UNIVERSAL RIGIDITY AND EDGE SPARSIFICATION FOR SENSOR NETWORK LOCALIZATION*

ZHISU ZHU†, ANTHONY MAN–CHO SO‡, AND YINYU YE§

Abstract. Owing to their high accuracy and ease of formulation, there has been great interest in applying convex optimization techniques, particularly that of semidefinite programming (SDP) relaxation, to tackle the sensor network localization problem in recent years. However, a drawback of such techniques is that the resulting convex program is often expensive to solve. In order to speed up computation, various edge sparsification heuristics have been proposed, whose aim is to reduce the number of edges in the input graph. Although these heuristics do reduce the size of the convex program and hence making it faster to solve, they are often ad hoc in nature and do not preserve the localization properties of the input. As such, one often has to face a tradeoff between solution accuracy and computational effort. In this paper, we propose a novel edge sparsification heuristic that can provably preserve the localization properties of the original input. At the heart of our heuristic is a graph decomposition procedure, which allows us to identify certain sparse *generically universally rigid* subgraphs of the input graph. Our computational results show that the proposed approach can significantly reduce the computational and memory complexities of SDP–based algorithms for solving the sensor network localization problem. Moreover, it compares favorably with existing speedup approaches, both in terms of accuracy and solution time.

Key words. sensor network localization, rigidity theory, semidefinite programming, edge sparsification

AMS subject classifications. 52C25, 90C22, 05C85, 68Q25

1. Introduction.

In recent years, the graph realization problem has attracted a lot of attention in many communities. Such an interest can be attributed to the fundamental nature and practical significance of the problem. Indeed, the graph realization problem arises in many different contexts, such as distance geometry [6], wireless sensor network localization [13], and molecular conformation [17]. In the context of wireless sensor network localization—which is the one we shall focus on in this paper—the problem can be formulated as follows. Let $d \geq 1$ be an integer (in practice, we usually have $d = 2$ or 3), and let $G = (V, E)$ be a graph. The vertices of G are partitioned into two sets: the set $V_s = \{1, \ldots, n\}$ of *sensors*, and the set $V_a = \{n+1, \ldots, n+m\}$ of *anchors*. Together, they induce two subsets $E_{ss}, E_{sa} \subset E$, which are defined as follows:

$$E_{ss} = \{(i,j) \in E : i, j \in V_s\} \quad \text{and} \quad E_{sa} = \{(i,j) \in E : i \in V_s,\, j \in V_a\}.$$

Now, each anchor $i \in V_a$ is given a position $a_i \in \mathbb{R}^d$. Moreover, each edge $(i,j) \in E_{ss}$ (resp. $(i,j) \in E_{sa}$) is given a positive weight d_{ij} (resp. $\bar{d}_{ij}$), which can be viewed as the distance between sensor i and sensor j (resp. sensor i and anchor j). In particular, the existence of an edge between $i, j \in V$ means that the distance between i and j is known. This allows us to assume without loss that $E_{aa} = \{(i,j) : n+1 \leq$

*A preliminary version of this paper has appeared in the *Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM 2010)*, 2010 [42]. This work was supported in part by the National Science Foundation (NSF) Grant GOALI 0800151, and the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No. CUHK 416908.

†Institute for Computational and Mathematical Engineering, Stanford University, Stanford, California 94305 (zhuzhisu@stanford.edu).

‡Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong (manchoso@se.cuhk.edu.hk).

§Department of Management Science and Engineering, Stanford University, Stanford, California 94305 (yinyu-ye@stanford.edu).

$i < j \leq n + m\} \subset E$, as the distance between any two anchors is trivially known. To summarize, an instance of the sensor network localization problem is given by a 4–tuple $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$, where $G = ((V_s, V_a), (E_{ss}, E_{sa}, E_{aa}))$ is the input graph, $\mathbf{d} = (d_{ij})_{(i,j)\in E_{ss}} \in \mathbb{R}_+^{|E_{ss}|}$ and $\bar{\mathbf{d}} = (\bar{d}_{ij})_{(i,j)\in E_{sa}} \in \mathbb{R}_+^{|E_{sa}|}$ are the vectors of edge weights, $\mathbf{a} = (a_i)_{i\in V_a} \in \mathbb{R}^{dm}$ is the vector containing the positions of the anchors, and $d \geq 1$ is the given dimension. For the sake of simplicity, suppose that the given distance data are exact. Our goal then is to determine the positions of the sensors so that they satisfy the given distance measurements. In other words, we are interested in finding an assignment of coordinates $\tilde{\mathbf{x}} = (\tilde{x}_1; \ldots; \tilde{x}_n) \in \mathbb{R}^{dn}$ to the vertices in V_s (which, together with $\mathbf{a} \in \mathbb{R}^{dm}$, is called a *localization* or *realization* of G) such that $\tilde{\mathbf{x}}$ satisfies the following system:

$$\begin{array}{rcll} \|x_i - x_j\|_2^2 & = & d_{ij}^2 & \text{for } (i,j) \in E_{ss}, \\ \|a_i - x_j\|_2^2 & = & \bar{d}_{ij}^2 & \text{for } (i,j) \in E_{sa}, \\ x_i & \in & \mathbb{R}^d & \text{for } i = 1, \ldots, n. \end{array} \tag{1.1}$$

Here, we use $(u; v)$ to denote the $(p + q)$–dimensional column vector obtained by stacking the p–dimensional column vector u on top of the q–dimensional column vector v. Problem (1.1) arises as an important sub–problem in many wireless sensor network applications, such as target detection and tracking, geographic routing, and collaborative signal processing [29]. In those applications, location–dependent data are collected either over a vast geographical area or in an indoor environment. Thus, it is typically infeasible or impractical to "solve" Problem (1.1) by installing GPS–like systems in the sensors. Furthermore, from a computational point of view, it is $\mathcal{NP}$–hard to decide whether Problem (1.1) has a feasible solution or not [32], even when the input graph is a so–called unit disk graph [4, 12]. As a result, many heuristics for solving Problem (1.1) have been proposed. One family of heuristics that has received much attention lately is the convex relaxation approach initiated by Doherty et al. [19] and Biswas and Ye [9]. The idea behind that approach is to develop efficiently solvable convex relaxations of Problem (1.1), so that approximate solutions to the original problem can be easily derived. As a concrete example, consider the following semidefinite programming (SDP) relaxation of Problem (1.1), which is first proposed by Biswas and Ye [9] and refines the convex relaxations proposed in Doherty et al. [19]:

$$\begin{array}{rcll} (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \bullet Z & = & d_{ij}^2 & \text{for } (i,j) \in E_{ss}, \\ (a_i; -e_j)(a_i; -e_j)^T \bullet Z & = & \bar{d}_{ij}^2 & \text{for } (i,j) \in E_{sa}, \\ Z & = & \begin{bmatrix} I & X \\ X^T & Y \end{bmatrix}, & \\ Z & \in & \mathcal{S}_+^{d+n}. & \end{array} \tag{1.2}$$

Here, $e_i \in \mathbb{R}^n$ is the i–th standard basis vector (where $i = 1, \ldots, n$), I is the $d \times d$ identity matrix, $\mathcal{S}_+^{d+n}$ is the set of $(d + n) \times (d + n)$ symmetric positive semidefinite matrices, and $A \bullet B = \mathrm{tr}(AB)$ is the trace inner product of two symmetric matrices A, B of the same dimension. As is well known, the SDP (1.2) can be solved (to any desired accuracy) efficiently using standard interior point algorithms [38]. Moreover, an approximate solution to Problem (1.1) can be extracted directly from the X component of the matrix Z (see [9, 35] for details). One advantage of the SDP

relaxation (1.2) (and in fact of most convex relaxations of Problem (1.1)) is that it can be applied to any dimension $d \geq 1$. This should be contrasted with some existing localization algorithms (see, e.g., [30, 11]), which relies heavily on the geometry of the two–dimensional Euclidean space and cannot be easily extended to higher dimensions. Moreover, as demonstrated in [9, 35, 34, 33], the SDP relaxation (1.2) not only can produce highly accurate results, but also possesses nice theoretical properties. However, a major drawback of (1.2) is that it is computationally demanding. Indeed, the SDP (1.2) contains $(d+n)(d+n+1)/2$ variables and $d(d+1)/2 + |E_{ss}| + |E_{sa}|$ equality constraints. As such, an instance of (1.2) with only a few hundred sensors is considered to be a challenge for standard SDP solvers [10]. Thus, there has been a lot of interest recently in designing faster algorithms for solving the SDP (1.2), as well as in developing other more computationally efficient convex relaxations of (1.1). Some of the proposed approaches include:

1. **Distributed Computation.** One natural way to handle a large instance of Problem (1.1) is to first divide the graph G into small subgraphs, then find a localization of each subgraph using, say, the SDP (1.2), and finally stitch the subgraphs back together to produce a localization of G. Such an approach is explored in various recent work (see, e.g., [10, 14, 8]), where the difference lies mainly in how the subgraphs are formed and how they are stitched back together. However, we note that the distributed computation approaches proposed so far tend to be ad hoc, and the error incurred in each subgraph often propagates throughout the entire graph. Thus, the solution produced by this approach can be quite inaccurate.
2. **Alternative Convex Relaxations.** Another way to obtain fast heuristics for solving Problem (1.1) is to consider other, possibly weaker, convex relaxations. For instance, Tseng [37] proposed a second–order cone programming (SOCP) relaxation of (1.1) and showed that it can be solved much faster than the SDP relaxation (1.2). Later, motivated by ideas in Fukuda et al. [20] and Nie [31], Wang et al. [40] proposed a further relaxation of (1.2), in which the $(d+n) \times (d+n)$ positive semidefinite cone constraint in (1.2) is replaced by a number of smaller positive semidefinite cone constraints that correspond to the local connectivity of the input graph. We remark that although the above relaxations can be solved much more efficiently than the SDP relaxation (1.2), they are not as tight as the SDP relaxation. Consequently, the solutions produced by the above relaxations may not be as accurate as those produced by the SDP relaxation (1.2).
3. **Exploiting Sparsity in SDPs.** In recent years, a lot of effort has been spent on improving the computational efficiency in solving large–scale SDPs. One particular approach is to exploit the sparsity in the data matrices to decompose a large positive semidefinite cone constraint into an *equivalent* system of smaller positive semidefinite cone constraints (see, e.g., [20, 39, 26]). (This should not be confused with the decomposition proposed in [40], which results in a *relaxation* of the original SDP.) In [27] the authors applied the above approach to solve the SDP relaxation (1.2) and another, tighter, SDP relaxation of Problem (1.1). The reported computational results are very promising. We remark that the techniques proposed in [20, 26] are very general and can be applied to any SDP. As such, they do not exploit the geometric structure of the sensor network localization problem.

