Chapter 1

COMPUTATIONAL EXPERIENCE WITH THE MOLECULAR DISTANCE GEOMETRY PROBLEM

Carlile Lavor

Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro – UERJ, Rua São Francisco Xavier 524, 6^{o} andar, bl. D, sala 6018, Rio de Janeiro 20550-900, Brazil.

carlile@ime.uerj.br

Leo Liberti

DEI, Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy.

liberti@elet.polimi.it

Nelson Maculan

COPPE, Universidade Federal do Rio de Janeiro – UFRJ, C.P. 68511, Rio de Janeiro 21945-970, Brazil.

maculan@cos.ufrj.br

Abstract In this work we consider the molecular distance geometry problem, which can be defined as the determination of the three-dimensional structure of a molecule based on distances between some pairs of atoms. We address the problem as a nonconvex least-squares problem. We apply three global optimization algorithms (spatial Branch-and-Bound, Variable Neighbourhood Search, Multi Level Single Linkage) to two sets of instances, one taken from the literature and the other new.

Keywords: molecular conformation, distance geometry, global optimization, spatial Branch-and-Bound, variable neighbourhood search, multi level single linkage.

Accepted for publication in J. Pintér (ed.), “Global Optimization: Selected Case Studies”, Springer (to appear).

1. Introduction

The Molecular Distance Geometry Problem (MDGP) is the problem of determining the three-dimensional structure of a molecule where a subset of the atomic distances is known. Formally, we need to find vectors $x_1, ..., x_n \in \mathbb{R}^3$, which describe the three-dimensional position of each atom in the molecule, such that:

$$\forall \{i,j\} \in S \quad (||x_i - x_j|| = d_{ij}),$$

where S is the subset of pairs of atoms $\{i,j\}$ whose distances d_{ij} are known. We address the problem in terms of finding the global minimizer of the function

$$f(x_1, \ldots, x_n) = \sum_{\{i,j\} \in S} \left(||x_i - x_j||^2 - d_{ij}^2 \right)^2 .$$

It is easy to verify that $x_1, \ldots, x_n \in \mathbb{R}^3$ solve the problem if and only if $f(x_1, \ldots, x_n) = 0$.

The MDGP is an important problem in molecular biology. The objective is to find a molecular conformation satisfying all the constraints imposed by the known distances (i.e., that $||x_i - x_j|| = d_{ij}$ for all $\{i,j\} \in S$). For some references, see [Crippen and Havel, 1988, Hendrickson, 1995, Moré and Wu, 1997, Moré and Wu, 1999, An, 2003].

The aim of this work is twofold. On the one hand, we present two different methods of generating MDGP instances, and we wish to test which of the methods generates the hardest instances. On the other hand, we want to assess the solution quality and efficiency of three well-known global optimization algorithms applied to the MDGP. The algorithms are: spatial Branch-and-Bound (sBB) [Ryoo and Sahinidis, 1995, Tawarmalani and Sahinidis, 2002, Adjiman et al., 1998, Smith and Pantelides, 1999, Hansen, 1992], Variable Neighbourhood Search (VNS) [Hansen and Mladenović, 2001, Mladenović et al., 2003], and Multi Level Single Linkage (MLSL) [Rinnooy-Kan and Timmer, 1987a, Rinnooy-Kan and Timmer, 1987b, Locatelli and Schoen, 1996, Schoen, 1998, Schoen, 1999, Locatelli and Schoen, 1999, Schoen, 2002, Kucherenko and Sytsko, 2004]. We test each of these algorithms on instances of varying sizes generated with the two generating methods, one taken from the literature [Moré and Wu, 1997] and the other new [Lavor, 2005].

Our computational results show that, in terms of user CPU time, VNS is the most efficient of the methods we tested. As the size of the instance grows, however, the performance difference between VNS and MLSL decreases. Whilst VNS and MLSL are stochastic algorithms, sBB is a deterministic algorithm. As such, it provides a guarantee of ε-global optimality, but at a practically high computational cost on most global optimization problems. With MDGP instances, however, sBB was found to be competitive with VNS and MLSL at least for small and medium-sized instances.

