# An SDP-Based Divide-and-Conquer Algorithm for Large-scale Noisy Anchor-free Graph Realization

SIAM J. SCI. COMPUT.
Vol. 31, No. 6, pp. 4351–4372


# AN SDP-BASED DIVIDE-AND-CONQUER ALGORITHM FOR LARGE-SCALE NOISY ANCHOR-FREE GRAPH REALIZATION*

NGAI-HANG Z. LEUNG† AND KIM-CHUAN TOH†‡

**Abstract.** We propose the DISCO algorithm for graph realization in $\mathbb{R}^d$, given sparse and noisy short-range intervertex distances as inputs. Our divide-and-conquer algorithm works as follows. When a group has a sufficiently small number of vertices, the basis step is to form a graph realization by solving a semidefinite program. The recursive step is to break a large group of vertices into two smaller groups with overlapping vertices. These two groups are solved recursively, and the subconfigurations are stitched together, using the overlapping atoms, to form a configuration for the larger group. At intermediate stages, the configurations are improved by gradient descent refinement. The algorithm is applied to the problem of determining protein moleculer structure. Tests are performed on molecules taken from the Protein Data Bank database. For each molecule, given 20–30% of the inter-atom distances less than 6Å that are corrupted by a high level of noise, DISCO is able to reliably and efficiently reconstruct the conformation of large molecules. In particular, given 30% of distances with 20% multiplicative noise, a 13000-atom conformation problem is solved within an hour with a root mean square deviation of 1.6Å.

**Key words.** SDP, molecular conformation, anchor-free graph realization, divide-and-conquer

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

**DOI.** 10.1137/080733103

## 1. Introduction.

A graph realization problem is to assign coordinates to vertices in a graph, with the restriction that distances between certain pairs of vertices are specified to lie in given intervals. Two practical instances of the graph realization problem are the molecular conformation and sensor network localization problems.

The *molecular conformation problem* is to determine the structure of a protein molecule based on pairwise distances between atoms. Determining protein conformations is central to research in biology, because knowledge of the protein structure aids in the understanding of protein functions, which could lead to further medical applications. In this problem, the distance constraints are obtained from knowledge of the sequence of constituent amino acids, from minimum separation distances (MSDs) derived from van der Waals interactions, and from nuclear magnetic resonance (NMR) spectroscopy experiments. We take note of two important characteristics of molecular conformation problems: the number of atoms may go up to tens of thousands, and the distance data may be very sparse and highly noisy.

The *sensor network localization problem* is to determine the location of wireless sensors in a network. In this problem, there are two classes of objects: anchors (whose locations are known a priori) and sensors (whose locations are unknown and to be determined). In practical situations, the anchors and sensors are able to communicate with one another, if they are not too far apart (say within radio range), in order to estimate the distance between them.

*Received by the editors August 19, 2008; accepted for publication (in revised form) October 2, 2009; published electronically December 16, 2009.
http://www.siam.org/journals/sisc/31-6/73310.html
†Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543, Singapore (deutsixfive@gmail.com, mattohkc@nus.edu.sg).
‡Singapore-MIT Alliance, 4 Engineering Drive 3, Singapore 117576, Singapore.

While the two problems are very similar, the key difference between molecular conformation and sensor network localization is that the former is anchor-free, whereas in the latter the positions of the anchor nodes are known a priori.

Recently, semidefinite programming (SDP) relaxation techniques have been applied to the sensor network localization problem [2]. While this approach was successful for moderate-size problems with sensors on the order of a few hundred, it was unable to solve problems with large numbers of sensors, due to limitations in SDP algorithms, software, and hardware. A distributed SDP-based algorithm for sensor network localization was proposed in [4], with the objective of localizing larger networks. One critical assumption required for the algorithm to work well is that there exist anchor nodes distributed uniformly throughout the physical space. The algorithm relies on the anchor nodes to divide the sensors into clusters, and solves each cluster separately using an SDP relaxation. In general, a divide-and-conquer algorithm must address the issue of combining the solutions of smaller subproblems into a solution for the larger subproblem. This is not an issue in the sensor network localization problem, because the solutions to the clusters automatically form a global configuration, as the anchors endow the sensors with global coordinates.

The distributed method proposed in [4] is not suitable for molecular conformations, since the assumption of uniformly distributed anchor nodes does not hold in the case of molecules. The authors of [4] proposed a distributed SDP-based algorithm (the DAFGL algorithm) for the molecular conformation problem [3]. The performance of the DAFGL algorithm is satisfactory when given 50% of pairwise distances less than 6Å that are corrupted by 5% multiplicative noise. The main objective of this paper is to design a robust and efficient distributed algorithm that can handle the challenging situation [26] when 30% of short-range pairwise distances are given and are corrupted with 10–20% multiplicative noise.

In this paper, we describe a new distributed approach, the DISCO (for DIStributed COnformation) algorithm, for the anchorless graph realization problem. By applying the algorithm to molecular conformation problems, we demonstrate its reliability and efficiency. In particular, for a 13000-atom protein molecule, we were able to estimate the positions to an RMSD (root mean square deviation) of 1.6Å given only 30% of the pairwise distances (corrupted by 20% multiplicative noise) less than 6Å.

Distributed algorithms (based on successive decomposition) similar to those in [3] have been proposed for fast manifold learning in [28, 29]. In addition, those papers considered recursive decomposition. The manifold learning problem is to seek a low-dimensional embedding of a manifold in a high-dimensional Euclidean space by modeling it as a graph-realization problem. The resulting problem has similar characteristics as the anchor-free graph-realization problem that we are considering in this paper, but there are some important differences which we should highlight. For the manifold learning problem, exact pairwise distances between any pairs of vertices are available, but for the problem considered in this paper, only a very sparse subset of pairwise distances are assumed to be given and are known only within given ranges. Such a difference implies that, for the former problem, any local patch will have a "unique" embedding (up to rigid body motion and certain approximation errors) computable via an eigenvalue decomposition, and the strategy to decompose the graph into subgraphs is fairly straightforward. In contrast, for the latter problem, given the sparsity of the graph and the noise in the distances data, the embedding problem itself requires a new method, not to mention that sophisticated decomposition strategies also need to be devised.

The remainder of the paper is organized as follows: section 2 describes existing molecular conformation algorithms, section 3 details the mathematical models for molecular conformation, section 4 explains the design of DISCO, section 5 contains the experiment setup and numerical results, and section 6 gives the conclusion.

The DISCO webpage [14] contains additional material, including the DISCO code, and a video of how DISCO solves the 1534-atom molecule 1F39.

In this paper, we adopt the following notational conventions. Lowercase letters, such as $n$, are used to represent scalars. Lowercase letters in bold font, such as $\boldsymbol{s}$, are used to represent vectors. Uppercase letters, such as $X$, are used to represent matrices. Uppercase letters in calligraphic font, such as $\mathcal{D}$, are used to represent sets. Cell arrays will be prefixed by a letter "c" and be in the math italic font, such as *cAest*. Cell arrays will be indexed by curly braces {}.

**2. Related work.** In this section, we give a brief tour of selected existing works. Since the focus of this work is on developing a molecular conformation algorithm that uses only pairwise interatom distances, our literature review will likewise be restricted to distance-based algorithms. Therefore, we do not discuss recent approaches that are heavily dependent on exploiting existing biological information using techniques such as integrating fragment libraries, scoring potentials that favor conformations with hydrophobic cores, or paired beta strands, among others.

Besides presenting the techniques used by various algorithms, we note that each algorithm was tested on different types of input data. For instance, some inputs were exact distances, while others were distances corrupted by low levels of noise, and yet others were distances corrupted with high levels of noise; some inputs consist of all the pairwise distances less than a certain cut-off distance, while others give only a proportion of the pairwise distances less than a certain cut-off distance. It is also the case that not all the authors used the same error measure. Although the accuracy of a molecular conformation is most commonly measured by the RMSD, some of the authors did not provide the RMSD error but only the maximum violation of lower or upper bounds for pairwise interatom distances. (We present more details about the RMSD measure in section 5.) Finally, because we aim to design an algorithm which is able to scale to large molecules, we make a note of the largest molecule which each algorithm was able to solve in the tests done by the authors. We summarize this information in Table 2.1.

**2.1. Methods using the inner product matrix.** It is known from the theory of distance geometry that there is a natural correspondence between inner product matrices and distance matrices [19]. Thus, one approach to the molecular conformation problem is to use a distance matrix to generate an inner product matrix, which can then be factorized to recover the atom coordinates. The methods we present in section 2.1 differ in how they construct the inner product matrix, but use the same procedure to compute the atom coordinates; we describe this procedure next. If we denote the atom coordinates by columns $\boldsymbol{x}_i$, and let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, then the inner product matrix $Y$ is given by $Y = X^T X$. We can recover approximate coordinates $\tilde{X}$ from a noisy $\tilde{Y}$ by taking the best rank-3 approximation $\tilde{Y} \approx \tilde{X}^T \tilde{X}$, based on the eigenvalue decomposition of $\tilde{Y}$.