In this paper, we present another novel approach for speeding up the solution time of the SDP (1.2). Our approach exploits both the combinatorial and geometric aspects of the sensor network localization problem and is motivated by various theoretical properties of (1.2) established in [35, 33]. Specifically, we first propose an edge sparsification algorithm that, given any graph $G = (V, E)$, returns an edge–sparsified subgraph $G' = (V, E')$ such that G and G' have essentially the same localization properties. In other words, under some mild assumptions, one can conclude that every localization of the sparsified instance $(G', (\mathbf{d}, \bar{\mathbf{d}})|_{G'}, \mathbf{a}, d)$ is also a localization of the original instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ (the converse is trivial, as G' is a subgraph of G). Here, $(\mathbf{d}, \bar{\mathbf{d}})|_{G'}$ denotes the restriction of $(\mathbf{d}, \bar{\mathbf{d}})$ onto the edges of G'. At the heart of our algorithm is a graph decomposition procedure, which allows us to identify certain sparse *generically universally rigid* subgraphs of the input graph. Our algorithm has polynomial complexity for any fixed $d \geq 1$, and the number of edges in the subgraph it produces can be bounded and is typically small. In the context of the SDP (1.2), our result implies that we can reduce the number of equality constraints *without changing the set of feasible solutions*. This should be contrasted with the edge sparsification algorithms proposed in [10, 14, 8, 40, 27], which do not have such a guarantee. In particular, the size–reduced SDPs produced by those algorithms are not necessarily equivalent to the original one. Recently, Krislock and Wolkowicz [28] have developed a speedup approach that has a similar flavor as ours. Their algorithm proceeds by identifying cliques in the input graph, and in some sense it can be viewed as a combinatorial algorithm for solving the SDP (1.2). However, the runtime of their algorithm is not clear. Moreover, as we shall see, our approach identifies a larger class of graphs (i.e., beyond cliques) for which edge sparsification is possible.

Next, we combine our edge sparsification algorithm with existing speedup techniques (see, e.g., [27]) to develop a fast SDP–based algorithm for the sensor network localization problem. The proposed SDP–based algorithm not only guarantees to find a solution to the original SDP (1.2) (i.e., with the full set of edge–distance data), but also runs significantly faster and has a much lower memory requirement than the original SDP (1.2). Moreover, it compares very favorably with existing convex relaxation–based algorithms, both in terms of accuracy and solution time.

Although our work employs the notion of universal rigidity mainly as a tool to speed up SDP and other convex relaxation–based algorithms for the sensor network localization problem, it should be emphasized that the notion is interesting and important in its own right. For instance, the recent work of So and Ye [35, 34, 33], Alfakih [2, 1], and Gortler and Thurston [22] show that many questions about universally rigid instances can be answered using semidefinite programming. In this regard, our work can be viewed as a further demonstration of the close connection between universal rigidity and semidefinite programming.

The rest of the paper is organized as follows. In Section 2, we give the motivation of our approach and show how it can be formalized and studied under the framework of rigidity theory. Then, we show that there exists a family of sparse graphs (i.e., graphs with only $O(|V|)$ edges, where $|V|$ is the number of vertices in the graph) for which the SDP relaxation (1.2) is essentially exact. Such a family forms the basic building block of our edge sparsification algorithm, which we introduce in Section 3. Then, we analyze the properties of the proposed algorithm. In particular, we establish the precise conditions under which the size–reduced SDP is equivalent to the original SDP. In Section 4, we discuss some further techniques for improving both the accuracy and solution speed of the size–reduced SDP. We then report our computational results

and experiences in Section 5 and close with some concluding remarks in Section 6.

2. Preliminaries.

2.1. Motivation of Our Approach. Before we delve into the details of the edge sparsification algorithm, let us motivate our approach and introduce some relevant concepts. In recent years, there has been some interest in applying rigidity theory to study the sensor network localization problem (see, e.g., [3, 41]). Roughly speaking, the theory of rigidity is concerned with the problem of determining whether a given realization of a graph is unique (up to congruence) in a given dimension (see [23] for an introduction). In other words, given a realization $\mathbf{p} = (p_1; \ldots; p_l) \in \mathbb{R}^{dl}$ of an l–vertex graph $G = (V, E)$ in $\mathbb{R}^d$, where vertex $i \in V$ is assigned the coordinates $p_i \in \mathbb{R}^d$ for $i = 1, \ldots, l$, we would like to know whether there exists another realization $\mathbf{q} = (q_1; \ldots; q_l) \in \mathbb{R}^{dl}$ of G such that $\|p_i - p_j\|_2 = \|q_i - q_j\|_2$ for all $(i, j) \in E$, but $\|p_i - p_j\|_2 \neq \|q_i - q_j\|_2$ for some $(i, j) \notin E$. If such a $\mathbf{q}$ does not exist, then $\mathbf{p}$ is the unique (up to congruence) realization of G in $\mathbb{R}^d$, and the pair $(G, \mathbf{p})$ is said to be *globally rigid* in $\mathbb{R}^d$. We should point out that in the setting of rigidity theory, all vertices of G are considered as sensors, and hence one is interested only in non–congruent realizations.

At first sight, it seems natural to apply rigidity–theoretic techniques to tackle the sensor network localization problem. Afterall, there exist efficient algorithms that, given an integer $d \geq 1$ and an l–vertex graph G, decide whether $(G, \mathbf{p})$ is globally rigid in $\mathbb{R}^d$ for all *generic*[1] realizations $\mathbf{p} \in \mathbb{R}^{dl}$ [24, 16, 25, 21]. Now, if G is *generically globally rigid* in $\mathbb{R}^d$ (i.e., if $(G, \mathbf{p})$ is globally rigid in $\mathbb{R}^d$ for all generic realizations $\mathbf{p}$ of G in $\mathbb{R}^d$), then every instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ will admit a unique localization in $\mathbb{R}^d$ (equivalently, there is a unique solution to the system in (1.1)), provided that there are at least $d + 1$ anchors, and $(\mathbf{d}, \bar{\mathbf{d}})$ and $\mathbf{a}$ induce a generic realization of G in $\mathbb{R}^d$. Moreover, it is clear that for any generically globally rigid graph $G = (V, E)$ in $\mathbb{R}^d$, there exists a minimal subset $E' \subset E$ of edges such that $G' = (V, E')$ is also generically globally rigid in $\mathbb{R}^d$. Thus, in order to reduce the size of a given instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem, one could try to identify generically globally rigid subgraphs of G and sparsify them as much as possible while preserving their generic global rigidity. Unfortunately, there is a serious obstacle in the above approach, namely, we may not be able to find the actual realizations of those generically globally rigid subgraphs in an efficient manner! Indeed, Aspnes et al. [4] showed that unless $\mathcal{RP} = \mathcal{NP}$, there does not exist an efficient randomized algorithm that, given an instance of the sensor network localization problem with a unique localization in $\mathbb{R}^2$, finds that localization. In other words, the knowledge that an instance has a unique localization in $\mathbb{R}^d$ does not make the task of finding that localization any easier. Consequently, the applicability of rigidity theory to the sensor network localization problem is somewhat limited.

In order to circumvent the above difficulty, one could try instead to identify so–called uniquely d–localizable sub–instances of a given instance of the sensor network localization problem. The notion of unique d–localizability was introduced by So and Ye [35] and can be viewed as a computationally efficient alternative to global rigidity. Specifically, an instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem is said to be *uniquely d–localizable* if (i) the system in (1.1) has a unique solution $\tilde{\mathbf{x}} = (\tilde{x}_1; \ldots; \tilde{x}_n) \in \mathbb{R}^{dn}$, and (ii) for any $l > d$, $((\tilde{x}_1; \mathbf{0}); (\tilde{x}_2; \mathbf{0}); \ldots; (\tilde{x}_n; \mathbf{0})) \in \mathbb{R}^{ln}$ is

[1] We say that the point $\mathbf{p} = (p_1, \ldots, p_l) \in \mathbb{R}^{dl}$ is *generic* if there does not exist a non–zero polynomial $h : \mathbb{R}^{dl} \to \mathbb{R}$ with integer coefficients such that $h(p_1, \ldots, p_l) = 0$.

the unique solution to the following system:

$$\begin{aligned} \|x_i - x_j\|_2^2 &= d_{ij}^2 \qquad \text{for } (i,j) \in E_{ss}, \\ \|(a_i; \mathbf{0}) - x_j\|_2^2 &= \bar{d}_{ij}^2 \qquad \text{for } (i,j) \in E_{sa}, \\ x_i &\in \mathbb{R}^l \qquad \text{for } i = 1, \ldots, n. \end{aligned}$$

Geometrically, the above conditions state that the given instance has a unique localization in $\mathbb{R}^l$ for all $l \geq d$, and that localization is given by $(\tilde{\mathbf{x}}, \mathbf{a}) \in \mathbb{R}^{dn} \times \mathbb{R}^{dm}$. We should emphasize that the notion of unique d–localizability depends both on the combinatorial structure of G and the geometric information embedded in $(\mathbf{d}, \bar{\mathbf{d}}) \in \mathbb{R}_+^{|E_{ss}|} \times \mathbb{R}_+^{|E_{sa}|}$ and $\mathbf{a} \in \mathbb{R}^{dm}$. Moreover, unlike in the setting of rigidity theory, as long as the given instance of the sensor network localization problem has more than one distinct localization, that instance is not considered as uniquely d–localizable, even if all those distinct localizations are congruent to each other.

As shown in [35], an instance of the sensor network localization problem is uniquely d–localizable if and only if the maximum rank solution to the SDP (1.2) is d. In this case, the SDP (1.2) is an exact relaxation of (1.1), i.e., Problem (1.1) and Problem (1.2) are equivalent. Thus, the SDP (1.2) provides an efficient means to check whether an instance of the sensor network localization problem is uniquely d–localizable, and if so, find that unique localization. Now, in view of our earlier discussion and to further take advantage of the above results, we are led to the following questions:

1. Given a uniquely d–localizable instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem, can we identify a set of edges in G such that the instance obtained by removing those edges is still uniquely d–localizable?
2. More generally, given an arbitrary instance of the sensor network localization problem, can we decompose it into a number of uniquely d–localizable sub–instances and sparsify them, while at the same time preserving the localization properties of the original instance?

In other words, we are interested in sparsifying a given instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem, so that the total number of edges in E_{ss} and E_{sa} is small, and yet the localization properties of the original instance are preserved. This will then allow us to speed up the solution time of the SDP (1.2). As we shall see, both of the above questions can be answered in the affirmative to some extent. Our approach is to first identify a family $\mathcal{H}$ of sparse graphs (i.e., with only $O(|V|)$ edges) such that if $H \in \mathcal{H}$, then the instance $(H, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is uniquely d–localizable whenever there are at least $d+1$ anchors, and $(\mathbf{d}, \bar{\mathbf{d}}) \in \mathbb{R}_+^{|E_{ss}|} \times \mathbb{R}_+^{|E_{sa}|}$ and $\mathbf{a} \in \mathbb{R}^{dm}$ induce a generic realization of G in $\mathbb{R}^d$. Now, if $G' = (V, E') \in \mathcal{H}$ and if $G = (V, E)$ is such that $E' \subset E$, then we can remove the edges in $E \backslash E'$ from G without affecting its localization properties. Thus, given an arbitrary instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem, we can first search for those subgraphs of G that contain a graph from $\mathcal{H}$ as a spanning subgraph, and then remove any redundant edges in each of those subgraphs. By proceeding in such a manner, we can reduce the size of the given instance, whence its associated SDP (1.2) can be more efficiently solved.

