The Isomap Algorithm in Distance Geometry

Leo Liberti[1] and Claudia D'Ambrosio[2]

1 **CNRS LIX Ecole Polytechnique, Palaiseau, France**
liberti@lix.polytechnique.fr
2 **CNRS LIX Ecole Polytechnique, Palaiseau, France**
dambrosio@lix.polytechnique.fr

Abstract

The fundamental problem of distance geometry consists in finding a realization of a given weighted graph in a Euclidean space of given dimension, in such a way that vertices are realized as points and edges as straight segments having the same lengths as their given weights. This problem arises in structural proteomics, wireless sensor networks, and clock synchronization protocols to name a few applications. The well-known Isomap method is a dimensionality reduction heuristic which projects finite but high dimensional metric spaces into the "most significant" lower dimensional ones, where significance is measured by the magnitude of the corresponding eigenvalues. We start from a simple observation, namely that Isomap can also be used to provide approximate realizations of weighted graphs very efficiently, and then derive and benchmark six new heuristics.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Theory, F.2.1 Numerical Algorithms and Problems, J.3 Life and Medical Sciences

Keywords and phrases distance geometry problem, protein conformation, heuristics

Digital Object Identifier 10.4230/LIPIcs.SEA.2017.5

1 Introduction

The fundamental problem in Distance Geometry (DG) is as follows.

> Distance Geometry Problem (DGP). Given an integer $K \geq 1$ and a simple, edge-weighted, undirected graph $G = (V, E, d)$, where $d : E \to \mathbb{R}_+$, determine whether there exists *realization function* $x : V \to \mathbb{R}^K$ such that:
>
> $$\forall \{i, j\} \in E \quad \|x_i - x_j\| = d_{ij}. \tag{1}$$

The DGP arises in many applications, for various values of K. Two important applications for $K = 3$ are the determination of protein structure from distance data [36], and the localization of a fleet of unmanned submarine vehicles [2]. The localization of mobile sensors in a wireless network is a well-studied application for the case $K = 2$ [11, 5, 16, 9]. The only engineering application we are aware of for the case $K = 1$ is to clock synchronization protocols in computer networks [32]. Although Equation (1) is actually a schema (since the norm is unspecified), most of the literature about the DGP uses the Euclidean norm (or 2-norm) [21, 19], which is also the focus of this paper. In this context, the name of the problem is Euclidean DGP (EDGP).

It is worth mentioning that, although the system of equations in Equation (1) involves square roots, the squared system

$$\forall \{i, j\} \in E \quad \|x_i - x_j\|^2 = d_{ij}^2. \tag{2}$$

16th International Symposium on Experimental Algorithms (SEA 2017).
Editors: Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman; Article No. 5; pp. 5:1–5:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

has the same set of solutions as Equation (1) and is a polynomial system of degree two [12]. This makes it amenable to be studied using methods of algebraic geometry, for example [35]. It was shown in [29] that the EDGP is **NP**-hard, by reduction from Partition to the case $K = 1$. Another proof for $K = 2$ was sketched in [38], and further proofs for some variants in $K = 3$ and general K were given in [17] and [20]. If the dimensionality K is not given in advance, then the question is whether the given graph admits a realization in some dimension K. This problem is known as Euclidean Distance Matrix Completion Problem (EDMCP). This difference between being given a K as part of the input or obtaining K as part of the output is considerable: while the EDGP is **NP**-hard, we do not know whether the EDMCP is **NP**-hard or in **P** (or neither, assuming $\mathbf{P} \neq \mathbf{NP}$).

Isomap [34] is a well-known dimension reduction algorithm which is able to project a set $X \subset \mathbb{R}^n$ of high dimensional points belonging to a low-dimensional manifold to its intrinsic dimension (say, K).

In this paper, we describe an easy adaptation of the Isomap algorithm to solve the EDGP. The rationale for using Isomap on the EDGP is that finding realizations in high dimensional spaces is empirically easier than in a given dimension K. We then describe six heuristics for the EDGP based on Isomap, and evaluate them computationally on a test set consisting of protein instances of different sizes.

It is often remarked that the EDGP and EDMCP only serve as abstract models for real-life applications, since in most engineering and biological settings only interval estimates or distributions of the distances are known (rather than exact distance values). Although we do not treat the case of intervals or distributions here, we note that it is at least theoretically possible to extend our heuristic methods to the interval case without excessive trouble — the simplest way to do so is to run the same heuristics using the distribution average on each interval. A better approach would replace error measures based on exact distances by corresponding measures in intervals [21].

The rest of this paper is organized as follows. In Section 2, we give a very brief account of the history of DG, introducing some of the concepts which we shall use later on in the paper. In Section 3 we describe the Isomap algorithm and its relationship to EDMCP and EDGP. In Section 4 we define and motivate our new heuristics based on Isomap. Our computational results are discussed in Section 5.

2 A very short history of DG

DG was formally introduced by Karl Menger in [24, 25], at a time when, under Hilbert's drive [14], the concerted effort of many mathematicians (specially from *Mitteleuropa*) pushed towards the axiomatization of mathematics in general, and specifically of geometry [18]. Menger and some of his students (Gödel among them) were part of the Vienna Circle, but when this became politicized, Menger founded his famous *mathematisches Kolloquium*, which ran at the University of Vienna between 1928 and 1937. It is interesting that the only co-authored paper published by Gödel appears in the proceedings of Menger's *Kolloquium*, and is about DG [18, 22]. Menger's foundational work is an axiomatization of geometry which puts metric spaces at its core (e.g. convexity can be defined via *betweenness* of points). Its main achievement is to characterize the metric spaces according to the dimension of the Euclidean spaces they can be realized in. Menger's work was continued by his student Blumenthal [7], but remained firmly in the domain of pure mathematics.[1]

[1] Another useful application dating from ancient Greece was Heron's formula for computing the area of a triangle from the lengths of its sides, extensively used in agricultural measurements.

