SIAM J. SCI. COMPUT.
Vol. 30, No. 3, pp. 1251–1277

A DISTRIBUTED SDP APPROACH FOR LARGE-SCALE NOISY ANCHOR-FREE GRAPH REALIZATION WITH APPLICATIONS TO MOLECULAR CONFORMATION*

PRATIK BISWAS†, KIM-CHUAN TOH‡, AND YINYU YE§

Abstract. We propose a distributed algorithm for solving Euclidean metric realization problems arising from large 3-D graphs, using only noisy distance information and without any prior knowledge of the positions of any of the vertices. In our distributed algorithm, the graph is first subdivided into smaller subgraphs using intelligent clustering methods. Then a semidefinite programming relaxation and gradient search method are used to localize each subgraph. Finally, a stitching algorithm is used to find affine maps between adjacent clusters, and the positions of all points in a global coordinate system are then derived. In particular, we apply our method to the problem of finding the 3-D molecular configurations of proteins based on a limited number of given pairwise distances between atoms. The protein molecules, all with known molecular configurations, are taken from the Protein Data Bank. Our algorithm is able to reconstruct reliably and efficiently the configurations of large protein molecules from a limited number of pairwise distances corrupted by noise, without incorporating domain knowledge such as the minimum separation distance constraints derived from van der Waals interactions.

Key words. semidefinite programming, anchor-free graph realization, molecular conformation

AMS subject classifications. 49M27, 90C06, 90C22, 90C26, 92E10, 92-08

DOI. 10.1137/05062754X

1. Introduction. Semidefinite programming (SDP) relaxation techniques can be used for solving a wide range of Euclidean distance geometry problems, such as data compression, metric-space embedding, covering and packing, and chain folding [12, 20, 23, 38]. More recently, SDP has been applied to machine learning problems such as nonlinear dimensionality reduction [35].

One particular instance of the distance geometry problem arises in sensor network localization [5, 6, 8] where, given the positions of some sensors in a network (the sensor nodes' known positions are called anchors) and a set of pairwise distances between sensors and between sensors and anchors, the positions of all the sensor nodes in the network have to be computed. This problem can be abstracted into a 2-D or 3-D graph realization problem, that is, finding the positions of the vertices of a graph given some constraints on the edge lengths.

Another instance of the graph realization problem arises in molecular conformation, specifically, protein structure determination. It is well known that protein structure determination is of great importance for studying the functions and properties of proteins. In order to determine the structure of protein molecules, nuclear magnetic resonance (NMR) experiments are performed to estimate lower and upper bounds on interatomic distances [13, 18]. Additional knowledge about the bond angles and lengths between atoms also yields information about relative positions of atoms.

*Received by the editors March 24, 2005; accepted for publication (in revised form) November 27, 2007; published electronically March 21, 2008.
http://www.siam.org/journals/sisc/30-3/62754.html
†Electrical Engineering, Stanford University, Stanford, CA 94305 (pbiswas@stanford.edu).
‡Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543, Singapore (mattohkc@nus.edu.sg).
§Management Science and Engineering and, by courtesy, Electrical Engineering, Stanford University, Stanford, CA 94305 (yinyu-ye@stanford.edu).

In the simplest form, given a subset of all the pairwise distances between the atoms of a molecule, the objective is to find a conformation of all the atoms in the molecule such that the distance constraints are satisfied.

Since this problem is also an abstraction of graph realization, most of the concepts that were developed for the sensor network localization problem can be used directly for the molecular conformation problem. In fact, the terms localization and realization will be used interchangeably throughout this paper. However, some crucial improvements to the basic algorithm have to be made before it can be applied to the molecular conformation problem. In [8], the problem was described in two dimensions, although it can be extended to higher dimensions, as is illustrated in this paper. The improvements suggested in [5] provide much better performance for noisy distance data. However, the SDP methods described in [5] and [8] solved problems of relatively small sizes with the number of points typically below 100 or so. A distributed version of the algorithm was described in [7] for larger sets of points, in which the larger problem was broken down into smaller subproblems corresponding to local clusters of points. The assumption made in the large-scale sensor network problem was the existence of anchor nodes all across the network, i.e., points whose positions are known prior to the computation. These anchor nodes play a major role in determining the positions of unknown nodes in the distributed algorithm, since the presence of anchors in each cluster is crucial in facilitating the clustering of points and the process of stitching together different clusters. The anchor nodes are used to create clusters by including all sensors within one or two hops of their radio range. When the positions for unknown nodes are computed, they are already in the global coordinate system since the anchor positions have been incorporated into the computation. Furthermore, the presence of anchors helps to dampen the propagation of errors in the estimated positions to other clusters when the problem is solved by a distributed algorithm.

Without anchors, we need alternative methods of clustering the points and stitching the clusters together. In this paper, we propose a distributed algorithm for solving large-scale noisy anchor-free Euclidean metric realization problems arising from 3-D graphs, to address precisely the issues just mentioned. In the problem considered here, there are no a priori determined anchors, as is the case in the molecular conformation problem. Therefore the strategy of creating local clusters for distributed computation is different. We perform repeated matrix permutations on the sparse distance matrix to form local clusters within the structure. The clusters are built keeping in mind the need to maintain a reasonably high degree of overlap between adjacent clusters; i.e., there should be enough common points considered between adjacent clusters. This is used to our advantage when we combine the results from different clusters during the stitching process.

Within each cluster, the positions of the points are first estimated by solving an SDP relaxation of a nonconvex minimization problem that seeks to minimize the sum of errors between given and estimated distances. A gradient-descent minimization method, first described in [22], is used as a postprocessing step, after the SDP computation to further reduce the estimation errors. The refinement of errors by a local optimization method is especially important, as distance data in a cluster may be too sparse, or the noise in the distance measures may be too high, to determine a unique realization in the required dimensional space. In that case, the SDP solution is in a higher-dimensional space, and a simple projection of that solution into a lower-dimensional space does not yield correct positions. Fortunately, it can often serve as a good starting iterate for a local optimization method to obtain a lower-dimensional

realization that satisfies the given distance constraints. After the gradient-descent postprocessing step, poorly estimated points within each cluster are isolated, and their positions are recomputed when more points are correctly estimated.

The solution of each individual cluster yields different orientations in its local coordinate systems since there are no anchors to provide global coordinate information. The local configuration may be rotated, reflected, or translated while still respecting the distance constraints. This was not a problem in the case when anchors were available, as they would perform the task of ensuring that each cluster followed the same global coordinate system. Instead in this paper, we use a least squares–based affine mapping between local coordinates of common points in overlapping clusters to create a coherent conformation of all points in a global coordinate system.

We test our algorithm on protein molecules of varying sizes and configurations. The protein molecules, all with known molecular configurations, are taken from the Protein Data Bank [4]. Our algorithm is able to reliably reconstruct the configurations of large molecules with thousands of atoms quite efficiently and accurately based on given upper and lower bounds on limited pairwise distances between atoms. To the best of our knowledge, there are no computational results reported in the literature for determining molecular structures of this scale by using only sparse and noisy distance information. However, there is still room for improvement in our algorithm in the case of very sparse or highly noisy distance data.

For simplicity, our current SDP-based distributed algorithm does not incorporate the lower constraints generated from van der Waals (VDW) interactions between atoms. However, such constraints can naturally be incorporated into the SDP model. Given that our current algorithm performs quite satisfactorily without the VDW lower bound constraints, we are optimistic that with the addition of such constraints and other improvements in the future, our algorithm would perform well even for the more difficult case of highly noisy and very sparse distance data.

Section 2 of this paper describes related work in distance geometry, SDP relaxations and molecular conformation, and attempts to situate our work in that context. Section 3 elucidates the distance geometry problem and the SDP relaxation models. A preliminary theory for anchor-free graph realization is also developed. In particular, regularization ideas for improving the SDP solution quality in the presence of noise are discussed. The intelligent clustering and cluster stitching algorithms are introduced in sections 4 and 5, respectively. Postprocessing techniques to refine the SDP estimated positions are discussed in section 6. Section 7 describes the complete distributed algorithm. Section 8 discusses the performance of the algorithm on protein molecules from the Protein Data Bank [4]. Finally, in section 9, we conclude with a summary of the paper and outline some work in progress to improve our distributed algorithm.

2. Related work. Owing to their large applicability, a lot of attention has been paid to Euclidean distance geometry problems. The use of SDP relaxations to solve this class of problems involves relaxing the nonconvex quadratic distance constraints into convex linear constraints over the cone of positive semidefinite matrices. It is illustrated through sensor network problems in [8]. Similar relaxations have also been developed in [2, 21, 35].

As the number of points and pairwise distances increases, it becomes computationally intractable to solve the entire SDP in a centralized fashion. With special focus on anchored sensor network localization problems, a distributed technique is proposed in [7]. This involves solving smaller clusters of points in parallel and using information from points in different clusters in subsequent iterations to refine the

solutions.

Building on the ideas in [7], the authors in [11] proposed an adaptive rule-based clustering strategy to sequentially divide a global anchored graph localization problem (in two dimensions) into a sequence of subproblems. The technique in localizing each subproblem is similar to that in [7], but the clustering and stitching strategies are different. It is reported that the improved techniques can localize anchored graphs very efficiently and accurately.

Interestingly, not only does the SDP relaxation method solve for unknown points, but the solution matrix also provides an error measure for each estimation. Furthermore, the dual of the SDP relaxation gives insights into the localizability of the given set of points. In fact, the issue of localizability—that is, the existence of a unique configuration of points satisfying the distance constraints—is closely linked to the rigidity of the graph structure underlying the set of points. These issues are explored in detail in [28].

Many approaches have been developed for the molecular distance geometry problem. An overall discussion of the methods and related software is provided in [39]. Some of the approaches are briefly described below.

When the exact distances between all pairs of n points are given, a valid configuration can be obtained by computing the eigenvalue decomposition of an $(n-1)\times(n-1)$ matrix (which can be obtained through a linear transformation of the squared distance matrix). Note that if the configuration of n points can be realized in a d-dimensional space, then the aforementioned matrix must have rank d, and the eigenvectors corresponding to the d nonzero eigenvalues give a valid configuration. So a decomposition can be found and a configuration constructed in $O(n^3)$ arithmetic operations. The EMBED algorithm [13] exploits this idea for sparse and noisy distances by first performing bound smoothing, that is, preprocessing the available data to remove geometric inconsistencies and finding valid estimates for unknown distances. Then a valid configuration is obtained through the eigenvalue decomposition of the inner product matrix, and the estimated positions are then used as the starting iterate for local optimization methods on certain nonlinear least squares problems.