It is worth mentioning explicitly that, somewhat unusually for this type of problems, we included no smoothing techniques in our algorithms, as the aim of this test was to verify the applicability of general-purpose global optimization algorithms to the problems in original form. Similar tests, but with smoothing techniques included, are currently work in progress.

The rest of this paper is organized as follows: Section 2 describes the global optimization algorithms used; Section 3 describes the two sets of instances used to generate the experiments and discusses the computational results.

2. Global optimization methods

In this section, we shall briefly describe the three algorithms we used to solve the MDGP. All these methods are general-purpose, in the sense that they can be used without modification to solve all global optimization problems. In other words, they do not take into account the structure of the problem.

2.1 Spatial Branch-and-Bound

Spatial Branch-and-Bound (sBB) algorithms locate the global optimum by generating converging sequences of upper and lower bounds to the objective function. The upper bounds are obtained by locally solving the original (nonconvex) problem. The lower bounds are obtained by locally solving a convex (in this case, linear) relaxation of the original problem. Since any local solution of a convex problem is also global, locally solving the linear relaxation yields a valid lower bound to the original problem. The algorithm, first proposed in [Smith, 1996, Smith and Pantelides, 1999], is shown in Fig. 1.1. The implementation details, as well as many refinements and improvements with respect to the original algorithm, are given in [Liberti, 2004].

The most outstanding feature of sBB algorithm described in this section is the automatic construction of the convex relaxation via symbolic reformulation. This involves identifying all the nonconvex terms in the problem and replacing them with the respective convex relaxations. The algorithm that carries out this task is symbolic in nature as it has to recognize the nonconvex operators in any given function. The relaxation is built in two stages: first the problem is reduced to a standard form where the nonlinear terms are linearized. This means that each nonlinear term is replaced by a linearizing variable, and a constraint of type “linearizing variable = nonlinear term” is added to the problem formulation. Such constraints are called *defining equations*, or *defining constraints*. In the second stage of the linear relaxation each nonlinear term is replaced by the corresponding linear under- and over-estimators. Note that this process is wholly automatic, and part of the implementation software.

1 (Initialization) Initialize a list of regions to a single region comprising the entire set of variable ranges. Set the convergence tolerance $\varepsilon > 0$, the best objective function value found up to the current step as $U := \infty$ and the corresponding solution point as $x^* := (\infty, \ldots, \infty)$. Optionally, perform optimization-based bounds tightening.

2 (Choice of Region) If the list of regions is empty, terminate the algorithm with solution x^* and objective function value U. Otherwise, choose a region R (the "current region") from the list. Delete R from the list. Optionally, perform feasibility-based bounds tightening on R.

3 (Lower Bound) Generate a convex relaxation of the original problem in the selected region R and solve it to obtain an underestimation l of the objective function with corresponding solution $\bar{x}$. If $l > U$ or the relaxed problem is infeasible, go back to step 2.

4 (Upper Bound) Attempt to solve the original (generally nonconvex) problem in the selected region to obtain a (locally optimal) solution $\tilde{x}$ with objective function value u. If this fails, set $u := +\infty$ and $\tilde{x} = (\infty, \ldots, \infty)$.

5 (Pruning) If $U > u$, set $x^* = \tilde{x}$ and $U := u$. Delete all regions in the list that have lower bounds bigger than U as they cannot possibly contain the global minimum.

6 (Check Region) If $u - l \leq \varepsilon$, accept u as the global minimum for this region and return to step 2. Otherwise, we may not yet have located the region global minimum, so we proceed to the next step.

7 (Branching) Apply a branching rule to the current region to split it into sub-regions. Add these to the list of regions, assigning to them an (initial) lower bound of l. Go back to step 2.

Figure 1.1. The spatial Branch-and-Bound algorithm.