To carry out the above program, we need some theoretical preparations. These will be detailed in the next section.

2.2. d**–lateration Graphs and Their Properties.** To begin, let us introduce the family of d–lateration graphs, which will be the main object of interest in our

study.

DEFINITION 2.1. *Let $d, l \geq 1$ be integers with $l \geq d+1$. An l–vertex graph $G = (V, E)$ is called a* d–lateration graph *if there exists an ordering $\{1, 2, \ldots, l\}$ of the vertices in V (called a* d–lateration ordering*) such that (i) the first $d+1$ vertices $1, 2, \ldots, d+1$ form a complete graph, and (ii) every vertex $j \geq d+2$ is connected to at least $d+1$ of the vertices $1, 2, \ldots, j-1$.*

The family of d–lateration graphs was studied in [3], where it was shown that a d–lateration graph is generically globally rigid in $\mathbb{R}^d$. This implies that if G is an $(n+m)$–vertex d–lateration graph with $n \geq 1$ sensors and $m \geq d+1$ anchors, then the instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ has a unique localization in $\mathbb{R}^d$ (i.e., there is a unique solution to the system in (1.1)) whenever $(\mathbf{d}, \bar{\mathbf{d}})$ and $\mathbf{a}$ induce a generic realization of G in $\mathbb{R}^d$. We now sharpen this result by showing that the instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is in fact uniquely d–localizable, i.e., it has a unique localization in $\mathbb{R}^l$ for all $l \geq d$.

THEOREM 2.2. *Let G be an $(n+m)$–vertex d–lateration graph with $n \geq 1$ sensors and $m \geq d+1$ anchors. Given an instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem, let $\tilde{\mathbf{x}} \in \mathbb{R}^{dn}$ be a feasible realization of the sensors. Suppose that the realization $(\tilde{\mathbf{x}}, \mathbf{a}) \in \mathbb{R}^{dn} \times \mathbb{R}^{dm}$ of G is generic. Then, the instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is uniquely d–localizable, and $(\tilde{\mathbf{x}}, \mathbf{a})$ is its unique localization.*

Proof. Let $V_s = \{1, \ldots, n\}$ be the set of sensors and $V_a = \{n+1, \ldots, n+m\}$ be the set of anchors. Since G is a d–lateration graph, there exists an ordering π of the vertices that satisfies the conditions in Definition 2.1. Let us first consider the case where the first $d+1$ vertices under π are all anchors and show that the desired conclusion holds. Towards that end, consider the vertex $\pi(d+2)$, which is the $(d+2)$–nd vertex under π. If it is an anchor, then there is nothing to argue. Hence, suppose that it is a sensor. Since it is connected to vertices $\pi(1), \ldots, \pi(d+1)$, we have the following constraints in the SDP (1.2):

$$\|a_{\pi(i)}\|_2^2 - 2a_{\pi(i)}^T x_{\pi(d+2)} + Y_{\pi(d+2),\pi(d+2)} = \bar{d}_{\pi(i),\pi(d+2)}^2 \qquad \text{for } i = 1, \ldots, d+1,$$

where $x_{\pi(d+2)}$ is the $\pi(d+2)$–nd column of the matrix $X \in \mathbb{R}^{d \times n}$ in (1.2), and $Y_{\pi(d+2),\pi(d+2)}$ is the $(\pi(d+2), \pi(d+2))$–nd entry of the matrix $Y \in \mathbb{R}^{n \times n}$ in (1.2). Upon eliminating $Y_{\pi(d+2),\pi(d+2)}$, we obtain the following system of linear equations:

$$(a_{\pi(i)} - a_{\pi(d+1)})^T x_{\pi(d+2)} = \frac{1}{2}\left(\|a_{\pi(i)}\|_2^2 - \|a_{\pi(d+1)}\|_2^2 + \bar{d}_{\pi(d+1),\pi(d+2)}^2 - \bar{d}_{\pi(i),\pi(d+2)}^2\right)$$
$$\text{for } i = 1, \ldots, d.$$

Since $(\tilde{\mathbf{x}}, \mathbf{a})$ is generic, the vectors $\{a_{\pi(1)} - a_{\pi(d+1)}, \ldots, a_{\pi(d)} - a_{\pi(d+1)}\}$ are linearly independent. Thus, there is a unique solution to the above system, namely, $x_{\pi(d+2)} = \tilde{x}_{\pi(d+2)}$. This in turn implies that $Y_{\pi(d+2),\pi(d+2)} = \|\tilde{x}_{\pi(d+2)}\|_2^2$. Moreover, since $Y - X^T X \succeq \mathbf{0}$ by the Schur complement and $(Y - X^T X)_{\pi(d+2),\pi(d+2)} = 0$, we conclude that

$$Y_{\pi(d+2),l} = Y_{l,\pi(d+2)} = \tilde{x}_{\pi(d+2)}^T x_l \qquad \text{for } l = 1, \ldots, n.$$

Now, suppose that for $i = d+2, \ldots, j$ with $\pi(i) \in V_s$, we have $x_{\pi(i)} = \tilde{x}_{\pi(i)}$, $Y_{\pi(i),\pi(i)} = \|\tilde{x}_{\pi(i)}\|_2^2$, and $Y_{\pi(i),l} = Y_{l,\pi(i)} = \tilde{x}_{\pi(i)}^T x_l$ for $l = 1, \ldots, n$. Consider vertex $\pi(j+1)$. Without loss of generality, we may assume that it is a sensor. Then, we have the

following constraints in the SDP (1.2):

$$\begin{aligned}
\|\tilde{x}_{\pi(l)}\|_2^2 - 2\tilde{x}_{\pi(l)}^T x_{\pi(j+1)} + Y_{\pi(j+1),\pi(j+1)} &= d_{\pi(l),\pi(j+1)}^2 \\
&\quad \text{for } (\pi(l),\pi(j+1)) \in E_{ss},\ l = 1,\ldots,j, \\
\|a_{\pi(l)}\|_2^2 - 2a_{\pi(l)}^T x_{\pi(j+1)} + Y_{\pi(j+1),\pi(j+1)} &= \bar{d}_{\pi(j+1),\pi(l)}^2 \\
&\quad \text{for } (\pi(j+1),\pi(l)) \in E_{sa},\ l = 1,\ldots,j.
\end{aligned}$$

Since vertex $\pi(j+1)$ is connected to at least $d+1$ of the vertices $\pi(1),\ldots,\pi(j)$, there are at least $d+1$ linear equations in the above system. Since $(\tilde{\mathbf{x}},\mathbf{a})$ is generic, we can extract $d+1$ independent linear equations from it. Then, by using the same argument as before, we see that $x_{\pi(j+1)} = \tilde{x}_{\pi(j+1)}$, $Y_{\pi(j+1),\pi(j+1)} = \|\tilde{x}_{\pi(j+1)}\|_2^2$, and

$$Y_{\pi(j+1),l} = Y_{l,\pi(j+1)} = \tilde{x}_{\pi(j+1)}^T x_l \qquad \text{for } l = 1,\ldots,n.$$

Hence, the inductive step is completed. In particular, we conclude that $Y = \tilde{X}^T\tilde{X}$, where $\tilde{X} = [\ \tilde{x}_1 \quad \cdots \quad \tilde{x}_n\ ]$, is the unique solution to (1.2). It then follows from [35, Theorem 2] that $(G,(\mathbf{d},\bar{\mathbf{d}}),\mathbf{a},d)$ is uniquely d–localizable.

In summary, we have shown that if $\mathbf{p} \in \mathbb{R}^{dl}$ is a generic realization of an l–vertex d–lateration graph $G = (V,E)$ (where $l \geq d+1$), and if the positions of the first $d+1$ vertices under the d–lateration ordering of V are given, then $\mathbf{p}$ can be found by solving systems of linear equations whenever the distances $d_{ij} = \|p_i - p_j\|_2$ (for $(i,j) \in E$) are given. Moreover, it will be the unique realization of G (with respect to the positions of the first $d+1$ vertices) in $\mathbb{R}^h$ for all $h \geq d$.

Now, let us consider the case where the first $d+1$ vertices under π are not all anchors. Suppose that there are two generic realizations $(\tilde{\mathbf{x}},\mathbf{a})$ and $(\tilde{\mathbf{y}},\mathbf{a})$ of G. Note that by our remark in the preceding paragraph, both realizations must lie in $\mathbb{R}^d$ and are unique with respect to the positions of the first $d+1$ vertices under π. This implies that there exists an isometry $T : \mathbb{R}^d \to \mathbb{R}^d$ that maps $(\tilde{\mathbf{x}},\mathbf{a})$ to $(\tilde{\mathbf{y}},\mathbf{a})$. However, since the $m \geq d+1$ generically positioned anchors are fixed by T, it follows that T must be the identity. In particular, we have $(\tilde{\mathbf{x}},\mathbf{a}) = (\tilde{\mathbf{y}},\mathbf{a})$, and the desired conclusion in the theorem statement follows. □

One of the important consequences of Theorem 2.2 is that it establishes the existence of sparse uniquely d–localizable instances of the sensor network localization problem (i.e., those whose graphs have only $O(|V|)$ edges). Recall that for such instances, the SDP relaxation (1.2) is exact, i.e., the SDP (1.2) will produce the unique localization of such instances. Thus, our result refutes a common belief in the literature (see, e.g., [4, 3, 5, 29, 11]) that every uniquely d–localizable instance must have $\Omega(|V|^2)$ edges.

Another consequence of Theorem 2.2 is that it suggests a way to simultaneously reduce the size of a given instance $(G,(\mathbf{d},\bar{\mathbf{d}}),\mathbf{a},d)$ of the sensor network localization problem and preserve its localization properties. Specifically, we can first try to identify subgraphs of G that contain a d–lateration graph as a spanning subgraph, and then remove any redundant edges in each of those subgraphs. Before we investigate this approach in more detail, however, let us provide further theoretical justification for it and discuss how the notion of unique d–localizability fits in the framework of rigidity theory.