Classical multidimensional scaling (MDS) is the general class of methods that takes inexact distances as input and extracts a valid configuration from them based on minimizing the discrepancy between the inexact measured distances and the distances corresponding to the estimated configuration. The inexact distance matrix is referred to as a dissimilarity matrix in this framework. Since the distance data is also incomplete, the problem also involves completing the partial distance matrix. The papers [31, 32, 33] consider this problem of completing a partial distance matrix, as well as the more general problem of finding a distance matrix of prescribed embedding dimension that satisfies specified lower and upper bounds, for use in MDS-based algorithms. In [32], good conformation results were reported for molecules with a few hundred atoms each, under the condition that all the pairwise distances (specified in the form of lower and upper bounds with a gap of 0.02Å) below 7Å were given.

Also worth noting in this regard is the distance geometry program APA described in [26], which applies the idea of a data box, a rectangular parallelepiped of dissimilarity matrices that satisfy some given upper and lower bounds on distances. An alternating projection–based optimization technique is then used to solve for both a dissimilarity matrix that lies within the data box, and a valid embedding of the points, such that the discrepancy between the dissimilarity matrix and the distances from the embedding is minimized.

The ABBIE software package [19], on the other hand, exploits the concepts of graph rigidity to solve for smaller subgraphs of the entire graph defined by the points and distances and finally combining the subgraphs to find an overall configuration. It is especially advantageous to solve for smaller parts of the molecule and to provide certificates confirming that the distance information is not enough to uniquely determine the positions of certain atoms. Our approach tries to retain these advantages by solving the molecule in a distributed fashion, that is, solving smaller clusters and later assembling them together.

Some global optimization methods attempt to attack the problem of finding a conformation which fits the given data as a large nonlinear least squares problem. For example, a global smoothing and continuation approach is used in the DGSOL algorithm [24]. To prevent the algorithm from getting stuck at one of the large number of possible local minimizers, the nonlinear least squares problem (with an objective that is closely related to the refinement stage of the EMBED algorithm) is mollified to smoother functions so as to increase the chance of locating the global minimizer. However, it can still be difficult to find the global minimizer from various random starting points, especially with noisy distance information. More refined methods that try to circumvent such difficulties have also been developed in [25], though with limited success.

Another example is the GNOMAD algorithm [36], also a global optimization method, which takes special care to satisfy the physically inviolable minimum separation distance, or VDW constraints. For GNOMAD, the VDW constraints are crucial in reducing the search space in the case of very sparse distance data. Obviously, the VDW constraints can easily be incorporated into any molecular conformation problem that is modeled by an optimization problem. In [36], the success of the GNOMAD algorithm was demonstrated on four molecules (the largest one has 5591 atoms) under the assumption that 30–70% of the exact pairwise distances below 6Å were given. In addition, the given distances included those from covalently bonded atoms and those between atoms that share covalent bonds with the same atom.

Besides optimization-based methods, there are geometry-based methods proposed for the molecular conformation problem. The effectiveness of simple geometric build up (also known as triangulation) algorithms has been demonstrated in [14] and [37] for molecules when exact distances within a certain cut-off radius are all given. Basically, this approach involves using the distances between an unknown atom and previously determined neighboring atoms to find the coordinates of the unknown atom. The algorithm progressively updates the number of known points and uses them to compute points that have not yet been determined. However, the efficacy of such methods for large molecules with very sparse and noisy data has not yet been demonstrated.

In this paper, we will attempt to find the structures of molecules with sizes varying from hundreds to several thousands of atoms, given only upper and lower bounds on some limited pairwise distances between atoms. The approach described in this paper also performs distance matrix completion, similar to some of the methods described above. The extraction of the point configuration after matrix completion is still the same as the MDS methods. The critical difference lies in the use of an SDP relaxation for completing the distance matrix. Furthermore, our distributed approach avoids the issue of intractability for very large molecules by splitting the molecule into smaller subgraphs, much like the ABBIE algorithm [19], and then stitching together the different clusters. Some of the atoms which are incorrectly estimated are solved separately using the correctly estimated atoms as anchors. The latter approach bears

some similarities to the geometric build up algorithms. In this way, we have adapted and improved some of the techniques used in previous approaches, but also introduced new ideas generated from recent advances in SDP to attack the twin problems of dealing with noisy and sparse distance data, and the computational intractability of large-scale molecular conformation.

3. The SDP model. We first present a nonconvex quadratic programming formulation of the position estimation problem (in the molecular conformation context) and then introduce its SDP relaxation model.

Assume that we have m known points (called anchors), $a_k \in \mathcal{R}^d$ (note that $m = 0$ if no anchor exists), $k = 1, \ldots, m$, and n unknown points, $x_j \in \mathcal{R}^d$, $j = 1, \ldots, n$. Suppose that we know the upper and lower bounds on the Euclidean distances between some pairs of unknown points specified in the edge set $\mathcal{N}$, and the upper and lower bounds on the Euclidean distances between some pairs of unknown points and anchors specified in the edge set $\mathcal{M}$. For the rest of the point pairs, the upper and lower bounds would be the trivial bounds, ∞ and 0.

We define the lower bound distance matrices $\underline{D} = (\underline{d}_{ij})$ and $\underline{H} = (\underline{h}_{ik})$, where $\underline{d}_{ij}$ is specified if $(i, j) \in \mathcal{N}$, and $\underline{d}_{ij} = 0$ otherwise; and $\underline{h}_{ik}$ is specified if $(i, k) \in \mathcal{M}$, and $\underline{h}_{ik} = 0$ otherwise. The upper bound distance matrices $\overline{D} = (\bar{d}_{ij})$ and $\bar{H} = (\bar{h}_{ik})$ are defined similarly with $\bar{d}_{ij} = 0$ if $(i, j) \notin \mathcal{N}$, and $\bar{h}_{ik} = 0$ if $(i, k) \notin \mathcal{M}$. We let $D = (d_{ij})$ be the mean of $\underline{D}$ and $\overline{D}$, i.e., $d_{ij} = (\underline{d}_{ij} + \bar{d}_{ij})/2$.

The *realization problem* for the graph $(\{1, \ldots, n\}, \mathcal{N}; \{a_1, \ldots, a_m\}, \mathcal{M})$ is to determine the coordinates of the unknown points $x_1, \ldots, x_n$, given the upper and lower bound distance matrices, $\underline{D}$, $\overline{D}$, $\underline{H}$, and $\overline{H}$.

Let $X = [x_1\ x_2\ \ldots\ x_n]$ be the $d \times n$ matrix that needs to be determined. The realization problem just mentioned can be formulated as the following feasibility problem:

$$
\begin{aligned}
&\text{Find} \quad X \quad \text{s.t.} \\
&\underline{d}_{ij}^2 \le \|x_i - x_j\|^2 \le \bar{d}_{ij}^2 \quad \forall (i,j) \in \mathcal{N}, \\
&\underline{h}_{ik}^2 \le \|x_i - a_k\|^2 \le \bar{h}_{ik}^2 \quad \forall (i,k) \in \mathcal{M}.
\end{aligned} \tag{1}
$$

We can write

$$
\|x_i - x_j\|^2 = e_{ij}^T X^T X e_{ij}, \qquad \|x_i - a_k\|^2 = (e_i; -a_k)^T [X \ \ I]^T [X \ \ I] (e_i; -a_k),
$$

where $e_{ij} = e_i - e_j$. Here e_i is the ith unit vector of appropriate dimension, I is the $d \times d$ identity matrix, and $(e_i; -a_k)$ is the vector obtained by appending $-a_k$ to e_i. Let $Y = X^T X$ and $Z = [Y \ \ X^T; X \ \ I]$. Then problem (1) can be rewritten as

$$
\begin{aligned}
&\text{Find} \quad Z \quad \text{s.t.} \\
&\underline{d}_{ij}^2 \le e_{ij}^T Y e_{ij} \le \bar{d}_{ij}^2 \quad \forall (i,j) \in \mathcal{N}, \\
&\underline{h}_{ik}^2 \le (e_i; -a_k)^T Z (e_i; -a_k) \le \bar{h}_{ik}^2 \quad \forall (i,k) \in \mathcal{M}, \\
&Z = [Y \ \ X^T; X \ \ I], \quad Y = X^T X.
\end{aligned} \tag{2}
$$

The above problem (2) is unfortunately nonconvex. Our method is to relax it to a semidefinite program by relaxing the constraint $Y = X^T X$ to $Y \succeq X^T X$ (meaning that $Y - X^T X$ is positive semidefinite). The last matrix inequality is equivalent to (see Boyd et al. [9])

$$Z = \begin{pmatrix} Y & X^T \\ X & I \end{pmatrix} \succeq \mathbf{0}, \quad Z \text{ symmetric.}$$

Thus, the SDP relaxation of (2) can be written as the following standard SDP problem:

$$\begin{aligned} &\text{Find} \quad Z \quad \text{s.t.} \\ &\underline{d}_{ij}^2 \le e_{ij}^T Z e_{ij} \le \bar{d}_{ij}^2 \quad \forall\ (i,j) \in \mathcal{N}, \\ &\underline{h}_{ik}^2 \le (e_i; -a_k)^T Z (e_i; -a_k) \le \bar{h}_{ik}^2 \quad \forall\ (i,k) \in \mathcal{M}, \\ &e_i^T Z e_i = 1 \quad \forall\ n+1 \le i \le n+d, \\ &(e_i + e_j)^T Z (e_i + e_j) = 2 \quad \forall\ n+1 \le i < j \le n+d, \\ &Z \succeq \mathbf{0}. \end{aligned} \tag{3}$$

Note that the last two sets of equality constraints in (3) specify that the lower-right $d \times d$ block of Z is the identity matrix.

We note that if there are additional constraints of the form $\|x_i - x_j\| \ge L$ coming from knowledge about the minimum separation distance between any two points, such constraints can be included in the semidefinite program (3) by adding inequality constraints of the form $e_{ij}^T Z e_{ij} \ge L^2$. In molecular conformation, the minimum separation distances corresponding to the VDW interactions are used in an essential way to reduce the search space in the atom-based constrained optimization algorithm (GNOMAD) described in [36]. The minimum separation distance constraints are also easily incorporated into the MDS framework [26, 32].

For the anchor-free case where $\mathcal{M} = \emptyset$ (empty set), the SDP problem (3) can be reduced in dimension by replacing Z by Y and removing the last $d(d+1)/2$ equality constraints, i.e.,

$$\begin{aligned} &\text{Find} \quad Y \quad \text{s.t.} \\ &\underline{d}_{ij}^2 \le e_{ij}^T Y e_{ij} \le \bar{d}_{ij}^2 \quad \forall\ (i,j) \in \mathcal{N}, \\ &Y\mathbf{e} = \mathbf{0}, \\ &Y \succeq \mathbf{0}. \end{aligned} \tag{4}$$

This is because when $\mathcal{M} = \emptyset$, the $(1,2)$ and $(2,1)$ blocks of Z are always equal to zero if the starting iterate for the interior-point method used to solve (3) is chosen to be so. Note that we add the extra constraint $Y\mathbf{e} = \mathbf{0}$ to eliminate the translational invariance of the configuration by putting the center of gravity of the points at the origin, i.e., $\sum_{i=1}^n x_i = \mathbf{0}$.

