Geometric buildup-based error minimization algorithms for protein structure determination problems with sparse large-noisy distances

Zhenli Sheng, Yaxiang Yuan

December 2, 2014

Abstract Distance Geometry Problem has attracted a great deal attentions in recent years due to its wide applications in biology, environmental monitor and sensor-based applications, etc. Based on the fast Geometric Buildup algorithms developed by Wu and his coauthors [13, 36, 27], we propose enhanced Geometric Buildup algorithms (GBEM-LLS and GBEM-NLS). Our main contribution is two-fold. First, we analysis the importance of computation order and carefully design a buildup sequence to make our algorithms much more stable than before. Second, an novel framework is proposed such that error minimization technique is integrated in every buildup process to control error accumulation effectively, which is a dominant disadvantage of the original algorithms. Extensive numerical experiments show that our algorithms can give reasonably good results in short computation time. For instance, given inter-atomic distances which are less than 6Å and are corrupted with 10% multiplicative noise, a 5681-atom can be realized within 3 minutes with a root-mean-square-deviation (RMSD) of 0.24Å. The former state-of-the-art Geometric Buildup methods [27] can only handle at most 0.01% noise with distances under 6Å, our novel algorithms can deal with 10% noises, which makes the Geometric Buildup methods practical, hence can be applied in real problems.

1 Introduction

There exist many ways to describe *Distance Geometry Problem* (henceforth, DGP), we favor the graph language as following,

For a graph $G = (V, E)$, where V is the vertex set and E is the edge set, given some distances d_{ij}, for $(i, j) \in E$, the aim is to find $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$

(where d is prescribed), such that

$$\|x_i - x_j\| = d_{ij}, \quad (i,j) \in E. \tag{1}$$

In the ideal case, no errors exist in the distance d_{ij}, the problem is called *DGP with exact (noiseless) distances*, otherwise it is called *DGP with inexact (noisy) distances*. In an even more realistic case, rather than the distance d_{ij}, but the upper bound u_{ij} and lower bound l_{ij} are given for each edge in E, then we need to find coordinates $x_i \in \mathbb{R}^d$ satisfying the following inequalities instead,

$$l_{ij} \leq \|x_i - x_j\| \leq u_{ij}, \quad (i,j) \in E. \tag{2}$$

Problem in this version is called *DGP with distance bounds*. In the rest of this paper, we focus on *DGP with noisy distances*.

DGP with exact distances has been proved to be NP-hard in general [23], a proof the corresponding noisy case can be found in [36]. Therefore, it is impossible to find polynomial-time algorithms to solve all the instances of this problem unless $P = NP$, though some special instances like *K-trilateration graph* [18] do admit polynomial-time approach.

This mathematical problem has a large number of applications in different fields, here we just list a few of them. In graph drawing area [17], it is called *Graph Realization Problem*, in which the task is to draw a graph such that the distances between points reveal the given weights.

In this paper, we study an important application in computational biology, which is called *Protein Structure Determination* [9, 26, 32], in some literatures, it is also called *Molecular Conformation Problem* [11, 6, 16] or *Molecular Distance Geometry Problem* [12, 13, 14]. In this problem, a part of the pairwise distances between atoms either can be measured by Nuclear Magnetic Resonance (NMR) or X-ray Crystallography technique, or can be estimated based on biological information, such as bond angle and bond length. Many properties of proteins are heavily relied on the three-dimensional structure of the proteins, which makes the determination of the relative coordinates of the atoms so important (the distance remains the same if the whole structure is moved under rigid transformations including translations, rotations and reflections, this is why we call the locations are relative coordinates).

Another important application is called *Wireless Sensor Network Localization* [1, 10, 19, 37]. Sensor is an efficient tool for environmental monitoring, animal management, and even in military operations, since it is cheap and convenient. Usually every unit is equipped with a sensor, which can send radio signals, collect and process simple information. Determining the location of the sensors usually is the first step because many further applications are location-related. The sensors can send signals within some radio range, thus the distance of nearby sensors can be measured by the difference of arrival time or the decay of the signal, the latter is more accurate in general. A main feature of this application differing from others is that the exact location of a small portion of the sensors are known in advance, those are called *anchors*.

Notice that the distances in the last two applications both are local due to the limitation of the NMR techniques or the power of the sensors, which means only distances within some cutoff range are known. As a result, the distance information is very sparse as a whole, that is to say, the known distances is only a small portion of all the pairwise distances.

In the rest of this paper, we focus on the protein structure determination, that is to say, we consider the three-dimensional anchor-free DGP and the points are atoms. We first model the problem as an optimization problem in Section 2, then review some related works in Section 3. A somehow detailed review on the original Geometric Buildup methods is presented in Subsection 3.4. Following that we give the framework and details of our algorithm in Section 4. Problem setting and extensive numerical results are provided in Section 5. Further discussions and possible extensions are given in Section 6.

Notations Throughout this paper, vectors are column vectors and are denoted by lower-case letters, matrices are denoted by capital letters. For a vector $x \in \mathbb{R}^n$, $\|x\| = \sqrt{x^T x}$ is the Euclidean 2-norm, for a matrix $X \in \mathbb{R}^{m \times n}$, $\|X\|_F = \sqrt{tr(X^T X)}$ is the Frobenius norm. $A = (a_{ij})$ means that the elements in matrix A are a_{ij}. $A(:, 1:k)$ denotes the 1 to k columns of A in a Matlab fashion, $A(1:k, 1:k)$ is the upper left k-dimensional submatrix. e_i is the i-th column of the identity matrix. We use subscript i to denote the i-th component of a vector, and superscript k to denote the iteration number. Conformation, configuration, layout and location are used interchangeably to denote the coordinates of all the atoms in a certain protein.

2 Error Minimization Models

We have introduced DGP as a feasible problem of a collection of equations, but it is usually solved by modeling as an unconstrained optimization problem,

$$\min_{x_1, x_2, \ldots, x_n} f(x_1, x_2, \ldots, x_n), \tag{3}$$

where x_i are the coordinates of the points, $f(\cdot)$ is an error function which measures the deviations between computed distances and given ones. In the literatures, the most commonly used error functions for DGP are as follows,

- Stress function

$$Stress(x_1, x_2, \dots, x_n) = \sum_{(i,j)\in E} \omega_{ij}(\|x_i - x_j\| - d_{ij})^2, \tag{4}$$

- Smoothed Stress function

$$SStress(x_1, x_2, \dots, x_n) = \sum_{(i,j)\in E} \omega_{ij}(\|x_i - x_j\|^2 - d_{ij}^2)^2, \tag{5}$$

- Absolute Error function

$$AbsErr(x_1, x_2, \dots, x_n) = \sum_{(i,j)\in E} \omega_{ij} \left|\|x_i - x_j\|^2 - d_{ij}^2\right|, \tag{6}$$

where ω_{ij} are the weights on edges, which can be chosen properly to generate suitable models. For example, we choose all ω_{ij} to be 1 to treat all the distance data equally, or we can set it small if we do not trust a certain data or set it large to emphasis a certain data. One special choice is $\omega_{ij} = 1/d_{ij}^2$ in (4), thus the model measures the relative errors instead.

Among them, *AbsErr* is usually used by SDP-based models as described in Subsection 3.3. Rather than *Stress*, the gradient of *SStress* is continuous since the norm term—$\|x_i - x_j\|$ does not show up in the denominator, which is what earns this function its name.