2.2 Variable Neighbourhood Search

Variable Neighbourhood Search (VNS) is a relatively recent metaheuristic which relies on iteratively exploring neighbourhoods of growing size to identify better local optima [Hansen and Mladenović, 2001]. More precisely, VNS escapes from the current local minimum x^* by initiating other local searches from starting points sampled from a neighbourhood of x^* which increases its size iteratively until a local minimum better than the current one is found. These steps are repeated until a given termination condition is met.

VNS has been applied to a wide variety of problems both from combinatorial and continuous optimization. Its early applications to continuous problems were based on a particular problem structure. In the continuous location-allocation problem the neighbourhoods are defined according to the meaning of problem variables (assignments of facilities to customers, positioning of yet unassigned facilities and so on) [Brimberg and Mladenović, 1996]. In the bilinearly constrained bilinear problem the neighbourhoods are defined in terms of the applicability of the successive linear programming approach, where the problem variables can be partitioned so that fixing the variables in either set yields a linear problem; more precisely, the neighbourhoods of size k are defined as the vertices of the LP polyhedra that are k pivots away from the current vertex [Hansen and Mladenović, 2001]. In summary, none of the early applications of VNS to continuous problems solved problems in general form.

The first VNS algorithm targeted at problems with fewer structural requirements, namely, box-constrained NLPs, was given in [Mladenović et al., 2003] (the paper focuses on a particular class of box-constrained NLPs, but the proposed approach is general). The very same code used in [Mladenović et al., 2003] (which is different from that used in this paper) has been already applied to another molecular conformation problem with considerable success [Drazić et al., 2004]. Since the problem is assumed to be box-constrained, the neighbourhoods arise naturally as hyperrectangles of growing size centered at the current local minimum x^*. In the pseudocode algorithm in Fig. 1.2, the termination condition is taken to be $k > k_{\max}$. This is the most common behaviour, but not the only one (the termination condition can be based on CPU time or other algorithmic parameters). The definition of the neighourhoods may vary. If $N_k(x)$ is taken to be a hyperrectangle $H_k(x)$ of "size" k centered at x, sampling becomes easy; there is a danger, though, that sampled points will actually be inside a smaller hyperrectangular neighbourhood. A way to deal with this problem is to take $N_k(x) = H_k(x) \backslash H_{k-1}(x)$, although this makes it harder to sample a point inside the neighbourhood. For each $k \leq k_{\max}$ we define $H_k(x^*)$ to be hyper-rectangles similar to $x^L \leq x \leq x^U$, all centered at x^*, whose sides have been scaled by $\frac{k}{k_{\max}}$. More formally, let $H_k(x^*)$ be the hyper-rectangle

1 Set $k \leftarrow 1$, pick random point $\tilde{x}$, perform local descent to find a local minimum x^*.

2 Until $k > k_{\text{max}}$ repeat the following steps:

(a) define a neighbourhood $N_k(x^*)$;

(b) sample a random point $\tilde{x}$ from $N_k(x^*)$;

(c) perform local descent from $\tilde{x}$ to find a local minimum x';

(d) if x' is better than x^* set $x^* \leftarrow x'$ and $k \leftarrow 1$; go to step 2;

(e) set $k \leftarrow k + 1$

Figure 1.2. The VNS algorithm.

$y^L \leq x \leq y^U$ where, for all $i \leq n$:

$$\begin{aligned} y_i^L &= x_i^* - \frac{k}{k_{\text{max}}}(x_i^* - x_i^L) \\ y_i^U &= x_i^* + \frac{k}{k_{\text{max}}}(x_i^U - x_i^*). \end{aligned}$$

This construction forms a set of hyper-rectangular "shells" centered at x^*. In our computational experiments, we used $N_k(x) = H_k(x)$ for simplicity.