Note also that, in the anchor-free case, if the graph is localizable (defined in the next subsection), a realization $X \in \mathcal{R}^{d \times n}$ can no longer be obtained from the $(2,1)$ block of Z but needs to be computed from the inner product matrix Y by factorizing

it to the form $Y = X^T X$ via eigenvalue decomposition (as has been done in previous methods discussed in the literature review). In the noisy case, the inner product matrix Y would typically have rank greater than d. In practice, X is chosen to be the best rank-d approximation, by choosing the eigenvectors corresponding to the d largest eigenvalues. The configuration so obtained is a rotated or reflected version of the actual point configuration.

3.1. Theory of anchor-free graph realization. In order to establish the theoretical properties of the SDP relaxation, we will consider the cases where all the given distances in $\mathcal{N}$ are exact, i.e., without noise. A graph $G = (\{1, \ldots, n\}, D)$ is localizable in dimension d if (i) it has a realization X in $\mathcal{R}^{d\times n}$ such that $\|x_i - x_j\| = d_{ij}$ for all $(i, j) \in \mathcal{N}$; (ii) it cannot be realized (nontrivially) in a higher-dimensional space. We let $D = (d_{ij})$ be the $n \times n$ matrix such that its (i, j) element d_{ij} is the given distance between points i and j when $(i, j) \in \mathcal{N}$, and zero otherwise. It is shown for the exact distances case in [28] that if the graph with anchors is localizable, then the SDP relaxation will produce a unique optimal solution Z with its (1,1) block equal to $X^T X$. For the anchor-free case where $\mathcal{M} = \emptyset$, it is clear that the realization cannot be unique since the configuration may be translated, rotated, or reflected, and still preserve the same distances.

To remove the translational invariance, we will add an objective function to minimize the norm of the solution in the problem formulation:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} \|x_j\|^2 \\ \text{s.t.} \quad & \|x_i - x_j\|^2 = d_{ij}^2 \quad \forall\ (i, j) \in \mathcal{N}. \end{aligned} \tag{5}$$

What this minimization does is to translate the center of gravity of the points to the origin; that is, if $\bar{x}_j$, $j = 1, \ldots, n$, is the realization of the problem, then the realization generated from (5) will be $\bar{x}_j - \bar{x}$, $j = 1, \ldots, n$, where $\bar{x} = \frac{1}{n}\sum_{j=1}^{n} \bar{x}_j$, subject to only rotation and reflection. The norm minimization also helps the following SDP relaxation of (5) to have bounded solutions:

$$\begin{aligned} \text{minimize} \quad & \text{Trace}(Y) = I \bullet Y \\ \text{s.t.} \quad & e_{ij}^T Y e_{ij} = d_{ij}^2 \quad \forall\ (i, j) \in \mathcal{N}, \\ & Y \succeq \mathbf{0}, \end{aligned} \tag{6}$$

where $Y \in \mathcal{S}^n$ (the space of $n \times n$ symmetric matrices), I is the identity matrix, and $\bullet$ denotes the standard matrix inner product. We note that a model similar to (6) is also proposed in [1] but with the objective function replaced by $\|Y\|_F^2$. The dual of the SDP relaxation (6) is given by

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2 \\ \text{s.t.} \quad & I - \sum_{(i,j)\in\mathcal{N}} w_{ij} \cdot e_{ij} e_{ij}^T \succeq \mathbf{0}. \end{aligned} \tag{7}$$

Note that the dual is always feasible and has an interior, since $w_{ij} = 0$ for all $(i, j) \in \mathcal{N}$ is an interior feasible solution. Thus the primal optimal value in (6) is always attained. However, the dual optimal value in (7) may not always be attainable unless the primal

problem (6) is strictly feasible. From the standard duality theorem for SDP, we have the following proposition.

PROPOSITION 1. *Let $\bar{Y} \succeq 0$ be an optimal solution of* (6)*, and suppose that the dual optimal value in* (7) *is attained, with $\bar{U} = I - \sum_{(i,j)\in cN} \bar{w}_{ij} \cdot e_{ij} e_{ij}^T \succeq 0$ being an optimal slack matrix. Then we have the following*:

1. *The complementarity condition holds: $\bar{Y} \bullet \bar{U} = 0$ or $\bar{Y}\bar{U} = \mathbf{0}$.*
2. $\mathrm{Rank}(\bar{Y}) + \mathrm{Rank}(\bar{U}) \leq n$.

In general, a primal (dual) max-rank solution is a solution that has the highest rank among all solutions for primal (6) (dual (7)). It is known that various path-following interior-point algorithms compute the max-rank solutions for both the primal and dual in polynomial time.

We now investigate when the SDP (6) will have an exact relaxation, given that the partial distance data (d_{ij}) is exact. For the anchored case, it was proved in [28] that the condition of exact relaxation is equivalent to the rank of the SDP solution $\bar{Y}$ being d. However, for the anchor-free case, we are unable to prove this. Instead, we derive an alternative result.

DEFINITION 1. *Problem* (5) *is d-localizable if there is no $x_j \in \mathcal{R}^h,\ j = 1, \ldots, n$, where $h \neq d$, such that*

$$\|x_i - x_j\|^2 = d_{ij}^2 \quad \forall\ (i,j) \in \mathcal{N}.$$

For $h > d$, the condition should exclude the trivial case when we set $x_j = (\bar{x}_j; \mathbf{0})$ for $j = 1, \ldots, n$.

The d-localizability indicates that the distances cannot be embedded by a nontrivial realization in higher-dimensional space, and cannot be "flattened" to a lower-dimensional space either. We now develop the following theorem.

THEOREM 1. *If problem* (5) *is d-localizable, then the solution matrix, $\bar{Y}$, of* (6) *is unique, and its rank equals d. Furthermore, if $\bar{Y} = \bar{X}^T\bar{X}$ and the dual optimal value of* (7) *is attained, then $\bar{X} = (\bar{x}_1, \ldots, \bar{x}_n) \in \mathcal{R}^{d\times n}$ is the unique minimum-norm localization of the graph with $\sum_{j=1}^n \bar{x}_j = 0$ (subject to only rotation and reflection).*

Proof. Since problem (5) is d-localizable, by definition every feasible solution matrix Y of (6) has rank d. Thus, there is a rank-d matrix $\bar{V} \in \mathcal{R}^{d\times n}$ such that any feasible solution matrix Y can be written as $Y = \bar{V}^T P \bar{V}$, where P is a $d \times d$ symmetric positive definite matrix. We show that the solution is unique by contradiction. Suppose that there are two feasible solutions

$$Y^1 = \bar{V}^T P^1 \bar{V} \quad \text{and} \quad Y^2 = \bar{V}^T P^2 \bar{V},$$

where $P^1 \neq P^2$ and, without loss of generality, $P^2 - P^1$ has at least one negative eigenvalue. (Otherwise, if $P^1 - P^2$ has at least one negative eigenvalue, we can interchange the role of P^1 and P^2; the only case left to be considered is when all the eigenvalues of $P^2 - P^1$ are equal to zero, but this case is not possible since it implies that $P^1 = P^2$.) Let

$$Y(\alpha) = \bar{V}^T(P^1 + \alpha(P^2 - P^1))\bar{V}.$$

Clearly, $Y(\alpha)$ satisfies all the linear constraints of (6), and it has rank d for $0 \leq \alpha \leq 1$. But there is an $\alpha' > 1$ such that $P^1 + \alpha'(P^2 - P^1)$ is positive semidefinite but not positive definite; that is, one of the eigenvalues of $P^1 + \alpha'(P^2 - P^1)$ becomes zero. Thus, $Y(\alpha')$ has a rank less than d but feasible to (6), which contradicts the fact that the graph cannot be "flattened" to a lower-dimensional space.

Let $\bar{U}$ be an optimal dual matrix of (7). Then, any optimal solution matrix $\bar{Y}$ satisfies the complementarity condition $\bar{Y}\bar{U} = \mathbf{0}$. Note that $\bar{U}\mathbf{e} = \mathbf{e}$, where $\mathbf{e}$ is the vector of all ones. Thus, we have $\bar{X}^T\bar{X}\mathbf{e} = \bar{Y}\mathbf{e} = \bar{Y}\bar{U}\mathbf{e} = \mathbf{0}$, which further implies that $\bar{X}\mathbf{e} = \mathbf{0}$. □

Theorem 1 has an important implication in a distributed graph localization algorithm. It suggests that if a subgraph is d-localizable, then the subconfiguration is the same (up to translation, rotation, and reflection) as the corresponding portion in the global configuration. Thus, one may attempt to localize a large graph by finding a sequence of d-localizable subgraphs.

Theorem 1 also says that if the graph G is d-localizable, then the optimal solution of the SDP is given by $\bar{Y} = \bar{X}^T\bar{X}$ for some $\bar{X} = [\bar{x}_1, \ldots, \bar{x}_n] \in \mathcal{R}^{d\times n}$ such that $\bar{X}\mathbf{e} = 0$.

It is now clear that when G is d-localizable, we have $\bar{Y} = \bar{X}^T\bar{X}$, and hence $\bar{Y}_{jj} = \|\bar{x}_j\|^2$ for $j = 1, \ldots, n$. But in general, when the given distances are not exact but corrupted with noises, we have only $\bar{Y} - \bar{X}^T\bar{X} \succeq 0$. This inequality, however, may give a measure of the quality of the estimated positions. For example, the individual trace

$$T_j := \bar{Y}_{jj} - \|\bar{x}_j\|^2 \tag{8}$$

may give an indication of the quality of the estimated position $\bar{x}_j$, where a smaller trace indicates more accuracy in the estimation.

3.2. Regularization term. When the measured distances have errors, the distance constraints usually contradict each other, and so there is no localization in $\mathcal{R}^d$. In other words, $Y \neq X^TX$. However, since the SDP approach relaxes the constraint $Y = X^TX$ into $Y \succeq X^TX$, it is still possible to locate the points in a higher-dimensional space (or choose a Y with a higher rank) such that they satisfy the distance constraints exactly. The optimal solution in a higher-dimensional space always results in a smaller violation of the distance constraints than the one constrained in $\mathcal{R}^d$. Furthermore, the max-rank property [17] implies that solutions obtained through interior-point methods for solving SDPs converge to the maximum rank solutions. Hence, because of the relaxation of the rank requirement, the solution is "lifted" to a higher-dimensional space. For example, imagine a rigid structure consisting of the set of points in a plane (with the points having specified distances from each other). If we perturb some of the specified distances, the configuration may need to be readjusted by setting some of the points outside the plane.