A finite metric space (V, d) is a finite set V with an associated metric d. It is usually represented as a weighted complete graph or a distance matrix. The graph is simple, undirected and edge-weighted, say $G = (V, E, d)$ where $E = \{\{i, j\} \mid i < j \in V\}$ and $d : E \to \mathbb{R}_+$ such that $d(i, j)$ is the value of the metric on the edge $\{i, j\}$ of the underlying set V. The distance matrix is an $n \times n$ symmetric matrix D with zero diagonal, where $n = |V|$, such that the component d_{ij} is the value of the metric defined on i and j, for all $i < j \in V$. Given some positive integer K, a metric space (V, d) is *realized* in the Euclidean space $\mathbb{R}^K$ w.r.t. $\|\cdot\|$ if there exists a realization function from V to $\mathbb{R}^K$ w.r.t. $\|\cdot\|$. The main problem in DG, for Menger and Blumenthal, was that of categorizing finite metric spaces (V, d) according to the integers K such that V can be realized in $\mathbb{R}^K$.

A note [31] written by Schoenberg's in 1935 on a paper by Fréchet showed that any Euclidean Distance Matrix (EDM) $D = (d_{ij})$, i.e. when the metric is the 2-norm, can be efficiently transformed into the Gram matrix of a Euclidean realization of the underlying metric. Since a matrix X is Gram if and only if it is Positive Semidefinite (PSD), and since PSD matrices can be factored as $X = xx^\top$ where x is a matrix of rank $K \leq n$, this offers a method for finding a realization of D in $\mathbb{R}^K$ [33]. This result was subsequently adapted to work on wrong or approximate EDMs by replacing negative eigenvalues of D by zero, and resulted in the hugely successful *multidimensional scaling* (MDS) method [8]. A further refinement, obtained by using only at most K positive eigenvalues of D, called *principal component analysis* (PCA) was equally successful. This firmly establishes DG as a branch not only of pure, but also of applied mathematics.

The first explicit mention of the DGP appears to arise in a 1978 paper by Yemini [37], which calls the reader's attention to the problem of finding a realization in the plane of a set of mobile sensors where the distances are only known if two sensors are close enough.

3 The Isomap method in Distance Geometry

The Isomap algorithm projects a finite subset of points $X \subset \mathbb{R}^n$ to $\mathbb{R}^K$ (for some positive given $K < n$) as follows:

1. it computes all pairwise distances for X, yielding the distance matrix D
2. it selects a subset d of "short" Euclidean distances in D (usually up to a given threshold), yielding a simple connected weighted graph $G = (V, E, d)$ where $d : E \to \mathbb{R}_+$;
3. it computes all shortest paths in G, and produces an approximate distance matrix $\tilde{D}$, where $\tilde{D}_{ij} = d_{ij}$ for all $\{i, j\} \in E$ and $\tilde{D}_{ij}$ is the value of the shortest path from i to j otherwise;
4. it derives a corresponding approximate Gram matrix $\tilde{B}$ by setting

$$\tilde{B} = -\frac{1}{2} J \tilde{D}^2 J, \tag{3}$$

   where $J = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$;
5. it finds the (diagonal) eigenvalue matrix Λ of $\tilde{B}$ and the corresponding eigenvector matrix P, so that $\tilde{B} = P^\top \Lambda P$;
6. since $\tilde{B}$ is only an approximation of a Gram matrix, it might have some negative eigenvalues: Isomap replaces all the negative eigenvalues with zeroes;
7. in case there are still more than K positive eigenvalues, Isomap replaces the smallest ones, leaving only the largest K eigenvalues on the diagonal of a PSD matrix $\tilde{\Lambda}$;
8. finally, it sets $x = P^\top \sqrt{\tilde{\Lambda}}$.

Steps 4–8 are collectively known as PCA. Without Step 7, they are known as classic MDS [15].

3.1 Isomap and the EDMCP

How can Isomap apply to the EDGP? A simple explanation is as follows: solving the EDGP is hard, but solving the EDMCP is not as hard, and provides a realization x' of G in (generally) more than K dimensions, say in $\mathbb{R}^n$. At this point, Isomap could be applied to x' and give an approximate projection in $\mathbb{R}^K$.

Although it was mentioned in Section 1 that no-one knows yet whether the EDMCP is **NP**-hard or in **P**, that statement refers to the usual definition of these complexity classes in the the Turing Machine (TM) computational model. On the other hand, the fact that the EDMCP can be solved efficiently in practice can be made more precise.