2.3 Multi-Level Single Linkage

In this section we shall describe the main features of a Multi-Level Single Linkage (MLSL) stochastic algorithm for global optimization. The algorithm is called *SobolOpt* [Kucherenko and Sytsko, 2004]. Its main strength is that it employs certain Low-Discrepancy Sequences (LDSs) of sampling points called *Sobol' sequences* whose distributions in Euclidean space have very desirable uniformity properties. Uniform random distributions where each point is generated in a time interval (as is the case in practice when generating a sampling on a computer) are guaranteed to "fill the space" in infinite time with probability 1. In fact, these conditions are very far from the normal operating conditions. LDSs, and in particular Sobol' sequences, are guaranteed to fill the space as uniformly as possible even in finite time. In other words, for any integer $N > 0$, the first N terms of a Sobol' sequence does a very good job of filling the space evenly. One further very desirable property of Sobol' sequences is that any projection on any coordinate hyperplane of the Euclidean space $\mathbb{R}^n$ containing N n-dimensional points from a Sobol' sequence will still contain N projected

$(n-1)$-dimensional Sobol' points. This clearly does not hold with the uniform grid distribution where each point is located at a coordinate lattice point (in this case the number of projected points on any coordinate hyperplanes is $O(N^{\frac{n-1}{n}})$, as shown in Fig. 1.3). The comparison between grid and Sobol' points in $\mathbb{R}^2$ is shown in Fig. 1.4.

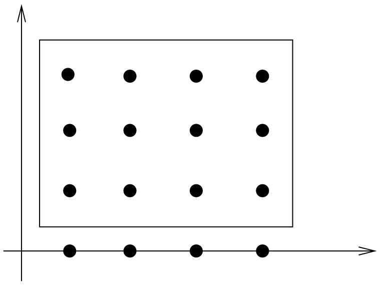

Figure 1.3. Projecting a grid distribution in $\mathbb{R}^2$ on the coordinate axes reduces the number of projected points. In this picture, $N = 12$ but the projected points are just 4.

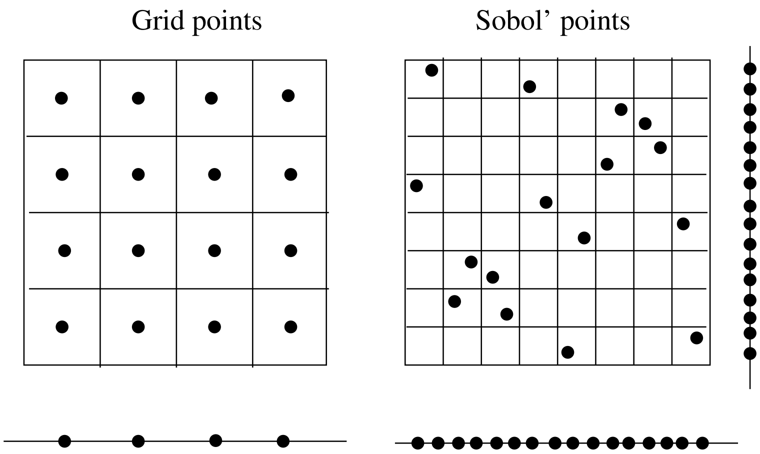

Figure 1.4. Comparison between projected distribution of grid points and Sobol' points in $\mathbb{R}^2$.

The regularity and uniformity properties of Sobol' sequences are exploited in the following MLSL algorithm. Let Q be the set of pairs of sampled points q together with their evaluation $f(q)$ (where f is the objective function). Let S be the list of all local minima found up to now.

The algorithm terminates with a list S of all the local minima found. Finding the global minimum is then a trivial matter of identifying the minimum with

1 (Initialization) Let $Q = \emptyset$, $S = \emptyset$, $k = 1$ and set $\varepsilon > 0$.

2 (Termination) If a pre-determined termination condition is verified, stop.

3 (Sampling) Sample a point q_k from a Sobol' sequence; add $(q_k, f(q_k))$ to Q.

4 (Clustering distance) Compute a distance r_k (which is a function of k and n; there are various ways to compute this distance, so this is considered as an "implementation detail" — one possibility is $r_k = \beta k^{-\frac{1}{n}}$, where β is a known parameter).

5 (Local phase) If there is no previously sampled point $q_j \in Q$ (with $j < k$) such that $||q_k - q_j|| < r_k$ and $f(q_j) \leq f(q_k) - \varepsilon$, solve the problem locally with q_k as a starting point to find a solution y with value $f(y)$. If $y \notin S$, add y to S. Set $k \leftarrow k + 1$ and repeat from step 2.