The above discussion leads us to the question of how to round the higher-dimensional (higher rank) SDP solution into a lower-dimensional (in this case, rank-3) solution. One way is to ignore the augmented dimensions and use the projection X^* as a suboptimal solution, which is the case in [8]. However, the projection typically leads to points getting "crowded" together. (Imagine the projection of the top vertex of a pyramid onto its base.) This is because a large contribution to the distance between two points could come from the dimensions we choose to ignore.

In [35], regularization terms have been incorporated into the SDP arising from kernel learning in nonlinear dimensionality reduction. The purpose is to penalize folding and try to find a stretched map of the set of points while respecting local distance constraints. Here we propose a similar strategy to ameliorate the difficulty of crowding.

Our strategy is to convert the feasibility problem (1) into a maximization problem using the following regularization term as the objective function:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x_j\|^2. \tag{9}$$

The new optimization problem is

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} \|x_i - x_j\|^2 \\ \text{s.t.} \quad & \underline{d}_{ij}^2 \le \|x_i - x_j\|^2 \le \bar{d}_{ij}^2 \quad \forall (i,j) \in \mathcal{N}, \\ & \sum_{i=1}^{n} x_i = 0, \end{aligned} \tag{10}$$

and the SDP relaxation is

$$\begin{aligned} \text{maximize} \quad & \langle I - (\mathbf{e}\mathbf{e}^T/n), Y \rangle \\ \text{s.t.} \quad & \underline{d}_{ij}^2 \le e_{ij}^T Y e_{ij} \le \bar{d}_{ij}^2 \quad \forall\ (i,j) \in \mathcal{N}, \\ & Y\mathbf{e} = 0, \quad Y \succeq 0, \end{aligned} \tag{11}$$

where $\mathbf{e}$ is the vector of all ones. As mentioned before, the constraints $\sum_{i=1}^{n} x_i = 0$ and $Y\mathbf{e} = 0$ are to remove the translational invariance of the configuration of points by putting the center of gravity at the origin.

The addition of a regularization term penalizes folding between the points and maximizes the separation between them, while still respecting local distance constraints. The idea of regularization has also been linked to tensegrity theory and realizability of graphs in lower dimensions; see [27]. The notion of stress is used to explain this. By maximizing the distance between some vertices in a graph, the graph gets stretched out, and there is a nonzero stress induced on the edges in the graph. For the configuration to remain in equilibrium, the total stress on a vertex must sum to zero. In order for the overall stress to cancel out completely, the graph must be in a low-dimensional space.

One important point to be noted is that in the case of very sparse distance data, often there may be two or more disjoint blocks within the given distance matrix; that is, the graph represented by the distance matrix may not be connected and may have more than one component. In that case, using the regularization term leads to an unbounded objective function since the disconnected components can be pulled as far apart as possible. Therefore, care must be taken to identify the disconnected components before applying (11) to the individual components.

4. Clustering. The SDP problem (11) is computationally intractable when there are several hundred points. Therefore we divide the entire molecule into smaller clusters of points and solve a separate SDP for each cluster. The clusters need to be chosen such that there should be enough distance information between the points in a cluster for it to be localized accurately, but at the same time only enough that it can also be solved efficiently. In order to do this, we make use of matrix permutations that reorder the points in such a way that the points which share the most distance information among each other are grouped together.

In the problem described in this paper, we have an upper and a lower bound distance matrix, but for simplicity, we will describe the operations in this section on just the partial distance matrix D. In the actual implementation, the operations described are performed on both the upper and lower bound distance matrices. This

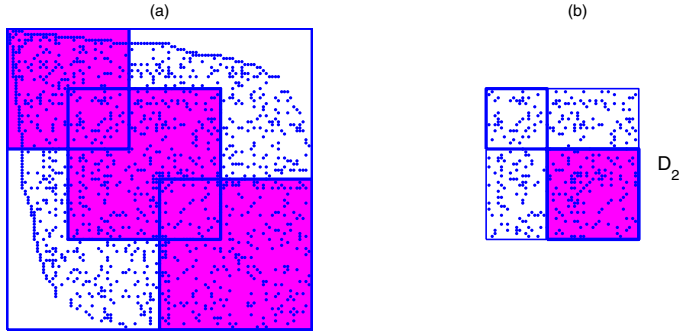


FIG. 1. (a) *Schematic diagram of the quasi-block-diagonal structure considered in the distributed SDP algorithm.* (b) *The shaded region is the* $(2,2)$ *block of* D_2 *that is not overlapping* D_1*. Points corresponding to this shaded block are reordered again via a symmetric reversed Cuthill–McKee permutation.*

does not make a difference, because the operations performed here basically exploit information only about the connectivity graph of the set of points.

We perform a symmetric permutation of the partial distance matrix D to aggregate the nonzero elements towards the main diagonal. Let $\tilde{D}$ be the permuted matrix. In our implementation, we used the function `symrcm` in MATLAB to perform the symmetric reverse Cuthill–McKee permutation [15] on D. In [40], the same permutation is also used in a domain decomposition method for fast manifold learning.

The permutated matrix $\tilde{D}$ is next partitioned into a quasi-block-diagonal matrix with variable block-sizes. Let the blocks be denoted by $D_1, \ldots, D_L$. A schematic diagram of the quasi-block-diagonal structure is shown in Figure 1. The size of each block (except the last) is determined as follows. Starting with a minimum block-size, say 50, we extend the block-size incrementally until the number of nonzero elements in the block is above a certain threshold, say 1000. We start the process of determining the size of each block from the upper-left corner and sequentially proceed to the lower-right corner of $\tilde{D}$. Observe that there are overlapping subblocks between adjacent blocks. For example, the second block overlaps with the first at its upper-left corner and overlaps with the third at its lower-right corner.

The overlapping subblocks serve an important purpose. For convenience, consider the overlapping subblock between the second and third blocks. This overlapping subblock corresponds to points that are common to the configurations defined by their respective distance matrices, D_2 and D_3. If the third block determines a localizable configuration X_3, then the common points in the overlapping subblock can be used to stitch the localized configuration X_3 to the current global configuration determined by the first two blocks. In general, if the kth block is localizable, then the overlapping subblock between the $k-1$st and kth blocks will be used to stitch the kth localized configuration to the global configuration determined by the aggregation of all the previous blocks.

Geometrically, the above partitioning process splits the job of determining the global configuration defined by $\tilde{D}$ into L smaller jobs, each of which tries to determine the subconfiguration defined by D_k and then assemble the subconfigurations sequentially from $k=1$ to L to reconstruct the global configuration.

As the overlapping subblocks are of great importance in stitching a subconfiguration to the global configuration, they should have as high a connectivity as possible.

In our implementation, we find the following strategy to be reasonably effective. After the blocks $D_1, \ldots, D_L$ are determined, starting with D_2, we perform a symmetric reverse Cuthill–McKee permutation on the (2,2) subblock of D_2 that is not overlapping D_1, and repeat the same process sequentially for all subsequent blocks. To avoid introducing excessive notation, we still use D_k to denote the permuted kth block.

It is also worth noting that, in the case of highly noisy or very sparse data, the size of the overlapping subblocks needs to be set higher for the stitching phase to succeed. The more common points there are between two blocks, the more robust the stitching between them. This is also true as the number of subblocks which need to be stitched is large (that is, the number of atoms in the molecules is large). However, increasing the number of common points also has an impact on the runtime, and therefore we must choose the overlapping subblock sizes judiciously. Experiments indicated that a subblock size of 15–20 was sufficient for molecules with less than 3000 atoms, but subblock sizes of 25–30 were more suitable for larger molecules.

5. Stitching. After all the individual localization problems corresponding to $D_1, \ldots, D_L$ have been solved, we have L subconfigurations that need to be assembled together to form the global configuration associated with $\tilde{D}$.

Suppose that the current configuration determined by the blocks D_i, $i = 1, \ldots, k-1$, is given by the matrix $X^{(k-1)} = [U^{(k-1)}, V^{(k-1)}]$. Suppose also that $F^{(k-1)}$ records the global indices of the points that are currently labeled as localized in the current global configuration. Let $X_k = [V_k, W_k]$ be the points in the subconfiguration determined by D_k. (For $k = 1$, V_k is the null matrix. For $k = 2$, V_k and W_k correspond to the unshaded and shaded subblocks, respectively, of D_2 in Figure 1(b).) Here $V^{(k-1)}$ and V_k denote the positions of the points corresponding to the overlapping subblock between D_{k-1} and D_k, respectively.

Let I_k be the global indices of the points in W_k. Note that the global indices of the unlocalized points for the blocks $D_1, \ldots, D_{k-1}$ are given by $J^{(k-1)} = \bigcup_{i=1}^{k-1} I_i \setminus F^{(k-1)}$.

We will now concentrate on stitching the subconfiguration D_k with the global index set I_k. Note that the points in the subconfiguration D_k, which have been obtained by solving the SDP on D_k, will most likely contain points that have been correctly estimated and points that have been estimated less accurately. It is essential that we isolate the badly estimated points, so as to ensure that their errors are not propagated when estimating subsequent blocks and that we may recalculate their positions when more points have been correctly estimated.

To detect the badly estimated points, we use a combination of two error measures. Let x_j be the position estimated for point j. Set $\widehat{F} \leftarrow F^{(k-1)}$. We use the trace error T_j from (8) and the local error

$$\widehat{E}_j = \frac{\sum_{i \in \widehat{\mathcal{N}}_j} (\|x_i - x_j\| - d_{ij})^2}{|\widehat{\mathcal{N}}_j|}, \tag{12}$$

where $\widehat{\mathcal{N}}_j = \{i \in \widehat{F} : i < j, \tilde{D}_{ij} \neq 0\}$. We require $\max\{T_j, \widehat{E}_j\} \leq T_\varepsilon$ as a necessary condition for the point to be correctly estimated.

In the case when we are just provided with upper and lower bounds on distances, we use the mean distance in the local error measure calculations, i.e.,

$$d_{ij} = (\bar{d}_{ij} + \underline{d}_{ij})/2. \tag{13}$$

The use of multiple error measures is crucial, especially in cases when the distance information provided is noisy. In the noise-free case, it is much easier to isolate points

which are badly estimated using only the trace error T_j. But in the noisy case, there are no reliable error measures. By using multiple error measures, we hope to identify all the bad points which might possibly escape detection when only a single error measure is used.

Setting the tolerances T_ε for the error is also an important parameter selection issue. For very sparse distance data and highly noisy cases, where even accurately estimated points may have significant error measure values, there is a trade-off between selecting the tolerance and the number of points that are flagged as badly estimated. If the tolerance is too tight, we might end up discarding too many points that are actually quite accurately estimated.

The design of informative error measures is still an open issue, and there is room for improvement. As our results will show, the stitching phase of the algorithm is one which is most susceptible to noise and inaccurate estimation, and we need better error measures to make it more robust.