2.3. Unique d–Localizability and Its Rigidity–Theoretic Counterpart. As mentioned in Section 2.1, when we consider the rigidity–theoretic aspects of a

graph, we do not make any distinction among its vertices. By contrast, in the definition of unique d–localizability, the distinction between sensor and anchor vertices is crucial. Thus, in order to discuss unique d–localizability in the context of rigidity theory, we need to somehow do away with the anchors. One possible way to achieve that is as follows:

DEFINITION 2.3. *Let $d, l \geq 1$ be integers. Let $G = (V, E)$ be an l–vertex graph, and let $\mathbf{p} = (p_1; \ldots; p_l) \in \mathbb{R}^{dl}$ be its realization in $\mathbb{R}^d$. We say that $(G, \mathbf{p})$ is* universally rigid *in $\mathbb{R}^d$ if for any realization $\mathbf{q} = (q_1; \ldots; q_l) \in \mathbb{R}^{hl}$ of G in $\mathbb{R}^h$ (where $h \geq 1$ is arbitrary), we have*

$$\Big[\|p_i - p_j\|_2 = \|q_i - q_j\|_2 \text{ for } (i,j) \in E\Big] \Longrightarrow \Big[\|p_i - p_j\|_2 = \|q_i - q_j\|_2 \text{ for } 1 \leq i < j \leq l\Big].$$

In other words, $\mathbf{p}$ is the unique (up to congruence) realization of G in any Euclidean space. If $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$ for all generic realizations $\mathbf{p} \in \mathbb{R}^{dl}$, then we say that G is generically universally rigid *in $\mathbb{R}^d$.*

Although the notion of universal rigidity was introduced long ago under the name *super stability* [15] (see also [2, 1, 22] and the references therein), there are still aspects of it that are not well understood. For instance, it is not entirely clear what the relationship is between unique d–localizability and universal rigidity in $\mathbb{R}^d$. The following theorem addresses this issue and makes the relationship explicit:

THEOREM 2.4. *Let $G = ((V_s, V_a), (E_{ss}, E_{sa}, E_{aa}))$ be a graph. Suppose that $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is a uniquely d–localizable instance of the sensor network localization problem, with $\mathbf{p} = (\tilde{\mathbf{x}}; \mathbf{a}) \in \mathbb{R}^{d(|V_s|+|V_a|)}$ being its unique localization in $\mathbb{R}^l$ for all $l \geq d$. Then, $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$.*

Conversely, let $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ be an instance of the sensor network localization problem, and let $\mathbf{p} = (\tilde{\mathbf{x}}; \mathbf{a}) \in \mathbb{R}^{d(|V_s|+|V_a|)}$ be a feasible realization of G in $\mathbb{R}^d$. Suppose that there exist $d+1$ affinely independent vectors in the family $\{a_i\}_{i \in V_a}$, and that $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$. Then, the instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is uniquely d–localizable.

Proof. Let $\mathbf{q} \in \mathbb{R}^{h(|V_s|+|V_a|)}$ be a realization of G in $\mathbb{R}^h$ for some $h \geq 1$. Since G contains the complete subgraph $G_a = (V_a, E_{aa})$, we can apply an isometry T to $\mathbf{q}$ so that $T(q_i) = a_i$ for all $i \in V_a$. However, since $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is uniquely d–localizable with $\mathbf{p} = (\tilde{\mathbf{x}}; \mathbf{a}) \in \mathbb{R}^{d(|V_s|+|V_a|)}$ being its unique localization in $\mathbb{R}^l$ for all $l \geq d$, we must have $T(\mathbf{q}) = \mathbf{p}$. Since T is an isometry, it follows that $\|p_i - p_j\|_2 = \|q_i - q_j\|_2$ for $1 \leq i < j \leq |V_s| + |V_a|$, whence $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$.

Conversely, suppose that $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$, and that $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is not uniquely d–localizable. Then, there exists another realization $\mathbf{p}' \in \mathbb{R}^{h|V|}$ of G in $\mathbb{R}^h$ for some $h \geq d$, such that $p'_i = (a_i; \mathbf{0}) \in \mathbb{R}^h$ for all $i \in V_a$. Since $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$, there exists an isometry T such that $T(\mathbf{p}') = \mathbf{p}$. On the other hand, since T fixes $\mathbf{a}$ and there are $d+1$ affinely independent vectors in $\{a_i\}_{i \in V_a}$, we conclude that T must be the identity. In particular, this implies that $\mathbf{p}' = \mathbf{p}$, which is a contradiction. Hence, the instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ is uniquely d–localizable, as desired. □

We remark that the affine independence assumption in the statement of Theorem 2.4 cannot be dropped. Indeed, consider the instance shown in Figure 2.1. Suppose that the anchors a_2, a_3, a_4 are collinear, and that there is an edge between the sensor x_1 and every anchor. Then, it is clear that the given instance is universally rigid in $\mathbb{R}^2$ for any placement of the anchors and sensor. However, it is not uniquely 2–localizable, as one can obtain another (congruent) realization by reflecting x_1 along the axis defined by the anchors a_2, a_3, a_4.

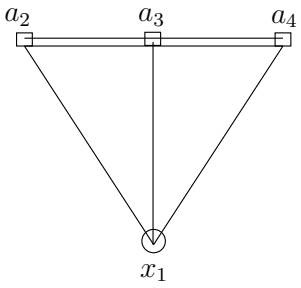
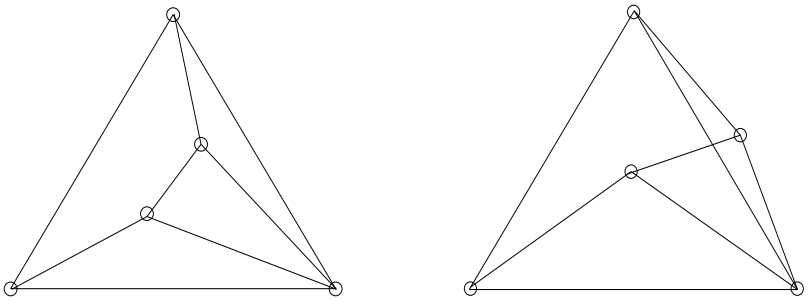


FIG. 2.1. *Importance of the affine independence assumption in Theorem 2.4.*

Theorem 2.4 has some interesting consequences. For instance, let $G = (V, E)$ be an l–vertex graph and $\mathbf{p} \in \mathbb{R}^{dl}$ be a realization of G in $\mathbb{R}^d$. Suppose that the family $\{p_i\}_{i \in V}$ contains $d + 1$ affinely independent vectors. Then, by Theorem 2.4 and the result in [35], we see that it is possible to check whether $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$ efficiently using SDP. Moreover, if $(G, \mathbf{p})$ is indeed universally rigid in $\mathbb{R}^d$, then the unique realization $\mathbf{p}$ can also be found efficiently using SDP. By contrast, Saxe [32] showed that it is $\mathcal{NP}$–hard to check whether $(G, \mathbf{p})$ is globally rigid in $\mathbb{R}^d$, and Aspnes et al. [4] showed that unless $\mathcal{RP} = \mathcal{NP}$, there does not exist an efficient algorithm for finding the unique realization $\mathbf{p}$ of a globally rigid instance $(G, \mathbf{p})$. Thus, it seems that the notion of universal rigidity is more amenable to algorithmic treatment than that of global rigidity. Furthermore, upon combining Theorems 2.2 and 2.4, we have the following corollary:

COROLLARY 2.5. *d–lateration graphs are generically universally rigid in $\mathbb{R}^d$ for all $d \geq 1$.*

In particular, if $G = (V, E)$ contains a spanning d–lateration subgraph $G' = (V, E')$, then the instances $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ and $(G', (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, d)$ have the same localization properties for almost all $(\mathbf{d}, \bar{\mathbf{d}})$ and $\mathbf{a}$. Such an observation forms the basis of our edge sparsification algorithm, which we will detail in the next section.



(a) An instance that is universally rigid in $\mathbb{R}^2$.

(b) An instance that is not universally rigid in $\mathbb{R}^2$.

FIG. 2.2. *Universal rigidity is not a generic property.*

We remark that universal rigidity is *not* a generic property [35, 33, 22]. In other words, given two realizations $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{dl}$ of an l–vertex graph G in $\mathbb{R}^d$, it is possible that $(G, \mathbf{p})$ is universally rigid in $\mathbb{R}^d$ while $(G, \mathbf{q})$ is not (see Figure 2.2 for an example). This should be contrasted with the notion of global rigidity, which is a generic

property [21]. Moreover, there is an efficiently computable characterization of generically globally rigid graphs, while no such characterization is known for generically universally rigid graphs. Naturally, it would be interesting to obtain an efficiently computable characterization of generically universally rigid graphs, and we leave this as an open question to the reader.

Before we leave this section, let us show how generically universally rigid graphs can be constructed in an incremental manner. We begin with a definition.

DEFINITION 2.6. *Let $d \geq 1$ be an integer. We say that a graph $G' = (V', E')$ is a* d–lateration extension *of another graph $G = (V, E)$, where $|V| \geq d+1$, if $V' = V \cup \{w\}$ with $w \notin V$, $E \subset E'$, and $|\{(w, v) \in E' : v \in V\}| \geq d + 1$. In other words, G' is obtained from G by adding a new vertex to G and connecting it to at least $d+1$ vertices of G.*

THEOREM 2.7. *Let $G = (V, E)$ be generically universally rigid in $\mathbb{R}^d$ with $|V| \geq d + 1$, and let $G' = (V', E')$ be a d–lateration extension of G. Then, G' is generically universally rigid in $\mathbb{R}^d$.*

Proof. Let $V = \{1, \ldots, l\}$ and $V' = \{1, \ldots, l, l + 1\}$. Define $\tilde{G}' = (V', \tilde{E}')$, where $\tilde{E}' = E' \cup \{(i, j) : 1 \leq i < j \leq l\}$. Note that since G is generically universally rigid in $\mathbb{R}^d$, any two generic realizations $\mathbf{p} = (p_i)_{i \in V'}$ and $\mathbf{q} = (q_i)_{i \in V'}$ of G' that satisfy $\|p_i - p_j\|_2 = \|q_i - q_j\|_2$ for all $(i, j) \in E'$ will also satisfy $\|p_i - p_j\|_2 = \|q_i - q_j\|_2$ for all $(i, j) \in \tilde{E}'$. Thus, in order to show that G' is generically universally rigid in $\mathbb{R}^d$, it suffices to show that $\tilde{G}'$ is generically universally rigid in $\mathbb{R}^d$. Towards that end, let $\mathbf{p} = (p_1; \ldots; p_{l+1}) \in \mathbb{R}^{d(l+1)}$ be a generic realization of $\tilde{G}'$ in $\mathbb{R}^d$, and consider the instance $(\tilde{G}', (\emptyset, \bar{\mathbf{d}}), \mathbf{a}, d)$ of the sensor network localization problem, where

$$\bar{\mathbf{d}} = (\|p_{l+1} - p_i\|_2)_{i \in V}, \quad \mathbf{a} = (p_i)_{i \in V}.$$

In other words, we treat the vertices $1, 2, \ldots, l$ as anchors and the vertex $l + 1$ as the only sensor in the instance. Upon following the argument in the proof of Theorem 2.2, we see that $(\tilde{G}', (\emptyset, \bar{\mathbf{d}}), \mathbf{a}, d)$ is uniquely d–localizable, with $\mathbf{p}$ being its unique localization in $\mathbb{R}^l$ for all $l \geq d$. Thus, by Theorem 2.4, we conclude that $(\tilde{G}', \mathbf{p})$ is universally rigid in $\mathbb{R}^d$. Since $\mathbf{p}$ is an arbitrary generic realization of $\tilde{G}'$ in $\mathbb{R}^d$, it follows that $\tilde{G}'$, and hence G', is generically universally rigid in $\mathbb{R}^d$. □

