefficients such that $f(p_1, p_2, \ldots, p_n) = 0$. Note that whether or not n points are in general positions can be checked in a time polynomial in n for any fixed dimension d, while the generic position condition is un-checkable. For convenience, let

$$A = (a_1, a_2, \cdots, a_n) = \begin{pmatrix} P \\ e^T \end{pmatrix} \in \mathbb{R}^{(d+1)\times n}, \tag{1}$$

where e is the vector of all ones. Then P is in general position if and only if every $(d+1)\times(d+1)$ square sub-matrix of A has rank $(d+1)$.

Graph G together with a position matrix P is called framework (G, P). If P is the only realizable position matrix in $\mathbb{R}^d$ up to a rigid motion, then the framework is called globally rigid. Furthermore, if P is the only realizable position matrix in all dimensions, then the framework is called universally rigid.

The notion of a stress matrix is closely related to that of a framework. A symmetric matrix $S \in \mathbb{R}^{n\times n}$ is called a stress matrix of (G, P) if and only if

$$AS = \mathbf{0}, \tag{2}$$

and

$$S_{ij} = 0, \ \forall (i,j) \notin E, \tag{3}$$

where E is the edge set of G. One can see that the highest possible rank of a stress matrix is $(n-d-1)$, and the all-zero matrix is a trivial stress matrix.

A question is: Given a framework (G, P) in $\mathbb{R}^d$, does (G, P) admit a non-trivial PSD stress matrix? The answer is yes if the (G, P) is universally rigid and $n > d+1$; and there is a polynomial-time algorithm to find a stress matrix with the maximum rank for any given instance by solving a pair of semidefinite programs as we illustrate in the next section (also see [17, 18, 4, 8]). Remarkably, Gortler *et al.* [11] further proved that a generic framework (all points are in generic positions) in $\mathbb{R}^d$ is universally rigid if and only if it admits a stress matrix that is PSD and of rank $(n-d-1)$ – the highest possible rank.

On the other hand, So [16] and Zhu *et al* [19] identified several classes of graphs that are universally rigid as long as they possess a position matrix in general position in $\mathbb{R}^d$. Note that the universal rigidity here does not depend on the position matrix but only the property of the graph. One such graph is the $(d+1)$-lateration graph – an n-point graph is a $(d+1)$-lateration graph if there is an ordering of points, say from 1 to n, such that

- the first $(d+1)$ points form a clique, and

- for each point $d+1 < j \leq n$, it connects $(d+1)$ points with their index set $N_j \subset \{1, 2, \cdots, j-1\}$.

However, it has been open whether or not the $(d+1)$-lateration framework in general position admits a PSD stress matrix with rank $(n-d-1)$, which property was observed in numerical computations. Note that the first $(d+2)$ points of the $(d+1)$-lateration graph also form a clique.

In this paper, we resolve this open question and prove our main result:

Theorem 1. *The $(d+1)$-lateration graph with a position matrix in general position admits a PSD and rank-$(n-d-1)$ stress matrix, and such a stress matrix can be computed in strongly polynomial time $O(n^3)$ arithmetic operations if the lateration ordering is known.*

As a corollary, we have

Corollary 1. *Any universal rigid framework (G, P) where the position matrix P in general position admits a PSD and rank $(n-d-1)$ stress matrix if G contains a $(d+1)$-lateration graph as a spanning subgraph, and such a stress matrix can be computed in polynomial time by solving the pair of SDPs (4) and (5).*

The corollary is true because we can ignore all other edges outside of the $(d+1)$-lateration spanning subgraph to prove the existence of a PSD and rank-$(n-d-1)$ stress matrix. Of course, we cannot actually construct such a matrix in $O(n^3)$ operations, since to find a $(d+1)$-lateration spanning subgraph needs at least $O(n^{d+2})$ operations. However, the interior-point SDP algorithms always compute a maximum rank stress matrix in a polynomial time, although it may not be strongly.

2 Stress and Pre-Stress Matrices

It turns out that the graph realization problem can be modeled as a semi-definite program, where $A^T A$ is a solution for the primal problem and the stress matrix is a solution to the dual problem. This can be seen as follows. Let the inner product of two matrices P and Q be defined by $P \cdot Q = \text{Trace}(P^T Q)$. The semidefinite programming relaxation problem for graph realization is to find a symmetric matrix $Y \in \mathbb{R}^{n \times n}$

$$\begin{array}{ll} \text{maximize} & \mathbf{0} \cdot Y \\ \text{subject to} & (e_i - e_j)(e_i - e_j)^T \cdot Y = d_{ij}^2 \ \ \forall \ (i < j, j) \in E \\ & Y \succeq \mathbf{0} \end{array} \tag{4}$$

where $e_j \in \mathbb{R}^n$ is the vector of all zeros except 1 at the j–th position and $Y \succeq \mathbf{0}$ means that Y needs to be symmetric positive semi-definite (PSD). One can see that

$$(e_i - e_j)(e_i - e_j)^T \cdot A^T A = \|a_i - a_j\|^2 = \|p_i - p_j\|^2 = d_{ij}^2, \ \forall \ (i < j, j) \in E,$$

so that $A^T A$ and $P^T P$ are both feasible solutions to the SDP relaxation problem.

