

METHODOLOGY ARTICLE

Open Access

Solving the molecular distance geometry problem with inaccurate distance data

Michael Souza^{1*}, Carlile Lavor², Albert Muritiba¹, Nelson Maculan³

From 8th International Symposium on Bioinformatics Research and Applications (ISBRA'12)
Dallas, TX, USA. 21-23 May 2012

Abstract

We present a new iterative algorithm for the molecular distance geometry problem with inaccurate and sparse data, which is based on the solution of linear systems, maximum cliques, and a minimization of nonlinear least-squares function. Computational results with real protein structures are presented in order to validate our approach.

Background

The knowledge of the protein structure is very important to understand its function and to analyze possible interactions with other proteins. Different methods can be applied to acquire protein structural information. Until 1984, the X-ray crystallography was the ultimate tool for obtaining information about protein structures, but the introduction of nuclear magnetic resonance (NMR) as a technique to obtain protein structures made it possible to obtain data with high precision in an aqueous environment much closer to the natural surroundings of living organism than the crystals used in crystallography [1].

The NMR technique provides a set of inter-atomic distances for certain pairs of atoms of a given protein. The molecular distance geometry problem (MDGP) arises in NMR analysis context. The MDGP consists of finding one set of atomic coordinates such that a given list of geometric constraints are satisfied [2]. Formally, the molecular distance geometry problem can be defined as the problem of finding Cartesian coordinates $x_1, \dots, x_n \in \mathbb{R}^3$ of atoms of a molecule such that $l_{ij} \leq \|x_i - x_j\| \leq u_{ij}, \forall (i, j) \in E$, where the bounds l_{ij} and u_{ij} for the Euclidean distances of pairs of atoms $(i, j) \in E$ are given *a priori* [3].

As suggested by Crippen and Havel [3], the MDGP can also be formulated as the global optimization problem of minimizing the function

$$f(x) = \sum_{(i,j) \in E} p_{ij}(x_i - x_j),$$

where the pairwise function $p_{ij} : \mathbb{R}^n \rightarrow \mathbb{R}$ is defined by

$$p_{ij}(x) = \max \left\{ \frac{l_{ij}^2 - \|x\|^2}{l_{ij}^2}, \frac{\|x\|^2 - u_{ij}^2}{u_{ij}^2}, 0 \right\}.$$

Clearly, $x = (x_1, \dots, x_n) \in \mathbb{R}^{3n}$ solves the MDGP if, and only if, x is a global minimizer of f and $f(x) = 0$.

An overview on methods applied to the MDGP is given in [4] and a very recent survey on distance geometry is given in [5].

Particular cases of the MDGP can be solved in a relatively easy way. For instance, when we know all distances $d_{ij} = \|x_i - x_j\|$, i.e., $d_{ij} = l_{ij} = u_{ij}$ and $E = \{1, 2, \dots, n\}^2$, a solution can be obtained by factoring the distance matrix $D = [d_{ij}]$. Assuming that $D = [d_{ij}]$ has the singular value decomposition $U\Sigma U^t = D$, then $x = U\Sigma^{1/2}$ is a solution for the exact MDGP defined by $l_{ij} = u_{ij} = d_{ij}$ [3]. Even in the case where the set of known distances is incomplete, i.e., when some entries of the distance matrix $D = [d_{ij}]$ is unknown, we can solve the MDGP in linear time using an iterative algorithm called **geometric buildup** [6]. First, this algorithm initializes a set \mathcal{B}

* Correspondence: michael@ufc.br

¹Department of Statistics and Applied Mathematics, Federal University of Ceará, Ceará, 60455-760, Brazil

Full list of author information is available at the end of the article

(base) with the index of four points, whose distances between all of them are known. Then, the coordinates of the points in \mathcal{B} are set using the singular value decomposition of the incomplete distance matrix D restricted to the base \mathcal{B} , and the remaining unset coordinates x_j are calculated by solving the linear system

$$\langle x_i, x_j \rangle = \frac{d_{i,1}^2 - d_{ij}^2 + d_{j,1}^2}{2}, \quad (1)$$

where $i \in \{i_1, i_2, i_3, i_4\} \subset \mathcal{B}$ and $d_{ij} = \|x_j - x_i\|$. The indexes i_1, i_2, i_3, i_4 can be chosen in an arbitrarily way, allowing us to choose another base subset when calculating the coordinate of the next x_j . At each iteration, the index j of the new coordinate x_j is inserted in the set \mathcal{B} increasing the number of subsets $\{i_1, i_2, i_3, i_4\}$ used as anchors to fix the remaining unset coordinates.