Another construction is based on taking the union of generically universally rigid graphs. Specifically, we prove the following theorem:

THEOREM 2.8. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two generically universally rigid graphs in $\mathbb{R}^d$ with $|V_1 \cap V_2| \geq d+1$. Then, the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ is also generically universally rigid in $\mathbb{R}^d$.*

Proof. Let $\mathbf{p} \in \mathbb{R}^{d|V_1 \cup V_2|}$ be a generic realization of G in $\mathbb{R}^d$, and let $\mathbf{q} \in \mathbb{R}^{h|V_1 \cup V_2|}$ be an arbitrary realization of G in $\mathbb{R}^h$ for some $h \geq 1$. Since G_1 is generically universally rigid in $\mathbb{R}^d$, there exists an isometry T_1 such that $T_1(\mathbf{q})|_{G_1} = \mathbf{p}|_{G_1}$. Here, $T_1(\mathbf{q})|_{G_1}$ (resp. $\mathbf{p}|_{G_1}$) denotes the restriction of $T_1(\mathbf{q})$ (resp. $\mathbf{p}$) onto the coordinates corresponding to the vertices of G_1. By a similar argument, there exists an isometry T_2 such that $T_2(\mathbf{q})|_{G_2} = \mathbf{p}|_{G_2}$. Now, observe that there exists an isometry T such that $T(T_1(\mathbf{q}))|_{G_2} = T_2(\mathbf{q})|_{G_2} = \mathbf{p}|_{G_2}$. Since T fixes the generically positioned vertices in $V_1 \cap V_2$ and since $|V_1 \cap V_2| \geq d + 1$ by assumption, it follows that T must be the identity. In particular, since $T_1(\mathbf{q})|_{G_1} = \mathbf{p}|_{G_1}$ and $T_1(\mathbf{q})|_{G_2} = \mathbf{p}|_{G_2}$, we conclude that $T_1(\mathbf{q}) = \mathbf{p}$. This completes the proof. □

3. Graph Sparsification and Reduction. Recall that given an instance of the sensor network localization problem, our goal is to speed up the solution time of its associated SDP relaxation (1.2). A natural way to achieve this is to reduce the

number of constraints in (1.2), which corresponds to removing edges from the input graph. Such an approach has indeed been pursued by several groups of researchers. For instance, Wang et al. [40] proposed to impose an upper bound $\lambda \geq 1$ on the degree of each vertex (i.e., both sensors and anchors) of the graph G and use an edge selection rule to determine which edge to keep, so that the degree at each vertex is bounded above by λ. In their simulations, they set $\lambda = 7$ for the case where $d = 2$, and the edges are chosen at random. Later, Kim et al. [27] adopted the same idea for edge reduction, except that their upper bound on the degree of each vertex has the form $\min\{\deg(u, E_{ss} \cup E_{sa}), \kappa\}$, where $\deg(u, E_{ss} \cup E_{sa})$ is the number of edges in $E_{ss} \cup E_{sa}$ that are incident to the vertex u, and $\kappa \geq d + 1$ is some integer. Their edge selection rule is also different, in that for each vertex u, the edges in E_{sa} that are incident to u are chosen before those in E_{ss}. Simulation results in [27] showed that such a heuristic does indeed perform better than the one proposed by Wang et al. [40].

Although the aforementioned edge sparsification heuristics do in general reduce the size of a given instance of the sensor network localization problem and hence speed up the solution time of its associated SDP relaxation, they are quite ad hoc in nature and may remove edges that are crucial to an accurate localization of the sensors. To illustrate this, consider the graph $G_n = (V, E)$, where $V = \{1, 2, \ldots, n + 3\}$ and

$$E = \{(i, j) : 1 \leq i < j \leq 3\} \cup \bigcup_{j=4}^{n+3} \{(i, j) : i = 1, 2, 3\}$$

(see Figure 3.1 for an illustration of the case where $n = 5$). We designate the first 3 vertices as anchors, and the rest as sensors. Suppose that the distances $\bar{\mathbf{d}} = (\bar{d}_{ij})_{(i,j)\in E}$ are given, and that the instance $(G, (\emptyset, \bar{\mathbf{d}}), \mathbf{a}, 2)$ has a generic realization $(\tilde{\mathbf{x}}, \mathbf{a}) \in \mathbb{R}^{2n} \times \mathbb{R}^{2\times 3}$. Since G_n is a 2–lateration graph, it is uniquely 2–localizable, and hence $(\tilde{\mathbf{x}}, \mathbf{a})$ is its unique localization in $\mathbb{R}^2$. However, if we remove *any* of the anchor–sensor edges (i.e., edges of the form (i, j), where $1 \leq i \leq 3$ and $4 \leq j \leq n + 3$), then the instance will no longer be uniquely 2–localizable. In particular, if we restrict the degree of each vertex in G_n to be at most a constant that is less than the maximum degree (as is done, e.g., in [40, 27]), then the SDP relaxation associated with the resulting sparsified instance will no longer be exact, and we will not be able to recover $(\tilde{\mathbf{x}}, \mathbf{a})$ from the SDP. In view of the above discussion, we are thus interested in developing edge sparsification heuristics that can preserve the localization properties of the input.

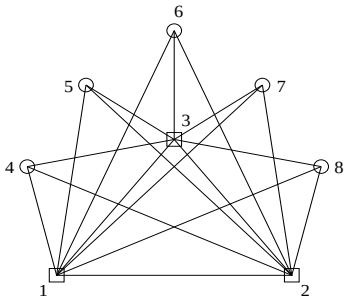


FIG. 3.1. *The graph* G_5.

3.1. The Equivalent Edge Sparsification Heuristic. Our edge sparsification heuristic is motivated by the results developed in Section 2. The high–level idea is

Algorithm 1 Equivalent Edge Sparsification (EES) Heuristic

Input: A graph $G = (V, E)$.
Output: An edge–sparsified graph EES(G).

1: Set $S \leftarrow \emptyset$, $T \leftarrow V$, $E' \leftarrow \emptyset$. The set E' contains the edges in E that will be included in the final output.
2: **if** there exists a 3–clique $K_3 = \{i_1, i_2, i_3\}$ in G **then**
3: set $S \leftarrow S \cup K_3$, $T \leftarrow T \backslash K_3$, $E' \leftarrow \{(i_{j_1}, i_{j_2}) : 1 \leq j_1 < j_2 \leq 3\}$
4: **else**
5: return G and stop
6: **end if**
7: **while** $|S| < |V|$ **do**
8: Let $i \in T$ be such that $|\{(i,j) \in E : j \in S\}| \geq 3$. Set

$$S \leftarrow S \cup \{i\}, \quad T \leftarrow T \backslash \{i\}, \quad E' \leftarrow E' \cup \{(i, j_k) : j_k \in S,\, k = 1, 2, 3\}.$$

The vertices $i \in T$ and $j_1, j_2, j_3 \in S$ are chosen according to some pre–specified rule.
9: **if** $|\{(i,j) \in E : j \in S\}| < 3$ for all $i \in T$ **then**
10: let $G_T = (T, E_T)$, where $E_T = \{(i,j) \in E : i, j \in T\}$, and compute $(T, E'_T) =$ EES(G_T)
11: set $E' \leftarrow E' \cup \{(i,j) \in E : i \in S,\, j \in T\} \cup E'_T$, $S \leftarrow S \cup T$, $T \leftarrow \emptyset$
12: **end if**
13: **end while**
14: return (V, E')

straightforward—decompose the input graph into a number of d–lateration subgraphs and remove any redundant edges in those subgraphs. For the sake of concreteness, let us describe our edge sparsification heuristic—which we call the *Equivalent Edge Sparsification (EES)* heuristic—for the case where $d = 2$ (which, of course, is a case of practical interest); see Algorithm 1. The generalization to $d \geq 3$ will be straightforward.

Note that in line 8 of the heuristic, there is some freedom in choosing which vertex i is to enter S and which three edges that connect i to S are to be included in E'. In Section 5, we shall discuss the specific selection rules used in our simulations. For now, let us assume that they are chosen in an arbitrary fashion.

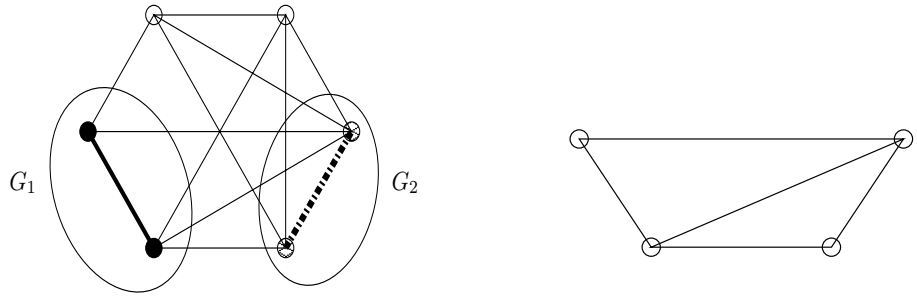
3.2. Analysis of the EES Heuristic. Naturally, we are interested in determining various theoretical properties of the EES heuristic. Towards that end, let us first analyze the structure of the graph produced by the heuristic. We begin with a definition:

Definition 3.1. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be vertex–disjoint subgraphs of a graph $G = (V, E)$. The* graph join *of G_1 and G_2 under G is the graph $G_1 +^G G_2$ defined by*

$$G_1 +^G G_2 = \big(V_1 \cup V_2, E_1 \cup E_2 \cup \{(u,v) \in E : u \in V_1,\, v \in V_2\}\big).$$

For an illustration of the definition, see Figure 3.2.

Theorem 3.2. *Let EES(G) be the output of the EES heuristic when applied to*



(a) The graph G and its vertex–disjoint subgraphs G_1 and G_2.

(b) The graph join $G_1 +^G G_2$.