It is easy to observe that, if the given distances are exact, finding the true coordinates is equivalent to finding the global solution of these error functions, whose function value is zero. However, all the error functions have extremely large number of local minimizers [34] (Page 14), and most of the local minimizers are meaningless to us, which makes the problem so challenging because conventional continuous optimization techniques like gradient descent or Newton's method can only guaranty to find a stationary point or a local minimizer. Consequently, it is necessary to take advantage of some other methods like SDP relaxation to obtain a coarse layout and minimize an error function to further improve the result [6, 16]. These papers emphasis the SDP process and use error minimization as a postprocess procedure. The optimization method to minimize error functions are somehow standard, thus the authors usually do not say much about it, which gives us an impression that this process is useful but do not stand at the center of the algorithm. However, we would like to view these techniques as a way to obtain a good initial point to accomplish our final task—finding the global or at least a good approximate solution to the error functions, that is to say, we emphasis the optimization procedure and view other techniques as auxiliary methods. Actually, this point of view gives us the motivation to introduce error function minimization procedure to every buildup step and finally achieve significant improvement, the details will become clear later.

We emphasis again that since we do not have anchor points, the whole configuration remains the same under rigid transformations, what we get from the algorithms are just the relative coordinates up to any rigid transformations.

3 Related works

The prior works about this problem trace back to [24] in 1935 and [7, 31]. After that, a large amount of algorithms have been proposed and each of them has its own pros and cons. We review some of them for our own interest in this section but have no attempt to cover them all. For example, we do not give the details of [22], which is also an interesting paper working on the rank constrained reduced distance matrix, based on an old result in [24]. For a brief summary of existing methods, see Table 13.1 in [16]. For those who have further interests, we refer them to [18], which is a recently published exhaustive survey paper in SIAM Review.

3.1 Matrix Decomposition Method

In [7], a matrix decomposition method is proposed to solve the DGP with full set of distances, by full set we mean all the pairwise distances are known. It is the oldest mature method for DGP to our knowledge, but is still very important because it is the foundation of many other methods. We give the details of this algorithm as below due to its fundamental position and its usage in our algorithms.

Since the whole structure is invariant under rigid transformations, without lose of generality, we set x_n at origin, i.e., $x_n = (0, 0, 0)^T$. Then we have $d_{in} = \|x_i - x_n\| = \|x_i\|$. Moreover, by taking the square of both sides of $\|x_i - x_j\| = d_{ij}$, we obtain

$$\|x_i\|^2 - 2x_i^T x_j + \|x_j\|^2 = d_{ij}^2, \quad \text{for } i, j = 1, 2, \ldots, n-1. \tag{7}$$

By substituting $\|x_i\|$ with d_{in}, and moving known items to one side, we have

$$x_i^T x_j = (d_{in}^2 - d_{ij}^2 + d_{jn}^2)/2, \quad \text{for } i, j = 1, 2, \ldots, n-1. \tag{8}$$

Let $X = (x_1, x_2, \ldots, x_{n-1})^T$ be the coordinate matrix, where each column of X^T is the coordinate of one point, and $B = (b_{ij}) = ((d_{in}^2 + d_{jn}^2 - d_{ij}^2)/2)$, then equations in (8) can be rewritten in a compact form as $XX^T = B$. This matrix equation can be solved by Singular Value Decomposition (SVD) as stated below,

Theorem 3.1 *(Eckart, Young [15]) Let $B = U\Sigma U^{\mathrm{T}}$ be singular value decomposition, where the singular values in Σ are arranged in descending order. Let $V = U(:, 1:k)$ and $\Lambda = \Sigma(1:k, 1:k)$, then $X = V\Lambda^{1/2}$ solves*

$$\min_{rank(X)\le k} \|XX^{\mathrm{T}} - B\|_F.$$

In our applications, if the distances are exact and not all the atoms are in the same plane, matrix B is rank-3, the result above is an exact solution. If some of the distances are noisy, the rank of B is usually larger than 3, then the rank-3 best approximation is obtained in this way.

About DGP with full set of exact distances, a linear-time algorithm was first proposed in [12], which is also the first paper about Geometric Buildup method for DGP, though the name—"Geometric Buildup" was first proposed in a subsequent paper [13], the details of this algorithm will be given in Subsection 3.4.

3.2 Global Continuation Algorithm

As we have mentioned, the error functions usually have prohibitively large amount of local minimizers, thus it is very difficult to solve these problems directly. In [20, 21], a Global Continuation Algorithm *dgsol* is developed to conquer this difficulty.

Two main techniques are exploited in this algorithm. First, a global smoothing process is applied. Specifically, the convolution of the original error function with a Gaussian density function is computed. Intuitively, this process replaces the original function value at a certain point by the average of function values around that point, with the weights specified by the Gaussian density function. After this process, many local minimizers are smoothed while the global minimizer is kept, thus it is much more easier to find the global solution to the new function. The degree of the smoothness is controlled by a parameter λ, the smoothed function approaches the original function as λ goes to zero. The second technique is so-called "continuation", where the smoothing parameter λ is gradually reduced to zero to trace back to the solution of the original problem. The philosophy behind the latter technique is closely related to homotopy method for solving nonlinear system of equations.

The authors test their algorithm in small protein fragments [21] and show that *dgsol* is significantly more reliable and efficient that multi-starts methods. This method is of theoretical interest but more investigations need be done for solving large-size real-world problems.

3.3 SDP-based algorithms

Semidefinite Programming (SDP) is a very hot research area in recent years. The theory about it has become very mature, as well as many user-friendly open-source softwares are developed, such as SeDuMi and SDPT3. To the best of our knowledge, SDP relaxation for DGP was first suggested in [28], and then further developed by many researchers [5, 6, 25, 16]. The main techniques involved are briefly summarized as below.

The basic model used is (6). By replacing $|x|$ with $x_+ - x_-$ $(x_+ \geq 0, x_- \geq 0)$, the nonsmooth part is removed. Let $X \in \mathbb{R}^{n\times d}$ be the coordinate matrix as defined before, then $\|x_i - x_j\|^2$ in (6) can be rewritten as $e_{ij}^T XX^T e_{ij}$, where $e_{ij} = e_i - e_j$. By relaxing $Y = XX^T$ with $Y \succeq XX^T$ [8], this linear matrix inequality is equivalent to

$$Z = \begin{pmatrix} Y & X \\ X^T & I \end{pmatrix} \succeq \mathbf{0}, \quad Z \text{ is symmetric.} \tag{9}$$

Furthermore, the bottom-right $d \times d$ submatrix is required to be the identity matrix. Thus, the SDP relaxation of (6) can be reformulated as a standard SDP problem in Z.

One charming advantage of SDP relaxation method is that there are no essential difficulties to generalize from equality case to inequality case, while this generalization is usually not trivial for other methods. One main disadvantage is that the computational cost to solve large-scale (more than several thousands) SDP problem is prohibitively high, due to the limitation of current solvers. Taking this into consideration, distributed technique is first introduced in [6], in which they divide the original graph into several subgraphs with overlaps according to the distance matrix, by exploiting the fact that the distances in real problems are sparse and local. This technique is also utilized in [16], where they further incorporate distance information derived from chemistry knowledge into the algorithm and design heuristic rule to detect whether the subgroup is well located or not, which is a very important step for distributed algorithms. Another way to conquer this difficulty is to further relax the SDP relaxation [33], in which a series of Node-based or Edge-based sub-SDP problems are solved, which are small-sized and can be solved efficiently.

3.4 Geometric Buildup algorithm

Our algorithm is based on the Geometric Buildup (GB) algorithm, which is first proposed in [12] and then generalized to sparse disntace data in [35]. The algorithm is further developed in [36, 27], where linear and nonlinear least square approximation are introduced to prevent the propagation of computation errors. For the completeness of this paper, we first briefly review several key steps in Buildup methods.