Unfortunately, in practice, the NMR experiments just provide a subset of distances between atoms spatially close and the data accuracy is limited. Thus in the real scenario, the set E is sparse and $l_{ij} < u_{ij}$. So, we just have bounds to some of the entries of the distance matrix D . In this situation, neither the singular value decomposition nor the **buildup** algorithm can be applied directly because they are both designed to deal with exact distances. In fact, the inaccurate and sparse instances of MDGP, where $l_{ij} < u_{ij}$, are much harder to solve as pointed by Moré and Wu who showed that the MDGP with inaccurate distances belongs to the NP-hard class of problems [7].

Our contribution is a new algorithm that can handle with inaccurate and sparse distance data. We propose an iterative method based on simple ideas: generate an approximated distance matrix D , take as base a clique in the graph that has D as a connectivity matrix, solve the system (1) and refine the solution using a nonlinear least-squares method. It needs to be pointed that the authors of the **buildup** algorithm and coworkers have done some modifications in the original form of the algorithm in order to handle inaccurate data [8,9]. However, the main advantage of our proposal is its simplicity and robustness. We have been able to find solutions with acceptable quality to instances of MDGP with inaccurate and sparse data, considering up to thousands of atoms.

The new iterative method

Defining the initial base

The set E of pairs (i, j) and the set of indexes $V = \{1, 2, \dots, n\}$ can be considered as a set of edges and a set of vertexes of a graph $G = (V, E)$, respectively. One may decide to use as base the biggest complete subgraph of G . The problem of calculating the biggest complete subgraph belongs to the NP-complete class and it has a

large number of applications (for a review in this subject consult [10]). We decided to use the algorithm **cliquer** proposed by Östergård in [11,12] mainly because its good behavior in graphs of moderately size and its availability on the Internet [13,14]. The **cliquer** algorithm uses a branch-and-bound algorithm developed by Östergård [15], which is based on an algorithm proposed by Carraghan and Pardalos [16].

Setting the coordinates

Once we have obtained the base \mathcal{B} associated with a complete subgraph using the algorithm **cliquer**, we need to set its coordinates. In order to generate an approximated Euclidean distance matrix (EDM) restricted to the points in the base, we define a matrix $D(t) = [d_{ij}(t)]$, where

$$d_{ij}(t) = (1 - t_{ij})l_{ij} + t_{ij}u_{ij} \quad (2)$$

for $t_{ij} \in [0, 1]$ for each $(i, j) \in E$. With this choice, we have $l_{ij} \leq d_{ij} \leq u_{ij}$, but D may not be an EDM with appropriated embedding dimension ($k = 3$). This may happen because the entries d_{ij} can violate the triangular inequality $d_{ij} \leq d_{ik} + d_{jk}$ for some indexes i, j, k , or because the rank of D is greater than 3. With this in mind, instead of considering the solution given by singular value decomposition directly, we take the columns (eigenvectors) of U associated with the 3 largest eigenvalues, getting the best 3-approximation rank of the solution to $xx^t = D(t)$ [17].

Refinement process

We should not expect great precision in x , because the matrix $D(t)$ is just an approximation. Then, we try to refine it by minimizing the nonlinear function

$$\min_x \phi_{\lambda, \tau}(x) = \sum_{(i,j) \in E: i,j \in \mathcal{B}} \phi_{\tau, \lambda}^{ij}(x, l, u), \quad (3)$$

where

$$\phi_{\tau, \lambda}^{ij}(x, l, u) = \lambda(l_{ij} - u_{ij}) + \theta_{\tau, \lambda}^{ij}(x, l) + \theta_{\tau, \lambda}^{ij}(x, u),$$

and

$$\theta_{\tau, \lambda}^{ij}(x, c) = \sqrt{\lambda^2(c - \sqrt{\|x_i - x_j\|^2 + \tau^2})^2 + \tau^2},$$

with $\lambda > 0, \tau > 0$. The parameter τ controls the smoothness degree and λ controls the intensity (weight) of the penalty function $\phi_{\lambda, \tau}$ (see Figure 1).