▶ **Theorem 1.** *The EDMCP can be solved in a polynomial number of basic steps in the Real RAM computational model [6].*

Proof. We first show that the EDMCP can be described by the following pure feasibility Semidefinite Program (SDP):

$$\left.\begin{array}{rcl} \forall\{i,j\} \in E \quad X_{ii} + X_{jj} - 2X_{ij} & = & d_{ij}^2 \\ X & \succeq & 0. \end{array}\right\} \tag{4}$$

Assume Equation (4) has a solution X^* for a given EDMCP instance. Then, since X^* is a PSD matrix, it is also a Gram matrix, which means that it can be factored as $X^* = YY^\top$. Consider a realization $x^* \in \mathbb{R}^n$ given by $x_i^* = Y_i$ for each $i \in V$, where Y_i is the i-th row of Y. Then we have

$$\|x_i^* - x_j^*\|^2 = \|Y_i - Y_j\|^2 = {Y_i}^\top Y_i + {Y_j}^\top Y_j - 2Y_iY_j = X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2$$

by the linear constraints in Equation (4). This means that Y is a valid realization for G in $\mathbb{R}^n$, i.e. the given EDMCP instance is YES. Assume now that Equation (4) is infeasible, but suppose that the given instance is YES: then it has a realization Y of G in $\mathbb{R}^n$, and it is immediate to verify that its Gram matrix $X^* = YY^\top$ satisfies Equation (4) providing a contradiction, so the given EDMCP instance must be NO, which concludes the first part of the proof.

Having established that solving the EDMCP is the same as solving Equation (4), we remark that the Interior Point Method (IPM) can be used to solve SDPs in polynomial time to any desired accuracy [1] in the TM computational model. If a primal path-following IPM based on Newton's steps could be run on a Real RAM machine, it would find an exact real solution for the EDMCP. ◀

In practice, the IPM can only compute approximate solutions to Equation (4) in floating point precision, which might not satisfy the constraints exactly. But SDP technology can be used in practice to solve EDMCPs efficiently to very good approximations.

3.2 Isomap and the EDGP

Although our basic idea is to solve an EDMCP instance in order to find a high dimensional realization of G as a pre-processing step to applying Isomap, it is easy to streamline this procedure better. We observe that Step 3 requires a weighted graph G as input, and that a weighted graph is part of the definition of any EDGP instance. It is therefore sufficient to start Isomap from Step 3. The *Isomap for DG* works as follows.

(A) Run the Floyd-Warshall all-shortest-paths algorithm [23] on the partial distance matrix D^G represented by $G = (V, E, d)$ and obtain $\tilde{D}$, a *completion* of D^G;

(B) find the (approximate) Gram matrix $\tilde{B}$ of $\tilde{D}$;
(C) find the PSD matrix B' closest to $\bar{B}$ by zeroing the negative eigenvalues, and then perform PCA to extract an approximate realization x in $\mathbb{R}^K$.

The interest in using the Isomap for DG is that it lends itself to the construction of many heuristics, through its combination with various pre- and post-processing algorithms. For example, Step 1, which essentially aims at solving an EDMCP instance by completing the corresponding graph using shortest paths, can be replaced by the solution of the SDP in Equation (4). Moreover, the final solution x obtained in Step 3 can be used as a starting point by a local Nonlinear Programming (NLP) solver.

4 Isomap heuristics

We list in this section six new heuristics based on Isomap for solving EDGPs.

(i) Isomap. This is the Isomap algorithm for DG as described in Section 3.2.
(ii) IsoNLP. This variant adds a post-processing phase consisting of a local NLP solver to improve the output of Isomap (see Section 4.1 below). Because of the importance of this phase, every following heuristic also uses it.
(iii) SPT. This variant, the name of which stands for *spanning tree*, replaces Step 1 of the Isomap algorithm definition given in Section 3.2 as follows: compute a realization $x' \in \mathbb{R}^K$ using a spanning tree of G (see Section 4.2 below for details). This realization is then used to obtain the EDM $\tilde{D}$. SPT also adds a post-processing local NLP solution phase.
(iv) SDP. This variant replaces Step 1: solve Equation (4) using a "natural" SDP formulation (see Section 4.3 below), obtain a realization x' in $\mathbb{R}^n$, and use it to compute the EDM $\tilde{D}$. SDP also adds a post-processing local NLP solution phase.
(v) Barvinok. This variant is similar to SDP, but it endows the SDP with an objective function designed to decrease the rank of the solution x' to $p = O(\sqrt{|E|})$ (see Section 4.4 below). Barvinok also adds a post-processing local NLP solution phase.
(vi) DGSol. This variant uses one of the first modern algorithms for solving EDGPs, `dgsol` [26], to compute an initial realization x' in $\mathbb{R}^K$, which it then uses to compute the EDM $\tilde{D}$ (see Section 4.5 below). DGSol also adds a post-processing local NLP solution phase.

4.1 Post-processing using a local NLP solver (IsoNLP)

Although Isomap can be used on its own, the quality of the realizations it obtains is greatly improved when the output is used as a starting point for a local NLP algorithm, such as active set or barrier algorithms [13]. Aside from Isomap, the rest of our heuristics all include this post-processing phase.

In the case of the SDP and Barvinok heuristics, this post-processing is backed by a theoretical result given in [4], also exploited in [10], which states that there is an SDP solution of Equation (4) which is asymptotically not too far from the manifold of solutions of Equation (2): hence, it makes sense to try a single local descent to reach that manifold.

The choice of solver was carried out through some preliminary computational experiments. Since most tests were carried out in Python, we limited ourselves to solvers offering a Python API. Among these, we decided to use the one which proved out to be *empirically* fastest, namely `lbfgs` from `scipy.optimize`, limited to 50 iterations. Code optimization might change this choice, specially in view of the fact that we only employed Python-enabled solvers, with APIs that shine more for ease of coding than efficiency.