Now we have two cases to consider in the stitching process:

(i) Suppose that there are enough points in the overlapping subblocks V_k and $V^{(k-1)}$ that are well estimated/localized; then we can stitch the kth subconfiguration directly to the current global configuration $X^{(k-1)}$ by finding the affine mapping that matches points in $V^{(k-1)}$ and V_k as closely as possible. Mathematically, we solve the following linear least squares problem:

$$\min \left\{ \|B(V_k - \alpha) - (V^{(k-1)} - \beta)\|_F \,:\, B \in \mathcal{R}^{d\times d} \right\}, \tag{14}$$

where α and β are the centroids of V_k and $V^{(k-1)}$, respectively. Once an optimal B is found, set

$$\widehat{X} = [U^{(k-1)}, V^{(k-1)}, \beta + B(W_k - \alpha)], \quad \widehat{F} \leftarrow F^{(k-1)} \bigcup I_k.$$

We should mention that in the stitching process it is very important to exclude points in $V^{(k-1)}$ and V_k that are badly estimated/unlocalized when solving (14) to avoid destroying the current global configuration. It should also be noted that there may be points in W_k that are incorrectly estimated in the SDP step. Performing the affine transformation on these points is useless, because they are in the wrong position in the local configuration to begin with. To deal with these points, we re-estimate the positions using those correctly estimated points as anchors. This procedure is exactly the same as that described in case (ii).

(ii) If there are not enough common points in the overlapping subblocks, then the stitching process described in (i) cannot be carried out successfully. In this case, the solution obtained from the SDP step for D_k is discarded. That is, the positions in W_k are discarded, and they are to be determined via the current global configuration $X^{(k-1)}$ point-by-point as follows.
Set $\widehat{X} \leftarrow X^{(k-1)}$ and $\widehat{F} \leftarrow F^{(k-1)}$. Let $T_\varepsilon = 10^{-4}$.
For $j \in J^{(k-1)} \bigcup I_k$, do the following:
   (a) Formulate the new SDP problem (3) with $\mathcal{N} = \emptyset$ and $\mathcal{M} = \{(i,j) : i \in \widehat{F},\ \tilde{D}_{ij} \neq 0\}$, where the anchor points are given by $\{\widehat{X}(:,i) : (i,j) \in \mathcal{M}\}$.
   (b) Let x_j and T_j be the newly estimated position and trace error from the previous step. Compute the local error measure $\widehat{E}_j$.

If $j \in J^{(k-1)}$, set $\widehat{X}(:,j) = x_j$; else, set $\widehat{X} = [\widehat{X}, x_j]$. If $\min\{T_j, \widehat{E}_j\} \leq T_\varepsilon$, then set $\widehat{F} \leftarrow \widehat{F} \cup \{j\}$, end.

Notice that in attempting to localize the points corresponding to W_k, we also attempt to estimate the positions of those previously unlocalized points, whose indices are recorded in $J^{(k-1)}$. Furthermore, we use previously estimated points as anchors to estimate new points. This not only helps in stitching new points into the current global configuration, but also increases the chances of correcting the positions of previously badly estimated points (since more anchor information is available when more points are correctly estimated).

6. Postprocessing refinement by a gradient descent method. The positions estimated from the SDP and stitching steps can be further refined by applying a local optimization method to the following problem:

$$\min_{X \in \mathcal{R}^{d\times n}} \left\{ f(X) := \sum_{(i,j)\in\mathcal{N}} (\|x_i - x_j\| - d_{ij})^2 + \sum_{(j,k)\in\mathcal{M}} (\|x_j - a_k\| - h_{jk})^2 \right\}. \tag{15}$$

The method we suggest to improve the current solution is to move every position along the negative gradient direction of the function $f(X)$ in (15) to reduce the error function value. Now, we will explain the gradient method in more detail.

Let $\mathcal{N}_j = \{i : (i,j) \in \mathcal{N}\}$ and $\mathcal{M}_j = \{k : (j,k) \in \mathcal{M}\}$. By using the fact that $\nabla_x \|x-b\| = (x-b)/\|x-b\|$ if $x \neq b$, it is easy to show that for the objective function $f(X)$ in (15), the gradient $\nabla_j f$ with respect to the position x_j is given by

$$\nabla_j f(X) \;=\; 2\sum_{i\in\mathcal{N}_j} \left(1 - \frac{d_{ij}}{\|x_j - x_i\|}\right)(x_j - x_i) + 2\sum_{i\in\mathcal{M}_j} \left(1 - \frac{h_{jk}}{\|x_j - a_k\|}\right)(x_j - a_k). \tag{16}$$

Notice that $\nabla_j f$ involves only points that are connected to x_j. Thus $\nabla_j f$ can be computed in a distributed fashion.

The gradient-descent method is a local optimization method that generally does not deliver the global optimal solution of a nonconvex problem, unless a good starting iterate is available. The graph realization problem described in this paper is a nonconvex optimization problem. Hence a pure gradient-descent method would not work. However, the SDP estimated solutions are generally close to the global minimum, and so they serve as excellent initial points to start the local optimization.

Different objective functions can also be used in the gradient-descent method; for example, one could have considered the objective function $\sum_{(i,j)\in\mathcal{N}}(\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(j,k)\in\mathcal{M}}(\|x_j - a_k\|^2 - h_{jk}^2)^2$ instead of the one in (15). But we have found that there are no significant differences in the quality of the estimated positions produced by the gradient method using either objective function. The one considered in (15) is easily computed and is a good indicator of the estimation error. In our implementation, we use the gradient refinement step quite extensively, both after the SDP step for a single block, and also after each stitching phase between two blocks. In the single block case, the calculation does not involve the use of any anchor points, but when used after stitching, we fix the previous correctly estimated points as anchors.

7. A distributed SDP algorithm for anchor-free graph realization. We will now describe the complete distributed algorithm for solving a large-scale

anchor-free graph realization problem. To facilitate the description of our distributed algorithm for anchor-free graph realization, we first describe the centralized algorithm for solving (1). It is important to note here that the terms localization and realization are used interchangeably.

CENTRALIZED GRAPH LOCALIZATION (CGL) ALGORITHM.

Input: $(\underline{D}, \overline{D}, \mathcal{N}; \{a_1, \ldots, a_m\}, \mathcal{M})$.

Output: Estimated positions, $[\bar{x}_1, \ldots, \bar{x}_n] \in \mathcal{R}^{d\times n}$, and corresponding accuracy measures; trace errors, $T_1, \ldots, T_n$, and local error measures, $\widehat{E}_1, \ldots, \widehat{E}_n$.

1. Formulate the optimization problem (10) and solve the resulting SDP. Let $X = [x_1, \ldots, x_n]$ be the estimated positions obtained from the SDP solution.
2. Perform the gradient-descent algorithm on (15) using the SDP solution as the starting iterate to get more refined estimated positions $[\bar{x}_1, \ldots, \bar{x}_n]$.
3. For each $j = 1, \ldots, n$, label the point j as localized or unlocalized based on the error measures T_j and $\widehat{E}_j$.

DISTRIBUTED ANCHOR-FREE GRAPH LOCALIZATION (DAFGL) ALGORITHM.

Input: Upper and lower bounds on a subset of the pairwise distances in a molecule.

Output: A configuration of all the atoms in the molecule that is closest (in terms of the RMSD error described in section 8) to the actual molecule (from which the measurements were taken).

1. Divide the entire point set into subblocks using the clustering algorithm described in section 4 on the sparse distance matrices.
2. Apply the CGL algorithm to each subblock.
3. Stitch the subblocks together using the procedure described in section 5. After each stitching phase, refine the point positions again using the gradient-descent method described in section 6 and update their error measures.

Some remarks are in order for the DAFGL algorithm. In step 3, we can solve each cluster individually, and the computation is highly distributive. In using the CGL algorithm to solve each cluster, the computational cost is dominated by the solution of the SDP problem (the SDP cost is in turn determined by the number of given distances). For a graph with n nodes and m given pairwise distances, the computational complexity in solving the SDP is roughly $O(m^3) + O(n^3)$, provided that sparsity in the SDP data is fully exploited. For a graph with 200 nodes and the number of given distances being 10% of the total number of pairwise distances, the SDP would have roughly 2000 equality constraints and matrix variables of dimension 200. Such an SDP can be solved on a Pentium IV 3.0 GHz PC with 2GB RAM in about 36 and 93 seconds using the general purpose SDP software SDPT3-3.1 [34] and SeDuMi-1.05 [29], respectively.

The computational efficiency in the CGL algorithm can certainly be improved in various ways. First, the SDP problem need not be solved to high accuracy. It is sufficient to have a low accuracy SDP solution if it is used only as a starting iterate for the gradient-descent algorithm. There are various highly efficient methods (such as iterative solver-based interior-point methods [30] or the SDPLR method of Burer and Monteiro [10]) to obtain a low accuracy SDP solution. Second, a dedicated solver based on a dual scaling algorithm can also speed up the SDP computation. Substantial speed up can be expected if the computation exploits the low rank structure present in the constraint matrices. However, as our focus in this paper is not on improving the computational efficiency of the CGL algorithm, we shall not discuss this issue further. In the numerical experiments conducted in section 8, we use the software packages SDPT3-3.1 and SeDuMi to solve the SDP in the CGL algorithm. An alternative is to

use the software DSDP5.6 [3], which is expected to be more efficient than SDPT3-3.1 or SeDuMi.

The stitching process in step 4 is sequential in nature. But this does not imply that the distributed DAFGL algorithm is redundant and that the centralized CGL algorithm is sufficient for computational purposes. For a graph with 10000 nodes and the number of given distances being 1% of the total number of all pairwise distances, the SDP problem that needs to be solved by the CGL algorithm would have 500000 constraints and matrix variables of dimension 10000. Such a large-scale SDP is well beyond the range that can be solved routinely on a standard workstation available today. By considering smaller blocks, the distributed algorithm does not suffer from the limitation faced by the CGL algorithm.

If there are multiple computer processors available, say p of them, the distributed algorithm can also take advantage of the extra computing power. The strategy is to divide the graph into p large blocks using step 2 of the DAFGL algorithm and apply the DAFGL algorithm to localize one large block on each processor.

We end this section with two observations on the DAFGL algorithm. First, we observed that usually the outlying points, which have low connectivity, are not well estimated in the initial stages of the method. As the number of well-estimated points grows gradually, more and more of these "loose" points are estimated by the gradient-descent algorithm. As the molecules get larger, the frequency of having nonlocalizable subconfigurations in step 4 also increases. Thus the point-by-point stitching procedure of the algorithm described in section 5 gets visited more and more often.