The function $\phi_{\tau, \lambda}$ is infinitely differentiable with respect to x , and therefore allows the application of classical optimization methods. The function $\phi_{\tau, \lambda}$ is a variation of the hyperbolic penalty technique used in [18,19].

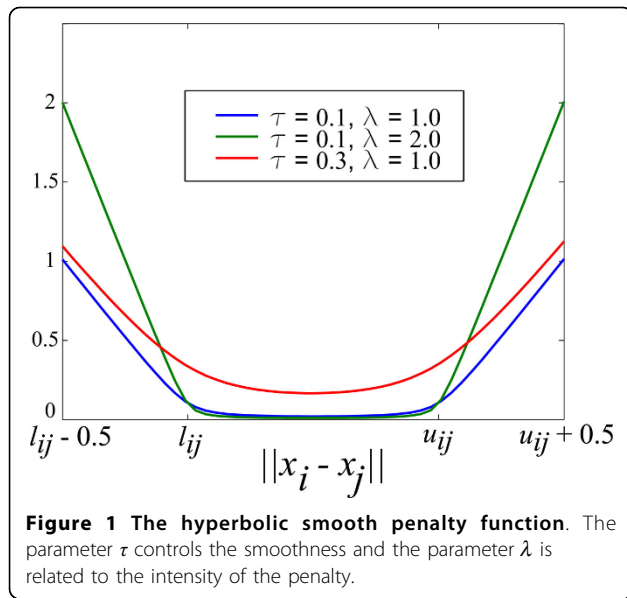


Figure 1 The hyperbolic smooth penalty function. The parameter τ controls the smoothness and the parameter λ is related to the intensity of the penalty.

In order to minimize the function $\phi_{\tau,\lambda}$, we used the local minimization routine **va35** encoded in FORTRAN and available at Harwell Subroutine Library. The routine **va35** implements the method BFGS with limited memory [20] (For additional information on this routine, see [21]).

Once we have refined the coordinates of the points in the base \mathcal{B} , we start to set the remaining (free) points. We begin with the points that have at least four constraints with the points in the base. In order to set the coordinate x_j , instead of using just four constraints involving the index j (like in the original version of the **buildup** algorithm), we use all constraints involving the index j and the indexes in the base. Explicitly, to set the coordinate x_j , we use the approximated distance matrix $D(t)$ for some $t \in [0, 1]^{|\mathcal{E}|}$, solve the linear system

$$\langle x_i, x_j \rangle = \frac{d_{i,1}^2 - d_{i,j}^2 + d_{j,1}^2}{2}, i \in \mathcal{B}, \quad (4)$$

and then we refine the solution by minimizing the function $\phi_{\lambda,\tau}(x)$ restricted to the index j and to the indexes in the base (see eq. (3)). Each newly calculated coordinate is included in the base. In the end, some points may not be fixed because they have less than four constraints involving the points in the base. In this case, we just position these points solving an undetermined system defined by constraints with points in the base. Our presented ideas are compiled in the algorithm **lsbuild** (see Additional file 1).

Methods

We have implemented our algorithm **lsbuild** in Matlab and tested it with a set of model problems on an Intel

Core 2 Quad CPU Q9550 2.83 GHz, 4GB of RAM and Linux OS-32 bits. In all experiments the parameters of the function $\phi_{\lambda,\tau}$ of the algorithm **lsbuild** were set at $\lambda = 1.0$ and at $\tau = 0.01$.

We compared our results with the algorithms **dgsol** and **buildup**. The algorithm **dgsol** proposed by Moré and Wu in [22] uses a continuation approach based on the Gaussian transformation

$$\langle f \rangle_{\lambda}(x) = \frac{1}{\pi^{n/2} \lambda^n} \int_{\mathbb{R}^n} f(y) \exp\left(-\frac{\|y-x\|^2}{\lambda^2}\right) dy$$

of the nonsmooth function

$$f(x) = \sum_{(i,j) \in \mathcal{E}} p_{ij}(x_i - x_j),$$

where the potentials p_{ij} are given by

$$p_{ij}(x) = \max \left\{ \frac{l_{ij}^2 - \|x\|^2}{l_{ij}^2}, \frac{\|x\|^2 - u_{ij}^2}{u_{ij}^2}, 0 \right\}^2.$$

The algorithm **dgsol** starts with an approximated solution and, given a sequence of smoothing parameters $\lambda_0 > \lambda_1 > \dots > \lambda_p = 0$, it determines a minimizer x_{k+1} of $\langle f \rangle_{\lambda}$. The algorithm **dgsol** uses the previous minimizer x_k as the starting point for the search. In this manner a sequence of minimizers x_1, \dots, x_{p+1} is generated, with the x_{p+1} a minimizer of f and the candidate for the global minimizer. In our experiments, we used the implementation of the algorithm **dgsol** encoded in language C and downloaded from [23].