4.2 Spanning tree realization heuristic (SPT)

A possible way to construct an approximate distance matrix $\tilde{D}$ based on an EDGP instance G consists in identifying a spanning subgraph of G for which the EDGP can be solved efficiently, and then use the corresponding realization to compute $\tilde{D}$. We use trees, which are a polynomial case of the EDGP.

Let T be a tree on V, and for each $v \in V$ let $N_T(v)$ be the set of vertices adjacent to v in T. The following algorithm realizes any tree in $K = 1$, and more specifically in the non-negative half-line $\mathbb{R}_+$.

1. Let r be a vertex with highest degree in G;
2. let $x_r = 0$;
3. let $Q = \{r\}$ be a priority queue containing vertices with their degrees w.r.t. $V \smallsetminus Q$ as priority;
4. pop the vertex u with highest priority from Q;
5. for each $v \in N_T(u)$ let $x_v = x_u + d_{uv}$ and add v to Q.

▶ **Lemma 2.** *The above algorithm is correct and runs in linear time.*

Proof. The important invariant of the algorithm is that every vertex u entering Q has a known realization x_u: this holds by Step 5 and because at the first iteration Q only contains r, realized at $x_r = 0$. It is also easy to see that, by connectedness of G, every vertex in V enters Q at least once. Moreover, since G is a tree, it has no cycles, which implies that no vertex can ever enter Q more than once. Since every vertex enters Q exactly once, the complexity of this algorithm is $\Theta(|V|)$. ◀

Once a tree is realized in a half-line, one can embed the realization in as many dimensions as needed, by embedding a congruent copy of the half-line in an appropriate Euclidean space.

The algorithm we use to construct a realization in $\mathbb{R}^K$ from a tree is based on the above one. It takes a general graph G as input, grows a largest-degree priority spanning tree, and realizes each vertex v as a uniformly chosen random point on the sphere centered at x_u with radius d_{uv} for each edge $\{u, v\}$ in the spanning tree. More precisely, we modify the above algorithm as follows: (a) we introduce a set Z (initialized to $\{r\}$) that records the vertices entering the tree and replace $N_T(u)$ by $N_G(u) \smallsetminus Z$; (b) we replace $x_v = x_u + d_{uv}$ by x_v sampled uniformly at random from $S^{K-1}(x_u, d_{uv})$.

4.3 Euclidean distance SDP objective (SDP)

Another way to compute $\tilde{D}$ is to solve the SDP in Equation (4), and obtain a realization Y having rank generally higher than K; $\tilde{D}$ is then set to the EDM corresponding to Y. IPM algorithms for SDP offer us some additional flexibility in that they solve optimization problems rather than pure feasibility problems such as Equation (4).

In the SDP heuristic, we simply rewrite the pure feasibility SDP as an optimization problem. First, we reformulate Equation (2) as follows:

$$\left.\begin{array}{rll} \min & \sum\limits_{\{i,j\} \in E} \|x_i - x_j\|^2 & \\ \forall \{i,j\} \in E & \|x_i - x_j\|^2 & \geq \; d_{ij}^2. \end{array}\right\} \tag{5}$$

The objective function of Equation (5) "pulls together" the realizations of the vertices, limited to the minimum possible value d_{ij} of each pairwise distance in E because of the

constraints. Notice that Equation (5) has a convex objective but reverse convex constraints, both linearized in the SDP relaxation below [10]:

$$\left.\begin{array}{rll} \min\limits_{X \succeq 0} & \sum\limits_{\{i,j\} \in E} (X_{ii} + X_{jj} - 2X_{ij}) & \\ \forall \{i,j\} \in E & X_{ii} + X_{jj} - 2X_{ij} \quad \geq & d_{ij}^2, \end{array}\right\} \tag{6}$$

Once the SDP solver finds a feasible solution X^* for Equation (6), one can either compute $\tilde{D}$ by inverting Equation (3) with $\tilde{B} = X^*$, or factor X^* into $YY^\top$ and then use the realization Y in $\mathbb{R}^n$ to compute $\tilde{D}$ as the corresponding EDM.

4.4 Barvinok's result (Barvinok)

The Barvinok heuristic is very similar to the SDP heuristic, but we solve a different SDP: more precisely, we endow Equation (4) with an objective function $\min F \bullet X$ for some "regular" matrix F (we sketch the regularity definition below). Barvinok proves in [3] that this SDP will generally provide a realization Y having rank $O(\sqrt{|E|})$ rather than $O(n)$. We recall that $F \bullet Y$ is the trace of the dot product of $F^\top$ and X. Write the columns of F one after the other to obtain a column vector in $\mathbb{R}^{n^2}$, and do the same for X: then $F \bullet X$ corresponds to the "ordinary" scalar product between these vectors.

More precisely, Barvinok proves that if a graph is realizable in $\mathbb{R}^t$ for some t which is generally $O(n)$, then it is also realizable for $t = \lfloor(\sqrt{8|E|+1} - 1)/2\rfloor$. The way he achieves this result is by showing that there exists a solution X to the SDP

$$\left.\begin{array}{rll} \min\limits_{X \succeq 0} & F \bullet X & \\ \forall \{i,j\} \in E & X_{ii} + X_{jj} - 2X_{ij} \quad = & d_{ij}^2, \end{array}\right\} \tag{7}$$

having rank $\leq t$, and that there exist matrices F which yield these low-rank solutions. Specifically, F must be "regular" with respect to the quadratic forms involved in the constraints of Equation (7). In other words, if we write $X_{ii} + X_{jj} - 2X_{ij}$ as $Q^{ij} \bullet X$ for some real symmetric $n \times n$ matrix Q^{ij} (for each $\{i,j\} \in E$), then F must be "regular" with respect to all the Q^{ij}s.