Figure 1.5. The SobolOpt algorithm.

lowest objective function value $f(y)$. Two of the most common termination conditions are (a) maximum number of sampled points and (b) maximum time limit exceeded. In our tests we accepted a default termination condition based on the number of local searches not exceeding 320.

Sobol' sequences are generated as follows. Let $P(x)$ be a primitive polynomial of degree q in $GF(2)[x]$, say $P(x) = \sum_{i=0}^{q} a_{q-i}x^i$ where $a_0 = a_q = 1$. Now for all $i > q$ define Q_i recursively as the result of the bitwise XOR operation on the following set of numbers: $\{2^k a_k Q_{i-k} \mid 1 \leq k \leq q-1\} \cup \{2^q Q_{i-q}, Q_{i-q}\}$ (this is a q-term recurrence relation; the first q terms of the sequence can be chosen as arbitrary odd integers respectively less than $2, \ldots, 2^q$). Let $V_i = \frac{Q_i}{2^i}$ for all i. For each integer j, let $\Lambda_j = \{V_i \mid \text{the } i\text{-th bit of } j \text{ is nonzero}\}$. We define X_j, the j-th element of the Sobol' sequence, as the result of the bitwise XOR operation on Λ_j. Multidimensional Sobol' sequences are obtained by building each vector out of a different primitive polynomial. For a full discussion on the implementation details, see [Press et al., 1997], p. 311.

3. Computational experiments

In this section we compare the results of the three global optimization algorithms mentioned in Section 2 on two sets of instances.

3.1 Instance generation methods

The first set of instances (Moré instances) was generated using the first model proposed in [Moré and Wu, 1997]. The model is based on a molecule with s^3 atoms ($s = 1, 2, 3, \ldots$) located in the three-dimensional lattice defined by

$$\{(i_1, i_2, i_3) \in \mathbb{R}^3 : 0 \leq i_k \leq s - 1,\ k = 1, 2, 3\}.$$

An order is defined for the atoms of the lattice by letting atom i be the atom at position (i_1, i_2, i_3), where

$$i = 1 + i_1 + s i_2 + s^2 i_3,$$

and the set S is defined by

$$S = \{\{i, j\} : |i - j| \leq s^2\}.$$

For example, for $s = 2$, the atoms located at $(0, 0, 0)$, $(1, 0, 0)$, and $(0, 1, 0)$, are the first ($i = 1$), second ($i = 2$), and third ($i = 3$) atoms.

The second set of instances (Lavor instances) was generated according to another model. This model considers a molecule as a chain of n atoms with Cartesian coordinates given by $x_1, \ldots, x_n \in \mathbb{R}^3$. For every pair of consecutive atoms i, j, let r_{ij} be the bond length which is the Euclidean distance between them. For every three consecutive atoms i, j, k, let θ_{ik} be the bond angle corresponding to the relative position of the third bead with respect to the line containing the previous two. Likewise, for every four consecutive atoms i, j, k, l, let ω_{il} be the angle, called the torsion angle, between the normals through the planes determined by the atoms i, j, k and j, k, l. The sets M_1, M_2 are the sets of pairs of atoms separated by one and two covalent bonds, respectively. The bond lengths and bond angles are set to $r_{ij} = 1.526$Å (for all $(i, j) \in M_1$) and $\theta_{ij} = 109.5°$ (for all $(i, j) \in M_2$), respectively. Torsion angles are obtained by selecting first one value ω from the set $\{60°, 180°, 300°\}$ and another one from the set $\{\omega + i : i = -5°, \ldots, 5°\}$. Both of these selections are random. To generate distances, it is necessary to calculate Cartesian coordinates for each atom of the chain. This can be done, for example, using the procedure described in [Phillips et al., 1996]. For each molecule, described by the selection of the torsion angles, we define the set S using a cut-off value of 4Å. That is, $(i, j) \in S$ if and only if $d_{ij} < 4$. The pairs of atoms (i, j) selected, associated to the distances d_{ij}, constitute an instance for the molecular distance geometry problem. For a complete description of this set of instances, see [Lavor, 2005].