We also compared our results with the ones obtained by the version of the algorithm **buildup** proposed by Sit, Wu and Yuan in [8]. The algorithm **buildup** starts defining a base set using four points whose distances between all of them are known (a clique of four points). Then, at each iteration, a new point x_k with known distances to at least four points in the base is selected. In order to avoid the accumulation of errors, instead of just positioning the new point, in the modified version of the algorithm **buildup** the entire substructure formed by the point x_k and its neighbors in the base is calculated by solving the nonlinear system

$$\langle x_i, x_j \rangle = \frac{d_{i,1}^2 - d_{i,j}^2 + d_{j,1}^2}{2}, \forall i, j \in B$$

with variables $x_i = (x_i^1, x_i^2, x_i^3), x_j = (x_j^1, x_j^2, x_j^3) \in \mathbb{R}^3$ and B being the set formed by the index k and the indexes of all neighbors of x_k in the current base set. The parameters d_{kj} are the given distances between the node x_k and its neighbors x_j in the base and, for the nodes x_j and x_i already in the base, if the distance between them is unknown, we consider $d_{ij} = \|x_i - x_j\|$.

Once the substructure is obtained, it is inserted in the original structure by an appropriated rotation and translation and the point x_k is included in the base. This process is repeated until all nodes are included in the base. We have implemented the **buildup** algorithm in Matlab.

Our decision to compare the **lsbuild** with the algorithms **dgso** and **buildup** is mainly motivated by their similarities with our proposal. In fact, the algorithm **dgso** uses a smooth technique in order to avoid the local minimizers and the algorithm **buildup** solves a sequence of systems which produce partial solutions and iteratively try to construct a candidate to global solution. Our algorithm combines some variations of these two ideas. We use a hyperbolic smooth technique to insert differentiability in the problem and a divide-and-conquer approach based in successive solutions of overdetermined linear systems in order to construct a candidate to global solution.

In our experiments, the distance data were derived from the real structural data from the Protein Data Bank (PDB) [24]. It needs to be pointed that each of the algorithms considered has a level of randomness, the algorithm **dgso** takes random start point and the algorithms **lsbuild** and **buildup** starts with an incomplete random matrix $D = [d_{ij}]$ where $l_{ij} \leq d_{ij} \leq u_{ij}$. So, in order to do a fair comparison, we run each test 30 times.

We considered two set of instances. The first one was proposed by Moré and Wu in order to validate the algorithm **dgso** [22]. This set is derived from the three-dimensional structure of the fragments made up of the first 100 and 200 atoms of the chain A of protein PDB:1GPV[25,26]. For each fragment, we generated a set of constraints considering only atoms in the same residue or the neighboring residues. Formally,

$$E = \{(i, j) : x_i \in R(k), x_j \in (R(k) \cup R(k + 1)), \forall k\},$$

where $R(k)$ represents the k -th residue.

In this set of instances, the bounds l_{ij} and u_{ij} were given by the equations

$$l_{ij} = (1 - \varepsilon)\hat{d}_{ij}, \quad u_{ij} = (1 + \varepsilon)\hat{d}_{ij},$$

where \hat{d}_{ij} is the real distance between the nodes x_i and x_j in the known structure x^* of protein PDB:1GPV. In this way, all distances between atoms in the same residue or neighboring residues were considered. We generated two instances for each fragment by taking ε equals to 0.00 and 0.08.

In order to measure the precision of the solutions just with respect to the constraints, without providing any information about the original structure x^* , we use the function

$$LDME = \left(\frac{1}{|E|} \sum_{(i,j) \in E} e_{ij}^2 \right)^{1/2}, \quad (5)$$

where

$$e_{ij} = \max\{l_{ij} - \|x_i - x_j\|, \|x_i - x_j\| - u_{ij}, 0\}$$

is the error associated to the constraint $l_{ij} \leq \|x_i - x_j\| \leq u_{ij}$. We also measured the deviation

of the solutions generated by each algorithm with respect to the original solution x^* in the PDB files, using the function

$$RMSD = \frac{1}{\sqrt{n}} \min_{Q,h} \|x^* - Q(x - h)\|_F, \quad (6)$$

with $h \in \mathbb{R}^{n \times 3}$ and $Q \in \mathbb{R}^{3 \times 3}$ orthogonal.