The central idea of the GB algorithms is based on the fact that in d-dimensional Euclidean space, usually $d+1$ distances to the fixed points can uniquely determine one point. For example, in 3-dimensional space, two distances to the fixed points determine one circle (assume two balls intersect but not tagent), three distances determine two points, with the fourth one can uniquely determine one point. Of course, we require that the four fixed points are not in the same plane. For general case, we require that the $d+1$ fixed points are not in the same hyperplane. With this observation, it is natural to come up with the GB Algorithms: Determine four points first, and determine the rest of points one by one.

3.4.1 Determine first four points

At the very beginning, we find a clique of four points, where clique means that all the pairwise distances are known. Recall that all the distances remain the same if the whole structure is moved by translations, rotations and reflections, without loss of generality, we set the first point at the origin—$(0,0,0)^T$, the second point at positive x-axis with coordinate $(d_{12},0,0)^T$, the third point at the first orthant of xy-plane and the fourth one between one of the two possible locations. Direct computation details can be found in [12].

3.4.2 Buildup step: determine one unknown point

The former determined four points are called base points, we then determine the rest of the points one by one, just like build up a house from some cornerstones.

As we have mentioned, four distances are needed to determine a new point. Suppose point j is to be determined, which has four known distances to the determined points with coordinates x_i $(i=1,2,3,4)$, i.e., we have

$$\|x_i - x_j\| = d_{ij}, \quad i = 1,2,3,4. \tag{10}$$

Square these equations to obtain

$$\|x_i\|^2 - 2x_i^T x_j + \|x_j\|^2 = d_{ij}^2, \quad i = 1,2,3,4. \tag{11}$$

Keep in mind that x_j is the variable, and x_i are fixed. Now we have four equations, subtracting one equation by its subsequent one (this is not the only way [13]) to obtain

$$2(x_{i+1} - x_i)^T x_j = (\|x_{i+1}\|^2 - \|x_i\|^2) - (d_{i+1,j}^2 - d_{ij}^2), \quad i = 1, 2, 3. \tag{12}$$

Let A be a matrix and b a vector, where

$$A = 2 \begin{bmatrix} (x_2 - x_1)^T \\ (x_3 - x_2)^T \\ (x_4 - x_3)^T \end{bmatrix}, \ b = \begin{bmatrix} (\|x_2\|^2 - \|x_1\|^2) - (d_{2j}^2 - d_{1j}^2) \\ (\|x_3\|^2 - \|x_2\|^2) - (d_{3j}^2 - d_{2j}^2) \\ (\|x_4\|^2 - \|x_3\|^2) - (d_{4j}^2 - d_{3j}^2) \end{bmatrix}, \tag{13}$$

the equations in (12) can be reformulated in a compact way

$$Ax_j = b. \tag{14}$$

A is nonsingular if the determined points are not in the same plane, then the exact location is $x_j = A^{-1}b$.

3.4.3 Linear Least-Squares

When the distances are noisy, the equation $Ax_j = b$ can not give us an accurate solution if only four distances are exploited. A natural way to deal with this situation is to take advantage of as much distances as possible to obtain a balanced solution. If we know l ($l \geq 4$) distances, then rather than a matrix equation (14), a linear least square problem

$$\min_{x_j} \|Ax_j - b\| \tag{15}$$

is solved, where A and b is constructed similar to (13), but both have $l - 1$ rows. This is the so-called Geometric Buildup with Linear Least-Squares (we refer it as GB-LLS) in [36], in which the detailed algorithm can be found.

3.4.4 Nonlinear Least-Squares

Notice that in GB-LLS, only the distances between the undetermined point to the determined points (bold solid lines in Figure 1) are exploited, but we usually know some distances among the determined points (normal solid lines in Figure 1). In order to take advantage these information, Geometric Buildup with Nonlinear Least-Squares (we refer it as GB-NLS) is proposed [36].

In GB-NLS, the first step is to compute the unmeasured distances among the fixed points through calculated coordinates (broken lines in Figure 1),

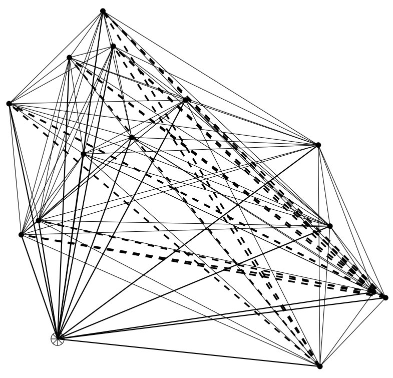

Figure 1: the subgraph in one step of GB-NLS. asterisk point: to be determined, dot points: determined, bold and normal solid lines: known distances, broken lines: computed distances

after that, all the distances among these $l+1$ points are known, therefore Matrix Decomposition Method introduced in Subsection 3.1 can be used to determine point j, and adjust these l determined points slightly at the same time. The adjustment usually is tiny but very important, because it provides the chance to mend the inaccurate localizations determined in previous steps when new distance information is involved, while in GB-LLS a point is permanently fixed once it is determined.

4 our Algorithm

Compare to SDP relaxation-based algorithms, Geometric Buildup methods (GB-LLS and GB-NLS) are extremely fast since small-sized subproblem is solved at each step. Nevertheless, the original Geometric Buildup algorithms can only tolerate very small noises in distances, usually less than 0.01%, even for the state-of-the-art one [27], which is not practical in real world applications. In particular, even for the exact distance case, accumulation of round error may destroy the conformation when the number of the atoms is large (more than several thousands), the situation is even worse in the noisy distances case. The initial motivation of this work is to develop some new techniques to overcome this shortage. The proposed algorithms can be viewed as enhanced GB algorithms. Another angle to view our algorithm is that our aim to solve a difficult error function minimization problem, which is a global minimization problem with tremendously many local minimizer, and the layout of GB algorithms serves as a good initial point. All in all, both the Buildup process and the optimization process are equivalently substantial to make the novel algorithm fast and accurate.

In the rest of this section, we first give the analysis of errors in the original GB algorithms, then our strategy to choose the new point and also the optimization subproblems are introduced. Finally a complete description of the novel algorithm is presented.

4.1 Why does numerical error matter?

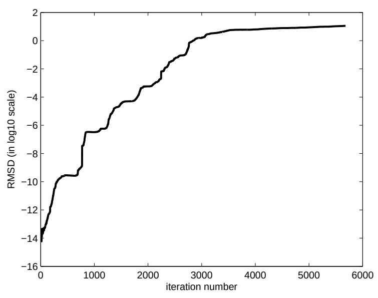

Figure 2: RMSD results when GB-LLS is applied to 1MQQ (5681 atoms) with exact distances (cutoff = 6Å)

The propagation of numerical errors is a big problem for Geometric Buildup algorithms [35, 36], which is also observed in our own numerical experiments. Figure 2 illustrates representative RMSD results (in log10 scale) when GB-LLS is applied to compute 1MQQ (5681 atoms) with exact distances. In theory, this algorithm is guaranteed to produce an accurate conformation [27]. However, though the RMSD is good at first several hundreds of iterations, it goes up step by step quickly, and becomes totally wrong after about 3500 iterations. Gap exists between theoretical results and computational experiences due to numerical errors, which asks for more robust algorithms to fix it.

4.2 Analysis of error accumulation in existing Geometric Buildup algorithms

Here we consider the simple case where the exact distances are given and we use linear least square to determine the new point.

Assume point j is to be determined, which has l distances to the determined points $1, 2, \ldots, l$. Let the ground truth be $\widehat{x}_i$, the computed coordinates be x_i, and denote location error of point i by $\delta x_i = x_i - \widehat{x}_i$. We further assume $\delta x_i = O(\delta x)$.