		sBB		VNS		SobolOpt	
Atoms	Variables	OF Value	Time	OF Value	Time	OF Value	Time
8	24	0	0.22	0	1.21	0	13.56
27	81	0	30.39	0	34.01	0	300.285
64	192	0	2237.73	0	398.875	0	2765.13

Table 1.1. Computational results for the Moré instances. Timings are in seconds of user CPU time.

		sBB		VNS		SobolOpt	
Atoms	Variables	OF Value	Time	OF Value	Time	OF Value	Time
5	15	0	0.02	0	0.48	0	0.57
10	30	0	1.12	0	7.06	0	69.71
20	60	0	2.25	0	49.99	0	411.152
30	90	0	488.87	0	352.06	0	1634.09
40	120	-	-	0.09	1258.13	0.547	2376.01
50	150	-	-	0	673.48	0	3002.88

Table 1.2. Computational results for the Lavor instances. Missing values are due to excessive computational requirements. Timings are in seconds of user CPU time.

3.2 Numerical results

All computations were performed on an Intel Xeon 2.8GHz with 2GB RAM running Linux. The local NLP optimization code we used to perform the local descents is SNOPT v.5 [Gill, 1999]. The global optimization algorithms were implemented as global solvers in the $oo\mathcal{OPS}$ optimization framework [Liberti et al., 2001, Liberti, 2005]. As such, they are not specially fine-tuned to solve the MDGP — this, together with the choice not to employ smoothing methods, explains the relatively small size of the largest molecules we can tackle.

The results are reported in Tables 1.1 (for the Moré instances) and 1.2 (for the new Lavor instances). The global optimum (with value 0) was found in all of the tested instances but the Lavor instance with 40 atoms. Three general trends emerge:

1 the Lavor instances, on average, are harder to solve than the Moré instances. In particular, one of the randomly generated Lavor instance (the one with 40 atoms), was so hard to solve that we could not reach the global optimum with any of the proposed global optimization algorithms;

2 the deterministic sBB algorithm is the fastest method for solving small to medium-sized instances. This result is rather surprising, as sBB is usually slower than heuristic methods;

3 both VNS and SobolOpt usually manage to find the correct solution, but VNS is faster. However, as the size of the molecule grows, the performance difference decreases.

Here are some notes and remarks about these computational experiments.

- The results obtained by the SobolOpt solver might be improved by careful tuning of parameters. Our tests were run with all the default parameter values. On the other hand, we did spend some time tuning the parameters of the VNS solver. It appears that for very hard instances (like the Lavor with 40 atoms), we can get nearer the global optimum by setting a very high $k_{\max}$ parameter (possibly in the region of 10^3) and a number of trials in each neighbourhood (i.e. maximum number of local searches to carry out in each neighbourhood) to something between 5 and 15. This slows the search down considerably, but it does produce better results.

- We also conducted a number of tests using a different VNS neighbourhood structure. In practice, we focused the search on the corners of each hyper-rectangle $H_k(x^*)$: this made it possible to sample starting points from disjoint neighbourhoods, but it affected the convergence properties of the VNS (certain regions were not sampled extensively). Surprisingly, VNS managed to locate the global optimum for all instances but the Lavor with 40 atoms, where it succeeded in locating a point with extremely low (albeit clearly non-zero) objective function value. This constitutes numerical evidence that the best minima are to be found near the extreme points of the hyper-rectangle.

- The convergence speed of the sBB solver can be improved by relaxing the ϵ tolerance (set by default to 1×10^{-3}).

- One of the reasons why sBB is so effective on this problem is that it has a known globally optimal value (0), and that the automatic convexification of sBB provides a tight lower bound (namely, 0 itself). Since the lower bound is so tight, many regions are discarded very soon in the Branch-and-Bound tree.