The dual of the SDP relaxation model is given by:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i<j,j)\in E} w_{ij} d_{ij}^2 \\ \text{subject to} \quad & S := \displaystyle\sum_{(i<j,j)\in N_x} w_{ij}(e_i - e_j)(e_i - e_j)^T \succeq \mathbf{0} \end{aligned} \tag{5}$$

Note that the dual is always feasible, as $w_{ij} = 0$ for all $(i, j) \in E$ will be a feasible solution. In fact, it is also optimal, since its objective value equals 0, the lowest possible from the weak duality theorem. From the duality theorem, for any optimal solution S of (5) and any feasible solution Y of (4), we must have $Y \cdot S = \mathbf{0}$, which implies that $A^T A \cdot S = ASA^T = 0$ or $AS = \mathbf{0}$. Moreover, $S_{ij} = 0$, $\forall (i, j) \notin E$, so that any dual optimal solution is an PSD stress matrix. We say that SDPs (4) and (5) admit a strictly complementary solution pair when there is pair, (Y, S), of the primal and dual such that $\text{rank}(Y) + \text{rank}(S) = n$.

Thus, the question whether or not there is a non-trivial PSD stress matrix becomes whether or not there is a non-trivial dual optimal solution given that the primal problem is feasible or there is a position matrix P that satisfies the distance constraints. In particular, when the framework is universally rigid in $\mathbb{R}^d$, then the primal (4) has a solution $Y = A^T A$ with rank $(d + 1)$. Hence, SDPs (4) and (5) admit a strictly complementary solution pair if and only if there is a rank-$(n - d - 1)$ dual optimal solution S for (5). We have

Proposition 1. *An n point universally rigid framework in $\mathbb{R}^d$, for $n > d + 1$, always admits a non-trivial PSD stress matrix.*

Proof. This simply follows from Theorem 6 in [1] which states that framework (G, P) in $\mathbb{R}^d$ admits a non-trivial PSD stress matrix if and only if there does not exist a framework (G, Q) in $\mathbb{R}^{n-1}$ such that $||q_i - q_j|| = ||p_i - p_j||$ for all $(i, j) \in E$. □

Theorem 1, which we would prove in the next section, tells that an n point $(d+1)$-lateration framework in $\mathbb{R}^d$ always admits a rank-$(n - d - 1)$ PSD stress matrix when the position matrix is in general position.

Now, even if there exists a non-trivial PSD stress matrix, could we find it? The following result was stated in [17]

Proposition 2. *There is a polynomial time interior-point algorithm to compute a primal solution Y for (4) that has the highest possible rank among all primal feasible solutions, and a dual solution S for (5) that has the highest possible rank among all dual optimal solutions.*

We now proceed to prove Theorem 1. We introduce another concept – call a matrix that satisfies condition (2) pre-stress matrix. Our constructive algorithm proof is to first generate a pre-stress matrix, PSD with rank $(n - d - 1)$; then go on to generate a true stress matrix, PSD with rank $(n - d - 1)$. The following result is a simple linear algebra exercise.

Proposition 3. *There always exists a pre-stress matrix, PSD with rank $(n - d - 1)$, for any framework in $\mathbb{R}^d$. In particular, a universally rigid framework in $\mathbb{R}^d$ with the complete graph admits a rank $(n - d - 1)$ PSD stress matrix.*

For example, the projection matrix

$$I - A^T(AA^T)^{-1}A$$

is a pre-stress matrix, PSD with rank $(n-d-1)$.

Under the general position assumption, one can find an $n \times (n-d-1)$ matrix L in the form

$$L = \begin{pmatrix} * & * & ... & * & * \\ . & . & . & . & . \\ * & . & . & . & . \\ 1 & * & ... & * & * \\ 0 & 1 & ... & * & * \\ . & . & ... & . & . \\ 0 & 0 & ... & 1 & * \\ 0 & 0 & ... & 0 & 1 \end{pmatrix},$$

that is, for column k of L, L_k $(k = 1, ..., (n-d-1))$, $L_{ik} = 1$ for $i = d+1+k$ and $L_{ik} = 0$ for $i > d+1+k$, such that

$$AL = \mathbf{0},$$

where A is the matrix in (1). Clearly, L has rank $(n-d-1)$, so that $S = LL^T$ is a PSD pre-stress matrix with rank $(n-d-1)$. L is called a Gale matrix of position matrix P [1] since its columns form a basis for the null space of A.

For the $(d+1)$-lateration graph, one can generate L_k, $k = 1, ..., (n-d-1)$, by solving a system of linear equations

$$\sum_{i \in N_k} L_{ik} a_i = -a_{d+1+k}, \tag{6}$$

where a_i is ith column of the given position matrix of (1), and assigning $L_{ik} = 0$ for all $i \notin N_k$.

Lemma 1. *Linear system (6) has a unique solution under the general position condition. Moreover, matrix*

$$S^n = LL^T = \sum_{k=1}^{n-d-1} L_k L_k^T \succeq \mathbf{0}$$

is a pre-stress matrix with rank $(n-d-1)$, and it can be computed in $O(nd^2)$ arithmetic operations.

3 A Purification Algorithm

We now prove Theorem 1 for the the $(d+1)$-lateration graph. If the pre-stress matrix S^n of Lemma 1 satisfies condition (3), then it is a desired stress matrix. This is true if the $(d+1)$-lateration graph is a $(d+1)$-tree graph, that is,

- the first $(d+1)$ points form a clique, and
- for each point $d+1 < j \leq n$, it connects $(d+1)$ points of a $(d+1)$-clique with their index set $N_j \subset \{1, 2, \cdots, j-1\}$.

Thus, any entry in $S^n = LL^T$ for $i < j$ and $i \notin N_j$ has zero value.

If S^n is not a stress matrix, we need to make those entries $S^n_{ij} \neq 0$, $i < j$ and $i \notin N_j$, equal 0. We do this in the reverse order, that is, to make $S^n_{in} \neq 0$, $i < n$ and $i \notin N_n$, equal 0 first. Then do $k = n-1$, $k = n-2$, ..., and so on down to $k = d+3$. In this "purification" process, the pre-stress matrix remains PSD and maintains rank $(n-d-1)$.