FIG. 3.2. *The graph join of G_1 and G_2 under G.*

the input graph $G = (V, E)$. Then, we have the following decomposition:

$$EES(G) = G_0 +^G G_1 +^G G_2 +^G \cdots +^G G_k. \tag{3.1}$$

Here, $k \geq 0$ is some integer, $G_i = (V_i, E_i)$ is a minimal 2–lateration graph (i.e., $G_i' = (V_i, E_i')$ is not a 2–lateration graph for any $E_i' \subsetneq E_i$) for $i = 0, 1, \ldots, k-1$, $G_k = (V_k, E_k)$ is either a minimal 2–lateration graph or a triangle–free graph, and $V_0, \ldots, V_k$ are mutually disjoint with $V = \cup_{i=0}^k V_i$. In particular, each G_i is a subgraph of G, and the number of edges in EES(G) is bounded above by

$$\begin{cases} O(k|V|) & \text{if } G_k \text{ is a minimal 2–lateration graph,} \\ O(k|V| + |V_k|^2) & \text{if } G_k \text{ is a triangle–free graph.} \end{cases}$$

Proof. The desired decomposition follows directly from the construction of the EES heuristic. To determine the number of edges in EES(G), we first note that the number of edges in an l–vertex minimal 2–lateration graph is $O(l)$. Thus, if G_k is a minimal 2–lateration graph, then the number of edges in EES(G) can be bounded above by

$$\sum_{i=0}^{k} O(|V_i|) + 2\sum_{j=1}^{k}\sum_{i=j}^{k} O(|V_i|) = O(k|V|),$$

because we have $|V| = \sum_{i=0}^k |V_i|$. On the other hand, if G_k is a triangle–free graph, then by using the trivial bound $|E_k| = O(|V_k|^2)$, we conclude that the number of edges in EES(G) is bounded above by

$$\sum_{i=0}^{k-1} O(|V_i|) + O(|V_k|^2) + 2\sum_{j=1}^{k-1}\left(\sum_{i=j}^{k-1} O(|V_i|) + O(|V_k|)\right) = O(k|V| + |V_k|^2).$$

This completes the proof. ∎

Theorem 3.2 shows that the graph returned by the EES heuristic does indeed have a small number of edges in general. Another important property of the EES heuristic is that it preserves the localization properties of the input graph. We first prove the following theorem:

THEOREM 3.3. *Let $G = ((V_s, V_a), (E_{ss}, E_{sa}, E_{aa}))$ be a graph, and consider an arbitrary feasible instance $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ of the sensor network localization problem. Suppose that $\mathbf{p} = (\tilde{\mathbf{x}}; \mathbf{a}) \in \mathbb{R}^{2(|V_s|+|V_a|)}$ is a generic realization of the instance $(EES(G), (\mathbf{d}, \bar{\mathbf{d}})|_{EES(G)}, \mathbf{a}, 2)$ in $\mathbb{R}^2$, where $(\mathbf{d}, \bar{\mathbf{d}})|_{EES(G)}$ is the restriction of $(\mathbf{d}, \bar{\mathbf{d}})$ onto the edges of $EES(G)$. Then, $\mathbf{p}$ is also a generic realization of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ in $\mathbb{R}^2$.*

Remark. Since $\mathrm{EES}(G)$ is a subgraph of G, *any* realization $\mathbf{p} \in \mathbb{R}^{2(|V_s|+|V_a|)}$ of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ will also be a realization of $(\mathrm{EES}(G), (\mathbf{d}, \bar{\mathbf{d}})|_{EES(G)}, \mathbf{a}, 2)$. Thus, the converse to Theorem 3.3 holds even without the genericity assumption.

Proof. Let $E \equiv E_{ss} \cup E_{sa} \cup E_{aa}$, and consider the decomposition of $\mathrm{EES}(G)$ given by (3.1). Recall that by Theorem 3.2, $G_0, \ldots, G_{k-1}$ are 2–lateration graphs. Thus, by Corollary 2.5 and the genericity of $\mathbf{p}$, we see that $(G_j, \mathbf{p}|_{G_j})$ is universally rigid in $\mathbb{R}^2$ for $j = 0, 1, \ldots, k-1$, where $\mathbf{p}|_{G_j}$ is the restriction of $\mathbf{p}$ onto the vertices of G_j. In particular, since $\mathbf{p}$ satisfies the distance constraints corresponding to the edges in E_j and $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ is feasible, we see that $\mathbf{p}$ must also satisfy the distance constraints corresponding to the edges in $(V_j \times V_j) \cap E$ for $j = 0, 1, \ldots, k-1$. This, together with the construction of the EES heuristic, implies that $\mathbf{p}$ satisfies the distance constraints corresponding to the edges in $((V \backslash V_k) \times V) \cap E$.

Now, if G_k is a 2–lateration graph, then by the above argument, we see that $\mathbf{p}$ also satisfies the distance constraints corresponding to the edges in $(V_k \times V) \cap E$. Otherwise, G_k is a triangle–free graph, and we have $E_k = (V_k \times V_k) \cap E$ by the construction of the EES heuristic. In particular, we see that $\mathbf{p}$ again satisfies the distance constraints corresponding to the edges in $(V_k \times V) \cap E$. Thus, in both cases, we can conclude that $\mathbf{p}$ satisfies the distance constraints corresponding to the edges in

$$\big[((V \backslash V_k) \times V) \cap E\big] \cup \big[(V_k \times V) \cap E\big] = (V \times V) \cap E = E.$$

In other words, $\mathbf{p}$ is a generic realization of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ in $\mathbb{R}^2$, as desired. □

In a similar fashion, we can prove the following theorem, which can be viewed as a partial converse to Theorem 3.3:

THEOREM 3.4. *Let $G = ((V_s, V_a), (E_{ss}, E_{sa}, E_{aa}))$ be a graph. Suppose that $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ is a uniquely 2–localizable instance of the sensor network localization problem, with $\mathbf{p} = (\tilde{\mathbf{x}}; \mathbf{a}) \in \mathbb{R}^{2(|V_s|+|V_a|)}$ being its unique localization in $\mathbb{R}^l$ for all $l \geq 2$. If $\mathbf{p}$ is generic, then the instance $(EES(G), (\mathbf{d}, \bar{\mathbf{d}})|_{EES(G)}, \mathbf{a}, 2)$ is also uniquely 2–localizable, and $\mathbf{p}$ is its unique localization in $\mathbb{R}^l$ for all $l \geq 2$.*

Proof. Let $\mathbf{q} \in \mathbb{R}^{h(|V_s|+|V_a|)}$ be an arbitrary localization in $\mathbb{R}^h$ of the instance $(\mathrm{EES}(G), (\mathbf{d}, \bar{\mathbf{d}})|_{EES(G)}, \mathbf{a}, 2)$, for some $h \geq 1$. Our goal is to show that $\mathbf{q}$ is also a feasible localization of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$, from which it would follow that $\mathbf{p} = \mathbf{q}$, as $\mathbf{p}$ is the unique localization of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ in $\mathbb{R}^l$ for all $l \geq 2$. Towards that end, we again use the decomposition of $\mathrm{EES}(G)$ given by (3.1) and the argument in the proof of Theorem 3.3 to conclude that $(G_j, \mathbf{p}|_{G_j})$ is universally rigid in $\mathbb{R}^2$ for $j = 0, 1, \ldots, k-1$. In particular, since $\|p_u - p_v\|_2 = \|q_u - q_v\|_2$ for all $(u, v) \in E_j$, we have $\|p_u - p_v\|_2 = \|q_u - q_v\|_2$ for all $(u, v) \in V_j \times V_j$. This implies that for all $i \in V_0 \cup \cdots \cup V_{k-1}$, we have

$$\|p_i - p_u\|_2 = \|q_i - q_u\|_2 \quad \text{for all } (i, u) \in E \equiv E_{ss} \cup E_{sa} \cup E_{aa}. \tag{3.2}$$

Now, if G_k is a 2–lateration graph, then by the above argument, we see that (3.2) holds for all $i \in V_k$ as well. Otherwise, G_k is a triangle–free graph, and we have $E_k = (V_k \times V_k) \cap E$ by the construction of the EES heuristic. In particular, we see

that (3.2) also holds in this case. Thus, in both cases, we have $\|p_u - p_v\|_2 = \|q_u - q_v\|_2$ for all $(u,v) \in E$, which implies that $\mathbf{q}$ is a feasible localization of $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$. Since $(G, (\mathbf{d}, \bar{\mathbf{d}}), \mathbf{a}, 2)$ is uniquely 2–localizable, we conclude that $\mathbf{p} = \mathbf{q}$, as desired. □

We remark that although Theorems 3.2 to 3.4 are stated for the case where $d = 2$, they can be generalized to other values of d in a straightforward manner.

Finally, let us discuss the implementation and analyze the computational complexity of the EES heuristic. At the beginning, we construct a global list $\mathcal{L}$ of all 3–cliques in the input graph G and maintain, for each vertex $i \in V$, a list $\mathcal{L}_i$ of 3–cliques to which it belongs. All of these can be done in $O(|V|^3)$ time. Lines 1 and 8 can be implemented in $O(|V|)$ and $O(|S| \cdot |V|)$ time, respectively. Given the list $\mathcal{L}$, line 2 can be implemented in $O(1)$ time. Now, if the condition in line 9 is satisfied, then before proceeding to line 10, we will first delete from $\mathcal{L}$ those 3–cliques that have at least one vertex in S using the lists $\{\mathcal{L}_i\}_{i \in S}$. Such a pruning step allows us to conclude that throughout the course of the algorithm, each element in $\mathcal{L}$ is accessed at most a constant number of times. As a result, the total runtime of the EES heuristic can be bounded by $O(|V|^3)$. Note that the complexity of the EES heuristic is much lower than that required for solving the SDP (1.2). This justifies the use of the EES heuristic as a preprocessing procedure for speeding up the solution time of (1.2).

4. Further Processing of the SDP Relaxation. In Section 3, we developed a novel edge sparsification heuristic that can provably preserve the localization properties of the input. To the best of our knowledge, this is the first heuristic with such a theoretical guarantee. The instance obtained after we apply the edge sparsification heuristic typically has fewer distance constraints, and hence its associated SDP relaxation can already be solved faster than the SDP relaxation associated with the original instance. However, using recently developed speedup techniques (see, e.g., [20, 26, 27]), one can further improve the computational efficiency of solving those SDP relaxations. To illustrate those techniques, consider the following general (primal) standard form SDP:

$$(P) \qquad \begin{array}{lll} \inf & A_0 \bullet X & \\ \text{subject to} & A_i \bullet X = b_i & \text{for } i = 1, \ldots, s, \\ & X \succeq \mathbf{0}, & \end{array}$$

where $A_0, A_1, \ldots, A_s$ and X are symmetric $n \times n$ matrices. One of the main ideas in [20, 26] is to formulate an equivalent SDP in which the $n \times n$ positive semidefinite cone constraint in (P) is replaced by a number of potentially smaller positive semidefinite cone constraints. The motivation for such a transformation is that the smaller positive semidefinite cone constraints can be handled much more efficiently by state–of–the–art interior–point algorithms. To formulate such an SDP, we first let $\mathcal{V} = \{1, \ldots, n\}$ be the set of row/column indices of the given data matrices $A_0, A_1, \ldots, A_s$, and define the so–called *aggregated sparsity pattern* $\mathcal{E}$ of $A_0, A_1, \ldots, A_s$ by (see [20])

$$\mathcal{E} = \big\{(i,j) \in \mathcal{V} : (A_k)_{ij} \neq 0 \text{ for some } k \in \{0, 1, \ldots, s\}\big\}.$$