Regularity, in this context, is defined by two conditions, one easy to explain and one harder. First, and easiest, F should be positive definite (PD), namely all its eigenvalues should be strictly positive. The hard part is as follows: consider the set of $n \times n$ matrices X that are feasible in (4): since each such X is PSD, it can be factored into $YY^\top$. As X ranges over the feasible region of Equation (7), Y ranges over possible realizations of G having various ranks $r \leq n$. We let this range be $\mathcal{Y} = \{Y \in \mathbb{R}^{r \times n} \mid YY^\top = X \text{ satisfies (4)} \wedge 1 \leq r \leq n\}$. Now we partition $\mathcal{Y}$ according to the values of r:

$$\forall r \leq n \quad \mathcal{Y}_r = \{Y \in \mathbb{R}^{n \times r} \mid YY^\top = X \text{ satisfies (4)}\}.$$

We then require that the map $\psi_F : \mathbb{R}^{|E|} \to \mathbb{R}^{n \times n}$ given by $\psi_F(z) = F - \sum\limits_{\{i,j\} \in E} z_{ij} Q^{ij}$ intersects each $\mathcal{Y}_r$ transversally, for each $r \leq n$. A map ϕ between smooth manifolds $A \to B$ *intersects a submanifold* $B' \subseteq B$ *transversally* if either $\phi(A)$ has no intersection with B' at all or, if it does, the intersection points are "well behaved", meaning their tangent spaces are non-singular in a certain way. Explaining this more precisely would require the introduction of too many new concepts; to give a suggestion, the curves $\psi(z) = z^2 + c$ intersect the manifold $z = 0$ transversally as long $c \neq 0$, since at $c = 0$ the tangent of $\psi(z)$ at the intersection point is parallel to the manifold $z = 0$.

In summary, F is regular if it is positive definite (PD) and the map $\psi_F(z)$ intersects each $\mathcal{Y}_r$ transversally. Since regular matrices prevent a corresponding (linear) map from displaying some type of singularity, most PD matrices are regular once the quadratic forms are fixed. Indeed, Barvinok's paper does not even suggest a way to sample or construct such matrices.

In [3, Example 4.1], Barvinok exploits the special structure of r-diagonal matrices to prove that the rank reduction is improved if the Q^{ij} are r-diagonal. Since our quadratic forms are fixed, and not r-diagonal in general, this is hardly relevant to our case. On the other hand, strictly diagonally dominant r-diagonal matrices are PD, so this suggests a good way to randomly generate regular matrices. We recall that a matrix F is r-diagonal if it consists of a diagonal band of width $2r+1$, i.e. an identity with r (small) nonzero entries to the left and right of the i-th diagonal entry. The off-diagonal nonzeros should be small enough for the matrix to be diagonally dominant and, in particular, PD.

4.5 Moré-Wu's `dgsol` algorithm

The `dgsol` algorithm has an outer iteration and an inner one [26]. The outer iteration starts from a smoothed convexified version of the penalty objective function

$$f(x) = \sum_{\{i,j\}\in E} \left(\|x_i - x_j\|_2^2 - d_{ij}^2\right)^2 \tag{8}$$

obtained via a Gaussian transform

$$\langle f\rangle_\lambda(x) = \frac{1}{\pi^{Kn/2}\lambda^{Kn}} \int\limits_{\mathbb{R}^{Kn}} f(y)\exp(-\|y-x\|_2^2/\lambda^2)\mathtt{d}y, \tag{9}$$

which tends to $f(x)$ as $\lambda \to 0$. For each fixed value of λ in the outer iteration, the inner iteration is based on the step

$$x^{\ell+1} = x^\ell - \alpha_\ell H_\ell \nabla\langle f\rangle_\lambda(x^\ell),$$

for $\ell \in \mathbb{N}$, where α_ℓ is a step size, and H_ℓ is an approximation of the inverse Hessian matrix of f. In other words, the inner iteration implements a local NLP solution method which uses the optimum at the previous value of λ as a starting point.

Overall, this yields a homotopy method which traces a trajectory as a function of $\lambda \to 0$, where a unique (global) optimum of the convex smoothed function $\langle f\rangle_\lambda$ for a high enough value of λ (hopefully) follows the trajectory to the global minimum of the multimodal, nonconvex function $\langle f\rangle_0 = f$.

The initial solution could theoretically be obtained by setting λ large enough so that $\langle f\rangle_\lambda(x)$ is convex, but DGSol's initial solution is computed by means of a spanning tree realization instead (see Section 4.2).

5 Computational assessment

We consider two test sets: a larger test set based on instances of various sizes, and a smaller test set with five very large sized instances. All instances are protein instances derived from Protein Data Bank (PDB) files, which contain realizations in $\mathbb{R}^3$. In order to obtain realistic instances, we computed the EDMs of these realizations, then kept the partial distance matrix consisting of all distances within a threshold of 5Å. This generates instances that are similar to the type of distance data produced by Nuclear Magnetic Resonance (NMR) experiments [30].

All of the heuristics have been coded in Python 2.7. All the tests have been obtained on a single core of an Intel Xeon processor at 2.53GHz with 48GB RAM. SDPs were solved using Mosek 7.1.0.41 [27] through the PICOS [28] Python-based API.