Recall that we need solve $\min_{x_j} \|Ax_j - b\|$, where

$$A = 2[(x_{i+1} - x_i)^T], b = [(\|x_{i+1}\|^2 - \|x_i\|^2) - (d_{i+1,j}^2 - d_{ij}^2)],\ i = 1, 2, \dots, l-1. \tag{16}$$

Then we have

$$\begin{aligned}\|x_{i+1}\|^2 - \|x_i\|^2 &= \|\widehat{x}_{i+1} + \delta x_{i+1}\|^2 - \|\widehat{x}_i + \delta x_i\|^2 &(17)\\ &= \|\widehat{x}_{i+1}\|^2 - \|\widehat{x}_i\|^2 + 2\widehat{x}_{i+1}\delta x_{i+1} - 2\widehat{x}_i \delta x_i + \delta x_{i+1}^2 - \delta x_i^2. &(18)\end{aligned}$$

Therefore, $A = \widehat{A} + O(\delta x), \quad b = \widehat{b} + O(\delta x)$

$$\begin{aligned}\|Ax_j - b\| &= \|(\widehat{A} + \delta A)(\widehat{x}_j + \delta x_j) - (\widehat{b} + \delta b)\| &(19)\\ &= \|\widehat{A}\delta x_j + O(\delta x)\|. &(20)\end{aligned}$$

From the analysis above, we know that the error in each buildup step comes from three aspects: (1) the error inherits from previous calculation, which is hidden in A and b and is the dominant error in most of the buildup steps. (2) the numerical error caused by ill-conditional coefficient matrix A, which arises only in a small number of steps but sometimes leads to large error. (3) round error existing in all limited-precision computations.

4.3 Strategy to choose the newly added point

Remind that the Geometric Buildup algorithms determine points one by one, which use the coordinates of the determined points and the distances between unknown point to determined points, hence it is obvious that the accuracy of the determined points affect that of subsequent determined points heavily, which makes the computation order very important. In the original GB methods, the default order of the atoms in PDB file (which will be explained in the numerical results part) is used. This kind of structure information is useful but unavailable for real world problems, in which only distances are measured or estimated [12]. We would like to design a rule that is efficient but solely depend on the given distances, therefore, it is realistic.

Based on the analysis and discussion before, we propose to exploit the following two kinds of information to choose point j to be determined.

- (structure information) Let $\mathcal{Y}$ be index set of determined points and $\mathcal{N} = \{1, 2, \dots, n\} \backslash \mathcal{Y}$ the undetermined ones, we define

$$p(i) = \sharp\{d_{ik} \neq 0 : k \in \mathcal{Y}\}, \quad for\ i \in \mathcal{N}, \tag{21}$$

and further define

$$\mathcal{I} = \{i : p(i) = \max_{k \in \mathcal{N}} \, p(k)\}, \tag{22}$$

where $p(i)$ is the the number of known distances to the determined points for undetermined point i, and $\mathcal{I}$ is the index set of points which have maximal number of known distances.

- (distance information) We choose j satisfying

$$j = \arg\min_{i \in \mathcal{I}} \sum_{k \in \mathcal{Y}} d_{ik}, \tag{23}$$

which is closest to the determined points in terms of total distances among all the points in $\mathcal{I}$.

Intuitively, we choose the point that has the maximal number of distances and is closest to the determined points at the same time. The reason why we need the second rule is that usually there are more than one point satisfying the first rule, randomly choose one point from these points makes the algorithm unstable and unpredictable.

The benefits of this computational order include but are not limited to the following several aspects. (1) The oscillation is removed. In the original GB algorithms, we often observe that the buildup process goes far away and then comes back, this phenomenon disappears under our rules. (2) As a result of (1), we actually have the potential to exploit more number of distances than the original algorithms, which is desired because more distances means we have larger chance to dismiss the measure errors existing in the distance data. (3) The whole algorithm is more stable, here by stable we mean that the condition number of the coefficient matrix in least-quare subproblem is reasonably good, therefore, the numerical difficulty of solving the subproblem is reduced. This is verified by the numerical tests.

We construct a simple 2D example in Figure 3 to illustrate the importance of the computational order. In this example, we have six points, whose serial numbers and locations are shown in the graph, where the line determined by point 1 and 2 is horizontal. What we know are the distances denoted by solid lines. We apply GB-LLS to this instance, points 1,2,3 are determined first as base points, and points 4,5,6 are then determined one by one. It seems perfect but has potential risks. Assume d_{34} is noisy such that point 4 is slightly perturbed under the horizontal line. If we only look at the first four points, the layout is still acceptable because only small error exist in point 4. We then determine point 5 based on the locations of points 1,2,4, it

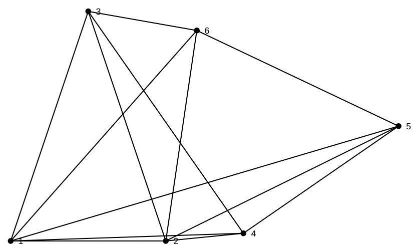

Figure 3: A simple example to illustrate the importance of computaional order

is also under horizontal line, which brings large error to this point. Finally point 6 can not be correctly determined either. The reason why small error perturbation in point 4 leads to large error in point 5 is that points 1,2,4 are almost in the same line, therefore the coefficient matrix A for determining point 5 is ill-conditioned.

Now we show how these risks disappear under our rules. The first two steps are the same, the difference is that point 6 is determined before point 5, since they both have three distances to the determined points—1,2,3,4, but point 6 is closer to them. After point 6 is determined, point 5 can also be determined stably because now four distances are used and points 1,2,4,6 make up a table base. This man-made instance shows the advantage of our strategies, which is also verified by the tests on real protein data.

4.4 Error function minimization procedures

4.4.1 Subproblems

We first introduce the concept of induced subgraph. Let $G = (V, E)$ be a graph, $V' \subseteq V$ be a subset of the vertex set, then $G(V') = (V', E')$ is the induced subgraph of G, where $E' = \{(i, j) \in E : i \in V', j \in V'\}$.

After each buildup step, we propose to add an optimization step on the subgraph $G(j \cup N(j))$, where $N(j) = \{k : d_{jk} \neq 0\}$ is the neighbourhood of point j, i.e., we further solve the following subproblem

$$\min f(x_{j \cup N(j)}), \tag{24}$$

where $f(\cdot)$ is a certain error function defined on the induced subgraph $G(j \cup N(j))$, the deviations are added on the edges in E'. We choose $f(\cdot)$ as $Stress(\cdot)$ in (4) with $\omega_{ij} = 1$ in this paper.

Notice that the objective function in (24) has the same form with the final problem but only consists of the given distances among point j and its neighbours. Since the problem is usually small-sized (no more than several tens in protein structure determination) and the buildup step produces a high-quality initial point, the subproblem (24) is usually solved within a small number of iterations to high accuracy, therefore, the computational cost brought by this optimization step is not too much.

From another point of view, the subproblem can be viewed as a tradeoff between linear and nonlinear least square in terms of the usage of distances information. As we have mentioned before, GB-LLS utilizes only l distances (too few) and GB-NLS utilizes not only given distances but also computed ones (too many), and we use a balanced approach between these two extreme methods. The benefits are obvious. We keep the advantage of GB-NLS—stability and the chance to adjust the determined points according to the freshly determined point and distances involved, and avoid the risks of accumulating old errors when use computed distances. Moreover, while matrix decomposition method works on the transformation of distance equations, our method works on squared deviations of the distances directly.

4.4.2 Solution algorithm

The derivative of of $Stress$ can be computed directed as following,

$$\frac{\partial Stress}{\partial x_i} = 2\sum_{j\in N(i)}\Big(1-\frac{d_{ij}}{\|x_i-x_j\|}\Big)(x_i-x_j), \tag{25}$$

where $N(i)$ is the neighbourhood of point i as defined before.