(The word "sparsity" here should not be confused with that used in the phrase "edge sparsification". The latter has the meaning of removing edges from a certain graph.) Upon treating the pair $(\mathcal{V}, \mathcal{E})$ as a graph $\mathcal{G}$, we consider one of its chordal extensions $\mathcal{G}'$,

i.e., $\mathcal{G}'$ is a chordal graph[2] obtained from $\mathcal{G}$ by adding edges to it. Let $C_1, \ldots, C_t \subset \mathcal{V}$ be a family of maximal cliques in $\mathcal{G}'$, i.e., each subset $C_j \subset \mathcal{V}$ of vertices induces a clique in $\mathcal{G}'$, and there does not exist a $C'_j \supsetneq C_j$ such that C'_j induces a clique in $\mathcal{G}'$ for $j = 1, \ldots, t$. Then, it can be shown [20] that the SDP

$$(P_c) \qquad \begin{array}{lll} \inf & A_0 \bullet X & \\ \text{subject to} & A_i \bullet X = b_i & \text{for } i = 1, \ldots, s, \\ & X_{C_j, C_j} \succeq \mathbf{0} & \text{for } j = 1, \ldots, t, \end{array}$$

is equivalent to (P). Here, X_{C_j,C_j} is the principal sub–matrix of X whose rows and columns are those indexed by C_j. The upshot of (P_c) is that the positive semidefinite cone constraints are typically smaller and hence (P_c) can be solved much faster than (P). Note, however, that the SDP (P_c) is not yet in standard form, as some variables may appear in several of the positive semidefinite cone constraints. To transform (P_c) into a standard form SDP, we may use either one of the following methods:

1. Treat each X_{C_j,C_j} as an independent matrix cone, and add linkage constraints for those variables that appear in multiple X_{C_j,C_j}'s. We refer the reader to [20, Section 4] for details.
2. View each X_{ij} as a dual variable, and treat (P_c) as an SDP in *dual* standard form. We refer the reader to [27, Section 4.4] for details.

The transformed SDP can then be solved by any standard SDP solver.

5. Simulation Results. In this section, we present some preliminary computational results to show the effectiveness of our edge sparsification approach to solving the sensor network localization problem. All test problems are solved on a Windows PC with 2.80GHz CPU and 2GB Memory using SeDuMi 1.2 [36].

5.1. Effectiveness of the EES Heuristic. To demonstrate the effectiveness of the EES heuristic, we randomly place 500 nodes over the unit square $[0, 1] \times [0, 1]$. Two vertices are connected by an edge if their distance is at most ρ, where $\rho = 0.05, 0.06, \ldots, 0.15$ (we shall refer to the parameter ρ as the *radio range* of the sensors). Moreover, in line 8 of the EES heuristic, we choose the vertices $i \in T$ and $j_1, j_2, j_3 \in S$ according to the following strategy. First, we label the vertices using the Reverse Cuthill–McKee Ordering [18] (this typically can speed up matrix computations). Then, we pick the vertices $i \in T$ and $j_1, j_2, j_3 \in S$ so that $\{j_1, j_2, j_3\}$ forms a 3–clique. If this is not possible, then we just pick the vertex $i \in T$ with the smallest label, and among the neighbors of i that are in S, we choose three that have the largest labels.

The percentage of edges kept by the EES heuristic is shown in Figure 5.1. For each value of ρ, the percentage is averaged over 10 randomly generated instances. As the plot demonstrates, the reduction in the number of edges becomes more substantial as the radio range increases. One particular example is shown in Figure 5.2, with $\rho = 0.1$. There are 3482 edges in the 500–node network before we apply the EES heuristic. After we apply the heuristic, only 1494 edges are left.

5.2. Performance of EES–Preprocessed SDP–Based Localization Algorithms. Next, we consider various SDP–based localization algorithms and study their performance when the input instance is preprocessed by two different edge sparsification heuristics, namely, the EES heuristic described in Section 3.1 and the edge

[2]Recall that an undirected graph is *chordal* if every cycle of length greater than 3 has a *chord*, i.e., an edge that connects two non–adjacent vertices on the cycle.

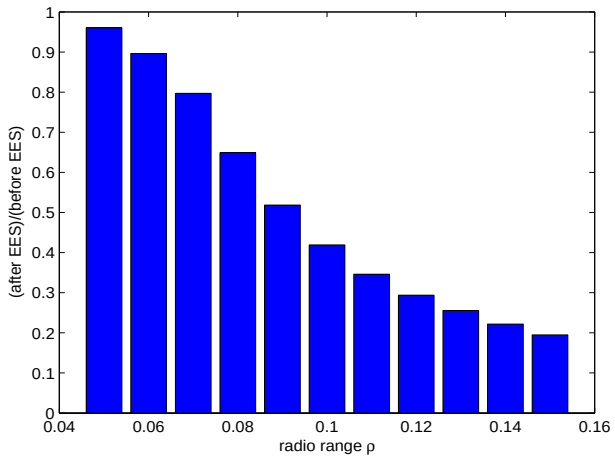


FIG. 5.1. *The ratios of the number of edges before and after applying the EES heuristic to randomly generated* 500*–node sensor networks.*

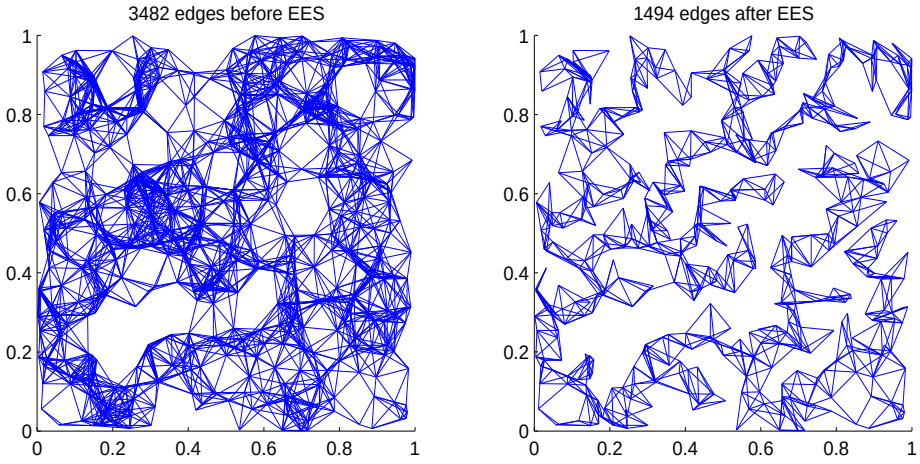


FIG. 5.2. *Effect of the EES heuristic on a randomly generated* 500*–node sensor network.*

sparsification heuristic of Kim et al. [27] (which we refer to as ES). Specifically, for $n = 100, 200, 400, 800, 1600$, we set $\rho = 2/\sqrt{n}$ and generate 5 random n–node sensor networks over the unit square with 90% sensors and 10% anchors (i.e., we generate 25 random sensor network localization instances in total). Then, we solve the resulting sensor network localization problems using six different schemes:

- FSDP: the SDP formulation (1.2) of Biswas and Ye [9]
- SFSDP: the so–called *sparse FSDP* formulation of Kim et al. [27]
- EES–FSDP: FSDP after applying the EES heuristic

- EES–SFSDP: SFSDP after applying the EES heuristic
- ES–FSDP: FSDP after applying the ES heuristic
- ES–SFSDP: SFSDP after applying the ES heuristic

5.2.1. Computation Time Comparison. We record the average computation time of each of the schemes above, where the average is taken over 5 instances for each $n = 100, 200, 400, 800, 1600$. The results are listed in Table 5.1. Note that the runtime of the EES heuristic is not included in the runtime of EES–FSDP or EES–SFSDP, and is listed separately in the last column of Table 5.1. As can be seen from the table, the solution time of the SDP formulations in question is significantly reduced after we preprocess the input with the EES heuristic. However, the schemes that use the EES heuristic typically run slower than those that use the ES heuristic. This is because the EES heuristic ensures that the removed edges will not affect the localization properties of the input. By contrast, the ES heuristic do not have such a guarantee.

n	FSDP	EES–FSDP	ES–FSDP	SFSDP	EES–SFSDP	ES–SFSDP	EES
100	5.0	3.9	4.8	1.4	1.0	1.1	0.0
200	29.0	24.8	26.8	3.5	2.5	2.0	0.3
400	265.0	206.9	211.9	11.3	5.2	4.3	2.0
800	2577.4	1648.3	1601.4	38.2	9.2	9.4	11.6
1600	*	*	*	163.9	19.9	23.8	51.7

TABLE 5.1
Computation time comparison (in seconds); "" = Out of Memory.*

5.2.2. Accuracy Comparison. Now, let us compare the accuracy of the various schemes. We follow [9, 37, 40, 27] and adopt the Root Mean Square Distance (rmsd) as a measure of discrepancy between the computed locations and true locations of the sensors:

$$\text{rmsd} = \left(\frac{1}{n} \sum_{i=1}^{n} \|x_i - \bar{x}_i\|^2 \right)^{1/2}.$$

Here, $x_i \in \mathbb{R}^d$ is the position of sensor i as computed by any one of the schemes above, and $\bar{x}_i \in \mathbb{R}^d$ is its true position. In Table 5.2, we record the average rmsd values of FSDP, EES–FSDP and ES–FSDP for the experiments conducted in Section 5.2.1. The discrepancy in the average rmsd values between FSDP and EES–FSDP can be attributed to the fact that the input instances are numerically different, thus causing the solutions produced by the interior–point method to be slightly different.

n	100	200	400	800
FSDP	2.55e-02	5.29e-03	1.18e-02	5.52e-03
EES–FSDP	2.56e-02	6.12e-03	1.26e-02	5.48e-03
ES–FSDP	8.32e-02	2.29e-02	1.80e-02	9.86e-03

TABLE 5.2
rmsd comparison among FSDP, EES–FSDP and ES–FSDP.

Since the EES heuristic preserves the localization properties of the input instance, the schemes that use it should perform better than those that use other, more ad hoc,

edge sparsification heuristics. As can be seen from Table 5.2, EES–FSDP indeed achieves a better accuracy than ES–FSDP in terms of the average rmsd. One particular example is shown in Figure 5.3, where we have a sensor network with 90 sensors, 10 anchors, and $\rho = 0.2$. We compare the sensor positions returned by EES–FSDP with those returned by ES–FSDP. From the figure, we see that EES–FSDP localizes the sensors quite accurately (rmsd=8.37e-05), while ES–FSDP fails to localize some of the sensors correctly (rmsd=8.96e-02).

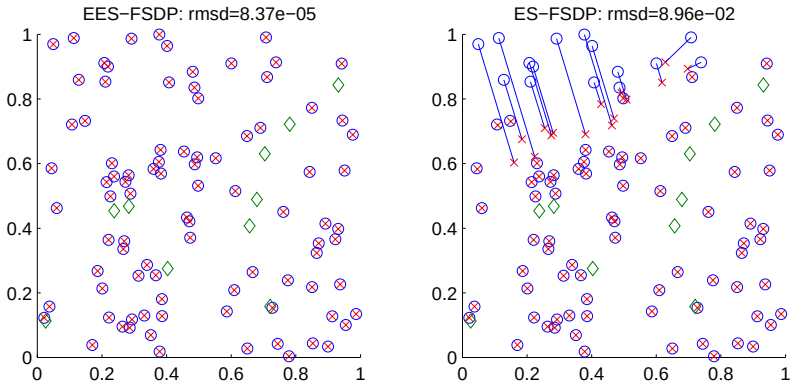


FIG. 5.3. *Comparison for EES and ES: localization (×) of a sensor network with* 90 *sensors (○),* 10 *anchors (◊) and* $\rho = 0.2$.