Second, for large molecules, the sizes of the overlapping blocks need to be larger for the stitching algorithm in section 5 to be robust (more common points generally lead to more accuracy in stitching). But to accommodate larger overlapping blocks, each subgraph in the DAFGL algorithm will correspondingly be larger, and that in turn increases the problem size of the SDP relaxation. In our implementation, we apply the idea of dropping redundant constraints to reduce the computational effort in selecting large subblock sizes of 100–150. This strategy works because many of the distance constraints are for some of the densest parts of the subblock, and the points in these dense sections can actually be estimated quite well with only a fraction of those distance constraints. Therefore in the SDP step, we limit the number of distance constraints for each point to fewer than 6. If there are more distance constraints, they are not included in the SDP step. This allows us to choose large overlapping block sizes while the corresponding SDPs for larger clusters can be solved without too much additional computational effort.

8. Computational results. To evaluate our DAFGL algorithm, numerical experiments were performed on protein molecules, with the number of atoms in each molecule ranging from a few hundreds to a few thousands. We conducted our numerical experiments in MATLAB on a single Pentium IV 3.0 GHz PC with 2GB of RAM. The known 3-D coordinates of the atoms were taken from the Protein Data Bank (PDB) [4]. These were used to generate the true distances between the atoms. Our goal in the experiments was to reconstruct as closely as possible the known molecular configuration for each molecule, using only distance information generated from a sparse subset of all the pairwise distances. This information was in the form of upper and lower bounds on the actual pairwise distances.

For each molecule, we generated the partial distance matrix as follows. If the distance between two atoms was less than a given cut-off radius R, the distance was kept; otherwise, no distance information was known about the pair. The cut-off

radius R is chosen to be 6Å ($1\text{Å} = 10^{-8}$cm), which is roughly the maximum distance that NMR techniques can measure between two atoms. Therefore, in this case, $\mathcal{N} = \{(i,j) : \|x_i - x_j\| \leq 6\text{Å}\}$.

We then perturb the distances to generate upper and lower bounds on the given distances in the following manner. Assuming that $\hat{d}_{ij}$ is the true distance between atom i and atom j, we set

$$\bar{d}_{ij} = \hat{d}_{ij}(1 + |\bar{\varepsilon}_{ij}|),$$

$$\underline{d}_{ij} = \hat{d}_{ij} \max(0, 1 - |\underline{\varepsilon}_{ij}|),$$

where $\bar{\varepsilon}_{ij}, \underline{\varepsilon}_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$.

By varying σ_{ij} (which we keep as the same for all pairwise distances), we control the noise in the data. This is a multiplicative noise model, where a higher distance value means more uncertainty in its measurement. For the rest of our discussion, we will refer to σ_{ij} in percentage values. For example, $\sigma_{ij} = 0.1$ will be referred to as 10% noise on the upper and lower bounds.

Typically, not all the distances below 6Å are known from NMR experiments. Therefore, we will also present results for the DAFGL algorithm when only a fraction of all the distances below 6Å are chosen randomly and used in the calculation.

Let $\mathcal{Q}$ be the set of orthogonal matrices in $\mathcal{R}^{d\times d}$ ($d = 3$). We measure the accuracy performance of our algorithm by the following criteria:

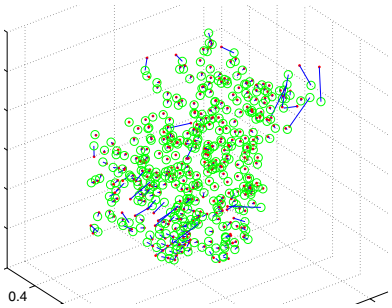
$$\text{RMSD} = \frac{1}{\sqrt{n}} \min\{\|X^{\text{true}} - QX - h\|_F \,|\, Q \in \mathcal{Q},\ h \in \mathcal{R}^d\}, \tag{17}$$

$$\text{LDME} = \left(\frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} \Big(\|x_i - x_j\| - d_{ij} \Big)^2 \right)^{1/2}. \tag{18}$$

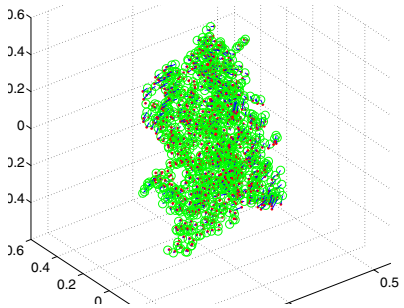
The first criterion RMSD (root mean square deviation) requires the knowledge of the true configuration, whereas the second does not. Thus the second criterion LDME (local distance matrix error) is more practical, but it is also less reliable in evaluating the true accuracy of the constructed configuration. The practically useful measure LDME gives lower values than the RMSD, and as the noise increases, it is not a very reliable measure.

When 90% of the distances (below 6Å) or more were given, and were not corrupted by noise, the molecules considered here were estimated very precisely with RMSD = 10^{-4}–10^{-6}Å. This goes to show that the algorithm performs remarkably well when there is enough exact distance data. In this regard, our algorithm is competitive compared to the geometric build-up algorithm in [37] which is designed specifically for graph localization with exact distance data. With exact distance data, we have solved molecules that are much larger and with much sparser distance data than those considered in [37].

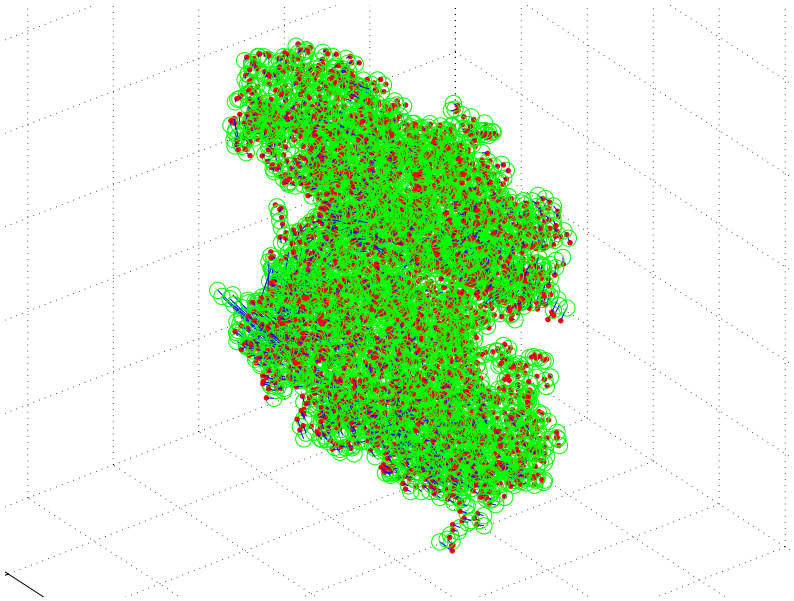
However, our focus in this paper is more on molecular conformation with noisy and sparse distance data. We will present results for only such cases. In Figure 2, the original true and the estimated atom positions are plotted for some of the molecules, with varying amounts of distance data and noise. The open green circles correspond to the true positions and solid red dots to their estimated positions from our computation. The error offset between the true and estimated positions for an individual



(a) 1PTQ(402 atoms) with 30% of distances below 6Å and 1% noise on upper and lower bounds, RMSD = 0.9858 Å.



(b) 1AX8(1003 atoms) with 70% of distances below 6Å and 5% noise on upper and lower bounds, RMSD = 0.8184 Å.



(c) 1TOA(4292 atoms) with 100% of distances below 6Å and 10% noise on upper and lower bounds, RMSD = 1.5058 Å.

FIG. 2. *Comparison of actual and estimated molecules.*

atom is depicted by a solid blue line. The solid lines, however, are not visible for most of the atoms in the plots because we are able to estimate the positions very accurately.

The plots show that even for very sparse data (as in the case of 1PTQ), the estimation for most of the atoms is accurate. The atoms that are badly estimated are the ones that have too little distance information to be localized. The algorithm also performs well for very noisy data, and for large molecules, as demonstrated for 1AX8 and 1TOA. The largest molecule we tried the algorithm on is 1I7W, with 8629 atoms. Figure 3 shows that even when the distance data is highly noisy (10% error

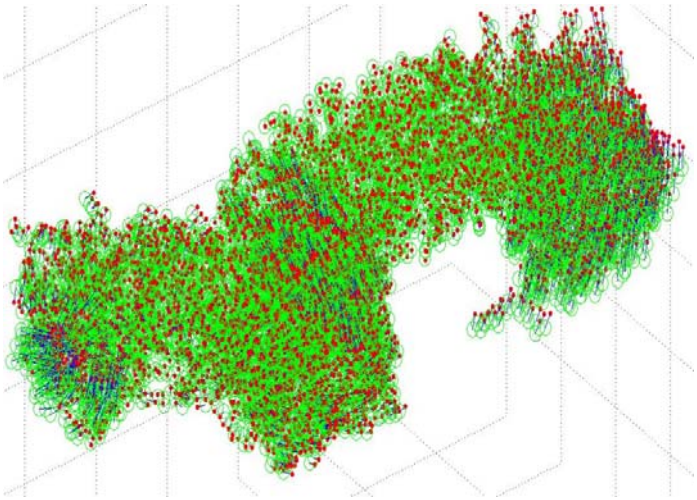


FIG. 3. *1I7W(8629 atoms) with 100% of distances below 6Å and 10% noise on upper and lower bounds, RMSD = 1.3842 Å.*

on upper and lower bounds), the estimation is close to the original with an RMSD error = 1.3842 Å.

However, despite using more common points for stitching, the DAFGL algorithm can sometimes generate estimations with high RMSD error for very large molecules, due to a combination of irregular geometry, very sparse distance data, and noisy distances. Ultimately, the problem boils down to being able to correctly identify when a point has been badly estimated. Our measures using trace error (8) and local error (12) are able to isolate the majority of the points that are badly estimated, but do not always succeed when many of the points are badly estimated. In fact, for such molecules, a lot of the computation time is spent in the point-by-point stitching phase, where we attempt to repeatedly solve for better estimations of the badly estimated points, and if that fails repeatedly, the estimations continue to be poor. If the number of badly estimated points is very high, it may affect the stitching of the subsequent clusters as well. In such cases, the algorithm more or less fails to find a global configuration. Examples of such cases are the 1NF7 molecule (5666 atoms) and the 1HMV molecule (7398 atoms) solved with 10% noise. While they are estimated correctly when there is no noise, the 1NF7 molecule estimation returns an RMSD of 25.1061 Å and the 1HMV molecule returns 28.3369 Å.

A more moderate case of stitching failure can be seen in Figure 4 for the molecule 1BPM with 50% of the distance below 6Å and 5% error on upper and lower bounds; the problem is, in particular, clusters (which are circled in the figure). Although they have correct local geometry, their positions with respect to the entire molecule do not. This indicates that the stitching procedure has failed because some of the common points are not estimated correctly and are then used in the stitching process, thus destroying the entire local configuration. So far, this is the weakest part of the algorithm, and future work is heavily focussed on developing better error measures to isolate the badly estimated points and to improve the robustness of the stitching process.