In this paper, we basically exploit gradient method with Barzilai-Borwein (BB) stepsize [3] to solve (25). We also incorporate with a Armijo backtracking line search [2, 29] if BB step makes no improvement in a succession of M steps to ensure the convergence.

Armijo Line search Let $\alpha = \delta\beta^i$, where $\delta > 0$ is the initial stepsize, i is the minimal nonnegative integer such that

$$f(x) - f(x+\delta\beta^i d) \geq -b\delta\beta^i d^T\nabla f(x), \tag{26}$$

where $b \in (0,1)$ is a parameter, and d is the search direction.

During the iterations, $\delta^k = \frac{\|s^{k-1}\|^2}{s^{k-1T}y^{k-1}}$ is the long BB stepsize in k-th iteration, where $s^k = x^k - x^{k-1}$, $y^{k-1} = g^k - g^{k-1}$, x^k is the iteration point and g^k is the gradient. The iteration scheme is given by

$$x^{k+1} = x^k + \alpha^k d^k. \tag{27}$$

4.5 Framework of our algorithms

Now we summarize the discussions before and give a complete description of our proposed algorithms.

Algorithm 1: Geometric Buildup-based Error Minimization (GBEM) algorithms for sparse noisy anchor-free Distance Geometry Problem

```
Step 1 Find a maximal clique that are not in the same plane, and use
       Matrix Decomposition Method (Subsection 3.1) to determine them.
       Set $\mathcal{T} = \{$index of points in the clique$\}$.

Step 2 Find index $j$ according to (21), (22) and (23). If $l = p(j) \geq 4$ and
       the points in $N(j)$ are not in the same plane, go to Step 3, go to
       Step 5 otherwise.

Step 3 Determine $x_j$ by linear least quare (Subsection 3.4.3) or nonlinear
       least square (Subsection 3.4.4), then solve (24) to adjust $x_{j \cup N(j)}$.
       Set $\mathcal{T} = \mathcal{T} \cup \{j\}$.

Step 4 If all the points are determined, go to Step 5, go to Step 2 otherwise.

Step 5 Solve (3) with variables in $\mathcal{T}$, stop.
```

According to different least square techniques we exploit in Step 3, we refer our proposed algorithms as GBEM-LLS and GBEM-NLS, respectively.

5 Numerical experiments

We have not found any open data sets about distance geometry problem. Even for the protein structure determination problem, different simulated data are used in different papers. For example, a part of the upper and lower bounds of distances under 6Å (1Å $= 10^{-10}$m) are used in DAFGL [6]. In the enhanced DISCO [16], 20% of pairwise noisy distances less than 6Å are tested, but they have incorporated distance information derived from chemistry knowledge. So it is not easy to compare our algorithms to many others because the problem settings are different and the simulated errors are added randomly. We only compare our algorithms with the original state-of-the-art Geometric Buildup algorithms in [27], especially show the ability to handle large noisy distances.

In this section, we will first introduce how we construct test problems to simulate real distance data in proteins, and then give the detailed numerical results. In addition, some conclusions and discussions are presented based on the experiment results.

5.1 Problem setting

We first download the protein data from Protein Data Bank (PDB) [4] (http://www.pdb.org), which is an archive of experimentally determined three-dimensional protein structures and contains 3D biological macromolecular structure data of proteins. According to the coordinates in the PDB files, we use *disk graph model* to construct the distance matrices, which is to say, we assume d_{ij} is known if it is less than or equal to the prescribed cutoff (usually 5Å or 6Å, where 6Å is roughly the maximum distance that NMR techniques can measure between two atoms [6]). In this way we simulate the real situation that only distances under a certain cutoff can be measured due to the limitation of biological techniques.

In real applications, the measured distances tend to be contaminated by noises, we simulate this by adding normally or uniformly distributed multiplicative perturbations to the exact distances as following,

$$d_{ij} = \bar{d}_{ij}(1 + \delta_{ij} * nl)$$

where $\bar{d}_{ij} = \|\bar{x}_i - \bar{x}_j\|$ is the exact distance between point i and point j, δ_{ij} is a random number obeying a certain given distribution. For example, $\delta_{ij} \sim N(0,1)$ if the standard normal distribution is used [6], and $\delta_{ij} \sim U[-1,1]$ if the uniform distribution is used [27]. We present results to deal with normally distributed noises in the paper, and omit the similar results for the latter case. nl is the noise level we set, which are $1\%, 5\%$, or 10% to represent low, median and high noise level, respectively.

5.2 Results and discussions

We apply GBEM-LLS and GBEM-NLS to the distance matrices. Notice that the only information we need in our algorithms is distances, and the correct coordinates of the points are only used to examine the accuracy of the results, which are measured by the Rooted Mean Squared Deviation (RMSD) value as following,

$$RMSD(X,Y) = \min_{Q,T}\{\|Y - XQ - T\|_F/\sqrt{n} : Q^{\mathrm{T}}Q = I\}, \tag{28}$$

where $X \in \mathbb{R}^{n\times 3}$, $Y \in \mathbb{R}^{n\times 3}$ are the computed coordinate matrix and true one, respectively, T is a translation vector and Q is an orthogonal matrix as the constraint states. Basically, we try to move X through rigid transformations (translations, rotations and possible reflections) to coincide with Y

as much as possible, and RMSD measures the minimal deviations. Notice that the RMSD criterion can only be used to examine the performance of the algorithm and is impractical in real applications because the ground truth coordinates Y is needed, which is unattainable for an unsolved problem. Instead, error function related criteria (such as function value or norm of the gradient) are used as stopping rules in various algorithms. One such kind of criterion is given in [6] as following,

$$LDME = \Big(\frac{1}{|E|} \sum_{(i,j) \in E} \big(\|x_i - x_j\| - d_{ij} \big)^2 \Big)^{1/2} \tag{29}$$

But just as the authors point out in paper [6], LDME (local distance matrix error) is more practical but less reliable in evaluating the true accuracy of the constructed configuration, which means smaller LDME does not necessarily mean more accurate localization, so we do not report function values in the main results tables, though we use it as a stopping criterion in the optimization step.

Our programs are written in Matlab, no MEX files are involved, though the most costly part is the gradient computation with loops, which can be speeded up by writing in C and compiling through MEX files. The programs are run in Matlab R2013b version 8.2 on a Dell PC, with 2.83 GHz CPU and 4.00 GB RAM.