5.1 Small to large sizes

We consider a set of 25 protein instances with sizes ranging from $n = 15$, $|E| = 39$ to $n = 488$, $|E| = 5741$, detailed in Table 1. For each instance and solution method we record the (scaled) mean (MDE) and largest (LDE) distance error of the solution, defined as:

$$\mathsf{mde}(x) = \frac{1}{|E|} \sum_{\{i,j\}\in E} \frac{|\,\|x_i - x_j\|_2 - d_{ij}\,|}{d_{ij}};$$

$$\mathsf{lde}(x) = \frac{1}{|E|} \max_{\{i,j\}\in E} \frac{|\,\|x_i - x_j\|_2 - d_{ij}\,|}{d_{ij}}$$

as a measure of the solution quality, as well as the CPU time taken by each method. All CPU times have been computed in Python using the `time` module. The last three lines of Table 1 contain geometric means, averages and standard deviations. Best results are emphasized in boldface.

According to Table 1, IsoNLP and Barvinok are the best heuristics with respect to both MDE and LDE, whereas Isomap is the fastest (but quality-wise among the worst).

5.2 Very large sizes

For the large sized instance benchmark, we considered a set of five very large protein datasets, detailed in Table 2. None of the heuristics above, aside from Isomap, was able to terminate within 12h of CPU time, the issue being that our post-processing phase based on Python's `scipy.optimize.lbfgs` local NLP solver is too slow. The results obtained by Isomap, however, are quite poor qualitywise, with MDE and LDE measures attaining values around 0.99 for all five instances.

We therefore decided to re-implement[2] IsoNLP using a fully compiled language (a mixture of pure C and Fortran 77). We use the Isomap algorithm to obtain a starting point for `dgopt`, the optimization engine used by `dgsol`. Since `dgopt` is a homotopy method rather than a simple NLP solver, we thought it would be fair to compare this heuristic with `dgsol` itself, which uses a spanning tree heuristic (see Section 4.2) to provide a starting point to `dgopt`. The results of our tests are presented in Table 2.

The results do not show a clear quality-wise dominance of either solution method. CPU time wise, DGSol has a clear advantage: this is easily explained since IsoNLP differs from DGSol only by the choice of the initial (approximate) realization, which takes $O(n^3)$ in IsoNLP (finding eigenvalues and eigenvectors within IsoMAP) and $O(n)$ in DGSol (Lemma 2). Given the large lde values, by our past experience the only reliable solution in Table 2 is the `water` realization obtained by IsoNLP (see Figure 1).

Methodologically, there is a generality vs. efficiency trade-off at play in evaluating IsoNLP versus DGSol. Whereas the former applies to the EDGP for any given K, the latter only solves EDGP instances with K fixed to the constant 3 (this trade-off does not concern our implementation, which calls parts of the `dgsol` program). Speficically, the evaluation of the integral in Equation (9) depends on a $\mathrm{d}y = dy_1 \cdots dy_K$ which explicitly depends on K.

[2] See `http://www.lix.polytechnique.fr/~liberti/isomapheur.zip`.

Table 1 Comparative results on small to large sized protein instances ($K = 3$).