For S^n constructed from L in the previous section, there is no need for purification for the last column (or row), Since any entry in LL^T for $i < n$ and $i \notin N_n$ has zero value. But for general pre-stress matrices, this may not be the case. Therefore, we first show how to purify the last column (or row) of an PSD pre-stress matrix with rank $(n-d-1)$. We construct vector $s^n \in \mathbb{R}^n$ such that

$$s^n_i = -S^n_{in} \ \forall i \notin N_n, \text{ and } s^n_n = 1,$$

and solve the system of linear equations for the rest entries in s^n:

$$\sum_{i \in N_n} s^n_i a_i = -\sum_{i \notin N_n} s^n_i a_i. \tag{7}$$

Again, linear system (7) has a unique solution under the general position condition. According to the construction, we have $As^n = \mathbf{0}$.

Lemma 2. *Let $S^{n-1} = S^n + s^n(s^n)^T$. Then*

- $AS^{n-1} = \mathbf{0}$.
- $S^{n-1} \succeq \mathbf{0}$ *and the rank of* S^{n-1} *remains* $(n-d-1)$.
- $S^{n-1}_{in} = 0$ *for all* $i < n,\ i \notin N_n$.

Proof. The first statement is from

$$AS^{n-1} = AS^n + As^n(s^n)^T = As^n(s^n)^T.$$

But $As^n = \mathbf{0}$ by the construction of s^n.

The second statement is due to $S^{n-1} \succeq S^n \succeq \mathbf{0}$.

The third statement is also from the construction. Note that, in the last column (or row) of $s^n(s^n)^T$, the ith entry, $i \neq n$ or $i \notin N_n$, is precisely $-S^n_{in}$, so that it would be canceled out in the last column (or row) of matrix $S^{n-1} = S^n + s^n(s^n)^T$. □

Then, we continue this purification process for $n-1$,...,k,..., down to $d+3$. Before the kth purification step, we have $S^k \succeq \mathbf{0}$, $AS^k = \mathbf{0}$, rank of S^k equals $(n-d-1)$, and

$$S^k_{ij} = 0,\ \forall j > k, i < j \text{ and } i \notin N_j$$

Then, we construct vector $s^k \in \mathbb{R}^n$ such that

$$s^k_i = -S^k_{ik} \ \forall i \notin N_k,\ s^k_k = 1, \text{ and } s^k_i = 0 \ \forall i > k,$$

and solve the system of linear equations for the rest entries in s^k:

$$\sum_{i \in N_k} s^k_i a_i = -\sum_{i \notin N_k} s^k_i a_i. \tag{8}$$

Again, we have $As^k = \mathbf{0}$.

Similarly, we have

Lemma 3. *Let* $S^{k-1} = S^k + s^k(s^k)^T$. *Then*

- $AS^{k-1} = \mathbf{0}$.
- $S^{k-1} \succeq \mathbf{0}$ *and the rank of* S^{k-1} *remains* $(n-d-1)$.
- $S^{k-1}_{ij} = 0$ *for all* $j \geq k$ *and* $i < j,\ i \notin N_j$.

Proof. The proof of the first two statements is identical to that in Lemma 2.

The third statement is also from the construction. Note that, in the kth column (or row) of $s^k(s^k)^T$, the ith entry, $i > k$ and $i \notin N_k$, is precisely $-S^k_{ik}$, so that it would be canceled out in the kth column (or row) of matrix $S^{k-1} = S^k + s^k(s^k)^T$. Furthermore, the $j(> k)$th column (or row) of $s^k(s^k)^T$ has all zero entries, so that the entries in $j(> k)$th column (or row) of S^{k-1} remain unchanged from S^k. □

After step $k = d+3$, we must have $AS^{d+2} = \mathbf{0}$ and $S^{d+2}_{ij} = 0$ for all $j \geq d+3$ and $i < j,\ i \notin N_j$, that is, $S^{d+2}_{ij} = 0$ for all $(i,j) \notin E$ (the first $(d+2)$ points form a clique in G). Furthermore, $S^{d+2} \succeq \mathbf{0}$ and the rank of S^{d+2} remains $(n-d-1)$. There are $(n-d-2)$ purification steps, and each step we compute a rank-one matrix $s^k(s^k)^T$ and form new pre-stress matrix $S^k + s^k(s^k)^T$ which needs at most $O(n^2)$ operations. Therefore, Theorem 1 is proved.

4 Strong Localizability of $(d+1)$-Lateration Graph with Anchors

In this section we study the stress matrix in sensor network localization, or graph localization with anchors. We are given $m(\geq d+1)$ anchor points whose positions, $\bar{p}_1, \ldots, \bar{p}_m \in \mathbb{R}^d$, are known, and n sensor points $x_1, \ldots, x_n \in \mathbb{R}^d$ whose locations are yet to be determined. Furthermore, we are given the Euclidean distance values $\bar{d}_{kj}$ between $\bar{p}_k$ and x_j for some k, j, and d_{ij} between x_i and x_j for some $i < j$. Specifically, let $N_a = \{(k,j) : \bar{d}_{kj} \text{ is specified}\}$ and $N_x = \{(i,j) : i < j,\ d_{ij} \text{ is specified}\}$. The problem is then to find a realization of $x_1, \ldots, x_n \in \mathbb{R}^d$ such that:

$$\begin{aligned} \|\bar{p}_k - x_j\|^2 &= \bar{d}^2_{kj} \quad \forall\ (k,j) \in N_a \\ \|x_i - x_j\|^2 &= d^2_{ij} \quad \forall\ (i,j) \in N_x \end{aligned} \tag{9}$$