The EMBED algorithm [11] was developed by Havel, Kuntz, and Crippen in 1983. Given lower and upper bounds on some pairwise distances as input, it attempts to find a feasible conformation as follows. It begins by using the triangle and tetrangle inequalities to compute distance bounds for all pairs of points. Then it chooses random numbers within the bounds to form an estimated distance matrix $\tilde{D}$, and checks

TABLE 2.1
*A summary of protein conformation algorithms.*

| Algorithms (year) | No. of atoms | Inputs | Output |
|---|---|---|---|
| EMBED (83), DISGEO (84), DG-II (91), APA (99) | 454 | All distance and chirality constraints needed to fix the covalent structure are given exactly. Some or all of the distances between hydrogen atoms less than 4Å apart and in different amino acid residues given as bounds. | RMSD 2.08Å |
| DGSOL (99) | 200 | All distances between atoms in the same and successive residues given as lying in $[0.84d_{ij}, 1.16d_{ij}]$. | RMSD 0.7Å |
| GNOMAD (01) | 1870 | All distances between atoms that are covalently bonded given exactly; all distances between atoms that share covalent bonds with the same atom given exactly; additional distances given exactly, so that 30% of the distances less than 6Å are given; physically inviolable minimum separation distance constraints given as lower bounds. | RMSD 2–3Å |
| MDS (02) | 700 | All distances less than 7Å were given as lying in $[d_{ij} - 0.01\text{Å}, d_{ij} + 0.01\text{Å}]$. | violations $< 0.01$Å |
| StrainMin (06) | 5147 | All distances less than 6Å are given exactly; a representative lower bound of 2.5Å is given for other pairs of atoms. | violations $< 0.1$Å |
| ABBIE (95) | 1849 | All distances between atoms in the same amino acid given exactly. All distances corresponding to pairs of hydrogen atoms less than 3.5Å apart from each other given exactly. | Exact |
| Geometric build-up (07) | 4200 | All distances between atoms less than 5Å apart from each other given exactly. | Exact |
| DAFGL (07) | 5681 | 70% of the distances less than 6Å were given as lying in $[\underline{d}_{ij}, \overline{d}_{ij}]$, where $\underline{d}_{ij} = \max(0, (1 - 0.05\lvert\underline{Z}_{ij}\rvert)d_{ij})$, $\overline{d}_{ij} = (1 + 0.05\lvert\overline{Z}_{ij}\rvert)d_{ij}$, and $\underline{Z}_{ij}$, $\overline{Z}_{ij}$ are standard normal random variables with zero mean and unit variance. | RMSD 3.16Å |

whether $\tilde{D}$ is close to a valid rank-3 Euclidean distance matrix. This step is repeated until a nearly valid distance matrix is found. As a postprocessing step, the coordinates are improved by applying local optimization methods.

The DISGEO package [12] was developed by Havel and Wüthrich in 1984, to solve larger conformation problems than what EMBED can handle. DISGEO works around the problem size limitation by using two passes of EMBED. In the first pass, coordinates are computed for a subset of atoms, subject to constraints inherited from the whole structure. This step forms a "skeleton" for the structure. The second pass of EMBED computes coordinates for the remaining atoms, building upon the skeleton computed in the first pass. They tested DISGEO on the BPTI protein [1], which has 454 atoms, using realistic input data consisting of distance (3290) and chirality (450) constraints needed to fix the covalent structure, and bounds (508) for distances between hydrogen atoms in different amino acid residues that are less than 4Å apart to simulate the distance constraints available from a nuclear Overhauser effect spectroscopy (NOESY) experiment. Havel's DG-II package [10], published in 1991, improves upon DISGEO by producing from the same input as DISGEO five structures having an average RMSD of 1.76Å from the crystal structure.

The alternating projections algorithm (APA) for molecular conformation was developed in 1990 [6, 18]. As in EMBED, APA begins by using the triangle inequality

to compute distance bounds for all pairs of points. The lower and upper bounds form a rectangular parallelepiped, which the authors refer to as a *data box*. Next, a random *dissimilarity matrix* $\Delta$ in the data box is chosen. (The dissimilarity matrix serves the same function as the estimated distance matrix in EMBED.) The dissimilarity matrix is adjusted so that it adheres to the triangle inequality. Next, $\Delta$ is projected onto the cone of matrices that are negative semidefinite on the orthogonal complement of $e = (1, 1, \ldots, 1)^T$, then back onto the data box. The alternating projections are repeated five times. The postprocessing step involves performing stress minimization on the resultant structure. In [18], APA was applied to the BPTI protein to compare the algorithm's performance to that of DISGEO and DG-II. Under the exact same inputs as DISGEO and DG-II, the five best structures out of thirty produced by APA had an average RMSD of 2.39Å compared with the crystal structure.

Classical multidimensional scaling (MDS) is a collection of techniques for constructing configurations of points from pairwise distances. Trosset has applied MDS to the molecular conformation problem [22, 23, 24] since 1998. Again, the first step is to use the triangle inequality to compute distance bounds for all pairs of points. Trosset's approach is to solve the problem of finding the squared dissimilarity matrix that minimizes the distances to the cone of symmetric positive semidefinite matrices of rank less than $d$, while satisfying the squared lower and upper bounds. In [24], MDS is applied to five molecules with less than 700 atoms. For points with pairwise distances $d_{ij}$ less than 7Å, lower and upper bounds of the form $(d_{ij} - 0.01\text{Å}, d_{ij} + 0.01\text{Å})$ are given; for pairwise distances greater than 7Å, a lower bound of 7Å is specified. The method was able to produce estimated configurations that had a maximum bound violation of less than 0.1Å. The author did not report the RMSD of the computed configurations.

More recently, in 2006, Grooms, Lewis, and Trosset proposed a dissimilarity parameterized approach called *StrainMin* [8] instead of a coordinate-based approach. Although the latter has fewer independent variables, the former seems to converge to "better" local minimizers. They propose to minimize an objective function which is the sum of the fit of the dissimilarity matrix $\Delta$ to the data and the distance of $\Delta$ to the space of rank-$d$ positive semidefinite matrices. The approach was tested on input data that consists of exact distances between atoms less than 6Å apart, and a 2.5Å lower bound as a representative van der Waal radius for atoms whose distance is unknown. They were able to satisfy the distance bounds with a maximum violation of 0.2Å, for an ensemble of six protein molecules. However, the RMSD errors were not reported.

The DAFGL algorithm of Biswas, Toh, and Ye in 2008 [3] is a direct ancestor of this work. DAFGL differs from the previous methods in that it applies SDP relaxation methods to obtain the inner product matrix. Due to limitations in SDP algorithms, software, and hardware, the largest SDP problems that can be solved are on the order of a few hundred atoms. In order to solve larger problems, DAFGL employs a distributed approach. It applies the symmetric reverse Cuthill–Mckee matrix permutation to divide the atoms into smaller groups with overlapping atoms (see Figure 4.2). Each group is solved using SDP, and the overlapping groups are used to align the local solutions to form a global solution. Tests were performed on 14 molecules with numbers of atoms ranging from 400–5600. The input data consists of 70% of the distances $d_{ij}$ below 6Å, given as lying in intervals $[\underline{d}_{ij}, \overline{d}_{ij}]$, where $\underline{d}_{ij} = \max(0, (1-0.05|\underline{Z}_{ij}|)d_{ij})$, $\overline{d}_{ij} = (1+0.05|\overline{Z}_{ij}|)d_{ij}$, and $\underline{Z}_{ij}, \overline{Z}_{ij}$ are standard normal random variables with zero mean and unit variance. Given such input, DAFGL is able to produce a conformation for most molecules with an RMSD of 2–3Å.

**2.2. Buildup methods.** The ABBIE program [13] was developed by Hendrickson in 1995 to solve molecular conformation problems given exact distance data. ABBIE aims to divide the problem into smaller pieces by identifying *uniquely realizable subgraphs*—subgraphs that permit a unique realization. Hendrickson tested ABBIE on the protein molecule 7RSA [1], which has 1849 atoms after discarding end chains. The input data included the exact distances between all pairs of atoms in the same amino acid (13879), and 1167 additional distances between hydrogen atoms less than 3.5 Å apart. Although it was not explicitly mentioned in the paper, we presume that the algorithm was able to compute an exact solution up to roundoff error.

Dong and Wu [5] (see also [27]) presented their geometric buildup algorithm in 2003, which also relies on having exact distances. The essential idea of this algorithm is that if four atoms form a four-clique—four atoms with distances between all pairs known—then their positions are fixed relative to one another. The algorithm starts by finding a four-clique and fixing the coordinates of the four atoms. The other atom positions are determined atom-by-atom; when the distance of an atom to four atoms with determined coordinates is known, that atom's position can be uniquely determined. When given all the distances less than 8Å, the algorithm was able to accurately estimate all ten molecules tested; when given all the distances less than 5Å, it was able to accurately estimate nine of the ten molecules.

**2.3. Global optimization methods.** For an introduction to optimization-based methods for molecular conformation, see [15]. Here we briefly describe two such methods.

The DGSOL code [16, 17] by Moré and Wu in 1999 treats the molecular conformation problem as a large nonlinear least squares problem with Gaussian smoothing applied to the objective function to increase the likelihood of finding the global minima. They applied DGSOL to two protein fragments consisting of 100 and 200 atoms, respectively. Distances were specified for atoms in the same or neighboring residues, and given as lower bounds $\underline{d}_{ij} = 0.84 d_{ij}$ and upper bounds $\overline{d}_{ij} = 1.16 d_{ij}$, where $d_{ij}$ denotes the true distance between atoms $i$ and $j$. DGSOL was able to compute structures with an average RMSD of 1.0Å and 2.9Å for 100 and 200 atoms, respectively.

The GNOMAD algorithm [26] by Williams, Dugan, and Altman in 2001 attempts to satisfy the input distance constraints as well as MSD constraints. Their algorithm applies to the situation when we are given sparse but exact distances. Since it is difficult to optimize all the atom positions simultaneously, GNOMAD updates the positions one at a time. The authors tested GNOMAD on the protein molecule 1TIM [1], which has 1870 atoms. Given all the covalent distances and distances between atoms that share covalent bonds to the same atom, as well as 30% of short-range distances less than 6Å, they were able to compute estimated positions with an RMSD of 2–3Å.[1]

We end this section by noting that while the GNOMAD algorithm would increasing get stuck in an unsatisfactory local minimum with more stringent MSD constraints, the addition of such lower bound constraints is highly beneficial for the DISCO algorithm proposed in this paper.