In the second experiment, we use a more realistic set of instances with larger proteins proposed by Biswas in [17]. Typically, just distances below 6Å ($1\text{Å} = 10^{-8}$ cm) between some pair of atoms can be measured by NMR techniques. So, in order to produce more realistic data, we considered only 70% of the distances lower than $R = 6$ Å. To introduce noise in the model, we set the bounds using the equations

$$l_{ij} = d_{ij}^* \max(0, 1 - |\varepsilon_{ij}|), \quad u_{ij} = d_{ij}^* (1 + |\varepsilon_{ij}|), \quad (7)$$

where d_{ij}^* is the true distance between atom i and atom j and $\varepsilon_{ij}, \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$ (normal distribution). With this model, we generate a sparse set of constraints and introduce a noise in the distances that are not so simple as the one used in the instances proposed by Moré and Wu.

Results and discussion

In Table 1 we can see the results of the first experiment defined from the protein PDB:1GPV and all distances in the same or neighboring residues. The values show that the algorithms **buildup** and **lsbuild** worked better (lower *LDME* and *RMSD* and CPU time) than the algorithm **dgso** in all instances. The algorithms **buildup** performed slightly better than the algorithm **lsbuild** being the fastest algorithm. Despite its simplicity, this set of instances worked as an indication of the correctness of our implementation of the **buildup** algorithm.

Table 2 shows the results of the second experiment with more realistic data. We can see that our approach was more efficient than the algorithms **buildup** and **dgso** that were not able to find good solutions in these harder instances. In this table, $|V|$ is the number of atoms in the instance, and CPU time is given in seconds. We also point out that *LDME* was low and the *RMSD* was lower than 3.5Å in all instances, which

Table 1 RMSD, LDME and the CPU time in seconds for PDB:1GPV protein.

	Fragment with 100 atoms					
	$\epsilon = 0.00$			$\epsilon = 0.08$		
	$\langle LDME \rangle$	$\langle RMSD \rangle$	$\langle TIME \rangle$	$\langle LDME \rangle$	$\langle RMSD \rangle$	$\langle TIME \rangle$
dgsol	8.29E-03	3.93E-01	3.61E+00	3.31E-03	8.25E-01	4.40E+00
buildup	3.50E-15	1.46E-14	1.08E-01	0.00E+00	3.13E-01	1.08E-01
lsbuild	6.47E-15	1.20E-14	1.51E-01	0.00E+00	7.77E-02	1.33E-01
	Fragment with 200 atoms					
	$\epsilon = 0.00$			$\epsilon = 0.08$		
	$\langle LDME \rangle$	$\langle RMSD \rangle$	$\langle TIME \rangle$	$\langle LDME \rangle$	$\langle RMSD \rangle$	$\langle TIME \rangle$
dgsol	3.18E-02	2.58E+00	1.48E+01	4.00E-03	2.45E+00	1.73E+01
buildup	4.85E-15	2.45E-14	3.11E-01	0.00E+00	5.18E-01	3.11E-01
lsbuild	1.90E-14	5.21E-14	6.01E-01	0.00E+00	4.21E-01	5.25E-01

Results for the fragments made up with the first 100 and 200 atoms of protein PDB:1GPV. The $\langle LDME \rangle$ and $\langle RMSD \rangle$ represent the LDME and RMSD measures respectively and $\langle TIME \rangle$ represents the mean time in seconds.

Table 2 RMSD and LDME for the larger instance set.