We have tested GBEM-LLS and GBEM-NLS on all the 17 proteins mentioned in [6] and [27], where the number of atoms ranging from 402 to 8629. We summarize the problems information in Table 1. The first column in the table is PDB ID, which is the unique identity of a certain protein in PDB. The second column is the number of atoms in each protein. In the third column, we list the sparsity of distance matrix as [6], which is the percentage of the known distances among all the pairwise distances. However, we do not think this is an appropriate criterion to describe the difficulty of the problem. For example, if we make nine copies of a graph (and add a few necessary edges to make it a connected graph), the available distances will be about ten times more (in linear scale), but the total pairwise distance is in order n^2, i.e., about 100 times more, hence the sparsity of the distance is significantly increased, while the real difficulty to solve the problem remains almost the same, except for the time the algorithms take. This example shows that it contradicts with the intuition that the sparser the distance, the much more difficult the problem. Actually, we think sparsity is a meaningful criterion only when the number of the points in different graphs are the same or in the same order. Therefore, we also list the maximal, minimal and average degree of the graphs in the subsequent three columns. Degree

Table 1: Problems information summary

ID	Num	cutoff = 5Å				cutoff = 6Å			
			degree				degree		
		per	max	min	avr	per	max	min	avr
1PTQ	402	5.46	38	4	21.9	8.79	61	6	35.3
1HOE	558	4.05	38	6	22.6	6.55	65	11	36.5
1LFB	641	3.40	40	5	21.8	5.57	59	8	35.7
1PHT	811	3.35	48	5	27.1	5.37	75	7	43.5
1POA	914	2.51	39	4	22.9	4.07	67	8	37.2
1AX8	1003	2.30	39	5	23.0	3.74	59	7	37.5
4MBA	1083	2.17	39	5	23.5	3.56	60	5	38.5
1F39	1534	1.47	40	5	22.6	2.43	62	7	37.2
1RGS	2015	1.12	41	3	22.6	1.87	66	4	37.7
1KDH	2846	0.83	43	4	23.6	1.36	64	5	38.8
1BPM	3671	0.66	42	3	24.4	1.12	64	4	40.9
1RHJ	3740	0.65	40	4	24.4	1.10	61	5	41.2
1HQQ	3944	0.60	40	3	23.7	1.00	64	5	39.5
1TOA	4292	0.56	39	3	24.0	0.94	62	4	40.1
1MQQ	5681	0.44	44	5	25.2	0.75	66	7	42.4
1HMV	7398	0.32	42	3	23.3	0.52	67	4	38.7
1I7W	8629	0.29	48	3	24.7	0.47	73	5	40.9

ID—protein ID in PDB, Num—number of atoms in each protein, per—percentage of the known distances among all the pairwise distances, degree—maximum/minimum/average degree of all the points

information can properly reveal the difficulty of the problem, especially for our algorithms. The smaller the degrees, the few information we can exploit for each individual point, thus the more difficult the problem. The last four columns are similar, where we increase the cutoff from 5Å to 6Å, hence more distances are known.

We first compare Geometric Buildup algorithms equipped with the proposed computation order (we refer it as GBnew) with the original Geometric Buildup algorithms to show the efficiency of our strategies to define the computation order. Notice that the optimization steps are omitted in these tests. The RMSD results are reported in Table 2, where results of the original GB algorithms are copied from Table 2 and Table 3 in [27], and '-' means that the proteins are not tested by them. From the table, we can see that GBnew achieves better results than the original algorithms, especially when the data are very sparse (cutoff=5Å), the original algorithms can not generate meaningful layout for many proteins but our algorithms can. We have also tested proteins with cutoff 7Å and 8Å as [27], all the instances are deter-

Table 2: RMSD values (in Å) of simulated proteins with exact distances

ID	Num	cutoff = 5Å				cutoff = 6Å			
		LLS		NLS		LLS		NLS	
		GB	GBnew	GB	GBnew	GB	GBnew	GB	GBnew
1PTQ	402	1.4e+00	6.5e−13	5.5e−14	5.1e−14	2.6e−09	2.8e−14	5.0e−14	2.9e−14
1HOE	558	5.8e−02	3.0e−13	1.6e−13	2.4e−14	3.1e−09	3.8e−14	2.7e−13	2.0e−14
1LFB	641	2.0e−02	9.3e−14	9.5e−14	3.1e−14	2.1e−10	4.6e−14	5.5e−14	2.1e−14
1PHT	811	1.2e+01	2.0e−12	1.1e−13	6.3e−14	8.2e−09	9.3e−14	1.8e−13	5.7e−14
1POA	914	6.6e+00	8.2e−13	3.2e−13	7.4e−14	1.9e−09	2.8e−13	1.5e−13	4.7e−14
1AX8	1003	5.2e+00	1.1e−11	4.0e−13	3.3e−14	1.8e−05	5.4e−14	4.6e−12	3.0e−14
4MBA	1083	4.9e+00	3.6e−12	1.8e−13	8.2e−14	3.8e−06	1.2e−13	2.6e−13	6.9e−14
1F39	1534	1.4e+00	6.7e−13	7.9e−13	5.2e−14	6.3e−08	2.1e−13	1.9e−13	4.7e−14
1RGS	2015	2.0e+01	2.5e−10	8.3e−12	2.2e−13	1.1e−01	2.1e−12	2.4e−12	1.7e−13
1KDH	2846	-	7.2e−11	-	1.9e−13	-	5.6e−13	-	1.5e−13
1BPM	3671	6.4e+04	9.6e−10	8.1e−11	1.3e−13	3.6e−02	3.3e−13	1.0e−11	1.0e−13
1RHJ	3740	-	3.7e−09	-	6.4e−14	-	3.9e−13	-	8.2e−14
1HQQ	3944	-	1.3e−09	-	5.3e−14	-	9.0e−13	-	6.8e−14
1TOA	4292	-	3.4e−09	-	9.5e−14	-	4.9e−12	-	2.2e−13
1MQQ	5681	-	6.8e−11	-	1.6e−13	-	6.1e−13	-	6.8e−14
1HMV	7398	1.2e+03	6.1e−07	1.1e−08	3.8e−13	3.5e+01	5.6e−11	5.5e−07	6.0e−13
1I7W	8629	-	2.0e−06	-	3.8e−13	-	6.4e−12	-	2.7e−13

ID—protein ID in PDB, Num—number of atoms in each protein, LLS—Linear Least Square, NLS—Nonlinear Least Square, GB—the original Geometric Buildup algorithms, GBnew—Geometric Buildup algorithms with the proposed computation order

mined accurately, even for those they can not determine. Since these settings are not practical in real problems, we do not report the details. Moreover, the increased CPU time are negligible compared to the original algorithms. For instance, a 8629 atoms protein—1I7W with cutoff 6Å is determined by GBnew-LLS in 19.0 seconds.

Before presenting the numerical results on large noisy distances, we have to point out that the results are related to the parameter settings in the algorithms. For example, different maximum iteration number or different small tolerance (for example 10^{-2} or 10^{-3}) the gradient achieves usually give different RMSD and CPU time results. In most of the cases, smaller tolerance or larger maximum iteration number gives more accurate (i.e., smaller RMSD) results, but takes longer time. One counterexample we encounter is when we compute 1MQQ with cutoff 5Å and noise level 5%, we increase the tolerance from 10^{-2} to 10^{-3}, then we obtain much smaller function value, but larger RMSD, and of course the time it takes is longer. Usually we can always adjust the parameters to get better results for one particular protein, but we do not do that because for real problems, what we can do is setting the parameters and obtain results, not the opposite. Throughout the paper, the results are generated by uniform parameter settings in all the noisy cases.