**3. Mathematics of molecular conformation.** We begin this section with the SDP models for molecular conformation, in section 3.1. Then we introduce the

[1] The RMSD of 1.07Å reported in Figure 11 in [26] is inconsistent with that appearing in Figure 8. It seems that the correct RMSD should be about 2–3 Å.

gradient descent method for improving sensor positions, in section 3.2. Finally, we present the alignment problem in section 3.3.

**3.1. SDP models for molecular conformation.** The setting of the molecular conformation problem is as follows. We wish to determine the coordinates of $n$ atoms, $\boldsymbol{s}_i \in \mathbb{R}^d,\ i = 1, \ldots, n$, given measured distances or distance bounds for some of the pairwise distances $\|\boldsymbol{s}_i - \boldsymbol{s}_j\|$ for $(i,j) \in \mathcal{N}$. Note that in the case of the sensor network localization problem, we are also given a set of $n_a$ anchor nodes with known coordinates $\boldsymbol{a}_i \in \mathbb{R}^d$, $i = 1, \ldots, n_a$, and some pairwise distances $\|\boldsymbol{a}_i - \boldsymbol{s}_j\|$ for $(i,j) \in \mathcal{N}^a$. For brevity, we will only briefly describe the anchorless case in this paper, but refer the reader to [2] for details on the sensor network problem.

In the "measured distances" model, we have measured distances for certain pairs of nodes,

$$\tilde{d}_{ij} \approx \|\boldsymbol{s}_i - \boldsymbol{s}_j\|, \qquad (i,j) \in \mathcal{N}. \tag{3.1}$$

In this model, the unknown positions $\{\boldsymbol{s}_i\}_{i=1}^n$ are the best fit to the measured distances, obtained by solving the following nonconvex minimization problem:

$$\min \left\{ \sum_{(i,j)\in\mathcal{N}} |\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 - \tilde{d}_{ij}^2| \ :\ \boldsymbol{s}_i,\ i = 1, \ldots, n \right\}. \tag{3.2}$$

In the "distance bounds" model, we have lower and upper bounds on the distances between certain pairs of nodes,

$$\underline{d}_{ij} \le \|\boldsymbol{s}_i - \boldsymbol{s}_j\| \le \overline{d}_{ij}, \qquad (i,j) \in \mathcal{N}. \tag{3.3}$$

In this model, the unknown positions $\{\boldsymbol{s}_i\}_{i=1}^n$ are the best fit to the measured distance bounds, obtained by solving the following nonconvex minimization problem:

$$\min \left\{ \sum_{(i,j)\in\mathcal{N}} \left(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 - \underline{d}_{ij}^2\right)_- + \left(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 - \overline{d}_{ij}^2\right)_+ :\ \boldsymbol{s}_i,\ i = 1, \ldots, n \right\}, \tag{3.4}$$

where $\alpha_+ = \max\{0, \alpha\}, \alpha_- = \max\{0, -\alpha\}$.

In order to proceed to the SDP relaxation of the problem, we need to consider the matrix

$$Y = X^T X, \quad \text{where} \quad X = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n]. \tag{3.5}$$

By denoting the $i$th unit vector in $\mathbb{R}^{n_s}$ by $\boldsymbol{e}_i$ and letting $\boldsymbol{e}_{ij} = \boldsymbol{e}_i - \boldsymbol{e}_j$, we have $\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 = \boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij}$. We can therefore conveniently express the constraints (3.1) as $\tilde{d}_{ij}^2 \approx \boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij},\ (i,j) \in \mathcal{N}$. Similar expressions hold also for (3.3). In the SDP relaxation, we relax the constraint $Y = X^T X$ in (3.5) to $Y \succcurlyeq X^T X$, and the relaxed model for (3.2) is given by

$$\min \left\{ \sum_{(i,j)\in\mathcal{N}} \left| \boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij} - \tilde{d}_{ij}^2 \right| \ :\ Y \succeq 0 \right\}. \tag{3.6}$$

The problem (3.4) can be relaxed in terms of $Y$ similarly. Once we have obtained a matrix $Y$ by solving (3.6), we recover the estimated sensor positions $X = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n_s}]$

from $Y$ by taking $X = D_1^{1/2} V_1^T$, where the eigenpair $(V_1, D_1)$ corresponds to the best rank-$d$ approximation of $Y$.

It is clear that the atom positions derived from (3.2) or (3.4) would have translational, rotational, and reflective freedom. This can cause numerical difficulties when solving the SDP relaxed problem. The difficulties can be ameliorated when we remove the translational freedom by introducing a constraint that corresponds to setting the center of mass at the origin, i.e., $\langle Y, E\rangle = 0$, where $E$ is the matrix of all ones.

For the sensor network localization problem, So and Ye [20] have shown that if the distance data is uniquely localizable, then the SDP relaxation (3.6) is able to produce the exact sensor coordinates up to rounding errors. We refer the reader to [20] for the definition of "unique localizability." Intuitively, it means that there is only one configuration in $\mathbb{R}^d$ (perhaps up to translation, rotation, reflection) that satisfies all the distance constraints. The result of So and Ye gives us a degree of confidence that the SDP relaxation technique is a strong relaxation.

We now discuss what happens when the distance data is sparse and/or noisy, so that there is no unique realization. In such a situation, it is not possible to compute the exact coordinates. Furthermore, the matrix $X$ extracted from the solution $Y$ of the SDP (3.6) will not satisfy $Y = X^T X$, and $Y$ will be of dimension greater than $d$. We present an intuitive explanation for this phenomenon. Suppose that we have points in the plane, and certain pairs of points are constrained so that the distances between them are fixed. If the distances are perturbed slightly, then some of the points may be forced out of the plane in order to satisfy the distance constraints. Therefore, under noise, $Y$ will tend to have a rank higher than $d$. Another reason for $Y$ to have a higher rank is that if there are multiple solutions, the interior-point methods used by many SDP solvers converge to a solution with maximal rank [9].

This situation presents us with potential problems. If $Y$ has a higher rank than $d$, then the solution $X$ extracted from $Y$ is unlikely to be accurate. To ameliorate this situation, we add the regularization term, $-\gamma\langle I, Y\rangle$, to the objective function, where $\gamma$ is a positive regularization parameter. This term spreads the atoms further apart and induces them to exist in a lower-dimensional space. We refer interested readers to [2] for details on the derivation of the regularization term. Thus the measured distances model (3.6) and distance bounds model become

$$\min \left\{ \sum_{(i,j)\in\mathcal{N}} \left| \boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij} - \tilde{d}_{ij}^2 \right| - \gamma\langle I, Y\rangle \,:\, \langle E, Y\rangle = 0,\ Y \succeq 0 \right\}, \tag{3.7}$$

$$\min \Big\{ -\langle I, Y\rangle \,:\, \underline{d}_{ij}^2 \le \boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij} \le \overline{d}_{ij}^2 \ (i,j) \in \mathcal{N},\ \langle E, Y\rangle = 0,\ Y \succeq 0 \Big\}. \tag{3.8}$$

**3.2. Coordinate refinement via gradient descent.** If we are given measured pairwise distances $\tilde{d}_{ij}$, then the atom positions can be computed as the minimizer of

$$\min f(X) := \sum_{(i,j)\in\mathcal{N}} \big( \|\boldsymbol{s}_i - \boldsymbol{s}_j\| - \tilde{d}_{ij} \big)^2. \tag{3.9}$$

Note that the above objective function is different from that of (3.2). Similarly, if we are given bounds for pairwise distances $\underline{d}_{ij}$ and $\overline{d}_{ij}$, then the configuration can be computed as the solution of

$$\min f(X) := \sum_{(i,j)\in\mathcal{N}} \big( \|\boldsymbol{s}_i - \boldsymbol{s}_j\| - \underline{d}_{ij} \big)_-^2 + \big( \|\boldsymbol{s}_i - \boldsymbol{s}_j\| - \overline{d}_{ij} \big)_+^2. \tag{3.10}$$

Again, note that the objective function is different from that of (3.4). We can solve (3.9) or (3.10) by applying local optimization methods. For simplicity, we choose to use a gradient descent method with backtracking line search. The implementation of this method is straightforward.

The problems (3.9) and (3.10) are highly nonconvex problems with many local minimizers. If the initial iterate $X_0$ is not close to a good solution, then it is extremely unlikely that the local optimum obtained from a local optimization method will be a good solution. In our case, however, when we set $X_0$ to be the conformation produced by solving the SDP relaxation, local optimization methods are often able to produce an $X$ with higher accuracy than the original $X_0$.

**3.3. Alignment of configurations.** The molecular conformation problem is anchor-free, and thus the configuration has translational, rotational, and reflective freedom. Nevertheless, we need to be able to compare two configurations, to determine how similar they are. In particular, we need to compare a computed configuration to the true configuration. In order to perform a comparison of two configurations, it is necessary to align them in a common coordinate system. We can define the "best" alignment as the affine transformation $T$ that minimizes [7]

$$\min \left\{ \sum_{i=1}^{n} \|T(\boldsymbol{a}_i) - \boldsymbol{b}_i\| : T(\boldsymbol{x}) = Q\boldsymbol{x} + \boldsymbol{c}, Q \in \mathbb{R}^{d\times d},\ Q \text{ is orthogonal} \right\}. \tag{3.11}$$

The constraint on the form of $T$ restricts it to be a combination of translation, rotation, and reflection. In the special case when $A$ and $B$ are centered at the origin, (3.11) reduces to an orthogonal procrustes problem $\min\{\|QA - B\|_F : Q \in \mathbb{R}^{d\times d}, Q \text{ is orthogonal}\}$. It is well known that the optimal $Q$ can be computed from the singular value decomposition of $AB^T$.