PDB	V	$\langle LDME \rangle$			$\langle RMSD \rangle$		
		lsbuild	buildup	dgsol	lsbuild	buildup	dgsol
1PTQ	402	2.61E-03	1.80E+00	5.41E-01	1.31E-02	9.49E+00	6.89E+00
1LFB	641	2.03E-04	1.84E+00	3.91E-01	4.19E-03	1.23E+01	5.48E+00
1AX8	1003	2.00E-04	1.83E+00	4.33E-01	1.62E-02	1.35E+01	7.95E+00
1F39	1534	3.03E-02	1.89E+00	4.74E-01	4.22E-01	1.79E+01	1.28E+01
1RGS	2015	1.08E-01	1.87E+00	4.73E-01	1.74E+00	1.92E+01	1.35E+01
1KDH	2846	1.39E-02	1.86E+00	5.19E-01	9.43E-02	2.11E+01	1.61E+01
1BPM	3671	2.20E-02	1.90E+00	5.14E-01	7.86E-02	2.29E+01	1.55E+01
1TOA	4292	6.90E-03	1.89E+00	6.75E-01	2.56E-01	2.52E+01	2.39E+01
1MQQ	5681	1.93E-02	1.91E+00	8.86E-01	1.89E-01	2.50E+01	2.50E+01

Results with instances considering just 70% of the distances below 6Å. The $\langle LDME \rangle$ and $\langle RMSD \rangle$ represent the mean LDME and mean RMSD respectively.

means that the algorithm is robust and able to find protein structures very similar to the original ones [1]. The results in Table 3 shows that the **buildup** algorithm was again the fastest. The CPU time of the algorithm **lsbuild** was in the average around to 2.45 times the time consumed by the algorithm **buildup**, this fact must be mitigated by the better quality of the solutions obtained by the algorithm **lsbuild**.

Table 3 TIME for the larger instance set.

PDB	$\langle TIME \rangle$		
	lsbuild	buildup	dgsol
1PTQ	9.99E-01	5.34E-01	1.03E+01
1LFB	1.86E+00	1.01E+00	2.55E+01
1AX8	2.98E+00	1.70E+00	4.36E+01
1F39	7.21E+00	3.57E+00	8.59E+01
1RGS	1.43E+01	4.70E+00	1.33E+02
1KDH	2.12E+01	7.28E+00	2.09E+02
1BPM	2.47E+01	8.04E+00	2.99E+02
1TOA	3.93E+01	1.14E+01	7.03E+02
1MQQ	3.93E+01	1.82E+01	7.63E+02

The mean CPU time in seconds with the instances considering just 70% of the distances below 6Å.

Finally, the results of both set of instances indicate that our algorithm **lsbuild** based on the combination of the resolution of linear systems, derived from the approximated EDM matrices, and the refinement process based on hyperbolic smoothing penalty is a very effective strategy to solve MDGP instances with sparse and inaccurate data.

Conclusions

We presented a new algorithm to solve molecular distance geometry problems with inaccurate distance data. These problems are related to molecular structure calculations using data provided by NMR experiments which, in fact, are not precise. Our algorithm combines the divide-and-conquer framework and a variation of the hyperbolic smoothing technique. The computational results show that the proposed algorithm is an effective strategy to handle uncertainty in the data.

Additional material

Additional file 1: Algorithm lsbuild.

Authors' contributions

MS, AM and CL participated in the development of the ideas presented in the design of the proposed algorithm. MS and CL drafted the manuscript. CL and NM gave final approval of the version to be published. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

We are grateful to the anonymous referees for improving this paper and the Brazilian Research Agencies FAPESP and CNPq by their support. This article has been published as part of *BMC Bioinformatics* Volume 14 Supplement 9, 2013: Selected articles from the 8th International Symposium on Bioinformatics Research and Applications (ISBRA'12). The full contents of the supplement are available online at <http://www.biomedcentral.com/bmcbioinformatics/supplements/14/S9>.

Declarations

The publication of this article is supported by the Brazilian Research Agencies FAPESP and CNPq.

Author details

¹Department of Statistics and Applied Mathematics, Federal University of Ceará, Ceará, 60455-760, Brazil. ²Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, Campinas, 13083-859, Brazil. ³Institute Alberto Luiz Coimbra, Federal University of Rio de Janeiro, Rio de Janeiro, 21941-972, Brazil.