Instance			mde						lde						CPU					
Name	n	$\|E\|$	Isomap	IsoNLP	SPT	SDP	Barvinok	DGSol	Isomap	IsoNLP	SPT	SDP	Barvinok	DGSol	Isomap	IsoNLP	SPT	SDP	Barvinok	DGSol
C0700odd.1	15	39	0.585	0.001	0.190	0.068	**0.000**	0.135	0.989	0.004	0.896	0.389	**0.001**	0.634	**0.002**	1.456	1.589	0.906	1.305	1.747
C0700odd.2	15	39	0.599	**0.000**	0.187	0.086	**0.000**	0.128	0.985	**0.002**	0.956	0.389	0.009	1.000	**0.003**	1.376	1.226	1.002	1.063	0.887
C0700odd.3	15	39	0.599	**0.000**	0.060	0.086	**0.000**	0.128	0.985	**0.002**	0.326	0.389	0.009	1.000	**0.003**	1.259	1.256	0.861	1.167	0.877
C0700odd.4	15	39	0.599	**0.000**	0.283	0.086	0.001	0.128	0.985	**0.002**	2.449	0.389	0.008	1.000	**0.003**	1.347	1.222	0.976	1.063	1.033
C0700odd.5	15	39	0.599	**0.000**	0.225	0.086	0.000	0.128	0.985	**0.002**	0.867	0.389	0.007	1.000	**0.003**	1.284	1.157	0.987	1.100	0.700
C0700odd.6	15	39	0.599	**0.000**	0.283	0.086	0.000	0.128	0.985	**0.002**	1.520	0.389	**0.002**	1.000	**0.002**	1.372	1.196	0.998	1.305	0.909
C0700odd.7	15	39	0.585	0.001	0.080	0.068	**0.000**	0.135	0.989	0.004	0.361	0.389	**0.001**	0.634	**0.003**	1.469	1.322	0.894	1.093	1.719
C0700odd.8	15	39	0.585	0.001	0.056	0.068	**0.000**	0.135	0.989	0.004	0.275	0.389	**0.003**	0.634	**0.003**	1.408	1.306	0.692	1.079	1.744
C0700odd.9	15	39	0.585	0.001	0.057	0.068	**0.000**	0.135	0.989	0.004	0.301	0.389	**0.002**	0.634	**0.002**	1.430	1.172	0.791	1.093	1.745
C0700odd.A	15	39	0.585	0.001	0.043	0.068	**0.000**	0.135	0.989	**0.004**	0.316	0.389	**0.004**	0.634	**0.002**	1.294	1.269	0.722	1.220	1.523
C0700odd.B	15	39	0.585	0.001	0.151	0.068	**0.000**	0.135	0.989	**0.004**	1.022	0.389	**0.004**	0.634	**0.002**	1.297	1.279	0.871	1.111	1.747
C0700odd.C	15	39	0.835	**0.022**	0.033	0.039	0.031	0.025	1.012	**0.147**	0.393	0.211	0.294	0.167	**0.004**	6.803	6.369	7.371	7.030	7.000
C0700odd.D	36	242	0.835	**0.022**	0.041	0.039	0.042	0.025	1.012	**0.147**	0.423	0.211	0.268	0.167	**0.006**	6.806	6.575	7.422	7.603	7.095
C0700odd.E	36	242	0.835	**0.022**	0.064	0.039	0.031	0.025	1.012	**0.147**	0.894	0.211	0.260	0.167	**0.006**	6.911	6.638	7.365	6.979	7.008
C0700odd.F	36	242	0.599	**0.000**	0.047	0.086	**0.000**	0.128	0.985	**0.002**	0.308	0.389	0.005	1.000	**0.002**	1.299	1.310	1.008	1.100	1.040
C0150alter.1	37	335	0.786	0.058	0.066	0.014	0.015	**0.010**	0.992	0.571	0.693	0.256	0.285	**0.253**	**0.004**	9.492	9.456	10.276	10.120	9.272
C0080create.1	60	681	0.887	0.053	0.083	**0.024**	**0.024**	0.054	1.967	0.949	0.789	**0.511**	0.516	0.718	**0.012**	18.835	19.720	21.247	20.906	19.962
C0080create.2	60	681	0.887	0.053	0.047	**0.024**	**0.024**	0.054	1.967	0.949	0.585	**0.511**	0.512	0.718	**0.008**	18.791	20.009	21.728	20.885	19.740
C0020pdb	107	999	0.939	0.110	0.119	**0.059**	0.060	0.103	1.242	1.113	1.349	1.082	1.138	**0.798**	**0.035**	29.024	27.772	35.273	35.486	32.479
1guu	150	955	0.986	0.068	0.069	**0.057**	**0.057**	0.061	0.999	0.854	0.830	**0.735**	0.751	0.768	**0.048**	30.869	28.784	41.488	41.852	37.848
1guu-1	150	959	0.986	0.061	0.063	0.058	**0.057**	0.060	1.000	**0.711**	0.855	0.805	0.829	0.778	**0.053**	31.322	31.442	42.308	41.590	37.218
1guu-4000	150	968	0.974	0.081	0.080	0.072	**0.065**	0.079	1.000	0.901	**0.728**	0.760	0.961	0.826	**0.050**	30.352	29.856	42.330	39.832	42.015
C0030pkl	198	3247	0.961	0.112	0.160	**0.076**	0.077	0.137	**1.197**	1.354	2.230	1.995	2.054	1.401	**0.091**	105.175	104.775	149.192	146.360	111.859
1PPT	302	3102	0.984	**0.121**	0.129	0.128	0.129	0.123	**1.000**	1.519	1.219	1.944	1.956	1.224	**0.356**	112.448	110.345	185.815	187.182	118.681
100d	488	5741	0.987	0.146	0.146	0.155	0.157	**0.137**	**1.000**	1.577	1.397	1.764	1.749	1.358	**0.828**	229.809	213.136	659.638	659.280	233.115
GeoMean			0.74	**0.00**	0.09	0.06	**0.00**	0.08	1.07	**0.04**	0.73	0.50	0.06	0.66	**0.01**	6.30	6.04	5.93	6.63	6.30
Avg			0.76	0.04	0.11	0.07	**0.03**	0.10	1.09	**0.44**	0.88	0.63	0.47	0.77	**0.06**	26.12	25.21	49.69	49.55	27.96
StDev			0.17	0.05	0.07	**0.03**	0.04	0.04	**0.27**	0.55	0.57	0.52	0.65	0.34	**0.18**	51.69	48.82	135.08	134.97	53.26

Table 2 Tests on large protein instances ($K = 3$).

Instance			mde		lde		CPU	
Name	$\|V\|$	$\|E\|$	IsoNLP	dgsol	IsoNLP	dgsol	IsoNLP	dgsol
water	648	11939	**0.005**	0.15	**0.557**	0.81	26.98	**15.16**
3al1	678	17417	0.036	**0.007**	0.884	**0.810**	**170.91**	210.25
1hpv	1629	18512	**0.074**	0.078	0.936	**0.932**	374.01	**60.28**
il2	2084	45251	**0.012**	0.035	**0.910**	0.932	465.10	**139.77**
1tii	5684	69800	0.078	**0.077**	0.950	**0.897**	7400.48	**454.375**

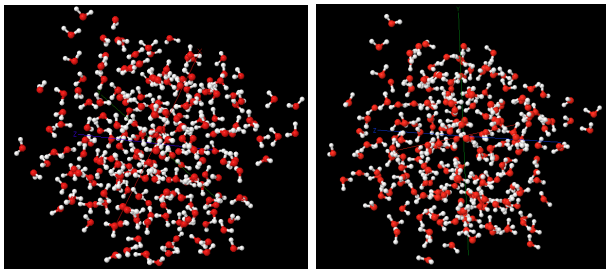

Figure 1 Comparison between water from the PDB (left) and the same structure reconstructed using IsoNLP (right).