The semidefinite programming relaxation model for (9) is to find a matrix

$$Z = \begin{pmatrix} I_d & X \\ X^T & Y \end{pmatrix} \succeq \mathbf{0} \tag{10}$$

such that

$$\begin{aligned} \text{maximize} \quad & \mathbf{0} \cdot Z \\ \text{subject to} \quad & Z_{1:d,1:d} = I_d \\ & (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \cdot Z = d^2_{ij} \ \ \forall\ (i,j) \in N_x \\ & (-\bar{p}_k; e_j)(-\bar{p}_k; e_j)^T \cdot Z = \bar{d}^2_{kj} \qquad \forall\ (k,j) \in N_a \\ & Z \succeq \mathbf{0} \end{aligned} \tag{11}$$

where $(-\bar{p}_k; e_j) \in \mathbb{R}^{d+n}$ is the vector of $-\bar{p}_k$ on top of e_j. $Z_{1:d,1:d}$ is the $d \times d$ principal submatrix of Z and I_d is the d–dimensional identity matrix. $Z_{1:d,1:d} = I_d$ can be represented as $d(d+1)/2$ linear equality constraints.

The dual of the SDP relaxation model is given by:

$$\begin{aligned} \text{minimize} \quad & I_d \cdot V + \sum_{(i,j)\in N_x} w_{ij} d_{ij}^2 + \sum_{(k,j)\in N_a} \bar{w}_{kj} \bar{d}_{kj}^2 \\ \text{subject to} \quad & S := \begin{pmatrix} V & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \sum_{(i,j)\in N_x} w_{ij} (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \\ & + \sum_{(k,j)\in N_a} \bar{w}_{kj} (-\bar{p}_k; e_j)(-\bar{p}_k; e_j)^T \succeq \mathbf{0} \end{aligned} \tag{12}$$

Note that the dual is always feasible, as $V = \mathbf{0}$, $w_{ij} = 0$ for all $(i,j) \in N_x$ and $\bar{w}_{kj} = 0$ for all $(k,j) \in N_a$ is a feasible solution.

It has been shown in [17] that both SDPs (11) and (12) are feasible and solvable when there is at least one anchor point and the graph is connected, and there is no duality gap between SDPs (11) and (12). Let P be a position matrix for the n sensors satisfying constraints in (9). Then, the problem is said to be uniquely localizable if

$$Z = \begin{pmatrix} I_d & P \\ P^T & P^T P \end{pmatrix} \succeq \mathbf{0}$$

is the only solution to SDP model (11), which is similar to the concept of universal rigidity. The problem is said to be strongly localizable if there is feasible dual matrix S such that

- $ZS = \mathbf{0}$.
- $S \succeq \mathbf{0}$ and the rank of S equals n.

One can see the graph realization problem is a sensor network localization problem without anchors. The graph realization and sensor network localization problems are closely related but different. For example, unlike in SDP (4), $A^T A$ of (1) is not longer feasible while $P^T P$ is a feasible matrix solution; hence in the dual the stresses at anchors may not need to be balanced. Also, given 2 anchors and 1 sensor in $\mathbb{R}^2$ where the sensor is connected to the 2 anchors, the problem is not uniquely localizable in sensor network localization, but it is universally rigid in graph realization since the three points form a clique. However, if the sensor network localization problem has at least $d+1$ anchors in general positions, and the graph realization problem has a $(d+1)$-point clique also in general positions, then the unique localizability is equivalent to the universal rigidity, and the strong localizability is equivalent to that of a framework $(n+d+1)$ points admits an PSD stress matrix with rank n; see [16, 19]. The latter implies that the SDP pair (11) and (12) admits a strictly complementarity solution pair.

Theorem 2. *Let G, the graph of $m(\geq d+1)$ anchor points and n sensor points with edges given in N_x and N_a, be a $(d+1)$-lateration graph and $(\bar{P}, P)$ be in general positions. Then, the sensor localization problem is strongly localizable, and a rank n dual optimal matrix can be computed in strongly polynomial time from $(\bar{P}, P)$ using $O(n^3)$ arithmetic operations if the lateration ordering is known.*

Proof. In the proof, for simplicity, we assume that there are exactly $(d+1)$ anchors and they are the first $(d+1)$ points in the lateration graph, and all other points are sensor points ordered as $1, \cdots, n$, where only edges $(i < j, j)$ are included in N_x. (One can exchange the anchor-sensor position to make the resulting rank n matrix into a desired dual optimal matrix, since the framework is universally rigid; see [19].)

We need to show there exists a matrix $S \in \mathbb{R}^{(d+n)\times(d+n)}$ that can be computed in $O(n^3)$ operations, which is a feasible solution to the dual problem (12), and also satisfies $ZS = \mathbf{0},\ S \succeq \mathbf{0}$ and $\text{rank}(S) = n$. The proof is more complicated than that of Theorem 1, since anchor point positions appear explicitly in the dual stress matrix.

Note that the only primal solution matrix Z can be written as

$$Z = \begin{pmatrix} I_d & P \\ P^T & P^T P \end{pmatrix} = \begin{pmatrix} I_d \\ P^T \end{pmatrix} \begin{pmatrix} I_d & P \end{pmatrix}$$

so that the matrix $L = \begin{pmatrix} -P; I_n \end{pmatrix} \in \mathbb{R}^{(d+n)\times n}$ is in the nullspace of Z or matrix $(I_d \ \ P)$. Moreover, the matrix

$$S^n = LL^T = \begin{pmatrix} PP^T & -P \\ -P^T & I_n \end{pmatrix} \succeq \mathbf{0},$$

will also be in the nullspace of Z, where $\text{rank}(S^n) = \text{rank}(L) = n$.