**4. The DISCO algorithm.** Here we present the DISCO algorithm (for DIStributed COnformation). In section 4.1, we explain the essential ideas that are incorporated into the design of DISCO. We present the procedures for the recursive and basis cases in sections 4.2 and 4.3, respectively.

**4.1. The basic ideas of DISCO.** Prior to this work, it was known that the combination of the SDP relaxation technique and gradient descent is able to accurately localize moderately sized problems (problems with up to 500 atoms). However, many protein molecules have more than 10000 atoms. In this work, we develop techniques to solve large-scale problems.

A natural idea is to employ a divide-and-conquer approach, which will follow the general framework: **If** the number of atoms is not too large, then solve the atom positions via SDP, and utilize gradient descent refinement to compute improved coordinates; **Otherwise** break the atoms into two subgroups, solve each subgroup recursively, and align and combine them together, again postprocessing the coordinates by applying gradient descent refinement. This naturally gives rise to two questions.

*Question* 1: *How should we divide an atom group into two subgroups?* We would wish to minimize the number of edges between the two subgroups. This is because when we attempt to localize the first subgroup of atoms, the edges with atoms in the second subgroup are lost. Simultaneously, we wish to maximize the number of edges within a subgroup. The more edges within a subgroup, the more constraints on the atoms, and the more likely that the subgroup is localizable.

*Question* 2: *How should we join together the two localized subgroups to localize the atom group?* Our strategy is for the two subgroups to have overlapping atoms. If

the overlapping atoms are accurately localized in the estimated configurations, then they can be used to align the two subgroup configurations. If the overlapping atoms are *not* accurately localized, it would be disastrous to use them in aligning the two subgroup configurations. Therefore, DISCO incorporates a heuristic criterion (see section 4.2.2) for determining when the overlapping atoms are accurately localized.

It is important to bear in mind that not all the atoms in a group may be localizable, for instance, atoms with fewer than four neighbors in that group. This must be taken into account when we are aligning two subgroup configurations together. If a significant number of the overlapping atoms are not localizable in either of the subgroups, the alignment may be highly erroneous (see Figure 4.4). This problem can be avoided if we can identify and discard the unlocalizable atoms in a group. A heuristic algorithm (see section 4.3.2) is used by DISCO to identify atoms which are likely to be unlocalizable.

The pseudocode of the DISCO algorithm is presented in Algorithm 1. We illustrate how the DISCO algorithm solves a small molecule in Figure 4.1.

ALGORITHM 1. The DISCO algorithm.

```
procedure DISCO(L, U)
    if number of atoms < basis size then
        [cAest, cI] ← DISCOBASIS(L, U)
    else
        [cAest, cI] ← DISCORECURSIVE(L, U)
    end if
    return cAest, cI
end procedure
procedure DISCOBASIS(L, U)
    cI ← LIKELYLOCALIZABLECOMPONENTS(L, U)
    for i = 1, ..., LENGTH(cI) do
        cAest{i} ← SDPLOCALIZE(cI{i}, L, U)
        cAest{i} ← REFINE(cAest{i}, cI{i}, L, U)
    end for
    return cAest, cI
end procedure
procedure DISCORECURSIVE(L, U)
    [L1, U1, L2, U2] ← PARTITION(L, U)
    [cAest1, cI1] ← DISCO(L1, U1)
    [cAest2, cI2] ← DISCO(L2, U2)
    cAest ← [cAest1, cAest2]
    cI ← [cI1, cI2]
    repeat
        [cAest, cI] ← COMBINECHUNKS(cAest, cI)
        [cAest, cI] ← REFINE(cAest, cI, L, U)
    until no change
    return cAest, cI
end procedure
```

### 4.2. Recursive case: How to split and combine.

**4.2.1. Partitioning into subgroups.** Before we discuss DISCO's partitioning procedure, we briefly describe the procedure used by DISCO's parent, the DAFGL algorithm [3]. The DAFGL algorithm partitions the set of atoms into consecutive subgroups such that consecutive subgroups have overlapping atoms (see Figure 4.2). It then solves each subgroup separately and combines the solutions together. Partitioning in DAFGL is done by repeatedly applying the symmetric reverse Cuthill–McKee (RCM) matrix permutation to submatrices of the distance matrix. The RCM permutation is specially designed to cluster the nonzero entries of a matrix toward the

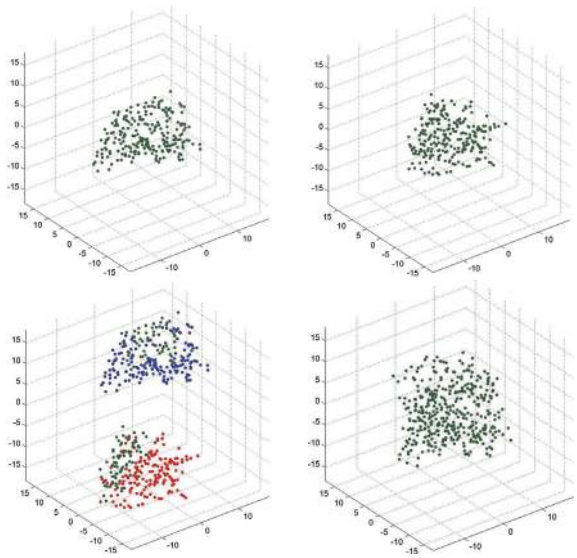

FIG. 4.1. *Since the number of atoms is too large ($n = 402 >$ basis size $= 300$), we divide the atoms into two subgroups; see Figure* 4.3. *We solve the subgroups independently (top left and right). The subgroups have overlapping atoms, which are colored in green (bottom left). The overlapping atoms allow us to align the two subgroups to form a realization of the molecule.*

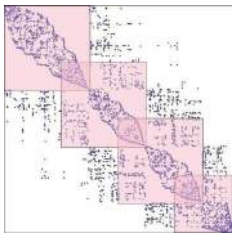

FIG. 4.2. *This permutation of the distance matrix illustrates DAFGL's partitioning strategy. The dots represent the known distances, and the shaded squares represent the subgroups.*

diagonal. We observe in Figure 4.2 that many of the edges are not available to any subgroup, as they lie outside all the shaded squares. We believe that DAFGL's partitioning procedure discards too many edges, and this is the reason why DAFGL performs poorly when the given distances are sparse, say less than 50% of pairwise distances being less than 6Å. The lesson learned from DAFGL's shortcoming is that DISCO's partitioning method must try to keep as many edges in its subgroups as possible.

Suppose that we wish to localize the set of atoms $\mathcal{A}$, but there are too many atoms in $\mathcal{A}$ for us to apply an SDP relaxation directly. We therefore divide $\mathcal{A}$ into two nonoverlapping subgroups $\mathcal{A}_1$ and $\mathcal{A}_2$. The two objectives in this division are that the number of edges between subgroups is approximately minimized, to maximize the chance that each subgroup will be localizable, and that the subgroups are approximately equal in size, so that the recursive procedure will be fast.

It is not immediately obvious, after localizing $\mathcal{A}_1$ and $\mathcal{A}_2$, how we should combine them to form a configuration for $\mathcal{A}$. Our method for merging two subgroups together is to make use of overlapping atoms between the subgroups. If the overlapping atoms are

localized in both groups, then the two configurations can be aligned via a combination of translation, rotation, and refection. Of course, $\mathcal{A}_1$ and $\mathcal{A}_2$ were constructed to have no overlapping atoms. Thus we need to enlarge them to subgroups $\mathcal{B}_1 \supset \mathcal{A}_1, \mathcal{B}_2 \supset \mathcal{A}_2$, which have overlapping atoms. We construct $\mathcal{B}_1$ by adding some atoms $\tilde{\mathcal{A}}_2 \subset \mathcal{A}_2$ to $\mathcal{A}_1$. Similarly, we add atoms $\tilde{\mathcal{A}}_1 \subset \mathcal{A}_1$ to $\mathcal{A}_2$ to create $\mathcal{B}_2$. The set of atoms $\tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2$ comprises auxiliary atoms added to $\mathcal{A}_1$ and $\mathcal{A}_2$ to create overlap. While $\mathcal{A}_1$ and $\mathcal{A}_2$ were constructed so as to *minimize* the number of edges $(i,j) \in \mathcal{N}$ with $i \in \mathcal{A}_1, j \in \mathcal{A}_2$, $\tilde{\mathcal{A}}_1$ and $\tilde{\mathcal{A}}_2$ are constructed so as to *maximize* the number of edges $(i,j) \in \mathcal{N}$ with $i \in \mathcal{A}_1, j \in \tilde{\mathcal{A}}_2$, and $(i,j) \in \mathcal{N}$ with $i \in \tilde{\mathcal{A}}_1, j \in \mathcal{A}_2$. The reason for this is that we want the set of atoms $\mathcal{B}_1 = \mathcal{A}_1 \cup \tilde{\mathcal{A}}_2$ and $\mathcal{B}_2 = \mathcal{A}_2 \cup \tilde{\mathcal{A}}_1$ to be localizable, so we want as many edges within $\mathcal{B}_1$ and $\mathcal{B}_2$ as possible.

We can succinctly describe the partitioning as splitting $A$ into two localizable groups $\mathcal{A}_1$ and $\mathcal{A}_2$, then growing $\mathcal{A}_1$ into $\mathcal{B}_1$ and $\mathcal{A}_2$ into $\mathcal{B}_2$ so that $\mathcal{B}_1$ and $\mathcal{B}_2$ are both likely to be localizable. The splitting step should minimize intergroup edges, to maximize the likelihood that $\mathcal{A}_1$ and $\mathcal{A}_2$ are localizable, while the growing step should maximize intergroup edges, to maximize the likelihood that $\tilde{\mathcal{A}}_2$ and $\tilde{\mathcal{A}}_1$ are localizable in $\mathcal{B}_1$ and $\mathcal{B}_2$.