6 Conclusion

This paper is concerned with Isomap-based heuristics for solving the Euclidean Distance Geometry Problem. It discusses the Isomap algorithm in the context of Distance Geometry, proposes six new heuristics, and benchmarks them on a set of protein conformation instances of various sizes.

References

1 F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

2 A. Bahr, J. Leonard, and M. Fallon. Cooperative localization for autonomous underwater vehicles. *International Journal of Robotics Research*, 28(6):714–728, 2009.

3 A. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete and Computational Geometry*, 13:189–202, 1995.

4 A. Barvinok. Measure concentration in optimization. *Mathematical Programming*, 79:33–53, 1997.

5 P. Biswas, T. Lian, T. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions in Sensor Networks*, 2:188–220, 2006.

6 L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: *NP*-completeness, recursive functions, and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

7 L. Blumenthal. *Theory and Applications of Distance Geometry*. Oxford University Press, Oxford, 1953.

8 T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, Boca Raton, 2001.

9 M. Cucuringu, Y. Lipman, and A. Singer. Sensor network localization by eigenvector synchronization over the Euclidean group. *ACM Transactions on Sensor Networks*, 8:1–42, 2012.

10 G. Dias and L. Liberti. Diagonally dominant programming in distance geometry. In R. Cerulli, S. Fujishige, and R. Mahjoub, editors, *International Symposium in Combinatorial Optimization*, volume 9849 of *LNCS*, pages 225–236, New York, 2016. Springer.

11 L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3 of *INFOCOM*, pages 1655–1663, Piscataway, 2001. IEEE.

12 I. Dokmanić, R. Parhizkar, J. Ranieri, and M. Vetterli. Euclidean distance matrices: Essential theory, algorithms and applications. *IEEE Signal Processing Magazine*, 1053-5888:12–30, Nov. 2015.

13 R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, second edition, 1991.

14 D. Hilbert. *Grundlagen der Geometrie*. Teubner, Leipzig, 1903.

15 I. Jolliffe. *Principal Component Analysis*. Springer, Berlin, 2nd edition, 2010.

16 N. Krislock and H. Wolkowicz. Explicit sensor network localization using semidefinite representations and facial reductions. *SIAM Journal on Optimization*, 20:2679–2708, 2010.

17 C. Lavor, L. Liberti, N. Maculan, and A. Mucherino. The discretizable molecular distance geometry problem. *Computational Optimization and Applications*, 52:115–146, 2012.

18 L. Liberti and C. Lavor. Six mathematical gems in the history of distance geometry. *International Transactions in Operational Research*, 23:897–920, 2016.

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

20 L. Liberti, C. Lavor, and A. Mucherino. The discretizable molecular distance geometry problem seems easier on proteins. In A. Mucherino, C. Lavor, L. Liberti, and N. Maculan, editors, *Distance Geometry: Theory, Methods, and Applications*, pages 47–60. Springer, New York, 2013.

21 L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, 18:33–51, 2010.

22 L. Liberti, G. Swirszcz, and C. Lavor. Distance geometry on the sphere. In *JCDCG*^{2}, LNCS, New York, accepted. Springer.

23 K. Mehlhorn and P. Sanders. *Algorithms and Data Structures*. Springer, Berlin, 2008.

24 K. Menger. Untersuchungen über allgemeine Metrik. *Mathematische Annalen*, 100:75–163, 1928.

25 K. Menger. New foundation of Euclidean geometry. *American Journal of Mathematics*, 53(4):721–745, 1931.

26 J. Moré and Z. Wu. Global continuation for distance geometry problems. *SIAM Journal of Optimization*, 7(3):814–846, 1997.

27 Mosek ApS. *The `mosek` manual, Version 7 (Revision 114)*, 2014. (`www.mosek.com`).

28 G. Sagnol. *PICOS: A Python Interface for Conic Optimization Solvers*. Zuse Institut Berlin, 2016. URL: `picos.zib.de`.

29 J. Saxe. Embeddability of weighted graphs in k-space is strongly **NP**-hard. *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.

30 T. Schlick. *Molecular modelling and simulation: an interdisciplinary guide*. Springer, New York, 2002.

31 I. Schoenberg. Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espaces distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics*, 36(3):724–732, 1935.

32 A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30:20–36, 2011.

33 M. Sippl and H. Scheraga. Cayley-Menger coordinates. *Proceedings of the National Academy of Sciences*, 83:2283–2287, 1986.

34 J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2322, 2000.

35 L. Wang, R. Mettu, and B. R. Donald. An algebraic geometry approach to protein structure determination from NMR data. In *Proceedings of the Computational Systems Bioinformatics Conference*, Piscataway, 2005. IEEE.

36 K. Wüthrich. Protein structure determination in solution by nuclear magnetic resonance spectroscopy. *Science*, 243:45–50, 1989.

37 Y. Yemini. The positioning problem – a draft of an intermediate summary. In *Proceedings of the Conference on Distributed Sensor Networks*, pages 137–145, Pittsburgh, 1978. Carnegie-Mellon University.

38 Y. Yemini. Some theoretical aspects of position-location problems. In *Proceedings of the 20th Annual Symposium on the Foundations of Computer Science*, pages 1–8, Piscataway, 1979. IEEE. doi:10.1109/SFCS.1979.39.