We now modify the elements of S^n in $O(n^3)$ operations, so that these conditions are maintained, and the resulting matrix becomes a feasible solution matrix to the dual problem (12). Note that the bottom right $n \times n$ submatrix of any dual feasible S, which corresponds to the sensor to sensor edges, will be

$$S_{i,j} = \begin{cases} -w_{ij}, & (i,j) \in N_x \\ 0, & (i,j) \notin N_x \\ \sum_{(i,j)\in N_x} w_{ij} + \sum_{(k,j)\in N_a} \bar{w}_{kj}, & i = j. \end{cases}$$

Thus, these diagonal elements are the negative sum of the edge weights of w_{ij} and $\bar{w}_{kj}$ for each sensor point j. The submatrix corresponding to the sensor to anchor is feasible if for $i \leq d, j > d$,

$$S_{i,j} = -\sum_{(k,j)\in N_a} \bar{w}_{kj} (\bar{p}_k)_i.$$

Note that since there are no constraints on V in the dual (12), any principal $d \times d$ submatrix is feasible as it remains positive semidefinite.

We start a step by modifying the last column, i.e. column $(d+n)$ of S^n, and make n total modification steps, each of which requires $O(n^2)$ operations. Let S^n_j denote the jth column of S^n. Then we construct a vector $s^n \in \mathbb{R}^{d+n}$ in the form,

$$s^n = \begin{pmatrix} p_n - \sum_{(k,n)\in N_a} \bar{w}_{kn} \bar{p}_k \\ e_n - \sum_{(i,n)\in N_x} w_{in} e_i \end{pmatrix}, \tag{13}$$

where $e_i \in \mathbb{R}^n$ is the ith unit vector. We show next how to decide weight variables $(\bar{w}_{kn}, w_{in})$ in s^n.

Let $S^{n-1} = S^n + s^n(s^n)^T$ be the modified matrix. Since the last element of s^n is 1, the last column of $s^n(s^n)^T$ is just s^n, so that

$$S^{n-1}_{(d+n)} = \begin{pmatrix} -p_n \\ e_n \end{pmatrix} + \begin{pmatrix} p_n - \sum_{(k,n)\in N_a} \bar{w}_{kn}\bar{p}_k \\ e_n - \sum_{(i,n)\in N_x} w_{in}e_i \end{pmatrix} = \begin{pmatrix} -\sum_{(k,n)\in N_a} \bar{w}_{kn}\bar{p}_k \\ 2e_n - \sum_{(i,n)\in N_x} w_{in}e_i \end{pmatrix}$$

Therefore, we would choose $\bar{w}_{kn}, \forall (k,n) \in N_a$, and $w_{in}, \forall (i,n) \in N_x$ such that S^{n-1}_{d+n} is in the nullspace of $(I_d \;\; P)$; and this will be true if s^n is in the nullspace of $(I_d \;\; P)$:

$$\begin{pmatrix} I_d & P \end{pmatrix} s^n = \begin{pmatrix} I_d & P \end{pmatrix} \begin{pmatrix} p_n - \sum_{(k,n)\in N_a} \bar{w}_{kn}\bar{p}_k \\ e_n - \sum_{(i,n)\in N_x} w_{in}e_i \end{pmatrix} = \mathbf{0}$$

or equivalently if,

$$\sum_{(k,n)\in N_a} \bar{w}_{kn}\bar{p}_k + \sum_{(i,n)\in N_x} w_{in}p_i = 2p_n \tag{14}$$

In order for this column to be feasible, we need to ensure the sum of the total edge weights in $S^{n-1}_{(d+n)}$ equal the diagonal element $S^{n-1}_{(d+n),(d+n)} = 2$, so that we also add

$$\sum_{(k,n)\in N_a} \bar{w}_{kn} + \sum_{(i,n)\in N_x} w_{in} = 2 \tag{15}$$

Equations (14) and (15) together exactly place $(d+1)$ linearly independent equations on $(d+1)$ weight variables, since $(\bar{P}, P)$ are in general position and G is a $(d+1)$-lateration graph, that is, sensor point n is connected to precisely $(d+1)$ points, sensors or anchors, before sensor point n. Thus, there always exists a unique solution for the values of $\bar{w}_{kn}$, and w_{in} that satisfy these equations. Now, the last column of S^{n-1} is feasible, and S^{n-1} satisfies

$$ZS^{n-1} = ZS^n + Zs^n(s^n)^T = \mathbf{0}, \quad S^{n-1} = S^n + s^n(s^n)^T \succeq S^n \succeq \mathbf{0}, \quad \text{and} \quad \operatorname{rank}(S^{n-1}) = n.$$

We continue to modify each column of the matrix, from column $(d+n)$ down to $(d+1)$, in a similar way. However, since step ℓ will add new elements to the columns and rows of $i < \ell$, so the modifications after the step $\ell = n$ will be slightly different. The ℓth modification step will modify entries, $S^\ell_{i,j}$, $i \le \ell, j \le \ell$, of S^ℓ while keep all other entries unchanged. Specifically, we want to modify the top $(d+\ell)$ entries of the $(d+\ell)$th column (or row) of S^ℓ to make it feasible while not alter the feasibility of columns $(d+\ell+1)$ to $(d+n)$ in S^ℓ. Here, we construct a vector $s^\ell \in \mathbb{R}^{d+n}$ such that $s^\ell_{d+\ell} = 1$, $s^\ell_i = 0$ for $i > d+\ell$, and the top $(d+\ell-1)$ entries are assigned to

$$s^\ell_{1:(d+\ell-1)} = \begin{pmatrix} -\sum_{(k,\ell)\in N_a} \bar{w}_{k\ell}\bar{p}_k \\ -\sum_{(i,\ell)\in N_x} w_{i\ell}e_i \end{pmatrix} - S^\ell_{1:(d+\ell-1),(d+\ell)}, \tag{16}$$

for weight variables $\bar{w}_{k\ell}$ and $w_{i\ell}$ yet to be determined, and let $S^{\ell-1} = S^\ell + s^\ell(s^\ell)^T$.