We report our first main numerical results on noisy distances in Table

Table 3: RMSD (in Å) and CPU time (in seconds) of simulated proteins with cutoff = 5Å, noise level = 1% and 5%

ID	Num	nDet	noise level = 1%				noise level = 5%			
			LLS		NLS		LLS		NLS	
			RMSD	CPU	RMSD	CPU	RMSD	CPU	RMSD	CPU
1PTQ	402	402	1.1e−01	1.0	6.0e−02	1.1	2.0e−01	2.1	2.1e−01	2.2
1HOE	558	558	1.6e−01	1.7	3.7e−02	1.6	1.9e−01	3.6	2.6e+00	4.1
1LFB	641	641	1.6e−01	1.9	9.6e−02	1.9	5.9e−01	3.1	3.6e+00	4.7
1PHT	811	806	2.2e−01	2.6	6.1e−02	3.0	3.2e−01	6.0	1.9e−01	5.8
1POA	914	914	2.0e−01	2.9	8.8e−02	2.7	2.5e−01	6.4	4.1e−01	5.9
1AX8	1003	1003	2.3e−01	3.8	7.9e−02	3.3	4.3e−01	8.8	8.0e+00	10.8
4MBA	1083	1080	2.1e−01	3.6	1.5e−01	4.3	2.6e−01	9.2	2.4e+00	10.7
1F39	1534	1534	3.4e−01	5.2	1.3e−01	5.0	6.3e−01	12.7	5.1e−01	9.5
1RGS	2015	2010	9.5e−01	10.2	1.8e−01	7.2	2.9e+00	17.7	7.9e+00	20.9
1KDH	2846	2846	3.4e−01	17.7	2.3e−01	17.0	2.6e+00	28.5	1.4e+01	34.6
1BPM	3671	3668	4.5e−01	23.3	9.5e−02	16.4	7.4e−01	35.4	2.6e−01	32.7
1RHJ	3740	3740	1.3e−01	18.2	9.4e−02	18.5	4.4e−01	36.8	1.4e+01	50.5
1HQQ	3944	3938	1.7e−01	19.0	1.0e−01	17.5	4.3e−01	37.2	4.5e−01	34.6
1TOA	4292	4280	4.0e−01	29.4	1.2e−01	22.4	1.0e+00	43.6	5.1e−01	34.4
1MQQ	5681	5681	3.3e−01	47.8	1.0e−01	35.0	5.1e+00	70.2	3.6e+00	73.0
1HMV	7398	7389	2.6e+00	70.7	3.8e−01	54.3	3.7e+00	83.0	9.7e+00	100.3
1I7W	8629	8624	1.9e+00	79.0	1.0e+00	69.8	2.3e+01	109.4	6.2e+00	121.1

ID—protein ID in PDB, Num—number of atoms in each protein, nDet—number of determined atoms by our algorithms, LLS—GBEM with Linear Least Square, NLS—GBEM with Nonlinear Least Square, RMSD—RMSD values of computed structures against original ones, CPU—toal CPU time the algorithms cost

3, where we set cutoff to be 5Å, and noise level to be 1% and 5%. Notice that nDet (number of determined atoms) column is a little smaller than Num column for some proteins, which means our algorithm can not determine all the atoms for these proteins, but only leave out very few, because the distance matrix is so sparse such that these atoms have at most three distances to the determined atoms, thus produces at least two options for the locations and can not be determined uniquely. The RMSD and CPU time results are reported only for the determined points.

Recall that the cutoff equals to 5Å, the layout is very accurate when RMSD (roughly the average deviation of all the atoms) is in order of 10^{-1} or even smaller, is not so good when it is larger than about 2Å, and is totally wrong when it is in order of 10^1 or larger. From the table, we find that GBEM-NLS generates more accurate configurations than GBEM-LLS when noise level is low—1%, which is the opposite when noise level is increased to

5%, where NLS inherits too much errors from previous steps when computing distances to make the subgraph complete. In a word, our algorithms can determine most of the proteins accurately but they also give wrong results in very few cases because the distances are too sparse and the noises are too large.

In term of CPU time the algorithms cost, both algorithms only take a couple of seconds to determine several thousands of points. Even for the two largest proteins with 7398 and 8629 atoms, they only take about one hundred seconds, which is much faster than SDP based algorithms [6]. Another observation is that GBEM-NLS takes shorter time to stop than GBEM-LLS if both algorithms have good results.

Table 4: RMSD (in Å) and CPU time (in seconds) of simulated proteins with cutoff = 6Å, noise level = 1%, 5% and 10% (All the points are determined)

ID	Num	noise level = 1%				noise level = 5%				noise level = 10%			
		LLS		NLS		LLS		NLS		LLS		NLS	
		RMSD	CPU	RMSD	CPU	RMSD	CPU	RMSD	CPU	RMSD	CPU	RMSD	CPU
1PTQ	402	4.1e−02	1.3	3.4e−02	2.2	1.6e−01	2.8	1.9e−01	4.3	3.1e−01	4.2	3.2e−01	5.8
1HOE	558	2.8e−02	1.9	2.3e−02	3.1	1.1e−01	4.2	1.0e−01	6.3	2.1e−01	5.9	2.0e−01	9.1
1LFB	641	4.2e−02	2.2	3.8e−02	3.5	1.6e−01	4.8	1.7e−01	8.6	3.3e−01	7.3	3.6e−01	8.7
1PHT	811	1.3e−01	3.4	4.8e−02	6.7	2.2e−01	7.3	1.6e−01	14.7	4.8e−01	10.4	4.7e−01	14.4
1POA	914	4.2e−02	3.3	3.7e−02	5.8	1.7e−01	7.4	4.6e−01	14.1	5.0e−01	10.0	5.3e−01	13.9
1AX8	1003	5.8e−02	4.0	3.7e−02	6.2	1.3e−01	8.4	1.2e−01	11.9	2.3e−01	11.9	2.9e−01	16.2
4MBA	1083	6.4e−02	4.8	2.9e−02	7.0	1.7e−01	9.2	1.2e−01	13.0	2.6e−01	13.4	2.4e−01	19.1
1F39	1534	4.6e−02	6.2	3.9e−02	9.7	2.5e−01	14.0	1.8e−01	19.3	6.1e−01	18.1	3.5e−01	24.0
1RGS	2015	1.2e−01	8.6	5.1e−02	13.6	2.9e−01	18.4	2.8e−01	27.2	6.0e−01	29.1	4.5e−01	37.3
1KDH	2846	1.0e−01	15.2	5.3e−02	22.7	1.6e−01	29.0	1.9e−01	40.9	3.9e−01	43.0	1.2e+00	51.2
1BPM	3671	9.4e−02	20.5	3.7e−02	34.0	1.5e−01	41.4	1.4e−01	67.8	2.4e−01	56.0	2.8e−01	73.0
1RHJ	3740	8.1e−02	22.0	2.8e−02	35.0	1.3e−01	39.9	1.2e−01	59.0	2.5e−01	58.3	2.8e−01	72.2
1HQQ	3944	5.8e−02	20.5	3.7e−02	36.2	1.8e−01	43.0	1.9e−01	63.4	3.4e−01	60.3	3.6e−01	82.7
1TOA	4292	8.4e−02	24.6	4.6e−02	45.3	1.8e−01	48.2	1.5e−01	72.3	5.6e−01	72.8	4.2e−01	89.2
1MQQ	5681	4.0e−02	35.5	3.2e−02	64.8	1.1e−01	67.7	1.2e−01	109.1	2.3e−01	94.4	2.7e−01	129.0
1HMV	7398	1.4e−01	57.7	1.0e−01	91.6	3.0e−01	105.9	4.0e−01	146.0	1.2e+00	132.5	5.8e−01	189.5
1I7W	8629	2.9e−01	64.5	9.3e−02	111.7	7.9e−01	120.4	3.7e−01	175.6	3.8e+00	165.7	6.7e−01	225.1

ID—protein ID in PDB, Num—number of atoms in each protein, LLS—GBEM with Linear Least Square, NLS—GBEM with Nonlinear Least Square, RMSD—RMSD values of computed structures against original ones, CPU—toal CPU time the algorithms cost

Numerical tests on proteins with cutoff 6Å are given in Table 4, different noise levels are considered. All the atoms are determined because the minimal degree is larger than or equal to four. From the table we can see than GBEM-NLS determines all the proteins very accurately, even when the noise level is increased up to 10%. GBEM-LLS determines most of the proteins accurately, except for 1HMV and 1I7W when noise level is 10%, where the localization quality is acceptable but not very good. As to the computation time, GBEM-NLS takes longer time in all the cases because many distances need be computed in the buildup step.