In Figure 5, we plotted the 3-D configuration (via Swiss PDBviewer [16]) of some of the molecules to the left and their estimated counterparts (with different distance

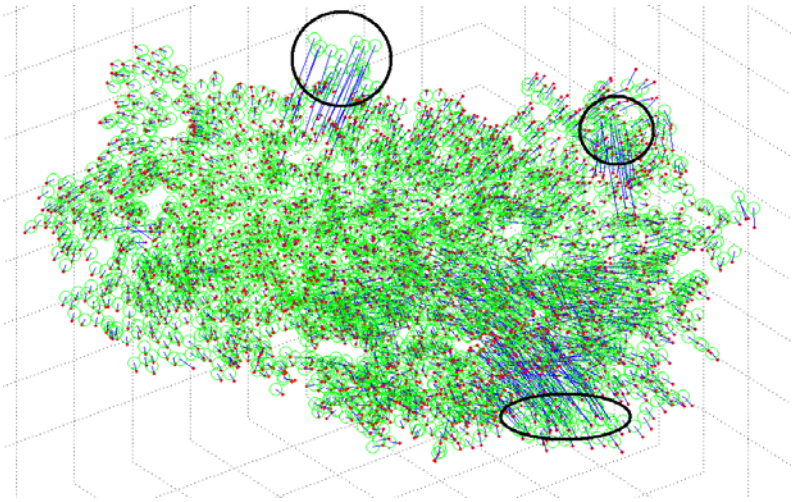
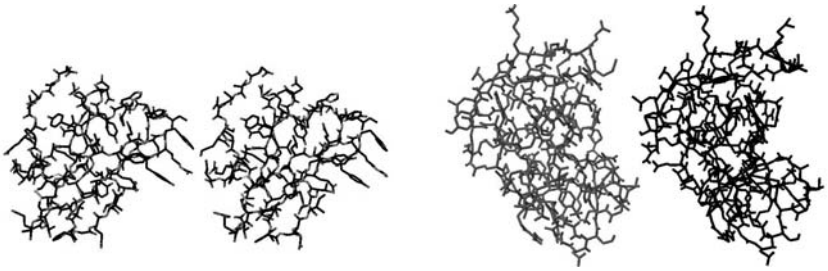
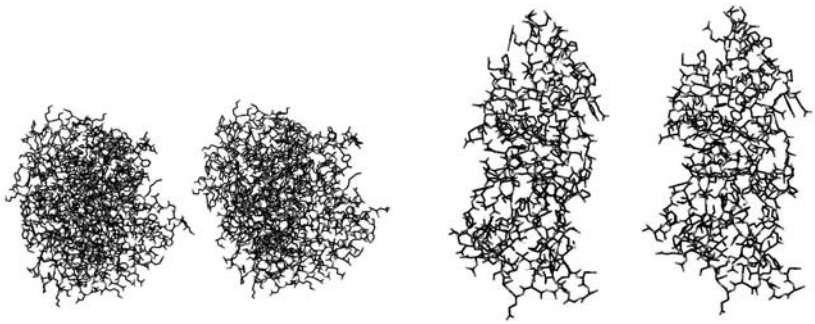


FIG. 4. *1BPM(3672 atoms) with 50% of distances below 6Å and 5% noise on upper and lower bounds, RMSD = 2.4360 Å.*



(a) 1HOE(558 atoms) with 40% of distances below 6Å and 1% noise on upper and lower bounds, RMSD = 0.2154 Å.

(b) 1PHT(814 atoms) with 50% of distances below 6Å and 5% noise on upper and lower bounds, RMSD = 1.2014 Å.



(c) 1RHJ(3740 atoms) with 70% of distances below 6Å and 5% noise on upper and lower bounds RMSD = 0.9535 Å.

(d) 1F39(1534 atoms) with 85% of distances below 6Å and 10% noise on upper and lower bounds, RMSD = 0.9852 Å.

FIG. 5. *Comparison between the original (left) and reconstructed (right) configurations for various protein molecules using Swiss PDB viewer.*

TABLE 1
Results for 100% of distances below 6Å, and 10% noise on upper and lower bounds.

PDB ID	No. of atoms	% of total pairwise distances used	RMSD(Å)	LDME(Å)	CPU time (secs)
1PTQ	402	8.79	0.1936	0.2941	107.7
1HOE	558	6.55	0.2167	0.2914	108.7
1LFB	641	5.57	0.2635	0.1992	129.1
1PHT	814	5.35	1.2624	0.2594	223.9
1POA	914	4.07	0.4678	0.2465	333.1
1AX8	1003	3.74	0.6408	0.2649	280.1
1F39	1534	2.43	0.7338	0.2137	358.0
1RGS	2015	1.87	1.6887	0.1800	665.9
1KDH	2923	1.34	1.1035	0.2874	959.1
1BPM	3672	1.12	1.1965	0.2064	1234.7
1RHJ	3740	1.10	1.8365	0.1945	1584.4
1HQQ	3944	1.00	1.9700	0.2548	1571.8
1TOA	4292	0.94	1.5058	0.2251	979.5
1MQQ	5681	0.75	1.4906	0.2317	1461.1

data inputs) to the right. As can be seen clearly, the estimated counterparts closely resemble the original molecules.

The results shown in the following tables are a good representation of the performance of the algorithm on different size molecules with different types of distance data sets. The numerical results presented in Table 1 are for the case when all distances below 6Å are used and perturbed with 10% noise on lower and upper bounds. Table 2 contains the results for the case when only 70% of distances below 6Å are used and perturbed with 5% noise, and also for the case when only 50% of distances below 6Å are used and perturbed with 1% noise. The results for 50% distance and 1% noise are representative of cases with sparse distance information and low noise, 100% distance and 10% noise represent relatively denser but highly noisy distance information, and 70% distance and 5% noise is a middle ground between the two extreme cases.

We can see from the values in Table 1 that LDME is not a very good measure of the actual estimation error as given by RMSD, since the former does not correlate well with the latter. Therefore we do not report the LDME values in Table 2.

From the tables, it can be observed that for relatively dense distance data (100% of all distances below 6Å), the estimation error stays below 2Å even when the upper and lower bounds are very loose. The algorithm is seen to be quite robust to high noise when there is enough distance information. The estimation error is also quite low for most molecules for cases when the distance information is very sparse but much more precise. In the sparse distance cases, it is the molecules that have more irregular geometries that suffer the most from lack of enough distance data and exhibit high estimation errors. The combination of sparsity and noise has detrimental impact on the algorithm's performance, as can be seen for the results with 70% distances and 5% noise.

In Figure 6, we plot the CPU times required by our DAFGL algorithm to localize a molecule with n atoms versus n (with different types of distance inputs). As the

TABLE 2
Results with 70% of distances below 6Å and 5% noise on upper and lower bounds, and with 50% of distances below 6Å and 1% noise on upper and lower bounds.

PDB ID	No. of atoms	70% Distances, 5% Noise		50% Distances, 1% Noise	
		RMSD(Å)	CPU time (secs)	RMSD(Å)	CPU time (secs)
1PTQ	402	0.2794	93.8	0.7560	22.1
1HOE	558	0.2712	129.6	0.0085	32.5
1LFB	641	0.4392	132.5	0.2736	41.6
1PHT	814	0.4701	129.4	0.6639	53.6
1POA	914	0.4325	174.7	0.0843	54.1
1AX8	1003	0.8184	251.9	0.0314	71.8
1F39	1534	1.1271	353.1	0.2809	113.4
1RGS	2015	4.6540	613.3	3.5416	308.2
1KDH	2923	2.5693	1641.0	2.8222	488.4
1BPM	3672	2.4360	1467.1	1.0502	384.8
1RHJ	3740	0.9535	1286.1	0.1158	361.5
1HQQ	3944	8.9106	2133.5	1.6610	418.4
1TOA	4292	9.8351	2653.6	1.5856	372.6
1MQQ	5681	3.1570	1683.4	2.3108	1466.2

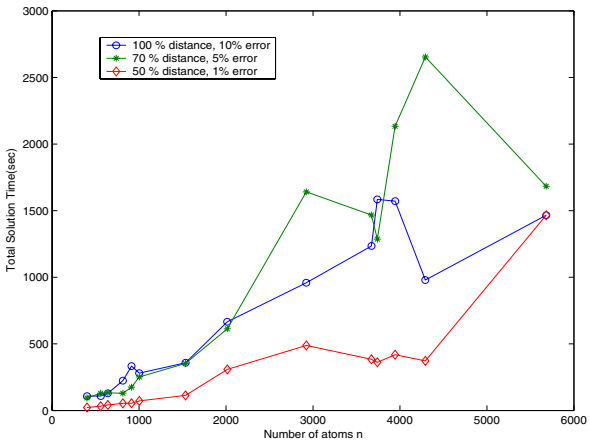


FIG. 6. *CPU time taken to localize a molecule with n atoms versus n.*

distance data becomes more and more sparse, the number of constraints in the SDP also reduce, and therefore they take less time to be solved in general. However, many points may be incorrectly estimated in the SDP phase, and so the stitching phase usually takes longer in these cases. This behavior is exacerbated by the presence of higher noise.

Our algorithm is reasonably efficient in solving large problems. The spikes that we see in the graphs for some of the larger molecules also correspond to cases with high RMSD error, in which the algorithm fails to find a valid configuration of points, either

due to very noisy or very sparse data. A lot of time is spent in recomputing points in the stitching phase, and many of these points are repeatedly estimated incorrectly. The number of badly estimated points grows at each iteration of the stitching process and becomes more and more time-consuming. But, given the general computational performance, we expect our algorithm to be able to handle even larger molecules if the number of badly estimated points can be kept low. On the other hand, we are also investigating methods that discard repeatedly badly estimated points from future calculations.

9. Conclusion and work in progress. An SDP-based distributed method to solve the distance geometry problem in three dimensions with incomplete and noisy distance data and without anchors is described. The entire problem is broken down into subproblems by intelligent clustering methods. An SDP relaxation problem is formulated and solved for each cluster. Matrix decomposition is used to find local configurations of the clusters, and a least squares–based stitching method is applied to finding a global configuration. Gradient-descent methods are also employed in intermediate steps to refine the quality of the solution.

The performance of the algorithm is evaluated by using it to find the configurations of large protein molecules with a few thousands atoms. The distributed SDP approach can solve large problems having favorable geometries with good accuracy and speed when 50–70% distances (corrupted by moderate level of 0–5% noise in both the lower and upper bounds) below 6Å are given.

The current DAFGL algorithm needs to be improved in order to work on very sparse (30–50% of all distances below 6Å) and highly noisy (10–20% noise) data, which is often the case for actual NMR data used to deduce molecular conformation. For the rest of this section, we outline some possible improvements that can be made to our current algorithm.