Note that this formula for s^n gives the same vector as was constructed above; and, by construction, adding $s^\ell(s^{(\ell)})^T$ to S^ℓ will not affect any column (or row) after column (or row) $d+\ell$. In particular, the top $(d+\ell-1)$ entries of the $(d+\ell)$th column (or row) of $S^{\ell-1}$ become

$$S^{\ell-1}_{1:(d+\ell-1),(d+\ell)} = s^\ell_{1:(d+\ell-1)} = \begin{pmatrix} -\sum_{(k,\ell)\in N_a} \bar{w}_{k\ell}\bar{p}_k \\ -\sum_{(i,\ell)\in N_x} w_{i\ell}e_i \end{pmatrix},$$

the $(d+\ell)$the entry

$$S^{\ell-1}_{(d+\ell),(d+\ell)} = 1 + S^{\ell}_{(d+\ell),(d+\ell)},$$

and the bottom entries

$$S^{\ell-1}_{(d+\ell+1):(d+n),(d+\ell)} = S^{\ell}_{(d+\ell+1):(d+n),(d+\ell)}.$$

We again choose weight variables $\bar{w}_{k\ell}$ and $w_{i\ell}$ in s^ℓ such that

$$\begin{pmatrix} I_d & P \end{pmatrix} s^\ell = \mathbf{0},$$

or equivalently, solve for $\bar{w}_{k\ell}$ and $w_{i\ell}$ from linear equations

$$-\sum_{(k,\ell)\in N_a} \bar{w}_{k\ell}\bar{p}_k - \sum_{(i,\ell)\in N_x} w_{i\ell}p_i + \left(1 + S^{\ell}_{(d+\ell),(d+\ell)}\right) p_\ell = S^{\ell}_{1:d,(d+\ell)} + \sum_{i=d+1}^{d+\ell-1} S^{\ell}_{i,\ell}p_i. \tag{17}$$

Similarly, to make the sum of total edge weighs in the column equal to the diagonal value $1 + S^{\ell}_{(d+\ell),(d+\ell)}$, we also add

$$\sum_{(k,\ell)\in N_a} \bar{w}_{k\ell} + \sum_{(i,\ell)\in N_x} w_{i\ell} = 1 + S^{\ell}_{(d+\ell),(d+\ell)} + \sum_{i=d+\ell+1}^{d+n} S^{\ell}_{i,(d+\ell)}. \tag{18}$$

Again, equations (17) and (18) are exactly $(d+1)$ linearly independent equations on the $(d+1)$ weight variables, and thus a unique solution always exists.

Now, the modified ℓth column of $S^{\ell-1}$ becomes feasible for a dual solution stress matrix, and $S^{\ell-1}$ satisfies

$$ZS^{\ell-1} = ZS^{\ell} + Zs^{\ell}(s^{\ell})^T = \mathbf{0},$$
$$S^{\ell-1} = S^{\ell} + s^{\ell}(s^{\ell})^T \succeq S^{\ell} \succeq \mathbf{0},$$

and hence $\mathrm{rank}(S^{\ell-1}) = n$.

Repeating this process on each column, from column $(d+n)$ down to column $(d+1)$ will result in an optimal dual solution matrix S^0 that satisfies $ZS^0 = \mathbf{0}$, $\mathrm{rank}(S^0) = n$, and $S^0 \succeq 0$. Therefore, Theorem 2 is proved. □

Similarly, we also have

Corollary 2. *Let G, the graph of $m(\geq d+1)$ anchor points and n sensor points with edges given in N_x and N_a, contain a $(d+1)$-lateration spanning subgraph and $(\bar{P}, P)$ be in general positions. Then, the sensor localization problem is strongly localizable, and a rank n dual optimal matrix can be computed in a polynomial time by solving SDPs (11) and (12).*

5 Examples

Consider a 3-lateration graph with the ordering 1 to $n = 7$. Note that $d = 2$ and the position matrix is given as

$$P = \begin{pmatrix} -1 & 1 & 0 & 2 & 1 & -1 & -2 \\ 1 & 1 & 0.5 & 0 & -1 & -1 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 7}.$$

Example 1 Let

$$N_4 = \{1, 2, 3\},\ N_5 = \{1, 3, 4\},\ N_6 = \{1, 2, 4\},\ N_7 = \{3, 4, 5\}.$$

In this example

$$L = \begin{pmatrix} 1.5000 & 5.0000 & -2.0000 & 0 \\ -0.5000 & 0 & 3.0000 & 0 \\ -2.0000 & -8.0000 & 0 & -1.6000 \\ 1.0000 & 2.0000 & -2.0000 & 1.4000 \\ 0 & 1.0000 & 0 & -0.8000 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

and

$$S^7 = LL^T = \begin{pmatrix} 31.2500 & -6.7500 & -43.0000 & 15.5000 & 5.0000 & -2.0000 & 0 \\ -6.7500 & 9.2500 & 1.0000 & -6.5000 & 0 & 3.0000 & 0 \\ -43.0000 & 1.0000 & 70.5600 & -20.2400 & -6.7200 & 0 & -1.6000 \\ 15.5000 & -6.5000 & -20.2400 & 10.9600 & 0.8800 & -2.0000 & 1.4000 \\ 5.0000 & 0 & -6.7200 & 0.8800 & 1.6400 & 0 & -0.8000 \\ -2.0000 & 3.0000 & 0 & -2.0000 & 0 & 1.0000 & 0 \\ 0 & 0 & -1.6000 & 1.4000 & -0.8000 & 0 & 1.0000 \end{pmatrix}.$$

Note that S^7 is already a stress matrix to meet edge condition (3), so that no "purification" algorithm is needed. This example is not interesting, since the graph is actually a 3-tree graph.