To make our description more concrete, we give the pseudocode of the partition algorithm as Algorithm 2. The operation of the algorithm is also illustrated in Figure 4.3. We elaborate on the details of the pseudocode below. The PARTITION method consists of three stages: SPLIT, REFINE, and OVERLAP.

ALGORITHM 2. The partition algorithm.

```
procedure PARTITION(D)
    [A_1, A_2] ← SPLIT(D)
    [A_1, A_2] ← REFINE(D, A_1, A_2)
    [B_1, B_2] ← OVERLAP(D, A_1, A_2)
    return B_1, B_2
end procedure
procedure SPLIT(D)
    P ← SYMRCM(D)
    A_1 ← p(1 : ⌊n/2⌋), A_2 ← p(⌊n/2⌋ + 1 : n)
    return A_1, A_2
end procedure
procedure REFINE(D, A_1, A_2)
    while exists a ∈ A_1 closer to A_2 do
        A_1 ← A_1 \ {a};   A_2 ← A_2 ∪ {a}
    end while
    while exists a ∈ A_2 closer to A_1 do
        A_2 ← A_2 \ {a};   A_1 ← A_1 ∪ {a}
    end while
end procedure
procedure OVERLAP(D, A_1, A_2)
    for i = 1, 2 do
        repeat
            a ← the closest point to A_i that is not in A_i
            A_i ← A_i ∪ {a}
        until A_i is of desired size
    end for
end procedure
```

In the SPLIT method, we compute the RCM permutation $\boldsymbol{p}$ of the rows and columns of the distance matrix $D$, that approximately minimizes the bandwidth of the matrix $D(\boldsymbol{p}, \boldsymbol{p})$. This is conveniently implemented as the `symrcm` command in MATLAB. The RCM permutation has the effect of clustering the nonzero entries toward the main diagonal, so that if we split the matrix with a vertical cut and horizontal cut through the center of the matrix, then the majority of the edges are in

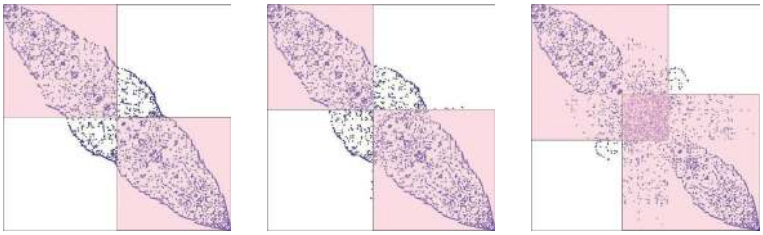

FIG. 4.3. *(left) An RCM permutation gives us a balanced cut, with few cross-edges (number of cross-edges = 317). (middle) Refining the split reduces the number of cross-edges (number of cross-edges = 266). (right) Expanding the nonoverlapping subgroups $\mathcal{A}_1, \mathcal{A}_2$ into the overlapping subgroups $\mathcal{B}_1, \mathcal{B}_2$.*

the (1,1) and (2,2) blocks, and only a few of the edges are in the (1,2) and (2,1) blocks. Thus if we select $\mathcal{A}_1 = \boldsymbol{p}(1 : \lfloor n/2 \rfloor)$ and $\mathcal{A}_2 = \boldsymbol{p}(\lfloor n/2 \rfloor + 1 : n)$, this approximately minimizes the number of edges from $\mathcal{A}_1$ to $\mathcal{A}_2$ (see Figure 4.3, left).

In the REFINE method, we can reduce the number of intergroup edges as follows: If an atom $a \in \mathcal{A}_1$ is "closer" to $\mathcal{A}_2$ than $\mathcal{A}_1$, then switch it over to $\mathcal{A}_2$. An atom is "closer" to group $\mathcal{A}_1$ rather than group $\mathcal{A}_2$ if one of two conditions holds: (1) it has more neighbors in group $\mathcal{A}_1$; (2) it has the same number of neighbors in groups $\mathcal{A}_1$ and $\mathcal{A}_2$, and its closest neighbor is in $\mathcal{A}_1$.

In the OVERLAP method, we compute $\mathcal{B}_1$ and $\mathcal{B}_2$. We begin by setting $\mathcal{B}_i$ to $\mathcal{A}_i$, then add to $\mathcal{B}_i$ the atom $a$ not in $\mathcal{B}_i$ that is closest to $\mathcal{B}_i$, repeating until $\mathcal{B}_i$ is has the desired number of atoms.

**4.2.2. Alignment of atom groups.** Here we describe how to combine the computed configurations for $\mathcal{B}_1$ and $\mathcal{B}_2$ to form a configuration for $\mathcal{A}$. We shall adopt the following notation to facilitate our discussion. Let $B_1, B_2$ be the coordinates for the atoms in $\mathcal{B}_1, \mathcal{B}_2$, respectively, and let $C_1, C_2$ be the coordinates for the overlapping atoms in $\mathcal{B}_1, \mathcal{B}_2$, respectively. If $a$ is an atom in $\mathcal{A}$, then let $\boldsymbol{a}$ denote its coordinates in the configuration for $\mathcal{A}$. If $a \in \mathcal{B}_1$ (resp., $a \in \mathcal{B}_2$), then let $\boldsymbol{b}_1$ (resp., $\boldsymbol{b}_2$) denote its coordinates in the configuration $B_1$ (resp., $B_2$).

The first method we used to produce a configuration for $\mathcal{A}$ was to consider the composition of translation, rotation, and reflection $T$ that best aligns $C_2$ to $C_1$. If $a$ is in $\mathcal{B}_1$ but not in $\mathcal{B}_2$, then we set $\boldsymbol{a} = \boldsymbol{b}_1$; if $a$ is in $\mathcal{B}_2$ but not in $\mathcal{B}_1$, then we set $\boldsymbol{a} = T(\boldsymbol{b}_2)$; if $a$ is in both $\mathcal{B}_1$ and $\mathcal{B}_2$, then we set $\boldsymbol{a} = (\boldsymbol{b}_1 + T(\boldsymbol{b}_2))/2$, the average of $\boldsymbol{b}_1$ and $T(\boldsymbol{b}_2)$. While this method is simple, it suffers from the disadvantage that a few outliers can have a high degree of influence on the alignment. If a significant number of the overlapping atoms are poorly localized, then the alignment may be destroyed.

The method used by DISCO is slightly more sophisticated, so as to be more robust. Our strategy is to use an iterative alignment process. We find the overlapping point such that the distance between its positions in the two configurations $\|\boldsymbol{b}_1 - T(\boldsymbol{b}_2)\|$ is the greatest. If it is greater than two times the mean distance over allover-lapping points, then it is likely that this point is not accurately localized in either of the two configurations, so we remove this outlier point and repeat the process; otherwise, we conclude that this point is not an outlier, and $T$ may give us a good alignment. By discarding points whose coordinates do not agree well, it is hoped that our alignment uses only points that are well localized in both groups. The linear transformation $T$ obtained from discarding outlier points goes through a second test. If the alignment of the remaining overlapping points has high error, that is, if the RMSD is greater

than a certain threshold, this indicates that it is not possible to accurately align $\mathcal{B}_1$ and $\mathcal{B}_2$ together, and we do not align them; otherwise, we proceed to align $\mathcal{B}_1$ and $\mathcal{B}_2$.

**4.3. Basis case: Localizing an atom group.** In this subsection, we explain that certain individual atoms or even larger clusters of atoms may be difficult or impossible to localize. If these "bad points" are not removed, oftentimes they will introduce errors into the computed structure. Therefore, DISCO temporarily removes these "bad points" and focuses on localizing only the remaining atoms which form the "core" of the molecule. Below, we will present the procedure used by DISCO to compute the likely localizable core of the molecule. Note that, after removing all the "bad points" from the main computation in DISCO to localize the core structure, we patch the "bad points" back to the core structure by applying a final round of the gradient descent method to the minimization problems in section 3.2.

**4.3.1. When DISCO fails.** A prototype of DISCO was able to accurately localize certain molecules, but would produce high-error structures for other molecules. Usually, the root of the trouble was that the configuration for one particular subgroup had high error. Unfortunately, aligning a good configuration and a bad configuration often produces a bad configuration, so that the error propagates up to the complete protein configuration (see Figure 4.4).

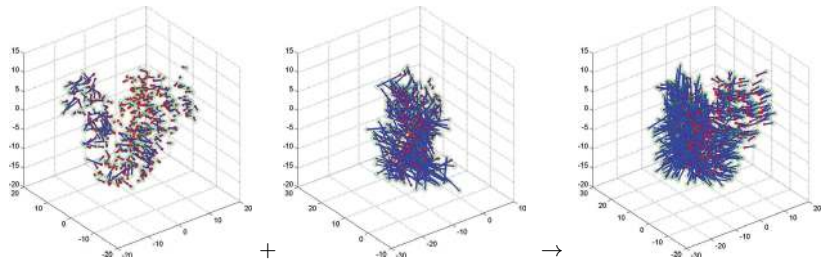

FIG. 4.4. *In each plot, a circle represents a true position, the red dot represents an estimated position, and the blue line joins the corresponding true and estimated positions. This figure shows how two subgroup configurations are aligned to produce a configuration for a larger group. Here, because one subgroup configuration is poorly localized, the resulting configurations formed from this poorly localized configuration are also unsatisfactory.*

There are several reasons for some atom groups to be badly localized. One obvious reason is that some of the atoms may have only three or fewer neighbors and so are not uniquely localizable in general. The second reason is more subtle. When we plotted the estimated positions against the true positions, we noticed that the badly localized groups often consisted of two subgroups; each subgroup was well localized relative to itself, but there were not many edges between the two subgroups, implying that the subgroups were not well positioned relative to each other.