One of the main difficulties we encounter in the current distributed algorithm is the propagation of position estimation errors in a cluster to other clusters during the stitching process when the given distances are noisy. Even though we have had some successes in overcoming this difficulty, it is not completely alleviated in the current paper. The difficulties faced by our current algorithm with very noisy or very sparse data cases are particularly noticeable for very large molecules (which correspond to a large number of subblocks that need stitching). Usually, the estimation for many of the points in some of the subblocks from the CGL step is not accurate enough in these cases. This is especially problematic when there are too few common points that can then be used for stitching with the previous block, or if the error measures used to identify the bad points are unable to filter out some of the badly estimated common points. This usually leads to the error propagating to subsequent blocks as well. Therefore, the bad point detection and stitching phases need to be made more robust.

To reduce the effect of inadvertently using a badly estimated point for stitching, we can increase the number of common points used in the stitching process, and at the same time, use more sophisticated stitching algorithms that not only stitch correctly estimated points but also isolate the badly estimated ones. As far as stitching is concerned, currently we use the coordinates of the common points between two blocks to find the best affine mapping that would bring the two blocks into the same coordinate system. Another idea is to fix the values in the matrix Y in (11) that correspond to the common points, based on the values that were obtained for it in solving the SDP for the previous block. By fixing the values, we are indirectly using the common points to anchor the new block with the previous one. The dual of the

SDP relaxation also merits further investigation, for possible improvements in the computational effort required and for more robust stitching results.

With regard to error measures, it would be useful to study whether the local error measure (12) can be made more sophisticated by including a larger set of points to check its distances with, as opposed to just its immediate neighborhood. In some cases, the distances are satisfied within local clusters, but the entire cluster itself is badly estimated (and the local error measure fails to filter out many of the badly estimated points in this scenario). Also, we need to investigate the careful selection of the tolerance T_ε in deciding which points have been estimated correctly and can be used for stitching.

As has been noted before, our current algorithm does not use the VDW minimum separation distance constraints of the form $\|x_i - x_j\| \geq L_{ij}$ described in [26] and [36]. The VDW constraints played an essential role in those previous works in reducing the search space of valid configurations, especially in the case of sparse data where there are many possible configurations fitting the sparse distance constraints. As mentioned in section 3, VDW lower bound constraints can be added to the SDP model, but we would need to keep track of the type of atom to which each point corresponds. However, one must be mindful that the VDW constraints should only be added when necessary so as not to introduce too many redundant constraints. If the VDW constraints between all pairs of atoms are added to the SDP model, then the number of lower bound constraints is of the order n^2, where n is the number of points. Thus even if n is below 100, the total number of constraints could be in the range of thousands. However, many of the VDW constraints are for two very remote points, and they are often inactive or redundant at the optimal solution. Therefore, we can adopt an iterative active constraint generation approach. We first solve the SDP problem by completely ignoring the VDW lower bound constraints to obtain a solution. Then we verify the VDW lower bound constraints at the current solution for all the points and add the violated ones into the model. We can repeat this process until all the VDW constraints are satisfied. Typically, we would expect only $O(n)$ VDW constraints to be active at the final solution.

We are optimistic that by combining the ideas presented in previous work on molecular conformation (especially incorporating domain knowledge such as the minimum separation distances derived from VDW interactions) with the distributed SDP-based algorithm in this paper, the improved distributed algorithm would likely be able to calculate the conformation of large protein molecules with satisfactory accuracy and efficiency in the future. Based on the performance of the current DAFGL algorithm, and the promising improvements to the algorithm we have outlined, a realistic target for us to set in the future is to correctly calculate the conformation of a large molecule (with 5000 atoms or more) given only about 50% of all pairwise distances below 6Å, and corrupted by 10–20% noise.

Acknowledgments. We would like to thank the anonymous referees for their insightful comments and suggestions that led to a major revision of the paper.

REFERENCES

[1] S. Al-Homidan and H. Wolkowicz, *Approximate and exact completion problems for Euclidean distance matrices using semidefinite programming*, Linear Algebra Appl., 406 (2005), pp. 109–141.

[2] A. Y. Alfakih, A. Khandani, and H. Wolkowicz, *Solving Euclidean distance matrix completion problems via semidefinite programming*, Comput. Optim. Appl., 12 (1999), pp. 13–30.

[3] S. J. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM J. Optim., 10 (2000), pp. 443–461.
[4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, *The protein data bank*, Nucleic Acids Res., 28 (2000), pp. 235–242.
[5] P. Biswas, T.-C. Liang, K.-C. Toh, T.-C. Wang, and Y. Ye, *Semidefinite programming approaches for sensor network localization with noisy distance measurements*, IEEE Trans. Automat. Sci. Engrg, Special Issue on Distributed Sensing, 3 (2006), pp. 360–371.
[6] P. Biswas, T.-C. Liang, T.-C. Wang, and Y. Ye, *Semidefinite programming based algorithms for sensor network localization*, ACM Trans. Sensor Networks, 2 (2006), pp. 188–220.
[7] P. Biswas and Y. Ye, *A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization*, in Mutiscale Optimization Methods and Applications, W. W. Hager, S.-J. Huang, P. M. Pardalos, and O. A. Prokopyev, eds., Springer-Verlag, NY, 2006, pp. 69–82.
[8] P. Biswas and Y. Ye, *Semidefinite programming for ad hoc wireless sensor network localization*, in Proceedings of the Third International Symposium on Information Processing in Sensor Networks, ACM Press, New York, 2004, pp. 46–54.
[9] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Stud. Appl. Math. 15, SIAM, Philadelphia, 1994.
[10] S. Burer and R. D. C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Programming, 95 (2003), pp. 329–357.
[11] M. W. Carter, H. H. Jin, M. A. Saunders, and Y. Ye, *Spaseloc: An adaptive subproblem algorithm for scalable wireless sensor network localization*, SIAM J. Optim., 17 (2006), pp. 1102–1128.
[12] B. Chazelle, C. Kingsford, and M. Singh, *The side-chain positioning problem: A semidefinite programming formulation with new rounding schemes*, in Proceedings of the Paris C. Kanellakis Memorial Workshop on Principles of Computing & Knowledge (PCK50), ACM Press, New York, 2003, pp. 86–94.
[13] G. Crippen and T. Havel, *Distance Geometry and Molecular Conformation*, Wiley, New York, 1988.
[14] Q. Dong and Z. Wu, *A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data*, J. Global Optim., 26 (2003), pp. 321–333.
[15] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall Professional Technical Reference, Prentice–Hall, Englewood Cliffs, NJ, 1981.
[16] N. Guex and M. C. Peitsch, *Swiss-model and the Swiss-Pdbviewer: An environment for comparative protein modeling*, Electrophoresis, 18 (1997), pp. 2714–2723.
[17] O. Güler and Y. Ye, *Convergence behavior of interior point algorithms*, Math. Programming, 60 (1993), pp. 215–228.
[18] T. F. Havel and K. Wüthrich, *An evaluation of the combined use of nuclear magnetic resonance and distance geometry for the determination of protein conformation in solution*, J. Molec. Biol., 182 (1985), pp. 281–294.
[19] B. A. Hendrickson, *The Molecular Problem: Determining Conformation from Pairwise Distances*, Ph.D. thesis, Department of Computer Science, Cornell University, Ithaca, NJ, 1991.
[20] G. Iyengar, D. Phillips, and C. Stein, *Approximation algorithms for semidefinite packing problems with applications to maxcut and graph coloring*, in Eleventh Conference on Integer Programming and Combinatorial Optimization, Berlin, 2005.
[21] M. Laurent, *Matrix completion problems*, The Encyclopedia of Optimization, 3 (2001), pp. 221–229.
[22] P. Biswas, T.-C. Liang, T.-C. Wang, and Y. Ye, *Semidefinite programming based algorithms for sensor network localization*, ACM Trans. Sensor Networks, 2 (2006), pp. 188–220.
[23] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica, 15 (1995), pp. 215–245.
[24] J. J. Moré and Z. Wu, *Global continuation for distance geometry problems*, SIAM J. Optim., 7 (1997), pp. 814–836.
[25] J. J. Moré and Z. Wu, *ϵ-optimal Solutions to Distance Geometry Problems via Global Continuation*, Preprint MCS–P520–0595, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1994.
[26] R. Reams, G. Chatham, W. K. Glunt, D. McDonald, and T. L. Hayden, *Determining protein structure using the distance geometry program APA*, Computers & Chemistry, 23 (1999), pp. 153–163.

[27] 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), Miami, FL, SIAM, Philadelphia, 2006, pp. 766–775.

[28] A. M.-C. So and Y. Ye, *Theory of semidefinite programming relaxation for sensor network localization*, Math. Programming, 109 (2007), pp. 367–384.

[29] J. F. Sturm, *Using SeDuMi* 1.02*, A MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11 & 12 (1999), pp. 625–633.

[30] K.-C. Toh, *Solving large scale semidefinite programs via an iterative solver on the augmented systems*, SIAM J. Optim., 14 (2003), pp. 670–698.

[31] M. W. Trosset, *Applications of multidimensional scaling to molecular conformation*, Comput. Sci. Statist., 29 (1998), pp. 148–152.

[32] M. W. Trosset, *Distance matrix completion by numerical optimization*, Comput. Optim. Appl., 17 (2000), pp. 11–22.

[33] M. W. Trosset, *Extensions of classical multidimensional scaling via variable reduction*, Comput. Statist., 17 (2002), pp. 147–162.

[34] R. H. Tütüncü, K. C. Toh, and M. J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT*3, Math. Programming, Ser. B, 95 (2003), pp. 189–217.

[35] K. Q. Weinberger, F. Sha, and L. K. Saul, *Learning a kernel matrix for nonlinear dimensionality reduction*, in Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04), ACM Press, New York, 2004, pp. 839–846.

[36] G. A. Williams, J. M. Dugan, and R. B. Altman, *Constrained global optimization for estimating molecular structure from atomic distances*, J. Comput. Biol., 8 (2001), pp. 523–547.

[37] D. Wu and Z. Wu, *An updated geometric build-up algorithm for molecular distance geometry problems with sparse distance data*, J. Global Optimization, 37 (2007), pp. 661–673.

[38] K. Varadarajan, S. Venkatesh, Y. Ye, and J. Zhang, *Approximating the radii of point sets*, SIAM J. Comput., 36 (2007), pp. 1764–1776.

[39] J.-M. Yoon, Y. Gad, and Z. Wu, *Mathematical Modeling of Protein Structure Using Distance Geometry*, in Numerical Linear Algebra and Optimization, Y. Yuan, ed., Scientific Press, Beijing, China, 2004.

[40] Z. Zhang and H. Zha, *Principal manifolds and nonlinear dimensionality reduction via tangent space alignment*, SIAM J. Sci. Comput., 26 (2004), pp. 313–338.