Example 2 Let

$$N_4 = \{1, 2, 3\},\ N_5 = \{1, 3, 4\},\ N_6 = \{2, 4, 5\},\ N_7 = \{1, 3, 6\}.$$

In this example

$$L = \begin{pmatrix} 1.5000 & 5.0000 & 0 & -1.2500 \\ -0.5000 & 0 & -1.0000 & 0 \\ -2.0000 & -8.0000 & 0 & 1.0000 \\ 1.0000 & 2.0000 & 2.0000 & 0 \\ 0 & 1.0000 & -2.0000 & 0 \\ 0 & 0 & 1.0000 & -0.7500 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

and

$$S^7 = LL^T = \begin{pmatrix} 28.8125 & -0.7500 & -44.2500 & 11.5000 & 5.0000 & 0.9375 & -1.2500 \\ -0.7500 & 1.2500 & 1.0000 & -2.5000 & 2.0000 & -1.0000 & 0 \\ -44.2500 & 1.0000 & 69.0000 & -18.0000 & -8.0000 & -0.7500 & 1.0000 \\ 11.5000 & -2.5000 & -18.0000 & 9.0000 & -2.0000 & 2.0000 & 0 \\ 5.0000 & 2.0000 & -8.0000 & -2.0000 & 5.0000 & -2.0000 & 0 \\ 0.9375 & -1.0000 & -0.7500 & 2.0000 & -2.0000 & 1.5625 & -0.7500 \\ -1.2500 & 0 & 1.0000 & 0 & 0 & -0.7500 & 1.0000 \end{pmatrix}.$$

While the last column (or row) of S^7 is fine with the edge condition (3), but the rest does not meet condition (3). We start the purification process from $k = 6$, where $S^6 = S^7$. The column vector s^6 is generated by first assigning

$$s_1^6 = -S_{1,6}^6 = -0.9375,\ s_3^6 = -S_{3,6}^6 = 0.75,\ s_6^6 = 1,\ s_7^6 = 0,$$

and solve (s_2^6, s_4^6, s_5^6) from linear system (8) and get

$$s^6 = \begin{pmatrix} -0.9375 \\ -0.0625 \\ 0.7500 \\ 0.8750 \\ -1.6250 \\ 1.0000 \\ 0 \end{pmatrix},$$

and

$$S^5 = S^6 + s^6(s^6)^T = \begin{pmatrix} 29.6914 & -0.6914 & -44.9531 & 10.6797 & 6.5234 & 0 & -1.2500 \\ -0.6914 & 1.2539 & 0.9531 & -2.5547 & 2.1016 & -1.0625 & 0 \\ -44.9531 & 0.9531 & 69.5625 & -17.3438 & -9.2188 & 0 & 1.0000 \\ 10.6797 & -2.5547 & -17.3438 & 9.7656 & -3.4219 & 2.8750 & 0 \\ 6.5234 & 2.1016 & -9.2188 & -3.4219 & 7.6406 & -3.6250 & 0 \\ 0 & -1.0625 & 0 & 2.8750 & -3.6250 & 2.5625 & -0.7500 \\ -1.2500 & 0 & 1.0000 & 0 & 0 & -0.7500 & 1.0000 \end{pmatrix}.$$

Next the column vector s^5 is generated by first assigning

$$s_2^5 = -S_{2,5}^5 = -2.1016,\ s_5^5 = 1,\ s_6^5 = s_7^5 = 0,$$

and solve (s_1^5, s_3^5, s_4^5) from linear system (8)

$$s^5 = \begin{pmatrix} 11.3047 \\ -2.1016 \\ -16.4063 \\ 6.2031 \\ 1.0000 \\ 0 \\ 0 \end{pmatrix},$$

and

$$S^4 = S^5 + s^5(s^5)^T = \begin{pmatrix} 157.4874 & -24.4489 & -230.4207 & 80.8041 & 17.8281 & 0 & -1.2500 \\ -24.4489 & 5.6705 & 35.4319 & -15.5909 & 0 & -1.0625 & 0 \\ -230.4207 & 35.4319 & 338.7275 & -119.1138 & -25.6250 & 0 & 1.0000 \\ 80.8041 & -15.5909 & -119.1138 & 48.2444 & 2.7813 & 2.8750 & 0 \\ 17.8281 & 0 & -25.6250 & 2.7813 & 8.6406 & -3.6250 & 0 \\ 0 & -1.0625 & 0 & 2.8750 & -3.6250 & 2.5625 & -0.7500 \\ -1.2500 & 0 & 1.0000 & 0 & 0 & -0.7500 & 1.0000 \end{pmatrix}.$$

One can see that S^4 is now a desired stress matrix for Example 2.

Example 3 The graph is identical to that in Example 2, but the first three points are anchors.

We use the same 3-lateration graph as in the previous examples, where $n = 7$, and $d = 2$, with the position matrix

$$P = \begin{pmatrix} -1 & 1 & 0 & 2 & 1 & -1 & -2 \\ 1 & 1 & 0.5 & 0 & -1 & -1 & 0 \end{pmatrix}$$

and

$$N_4 = \{1, 2, 3\}, N_5 = \{1, 3, 4\}, N_6 = \{2, 4, 5\}, N_7 = \{1, 3, 6\}$$

where nodes 1,2,3 are anchors. The matrix L in this case is

$$L = \begin{pmatrix} -2 & -1 & 1 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and S^4 is

$$S^4 = LL^T = \begin{pmatrix} 10 & 0 & -2 & -1 & 1 & 2 \\ 0 & 2 & 0 & 1 & 1 & 0 \\ -2 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We first form s^4 according to Equation (13) for s^n, which gives

$$s^4 = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T$$

Moreover, this vector is in the nullspace of Z, and satisfies both equations (14) and (15). The updated matrix is then