**4.3.2. Identifying a likely localizable core.** As we may observe from Figure 4.4, if one subgroup is poorly localized, the complete protein configuration could be destroyed. Thus we must make an extra effort to ensure that we are able to accurately localize each subgroup.

Here we make a slight digression to a related result. In the case when exact distances are given, Hendrickson [13] established sufficient conditions for unique localizability in $\mathbb{R}^3$. These conditions are not of great import to us, and so we give only a flavor of the conditions: (1) vertex 4-connectivity, (2) redundant rigidity—the

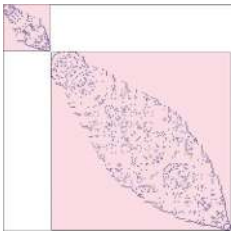

FIG. 4.5. *Finding the cut that minimizes the number of edges between subgroups.*

graph is rigid after the removal of any edge, (3) stress testing—the null space of the so-called "stress matrix" has dimension 4.

Unfortunately, Hendrickson's results, while interesting, are not immediately applicable to our situation. Imagine a conformation that is kept rigid by a set of edges, which are constrained to be of specified distances. If the specified distances are relaxed to distance bounds, it is possible that the conformation will have freedom to flex or bend into a shape that is drastically different from the original. Thus to get a good localization with noisy distances requires stricter conditions than to get a good localization with exact distances.

To ensure that we can get a good localization of a group, we may have to discard some of the atoms, or split the group into several subgroups (see section 4.3.1). Atoms with fewer than four neighbors should be removed, because we have no hope of localizing them accurately. We should also check whether it is possible to split the atoms into two subgroups, both larger than the `MinSplitSize`,[2] which have fewer than `MinCrossEdges` edges between them. If this were the case, it may not be possible to localize both subgroups together accurately, but if we split the subgroups, it may be possible to localize each of them accurately. The optimal choice of these parameters is not known, but we have found that the value of 20 for `MinSplitSize` and 50 for `MinCrossEdges` seems to work well in practice. With regard to our choice for `MinCrossEdges`, in the case of exact distances, in general six edges are needed to align two rigid subgroups. However, in our case the distance data may be very noisy, so we may need many more edges to align the two groups well. This is why the somewhat conservative value of 50 is chosen.

How should we split the atoms into two subgroups, both subgroups with at least `MinSplitSize` atoms, so that there are as few edges between them as possible? This problem is familiar to us, because it bears resemblance to the partitioning problem that was discussed in section 4.2.1. This suggests that we could adapt the partitioning algorithm to this splitting problem. The difference between the two problems is that in the partitioning problem, we would like the two subgroups to have approximately the same size, while in this situation we would like both subgroups to have at least `MinSplitSize` atoms. DISCO finds the approximate minimum split by first applying the RCM permutation to permute the rows and columns of the distance matrix and cluster the nonzero entries towards the diagonal. It then tries values of $p$ from `MinSplitSize` to $n - $ `MinSplitSize` $+ 1$, to see which is the cut such that the number of edges between atoms $1 : p$ and $(p + 1) : n$ is minimized (see Figure 4.5).

[2] We are looking for two rigid subgroups, which have few edges between them. The rigid subgroups should not be a very small group of atoms.

To present these ideas more concretely, we show the pseudocode of DISCO's localizable components algorithm in Algorithm 2.

ALGORITHM 2. Computing the likely localizable core.

```
procedure LocalizableComponents(A)
    Remove atoms with fewer than 4 neighbors from A
    [nCrossEdges, A1, A2] ← MinSplit(A)
    if nCrossEdges < MinCrossEdges then
        cI1 ← LocalizableComponents(A1)
        cI2 ← LocalizableComponents(A2)
        return [cI1, cI2]
    else
        return A
    end if
end procedure
procedure MinSplit(A)
    p ← SymRcm(D)
    for i = MinSplitSize, ..., n − MinSplitSize − 1 do
        nCrossEdges{i} ← nCrossEdges(D, p(1 : i), p(i + 1 : n))
    end for
    i ← MinIndex(nCrossEdges)
    nCrossEdges ← nCrossEdges{i}
    A1 ← A(1 : i);   A2 ← A(i + 1 : n)
end procedure
```

**5. Computational experiments.** In section 5.1, we explain computational issues in the DISCO algorithm. In section 5.2, we present the experimental setup. In section 5.3, we discuss the numerical results.

### 5.1. Computational issues.

**5.1.1. SDP localization.** In section 3, we presented the "measured distances" and "distance bounds" SDP models for the graph-realization problem. We now have to decide which model is more suitable for DISCO. We decided to use the "measured distances" model for DISCO, because the running time is superior, while the accuracy is comparable to that of the "distance bounds" model. With regards to the running time, DISCO uses the software SDPT3 [24] to solve the SDPs. The running time of SDPT3 is on the order of $O(mn^3) + O(m^2n^2) + \Theta(m^3) + \Theta(n^3)$, where $m$ is the number of constraints and $n$ is the dimension of the SDP matrix. In our case, $m$ corresponds to the number of distances/bounds, and $n$ corresponds to the number of atoms. As the "distance bounds" model has (roughly) twice as many constraints as the "measured distances" model, in practice, it may be 3–8 times slower on the same input.

In the "measured distances" model, the regularization parameter $\gamma$ has to be chosen judiciously. The parameter affects the configuration in the following way: the larger the parameter, the more separated the computed configuration. In the extreme case when the parameter is very large, the regularization term will dominate the distance error term to the extent that the objective value goes to minus infinity because the atoms move as far apart as possible rather than fitting the distance constraints. We have found that the value $\gamma = \bar{\gamma} := m/25n$ seems to work well in practice. We present our intuition for choosing such a value in the following. It is expected that if the value of the distance term $\sum_{(i,j)\in\mathcal{N}} |\boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij} - \tilde{d}_{ij}^2|$ and the value of the separation term $\gamma\langle I, Y\rangle$ in (3.7) are balanced, then the computed configuration will be neither too compact nor too separated. If we let $r$ denote the half-diameter of

the chunk, and we make the very crude estimates that

$$|\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 - \tilde{d}_{ij}^2| \approx \sum_{(i,j)\in\mathcal{N}} |\boldsymbol{e}_{ij}^T Y \boldsymbol{e}_{ij} - \tilde{d}_{ij}^2| \approx r^2/25, \qquad Y_{ii} \approx r^2,$$

then this gives rise to the choice of $\gamma = m/25n$. The factor $m/n$ comes from the fact that there are $m$ edges and $n$ diagonal terms. In our computational experiments, we have found that values $\gamma \in (\frac{1}{4}\bar{\gamma}, 4\bar{\gamma})$ seem to work well in practice, so the SDP model works well for a reasonably wide range of $\gamma$.

Let

$$\sigma(X) = \frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} \frac{\|\boldsymbol{s}_i - \boldsymbol{s}_j\|}{\|\boldsymbol{s}_i^{\text{true}} - \boldsymbol{s}_j^{\text{true}}\|}, \quad \tau(X) = \frac{1}{|\mathcal{N}|} \sum_{(i,j)\in\mathcal{N}} \frac{\|\boldsymbol{s}_i - \boldsymbol{s}_j\|}{\tilde{d}_{ij}^s}. \tag{5.1}$$

It would be useful to be able to quantify how "separated" an estimated configuration is, compared to the true configuration. Ideally, we define the *separation* of a computed configuration as $\sigma(X)$ in (5.1), which requires us to know the true configuration. Since we do not have the true configuration available, it is more appropriate to use the *approximate separation* of a computed configuration, defined as $\tau(X)$ in (5.1). The approximate separation of the computed configuration is a metric that helps us to decide whether the SDP should be resolved with a different regularization parameter. If the approximate separation indicates that the computed solution is "too compact" ($\tau(X) < 0.8$), then resolving the SDP with a larger $\gamma$ (doubling $\gamma$) may produce a "more separated" configuration that is closer to the true configuration. Similarly, if the computed solution is "too separated" ($\tau(X) > 1.1$), then resolving the SDP with a smaller $\gamma$ (halving $\gamma$) may produce a more accurate configuration.

The inclusion of MSD constraints can also help us to compute a better configuration from the SDP model. For physical reasons, there is a minimum separation distance between any two atoms $i$ and $j$, which we shall denote by $\alpha_{ij}$. After solving the SDP (3.7), we check to see if the minimum separation condition

$$\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 \approx Y_{ii} + Y_{jj} - 2Y_{ij} \geq \alpha_{ij}^2 \tag{5.2}$$

is satisfied for all pairs $(i, j)$. If not, then we let $\mathcal{E}$ be the set of pairs $(i, j)$ which violate the condition. We then resolve the SDP (3.7), with the additional constraints (5.2) for all $(i, j) \in \mathcal{E}$. We observed that imposing the minimum separation constraint improves the quality of the SDP solution. While it was reported in [26, p. 526] that the minimum separation constraints pose a significant global optimization challenge for molecular structure determination, we believe that the minimum separation constraints may in fact be advantageous for finding a lower rank SDP solution from (3.7).

In this paper, we set the minimum separation distance $\alpha_{ij}$ to 1Å uniformly, regardless of the types of the $i$th and $j$th atoms. In a more realistic setting, it is desirable to set $\alpha_{ij}$ as the sum of the van der Waals radii of atoms $i, j$ if they are not covalently bonded. In that setting, the MSD $\alpha_{ij}$ is usually larger than 1Å. For example, $\alpha_{ij} \approx 3.2$Å for the carbon and oxygen atoms if they are not covalently bonded.