As a further illustration, Figure 5.4 shows the result when we first apply EES–SFSDP to one of the input instances, and then refine the solution using the gradient descent procedure in [7]. We use "○" to denote the true location of a sensor (whose position is not known to the SDP) and "×" to denote the location of a sensor computed by the algorithm. From the figure, we see that EES–SFSDP with gradient descent can give a very accurate localization of the sensors.

6. Conclusion and Future Directions. In this paper, we developed a novel edge sparsification heuristic for the sensor network localization problem and showed that it possesses some nice theoretical properties. Specifically, we showed that our heuristic can reduce the number of edges (and hence distance measurements) in the input network, while at the same time preserving its localization properties. An important component in our heuristic is a graph decomposition procedure, which allows us to identify certain sparse generically universally rigid subgraphs of the input graph. We then used our edge sparsification heuristic to speed up existing convex relaxation–based localization algorithms, and simulation results showed that our approach is promising. Moreover, our speedup technique can be applied in conjunction with existing ones (see, e.g., [20, 26, 27]), and they complement each other well.

The notion of universal rigidity plays an important role in the development of our edge sparsification heuristic. In the future, it would be interesting to further investigate the properties of universally rigid instances, as they are still not fully understood. Moreover, it would be nice to develop more efficient algorithms for verifying and realizing universally rigid instances. Finally, in view of the complexity results in [32, 4, 3, 12], it is worth identifying other classes of efficiently realizable *globally rigid* instances. Such instances could potentially lead to more efficient edge sparsification

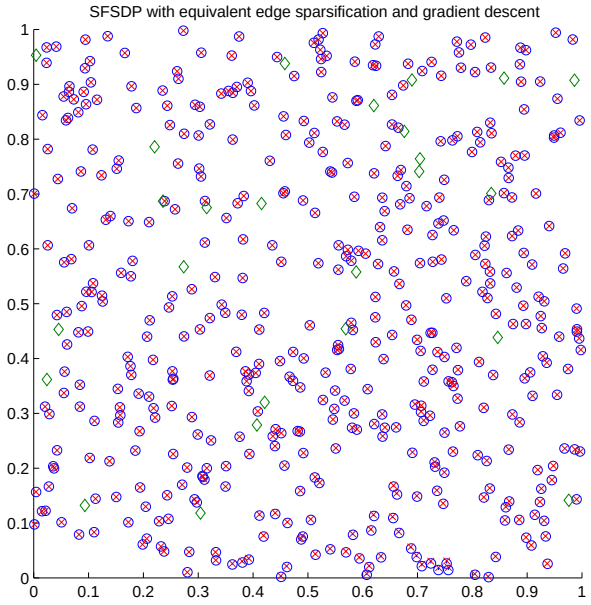


FIG. 5.4. *Localization (×) of a sensor network with* 475 *sensors (∘),* 25 *anchors (◊) and* $\rho = 0.1$*, computed by EES–SFSDP with gradient descent.*

heuristics, and hence faster algorithms for solving the sensor network localization problem.

Acknowledgments. The authors would like to thank Professor Wenan Zang for discussions related to the d–lateration graphs, as well as the referees for their insightful comments and suggestions. Part of this work was done when Anthony Man–Cho So was visiting the Department of Statistics and Decision Support Systems (ISDS) of the University of Vienna. He would like to thank Professor Immanuel Bomze for his hospitality during the visit.

REFERENCES

[1] A. Y. Alfakih. On Bar Frameworks, Stress Matrices and Semidefinite Programming. Preprint, 2009.

[2] A. Y. Alfakih. On the Universal Rigidity of Generic Bar Frameworks. *Contributions to Discrete Mathematics*, 5(1):7–17, 2010.

[3] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur. A Theory of Network Localization. *IEEE Transactions on Mobile Computing*, 5(12):1663–1678, 2006.

[4] J. Aspnes, D. Goldenberg, and Y. R. Yang. On the Computational Complexity of Sensor Network Localization. In S. Nikoletseas and J. D. P. Rolim, editors, *Proceedings of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004)*, volume 3121 of *Lecture Notes in Computer Science*, pages 32–44. Springer–Verlag, 2004.

[5] A. Basu, J. Gao, J. S. B. Mitchell, and G. Sabhnani. Distributed Localization Using Noisy Distance and Angle Information. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, pages 262–273, 2006.

[6] M. Belk and R. Connelly. Realizability of Graphs. *Discrete and Computational Geometry*, 37(2):125–137, 2007.

[7] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite Programming Based Algorithms

for Sensor Network Localization. *ACM Transactions on Sensor Networks*, 2(2):188–220, 2006.

[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] P. Biswas and Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, pages 46–54, 2004.

[10] P. Biswas and Y. Ye. A Distributed Method for Solving Semidefinite Programs Arising from Ad Hoc Wireless Sensor Network Localization. In W. W. Hager, S.-J. Huang, P. M. Pardalos, and O. A. Prokopyev, editors, *Multiscale Optimization Methods and Applications*, volume 82 of *Nonconvex Optimization and Its Applications*, pages 69–84. Springer Science+Business Media, Inc., New York, 2006.

[11] J. Bruck, J. Gao, and A. A. Jiang. Localization and Routing in Sensor Networks by Local Angle Information. *ACM Transactions on Sensor Networks*, 5(1):Article 7, 2009.

[12] M. Bădoiu, E. D. Demaine, M. Hajiaghayi, and P. Indyk. Low–Dimensional Embedding with Extra Information. *Discrete and Computational Geometry*, 36(4):609–632, 2006.

[13] N. Bulusu, J. Heidemann, and D. Estrin. GPS–less Low–Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7(5):28–34, 2000.

[14] M. W. Carter, H. H. Jin, M. A. Saunders, and Y. Ye. SPASELOC: An Adaptive Subproblem Algorithm for Scalable Wireless Sensor Network Localization. *SIAM Journal on Optimization*, 17(4):1102–1128, 2006.

[15] R. Connelly. Tensegrity Structures: Why are They Stable? In M. F. Thorpe and P. M. Duxbury, editors, *Rigidity Theory and Applications*, Fundamental Materials Research, pages 47–54. Kluwer Academic/Plenum Publishers, New York, 1999.

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

[17] G. M. Crippen and T. F. Havel. *Distance Geometry and Molecular Conformation*, volume 15 of *Chemometrics Series*. Research Studies Press Ltd., Taunton, Somerset, England, 1988.

[18] E. Cuthill and J. McKee. Reducing the Bandwidth of Sparse Symmetric Matrices. In *Proceedings of the 1969 24th National Conference*, pages 157–172, 1969.

[19] L. Doherty, K. S. J. Pister, and L. El Ghaoui. Convex Position Estimation in Wireless Sensor Networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 3, pages 1655–1663, 2001.

[20] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework. *SIAM Journal on Optimization*, 11(3):647–674, 2000.

[21] S. J. Gortler, A. D. Healy, and D. P. Thurston. Characterizing Generic Global Rigidity. *American Journal of Mathematics*, 132(4):897–939, 2010.

[22] S. J. Gortler and D. P. Thurston. Characterizing the Universal Rigidity of Generic Frameworks. Preprint, 2009.

[23] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*, volume 2 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 1993.

[24] B. Hendrickson. The Molecule Problem: Exploiting Structure in Global Optimization. *SIAM Journal on Optimization*, 5(4):835–857, 1995.

[25] B. Jackson and T. Jordán. Connected Rigidity Matroids and Unique Realizations of Graphs. *Journal of Combinatorial Theory, Series B*, 94(1):1–29, 2005.

[26] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita. Exploiting Sparsity in Linear and Nonlinear Matrix Inequalities via Positive Semidefinite Matrix Completion. Technical Report B–452, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh–Okayama, Meguro–ku, Tokyo 152–8552, Japan, 2009.

[27] S. Kim, M. Kojima, and H. Waki. Exploiting Sparsity in SDP Relaxation for Sensor Network Localization. *SIAM Journal on Optimization*, 20(1):192–215, 2009.

[28] N. Krislock and H. Wolkowicz. Explicit Sensor Network Localization Using Semidefinite Representations and Clique Reductions. Technical Report CORR 2009–04, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, 2009.

[29] X. Li. *Wireless Ad Hoc and Sensor Networks: Theory and Applications*. Cambridge University Press, New York, 2008.

[30] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS) Using AOA. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, volume 3, pages 1734–1743, 2003.

[31] J. Nie. Sum of Squares Method for Sensor Network Localization. *Computational Optimization and Applications*, 43(2):151–179, 2006.

[32] J. B. Saxe. Embeddability of Weighted Graphs in k–Space is Strongly NP–Hard. In *Proceedings of the 17th Allerton Conference in Communication, Control, and Computing*, pages 480–489, 1979.

[33] A. M.-C. So. *A Semidefinite Programming Approach to the Graph Realization Problem: Theory, Applications and Extensions*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, 2007.

[34] A. M.-C. So and Y. Ye. A Semidefinite Programming Approach to Tensegrity Theory and Realizability of Graphs. In *Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2006)*, pages 766–775, 2006.

[35] A. M.-C. So and Y. Ye. Theory of Semidefinite Programming for Sensor Network Localization. *Mathematical Programming, Series B*, 109(2):367–384, 2007.

[36] J. F. Sturm. Using SeDuMi 1.02, a MATLAB Toolbox for Optimization over Symmetric Cones. *Optimization Methods and Software*, 11(1–4):625–653, 1999.

[37] P. Tseng. Second–Order Cone Programming Relaxation of Sensor Network Localization. *SIAM Journal on Optimization*, 18(1):156–185, 2007.

[38] L. Vandenberghe and S. Boyd. Semidefinite Programming. *SIAM Review*, 38(1):49–95, 1996.

[39] H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of Squares and Semidefinite Program Relaxations for Polynomial Optimization Problems with Structured Sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.

[40] Z. Wang, S. Zheng, Y. Ye, and S. Boyd. Further Relaxations of the Semidefinite Programming Approach to Sensor Network Localization. *SIAM Journal on Optimization*, 19(2):655–673, 2008.

[41] Z. Yang, Y. Liu, and X.-Y. Li. Beyond Trilateration: On the Localizability of Wireless Ad–Hoc Networks. In *Proceedings of the 28th Conference on Computer Communications (INFOCOM 2009)*, pages 2392–2400, 2009.

[42] Z. Zhu, A. M.-C. So, and Y. Ye. Universal Rigidity: Towards Accurate and Efficient Localization of Wireless Networks. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010)*, 2010.