$$S^3 = S^4 + s^4(s^4)^T = \begin{pmatrix} 14 & 0 & -2 & -1 & 1 & 4 \\ 0 & 2 & 0 & 1 & 1 & 0 \\ -2 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Each next step creates a vector of weights that is solved for by a linear system, and satisfy the equations (17) and (18). The matrices are updated accordingly,

$$s^3 = \begin{pmatrix} -15 & 1 & 8 & 0 & 1 & 0 \end{pmatrix}^T$$

and

$$S^2 = S^3 + s^3(s^3)^T = \begin{pmatrix} 239 & -15 & -122 & -1 & -14 & 4 \\ -15 & 3 & 8 & 1 & 2 & 0 \\ -122 & 8 & 65 & 0 & 8 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ -14 & 2 & 8 & 0 & 2 & 0 \\ 4 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Next,

$$s^2 = \begin{pmatrix} 1 & -1 & 0 & -1 & 0 & 0 \end{pmatrix}^T$$

and

$$S^1 = S^2 + s^2(s^2)^T = \begin{pmatrix} 240 & -16 & -122 & -2 & -14 & 4 \\ -16 & 4 & 8 & 2 & 2 & 0 \\ -122 & 8 & 65 & 0 & 8 & 0 \\ -2 & 2 & 0 & 2 & 0 & 0 \\ -14 & 2 & 8 & 0 & 2 & 0 \\ 4 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

And lastly, we have

$$s^1 = \begin{pmatrix} 6.5 & 8.5 & 1 & 0 & 8.5 & 0 \end{pmatrix}^T$$

and

$$S = S^1 + s^1(s^1)^T = \begin{pmatrix} 282.2500 & 39.2500 & -115.5000 & -2.0000 & 41.2500 & 4.0000 \\ 39.2500 & 76.2500 & 16.5000 & 2.0000 & 74.2500 & 0 \\ -115.5000 & 16.5000 & 66.0000 & 0 & 16.5000 & 0 \\ -2.0000 & 2.0000 & 0 & 2.0000 & 0 & 0 \\ 41.2500 & 74.2500 & 16.5000 & 0 & 74.2500 & 0 \\ 4.0000 & 0 & 0 & 0 & 0 & 2.0000 \end{pmatrix}$$

This resulting matrix S is the desired dual stress matrix, and satisfies $ZS = 0$, $S \succeq 0$ and $\text{rank}(S) = n$.

References

[1] A. Y. Alfakih. On bar frameworks, stress matrices and semidefinite programming. *to appear in Mathematical Programming.*

[2] A. Y. Alfakih. Graph Rigidity via Euclidean Distance Matrices. *Linear Algebra and Its Applications* 310:149–165, 2000.

[3] A. Y. Alfakih. On Rigidity and Realizability of Weighted Graphs. *Linear Algebra and Its Applications* 325:57–70, 2001.

[4] A. Y. Alfakih. On the Universal Rigidity of Generic Bar Frameworks. *Contribution to Discrete Mathematics* 5(3):7–17, 2010.

[5] A. Y. Alfakih, A. Khandani, H. Wolkowicz. Solving Euclidean Distance Matrix Completion Problems Via Semidefinite Programming. *Comput. Opt. and Appl.* 12:13–30, 1999.

[6] James Aspnes, David Goldenberg, Yang Richard Yang. On the Computational Complexity of Sensor Network Localization. *ALGOSENSORS 2004*, in *LNCS* 3121:32–44, 2004.

[7] P. Biswas, Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. *Proc. 3rd IPSN* 46–54, 2004.

[8] P. Biswas, K. C Toh and Y. Ye. A Distributed SDP approach for Large-scale Noisy Anchor-free Graph Realization with Applications to Molecular Conformation. *SIAM Journal on Scientific Computing* 30(3): 1251–1277, 2008.

[9] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Moore, B. D. O. Anderson, P. N. Belhumeur. Rigidity, Computation, and Randomization in Network Localization. *Proc. 23rd INFOCOM*, 2004.

[10] R. Connelly. Generic Global Rigidity. *Discrete and Computational Geometry*, 33(4):549–563, 2005.

[11] S. J. Gortler and D. P. Thurston. Characterizing the universal rigidity of generic frameworks. *arXiv/1001.0172v1*, 2009.

[12] S. J. Gortler, A. D. Healy, D. P Thurston. Characterizing Generic Global Rigidity. *American Journal of Mathematics*, 2010.

[13] J. Graver, B. Servatius, H. Servatius. Combinatorial Rigidity. *AMS*, 1993.

[14] B. Hendrickson. Conditions for Unique Graph Realizations. *SIAM J. Comput.* 21(1):65–84, 1992.

[15] B. Jackson, T. Jordán. Connected Rigidity Matroids and Unique Realizations of Graphs. Preprint, 2003.

[16] A. M.-C. So. A Semidefinite Programming Approach to the Graph Realization Problem: Theory, Applications and Extensions. Ph.D. Thesis, Stanford University, 2007.

[17] A. M.-C. So and Y. Ye. Theory of Semidefinite Programming for Sensor Network Localization. *Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithm (SODA 2005)* 2005, pp. 405–414; and *Mathematical Programming, Series B*, vol. 109, no. 2, pp. 367–384, 2007.

[18] A. M.-C. So and Y. Ye. A Semidefinite Programming Approach to Tensegrity Theory and Realizability of Graphs. *Proceedings of the 18th Annual ACM–SIAM Symposium on Discrete Algorithm (SODA 2006)*, 2006, pp. 766–775.

[19] Z. Zhu, A. M-C So, Y. Ye. Universal Rigidity: Towards Accurate and Efficient Localization of Wireless Networks. *Proc. IEEE INFOCOM*, 2010.