**5.1.2. Gradient descent.** We have found that a regularized gradient descent refinement performs better than the nonregularized counterpart. Recall that the atom coordinates obtained via SDP localization are obtained by projecting $Y$ onto the space of rank-$d$ matrices. This tends to give rise to a configuration that is "too compact," because the discarded dimensions may make significant contributions to some of the

pairwise distances. Introducing a separation term in the objective function may induce the atoms to spread out appropriately.

Here we describe the regularized gradient descent. Let us denote the initial iterate by $X^0 = [\boldsymbol{s}_1^0, \ldots, \boldsymbol{s}_n^0]$, which we will assume is centered at the origin. The regularized objective function is

$$\sum_{(i,j)\in\mathcal{N}} \big(\|\boldsymbol{s}_i - \boldsymbol{s}_j\| - \tilde{d}_{ij}\big)^2 - \mu \sum_{i=1}^{n} \|\boldsymbol{s}_i\|^2 \quad \text{with } \mu = \frac{\sum_{(i,j)\in\mathcal{N}} \big(\|\boldsymbol{s}_i^0 - \boldsymbol{s}_j^0\| - \tilde{d}_{ij}\big)^2}{10 \sum_{i=1}^{n} \|\boldsymbol{s}_i^0\|^2}. \tag{5.3}$$

We have found that the above regularization parameter $\mu$ works well in practice.

We remark that choosing a suitable maximum number of iterations and tolerance level to terminate the gradient descent can significantly reduce the computational time of the gradient descent component of DISCO.

**5.2. Experimental setup.** The DISCO source code was written in MATLAB, and is freely available from the DISCO website [14]. DISCO uses the SDPT3 software package of Toh, Todd, and Tütüncü [21, 25] to solve SDPs arising from graph realization.

Our experimental platform was a dual-processor machine (2.40GHz Intel Core2 Duo processor) with 4GB RAM, running MATLAB version 7.3, which runs on only one core. We tested DISCO using input data obtained from a set of 12 molecules taken from the Protein Data Bank. The conformation of these molecules is already known, so that our computed conformation can be compared with the true conformation.

The sparsity of the interatom distances was modeled by choosing at random a proportion of the short-range interatom edges, subject to the condition that the distance graph be connected.[3] It is important to note that the structure of distances thus generated may not give rise to a localizable problem even given exact distances. Thus, it can be very challenging to produce a high-accuracy conformation under a high level of noise. We have chosen to define short-range interatom distances as those less than 6Å. The "magic number" of 6Å was selected because NMR techniques are able to give us distance information between some pairs of atoms if they are less than approximately 6Å apart. We have adopted this particular input data model because it is simple and fairly realistic [3, 26].

In realistic molecular conformation problems, the exact interatom distances are not given, but only lower and upper bounds on the interatom distances are known. Thus after selecting a certain proportion of short-range interatom distances, we add noise to the distances to give us lower and upper bounds. In this paper, we have experimented with "normal" and "uniform" noise. The noise level is specified by a parameter $\nu$, which indicates the expected value of the noise. When we say that we have a noise level of 20%, it means that $\nu = 0.2$. The bounds are specified by

$$\underline{d}_{ij} = \max\big(1, (1 - |\underline{Z}_{ij}|)d_{ij}\big), \quad \overline{d}_{ij} = (1 + |\overline{Z}_{ij}|)d_{ij}.$$

In the normal noise model, $\underline{Z}_{ij}, \overline{Z}_{ij}$ are independent normal random variables with zero mean and standard deviation $\nu\sqrt{\pi/2}$. In the uniform noise model, $\underline{Z}_{ij}, \overline{Z}_{ij}$ are independent uniform random variables in the interval $[0, 2\nu]$. We have defined the normal and uniform noise models in such a way that for both noise models the expected value of $|\underline{Z}_{ij}|, |\overline{Z}_{ij}|$ is $\nu$.

[3] The interested reader may refer to the code for the details of how the selection is done.

In addition to the lower and upper bounds, which are available for only some pairwise distances, we have MSDs between all pairs of atoms. For physical reasons, two atoms $i$ and $j$ must be separated by an MSD $\alpha_{ij}$, which depends on particular details such as the type of the pair of atoms (e.g., C-N, N-O), whether they are covalently bonded, etc. The MSD gives a lower bound for the distance between the two atoms. As mentioned in the previous subsection, for simplicity, we set the minimum separation distance to be 1Å uniformly, regardless of the types of atoms.

The error of the computed configuration is measured by the RMSD. If the computed configuration $X$ is optimally aligned to the true configuration $X^*$, using the procedure of section 3.3, then the RMSD is defined by the following formula:

$$\text{RMSD} = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{x}_i^*\|^2.$$

The RMSD is a measure of the "average" deviation of the computed atom positions from the true atom positions.

**5.3. Results and discussion.** To help the reader appreciate the difficulty of the molecular conformation problem under the setup we have just described, we solved two of the smaller molecules using sparse but exact distances. This information is presented in Table 5.1. Even if we solve a molecular problem in a centralized fashion without divide-and-conquer, due to the sparsity of the distance data, the problem is not localizable, and we can get only an approximate solution.

TABLE 5.1

*The molecular problem with sparse but exact distance data cannot always be solved exactly. We have denoted by $n$ the number of atoms in the molecule and by $\ell$ the number of atoms with degree less than* 4.

| Input data: Subsets of exact distances $\leq$ 6Å | | | | | |
|---|---|---|---|---|---|
| Molecule | $n$ | 30% distances | | 20% distances | |
| | | RMSD (Å) | $\ell$ | RMSD (Å) | $\ell$ |
| 1GM2 | 166 | 0.10 | 0 | 0.83 | 10 |
| 1PTQ | 402 | 0.39 | 9 | 1.16 | 38 |

The performance of DISCO is listed in Tables 5.2 and 5.3, for the case when the initial random number seed is set to zero, i.e., `randn('state',0)`, `rand('state',0)`. The RMSD plots across the molecules, with 10 runs given different initial random number seeds, is shown in Figure 5.1.

When given 30% of the short-range distances, which are corrupted by 20% noise, for each molecule, DISCO is able to compute an accurate structure. We have a final structure (core structure) with an RMSD of 0.9–1.6Å (0.6–1.6Å). The core structure is the union of thelikely localizable components. Typically, the core structure is solved to a slightly higher accuracy.

We believe these are the best numbers which we could hope for, and we present an intuitive explanation of why this is so. For simplicity, let us assume that the mean distance of any given edge is 3Å. This is reasonable because the maximum given distance is about 6Å. Given 20% noise, we give a bound of about 2.4–3.6Å for that edge. Thus each atom can move about 1.2Å. The RMSD should therefore be approximately 1.2Å.

When given 20% of the short-range distances, the conformation problems become more difficult, due to the extreme sparsity of available distances. For each problem,

TABLE 5.2

*We have denoted by $\ell$ the number of atoms with degree less than 4. The mean degree of an atom is 10.8–12.6. The approximate core of the structure consists typically of 94–97% of the total number of atoms. For large molecules, the SDP localization consumes about 70% of the running time, while the gradient descent consumes about 20% of the running time. The numbers in parentheses are the RMSD of the core structures.*

| Input data: 30% distances $\leq$ 6Å, corrupted by 20% noise | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Molecule | $n$ | $\ell$ | RMSD (Å) | | | | Time (h:m:s) | |
| | | | Normal | | Uniform | | Normal | Uniform |
| 1GM2 | 166 | 0 | 0.92 | (0.94) | 0.74 | (0.76) | 00:00:08 | 00:00:13 |
| 1PTQ | 402 | 9 | 1.08 | (0.96) | 1.00 | (0.85) | 00:00:23 | 00:00:18 |
| 1PHT | 814 | 15 | 1.45 | (0.69) | 1.15 | (0.56) | 00:01:22 | 00:01:00 |
| 1AX8 | 1003 | 16 | 1.35 | (1.17) | 1.00 | (0.80) | 00:01:31 | 00:01:07 |
| 1TIM | 1870 | 45 | 1.23 | (1.03) | 0.94 | (0.80) | 00:04:18 | 00:03:28 |
| 1RGS | 2015 | 37 | 1.52 | (1.36) | 1.51 | (1.41) | 00:05:25 | 00:03:53 |
| 1KDH | 2923 | 48 | 1.38 | (1.16) | 1.21 | (0.89) | 00:07:57 | 00:05:30 |
| 1BPM | 3672 | 36 | 1.10 | (1.03) | 0.79 | (0.73) | 00:11:24 | 00:08:08 |
| 1TOA | 4292 | 62 | 1.15 | (1.07) | 0.89 | (0.78) | 00:13:25 | 00:09:06 |
| 1MQQ | 5681 | 46 | 0.92 | (0.86) | 0.82 | (0.74) | 00:23:56 | 00:17:24 |
| 1I7W | 8629 | 134 | 2.45 | (2.34) | 1.51 | (1.40) | 00:40:39 | 00:31:26 |
| 1YGP | 13488 | 87 | 1.92 | (1.93) | 1.50 | (1.52) | 01:20:35 | 01:00:55 |