Published: 28 June 2013

References

- Schlick T: *Molecular modeling and simulation: an interdisciplinary guide*. second edition. New York: Springer Verlag; 2010.
- Crippen GM: **Linearized embedding: a new metric matrix algorithm for calculating molecular conformations subject to geometric constraints.** *Journal of Computational Chemistry* 1989, **10**(7):896-902.
- Crippen GM, Havel T: *Distance geometry and molecular conformation* New York: Wiley; 1988.
- Liberti L, Lavor C, Mucherino A, Maculan N: **Molecular distance geometry methods: from continuous to discrete.** *International Transactions in Operational Research* 2010, **18**:33-51.
- Liberti L, Lavor C, Maculan N, Mucherino A: **Euclidean distance geometry and applications.** *arXiv:1205.0349* 2012.
- Wu D, Wu Z: **An updated geometric build-up algorithm for solving the molecular distance geometry problems with sparse distance data.** *Journal of Global Optimization* 2007, **37**:661-673.
- Moré JJ, Wu Z: **Global continuation for distance geometry problems.** *SIAM Journal on Optimization* 1997, **7**:814-836.
- Sit A, Wu Z, Yuan Y: **A geometric buildup algorithm for the solution of the distance geometry problem using least-squares approximation.** *Bulletin of mathematical biology* 2009, **71**:1914-1933.
- Luo X, Wu Z: **Least-Squares Approximations in Geometric Buildup for Solving Distance Geometry Problems.** *Journal of Optimization Theory and Applications* 2011, **149**:580-598.
- Bomze I, Budinich M, Pardalos P, Pelillo M: **The maximum clique problem.** *Handbook of combinatorial optimization* 1999, **4**:1-74.
- Östergård P: **A new algorithm for the maximum-weight clique problem.** *Nordic J of Computing* 2001, **8**(4):424-436.
- Östergård P: **A fast algorithm for the maximum clique problem.** *Discrete Applied Mathematics* 2002, **120**:197-207.
- Niskanen S, Östergård P: **Cliquer User's Guide, Version 1.0,** Communications Laboratory, Helsinki University of Technology, Espoo. *Tech rep, Finland, Tech Rep* 2003.
- CLIQER: **Routines for Clique Searching.** [<http://users.tkk.fi/pat/cliquer.html>].
- Ostergard PR: **A fast algorithm for the maximum clique problem.** *Discrete Appl Math* 2002, **120**:195-205.
- Carraghan R, Pardalos PM: **An exact algorithm for the maximum clique problem.** *Operational Research Letters* 1990, **9**:375-382.
- Biswas P, Toh KC, Ye Y: **A Distributed SDP Approach for Large-Scale Noisy Anchor-Free Graph Realization with Applications to Molecular Conformation.** *SIAM Journal on Scientific Computing* 2008, **30**:1251-1277.
- Souza M, Xavier AE, Lavor C, Maculan N: **Hyperbolic smoothing and penalty techniques applied to molecular structure determination.** *Operations Research Letters* 2011, **39**:461-465.
- Xavier AE: **Hyperbolic Penalty: A New Method for Nonlinear Programming with Inequalities.** *International Transactions in Operational Research* 2001, **8**:659-671.
- Liu D, Nocedal : **On The Limited Memory BFGS Method For Large Scale Optimization.** *Tech Rep NA-03* Department of Electrical Engineering and Computer Science Northwestern University; 1988.
- Harwell Subroutine Library. [<http://www.hsl.rl.ac.uk>].
- Moré JJ, Wu Z: **Distance Geometry Optimization for Protein Structures.** *Journal of Global Optimization* 1999, **15**:219-234.
- DGsol: **Distance Geometry Optimization Software.** [<http://www.mcs.anl.gov/~more/dgso/>].
- Berman H, Westbrook J, Feng Z, Gilliland G, Bhat T, Weissig H, Shindyalov I, Bourne P: **The protein data bank.** *Nucleic acids research* 2000, **28**:235-242.
- Guan Y, Zhang H, Konings RNH, Hilbers CW, Terwilliger TC, Wang AHJ: **Crystal structure of Y41H and Y41F mutants of gene V suggest possible protein-protein interactions in the GVP-SSDNA complex.** *Biochemistry* 1994, **33**:7768.
- Skinner M, Zhang H, Leschnitzer D, Guan Y, Bellamy H, Sweet R, Gray C, Konings R, Wang A, Terwilliger T: **Structure of the gene V protein of bacteriophage F1 determined by multi-wavelength X-ray diffraction on the selenomethionyl protein.** *Proc Nat Acad Sci USA* 1994, **91**:2071.

doi:10.1186/1471-2105-14-S9-S7

Cite this article as: Souza et al.: Solving the molecular distance geometry problem with inaccurate distance data. *BMC Bioinformatics* 2013 **14**(Suppl 9):S7.

Submit your next manuscript to BioMed Central and take full advantage of:

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at
www.biomedcentral.com/submit