Based on Table 3 and Table 4, we make the following several conclusions. Our algorithms are fast and accurate in most cases of protein structure determination. They are robust to the noises, especially when the distances are relatively dense–cutoff is 6Å (still very sparse as a whole). GBEM-NLS is recommended for high accuracy localization when the data is relatively dense, otherwise GBEM-LLS is recommended.

At the end of this section, we give two lists of objective function values in Table 5 which give us some insight on the efficiency of our algorithms. One is *Fcomp*—function values when GBEM-NLS stops, the other is *Ftrue*—function values that we put the true coordinates into the objective function. Since the distances are noisy, *Fture* are not equal to 0. It is really surprising that *Fcomp* are smaller than *Ftrue* except for 1KDH, where the configuration is not very accurate. These results indicate that what we done is almost optimal, because the computed function values are even smaller than "true" ones, which means that it is almost the best we can do if no more information is considered except noisy distances. Significantly better results can be expected only when we change the models, which is as interesting direction for future work.

Table 5: Comparison of computed function values with the "true" values (cutoff = 6Å, noise level = 10%)

ID	1PTQ	1HOE	1LFB	1PHT	1POA	1AX8	4MBA	1F39	1RGS
Fcomp	308.44	430.74	498.13	786.12	746.17	804.56	900.50	1256.29	1673.50
Ftrue	369.81	515.73	593.65	906.38	878.44	958.83	1069.63	1497.36	1984.83

ID	1KDH	1BPM	1RHJ	1HQQ	1TOA	1MQQ	1HMV	1I7W
Fcomp	2844.25	3437.47	3540.00	3528.73	3906.33	5503.48	6564.94	7934.25
Ftrue	2839.00	4019.67	4120.30	4147.49	4560.61	6384.25	7524.43	9262.98

Fcomp—function values at the optimal solution of GBEM-NLS, Ftrue—function values that put the true coordinates into the objective function

6 Discussions and extensions

We have proposed two algorithms to solve anchor-free distance geometry problems with noisy distances. Though our algorithms and numerical tests are designed for 3-dimensional protein structure determination problems, they can be extended to any other dimensional anchor-free problems directly, for example, 2-dimensional anchor-free wireless sensor network localization problems, and dimension reduction problems in high-dimensional space. The Euclidean distances used by this paper can also be extended to other distance-like measurements, such as similarity/dissimilarity of two items, geodetic distances in manifolds, shortest path between two points in a graph [30], etc. As to the very sparse case (5Å in our tests), a rigid framework exploiting binary search is studied in [36] to further handle the small number of undetermined atoms, which can also be incorporated in our algorithms, but we do not bother to include this feature to our code currently.

Consider the distance geometry problems with anchors, where the locations of some points are known as prior knowledge, we believe this is good news to our algorithms because they are useful for the control of error accumulation. There are no essential theoretical difficulties to exploit these information, but the codes have to be changed to set these points as fixed determined points, we plan to finish this when we have time.

Currently we can only deal with noisy distances, rather than distance bounds considered in [6, 16, 26, 32], where usually an assemble of configurations exist to satisfy the given bounds. How to extend our algorithms to handle this general case is still an open problem to us, which is our future work. One possible way is to consider the average distance value of a pair of upper and lower bound, but it is not good when one bound is close to the real distance and another is far away from. In [26, 32], a generalized model is developed, which gives each point a maximal perturbation radius hence the feasible problem with multiple solutions is transformed to an deterministic optimization problem. In addition, a buildup-fashion algorithm is proposed to approximately solve the model, how to improve the efficiency of the algorithm is also an interesting research direction.

7 Acknowledgement

We greatly thank Prof. Zhijun Wu from Iowa State University for the discussion on analysis of error accumulation when he visited Chinese Academy of Sciences in the summer of 2014.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[2] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[3] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[5] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *Automation Science and Engineering, IEEE Transactions on*, 3(4):360–371, 2006.

[6] P. Biswas, K. C. Toh, and Y. Ye. A distributed sdp approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *Siam Journal on Scientific Computing*, 30(3):1251–1277, 2008.

[7] L. Blumenthal. Theory and applications of distance geometry, 1953.

[8] S. P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM, 1994.

[9] W. Braun. Distance geometry and related methods for protein structure determination from nmr data. *Quarterly reviews of biophysics*, 19(3-4):115–157, 1987.

[10] C.-Y. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[11] G. M. Crippen and T. F. Havel. *Distance geometry and molecular conformation*, volume 74. Research Studies Press Taunton, UK, 1988.

[12] Q. F. Dong and Z. J. Wu. A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization*, 22(1-4):365–375, 2002.

[13] Q. F. Dong and Z. J. Wu. A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. *Journal of Global Optimization*, 26(3):321–333, 2003.

[14] R. dos Santos Carvalho, C. Lavor, and F. Protti. Extending the geometric build-up algorithm for the molecular distance geometry problem. *Information Processing Letters*, 108(4):234–237, 2008.

[15] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[16] X. Fang and K.-C. Toh. *Using a distributed SDP approach to solve simulated protein molecular conformation problems*, pages 351–376. Springer, 2013.

[17] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Graph Drawing*, pages 239–250. Springer.

[18] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, 2014.

[19] G. Mao, B. Fidan, and B. Anderson. Wireless sensor network localization techniques. *Computer networks*, 51(10):2529–2553, 2007.

[20] J. J. More and Z. J. Wu. Global continuation for distance geometry problems. *Siam Journal on Optimization*, 7(3):814–836, 1997.

[21] J. J. More and Z. J. Wu. Distance geometry optimization for protein structures. *Journal of Global Optimization*, 15(3):219–234, 1999.

[22] H.-D. Qi and X. Yuan. Computing the nearest euclidean distance matrix with low embedding dimensions. *Mathematical Programming*, pages 1–39, 2012.

[23] J. B. Saxe. *Embeddability of weighted graphs in k-space is strongly NP-hard.* Carnegie-Mellon University, Department of Computer Science, 1979.

[24] I. J. Schoenberg. Remarks to maurice frechet's article"sur la definition axiomatique d'une classe d'espace distances vectoriellement applicable sur l'espace de hilbert. *Annals of Mathematics*, pages 724–732, 1935.

[25] D. Shamsi, N. Taheri, Z. Zhu, and Y. Ye. On sensor network localization using sdp relaxation. *arXiv preprint arXiv:1010.2262*, 2010.

[26] A. Sit and Z. Wu. Solving a generalized distance geometry problem for protein structure determination. *Bull Math Biol*, 73(12):2809–36, 2011.

[27] A. Sit, Z. Wu, and Y. Yuan. A geometric buildup algorithm for the solution of the distance geometry problem using least-squares approximation. *Bull Math Biol*, 71(8):1914–33, 2009.

[28] A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. *Mathematical Programming*, 109(2-3):367–384, 2006.

[29] W. Sun and Y.-X. Yuan. *Optimization theory and methods: nonlinear programming*, volume 1. springer, 2006.

[30] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[31] W. S. Torgerson. Theory and methods of scaling. 1958.

[32] Z. Voller and Z. Wu. *Distance geometry methods for protein structure determination*, pages 139–159. Springer, 2013.

[33] Z. Wang, S. Zheng, Y. Ye, and S. Boyd. Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM Journal on Optimization*, 19(2):655–673, 2008.

[34] S. Wright and J. Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[35] D. Wu and Z. Wu. An updated geometric build-up algorithm for solving the molecular distance geometry problems with sparse distance data. *Journal of Global Optimization*, 37(4):661–673, 2006.

[36] D. Wu, Z. J. Wu, and Y. X. Yuan. The solution of the distance geometry problem in protein modeling via geometric buildup. *Biomat 2007*, pages 50–88, 2008.

[37] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.