TABLE 5.3

*We have denoted by $\ell$ the number of atoms with degree less than 4. The mean degree of an atom is 7.4–8.6. The approximate core of the structure consists typically of 88–92% of the total number of atoms. For large molecules, the SDP localization consumes about 60% of the running time, while the gradient descent consumes about 30% of the running time.*

| Input data: 20% distances $\leq$ 6 Å, corrupted by 10% noise | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Molecule | $n$ | $\ell$ | RMSD (Å) | | | | Time (h:m:s) | |
| | | | Normal | | Uniform | | Normal | Uniform |
| 1GM2 | 166 | 7 | 1.44 | (1.25) | 0.92 | (0.77) | 00:00:04 | 00:00:04 |
| 1PTQ | 402 | 46 | 1.49 | (1.17) | 1.48 | (1.14) | 00:00:14 | 00:00:10 |
| 1PHT | 814 | 53 | 1.53 | (1.13) | 1.40 | (0.97) | 00:01:02 | 00:00:54 |
| 1AX8 | 1003 | 78 | 1.69 | (1.40) | 1.52 | (1.17) | 00:01:00 | 00:00:55 |
| 1TIM | 1870 | 143 | 1.77 | (1.41) | 1.84 | (1.50) | 00:02:43 | 00:02:33 |
| 1RGS | 2015 | 189 | 9.82 | (9.51) | 1.83 | (1.39) | 00:03:32 | 00:03:14 |
| 1KDH | 2923 | 210 | 1.74 | (1.31) | 1.63 | (1.13) | 00:04:28 | 00:04:45 |
| 1BPM | 3672 | 187 | 1.46 | (1.14) | 1.31 | (0.91) | 00:06:52 | 00:06:33 |
| 1TOA | 4292 | 251 | 1.67 | (1.26) | 1.58 | (1.16) | 00:08:39 | 00:08:49 |
| 1MQQ | 5681 | 275 | 1.17 | (0.95) | 1.17 | (0.92) | 00:14:31 | 00:14:21 |
| 1I7W | 8629 | 516 | 4.04 | (3.69) | 3.87 | (3.52) | 00:26:52 | 00:26:04 |
| 1YGP | 13488 | 570 | 1.83 | (1.63) | 1.70 | (1.46) | 01:01:21 | 00:57:31 |

the mean degree of each atom is 7.4–8.6, so the data is highly sparse. We set a lower level of 10% noise for these experiments. Even under such challenging input, DISCO is still able to produce a fairly accurate structure ($\approx$ 2Å) for all the molecules except 1RGS and 1I7W ($\approx$ 4Å) in Table 5.3.

In Figure 5.1, we plot the RMSDs for different random inputs of the same molecule. The graphs indicate that DISCO is able to produce an accurate conformation (RMSD $<$ 3Å) for most of the molecules over different random inputs. DISCO does not perform so well on the two molecules 1RGS and 1I7W, which have less regular geometries. Although we have designed DISCO with safeguards, DISCO will nevertheless occasionally make mistakes in aligning subconfigurations that are not well localized.

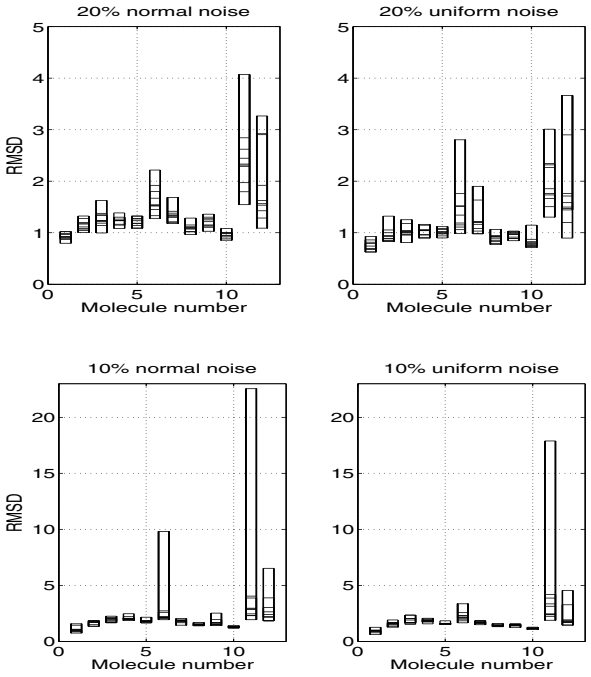


FIG. 5.1. *For each molecule, ten random inputs were generated with different initial random number seeds. We plot the RMSDs of the ten structures produced by DISCO against the molecule number. (top left)* 30% *short-range distances,* 20% *normal noise; (top right)* 30% *short-range distances,* 20% *uniform noise; (bottom left)* 20% *short-range distances,* 10% *normal noise; (bottom right)* 20% *short-range distances,* 10% *uniform noise.*

**6. Conclusion and future work.** We have proposed a novel divide-and-conquer, SDP-based algorithm for the molecular conformation problem. Our computational experiments demonstrate that DISCO is able to solve very sparse and highly noisy problems accurately, in a short amount of time. The largest molecule with more than 13000 atoms was solved in about one hour to an RMSD of 1.6Å, given only 30% of pairwise distances less than 6Å and corrupted by 20% multiplicative noise. We hope that, with the new tools and ideas developed in this paper, we will be able to tackle molecular conformation problems with highly realistic input data, as was done in [12].

REFERENCES

[1] 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.

[2] 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. Auto. Sci. Eng., 3 (2006), pp. 360–371.

[3] 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 J. Sci. Comput., 30 (2008), pp. 1251–1277.

[4] P. BISWAS AND Y. YE, *A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization*, in Multiscale Optimization Methods and Applications, Nonconvex Optim. Appl. 82, Springer, New York, 2006, pp. 69–84.

[5] Q. DONG AND Z. WU, *A geometric build-up algorithm for solving the molecular distance geometry problems with sparse distance data*, J. Global Optim., 26 (2003), pp. 321–333.

[6] W. Glunt, T. Hayden, S. Hong, and J. Wells, *An alternating projection algorithm for computing the nearest Euclidean distance matrix*, SIAM J. Matrix. Anal. Appl., 11 (1990), pp. 589–600.
[7] J. Gower and G. Dijksterhuis, *Procrustes Problems*, Oxford University Press, London, 2004.
[8] I. G. Grooms, R. M. Lewis, and M. W. Trosset, *Molecular embedding via a second order dissimilarity parameterized approach*, SIAM J. Sci. Comput., 31 (2009), pp. 2733–2756.
[9] O. Güler and Y. Ye, *Convergence behavior of interior point algorithms*, Math. Program., 60 (1993), pp. 215–228.
[10] T. F. Havel, *A evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance*, Progr. Biophys. Molec. Bio., 56 (1991), pp. 43–78.
[11] T. F. Havel, I. D. Kuntz, and G. M. Crippen, *The combinatorial distance geometry approach to the calculation of molecular conformation*, J. Theoret. Biol., 104 (1983), pp. 359–381.
[12] T. F. Havel and K. Wüthrich, *A distance geometry program for determining the structures of small proteins and other macromolecules from nuclear magnetic resonance measurements of* $1h$*-*$1h$ *proximities in solution*, Bull. Math. Biol., 46 (1984), pp. 673–698.
[13] B. Hendrickson, *The molecule problem: Exploiting structure in global optimization*, SIAM J. Optim., 5 (1995), pp. 835–857.
[14] N.-H. Z. Leung and K.-C. Toh, *The DISCO algorithm web page*, available online at http://www.math.nus.edu.sg/∼mattohkc/disco.html.
[15] M. Locatelli and F. Schoen, *Structure prediction and global optimization*, Optima (Mathematical Programming Society Newsletter), 76 (2008), pp. 1–8.
[16] J. J. Moré and Z. Wu, *Global continuation for distance geometry problems*, SIAM J. Optim., 7 (1997), pp. 814–836.
[17] J. J. Moré and Z. Wu, *Distance geometry optimization for protein structures*, J. Global Optim., 15 (1999), pp. 219–234.
[18] R. Reams, G. Chatham, W. Glunt, D. McDonald, and T. Hayden, *Determining protein structure using the distance geometry program apa*, Comput. Chem., 23 (1999), pp. 153–163.
[19] I. J. Schoenberg, *Remarks to Maurice Frechet's article "Sur la definition axiomatique d'une classe d'espaces vectoriels distancies applicables vectoriellement sur l'espace de Hilbert,"* Ann. Math., 36 (1935), pp. 724–732.
[20] A. M.-C. So and Y. Ye, *Theory of semidefinite programming for sensor network localization*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Vancouver, BC, 2005, SIAM, Philadelphia, 2005, pp. 405–414.
[21] K.-C. Toh, M. J. Todd, and R. H. Tutuncu, *SDPT*3*—A MATLAB software package for semidefinite programming*, Optim. Methods Softw., 11 (1999), pp. 545–581.
[22] M. W. Trosset, *Applications of multidimensional scaling to molecular conformation*, Comput. Sci. Statist., 29 (1998), pp. 148–152.
[23] M. W. Trosset, *Distance matrix completion by numerical optimization*, Comput. Optim. Appl., 17 (2000), pp. 11–22.
[24] M. W. Trosset, *Extensions of classical multidimensional scaling via variable reduction*, Comput. Statist., 17 (2002), pp. 147–163.
[25] R. H. Tutuncu, K.-C. Toh, and M. J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT*3, Math. Program. Ser. B, 95 (2003), pp. 189–217.
[26] 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.
[27] D. Wu and Z. Wu, *An updated geometric build-up algorithm for solving the molecular distance geometry problems with sparse distance data*, J. Global Optim., 37 (2007), pp. 661–673.
[28] Z. Zhang and H. Zha, *Principal manifolds and nonlinear dimensionality reduction via local tangent space alignment*, SIAM J. Sci. Comput., 26 (2004), pp. 313–338.
[29] Z. Zhang and H. Zha, *A domain decomposition method for fast manifold learning*, in Proceedings of the Nineteenth Annual Conference on Neural Information Processing Systems, Vancouver, BC, 2005; available online at http://books.nips.cc/papers/files/nips18/NIPS2005_0454.pdf.