4. Conclusion

In this paper we described computational experiments performed in globally solving instances of the molecular distance geometry problem. We discussed three global optimization methods: a deterministic one (spatial Branch-and-Bound) and two heuristic ones (Variable Neighbourhood Search and Multi Level Single Linkage with deterministic low-discrepancy sampling based on Sobol' sequences). We solved instances from two different classes: one taken from the literature and the other new. Rather surprisingly, sBB is clearly the best choice for small-scale problems, both because it provides a guarantee of ε-global optimality and because it is faster than the other methods. SobolOpt and VNS perform rather well for medium to large-scale problems, with VNS being faster than SobolOpt.

Acknowledgments

We wish to acknowledge the invaluable help of Dr. Sergei Kucherenko who provided the SobolOpt global solver. We also would like to thank CNPq and FAPERJ for their support. One of the authors (LL) is grateful to Prof. Nelson Maculan for financial support.

References

Adjiman, C.S., Dallwig, S., Floudas, C.A., and Neumaier, A. (1998). A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical Advances. *Computers & Chemical Engineering*, 22(9):1137–1158.

An, L.T. Hoai (2003). Solving large scale molecular distance geometry problems by a smoothing technique via the Gaussian transform and d.c. programming. *Journal of Global Optimization*, 27:375–397.

Brimberg, J. and Mladenović, N. (1996). A variable neighbourhood algorithm for solving the continuous location-allocation problem. *Studies in Location Analysis*, 10:1–12.

Crippen, G.M. and Havel, T.F. (1988). *Distance Geometry and Molecular Conformation*. Wiley, New York.

Drazić, M., Lavor, C., Maculan, N., and Mladenović, N. (2004). A continuous vns heuristic for finding the tridimensional structure of a molecule. *Le Cahiers du GERAD*, G-2004-22.

Gill, P.E. (1999). *User's Guide for SNOPT 5.3*. Systems Optimization Laboratory, Department of EESOR, Stanford University, California.

Hansen, E. (1992). *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc., New York.

Hansen, P. and Mladenović, N. (2001). Variable neighbourhood search: Principles and applications. *European Journal of Operations Research*, 130:449–467.

Hendrickson, B.A. (1995). The molecule problem: exploiting structure in global optimization. *SIAM Journal on Optimization*, 5:835–857.

Kucherenko, S. and Sytsko, Yu. ((to appear) 2004). Application of deterministic low-discrepancy sequences to nonlinear global optimization problems. *Computational Optimization and Applications*.

Lavor, C. ((to appear) 2005). On generating instances for the molecular distance geometry problem. In [Liberti and Maculan, 2005].

Liberti, L. (2004). *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK.

Liberti, L. ((to appear) 2005). Writing global optimization software. In [Liberti and Maculan, 2005].

Liberti, L. and Maculan, N., editors ((to appear) 2005). *Global Optimization: from Theory to Implementation*. Kluwer, Dordrecht.

Liberti, L., Tsiakis, P., Keeping, B., and Pantelides, C.C. (2001). *oo$\mathcal{OPS}$*. Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, 1.24 edition.

Locatelli, M. and Schoen, F. (1996). Simple linkage: Analysis of a threshold-accepting global optimization method. *Journal of Global Optimization*, 9:95–111.

Locatelli, M. and Schoen, F. (1999). Random linkage: a family of acceptance/rejection algorithms for global optimization. *Mathematical Programming*, 85(2):379–396.

Mladenović, N., Petrović, J., Kovačević-Vujčić, V., and Čangalović, M. (2003). Solving a spread-spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operations Research*, 151:389–399.

Moré, J.J. and Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7:814–836.

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

Pardalos, P.M. and Romeijn, H.E., editors (2002). *Handbook of Global Optimization*, volume 2. Kluwer Academic Publishers, Dordrecht.

Phillips, A.T., Rosen, J.B., and Walke, V.H. (1996). Molecular structure determination by convex underestimation of local energy minima. In Pardalos, P.M., Shalloway, D., and Xue, G., editors, *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding*, volume 23, pages 181–198, Providence. American Mathematical Society.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992, reprinted 1997). *Numerical Recipes in C, Second Edition*. Cambridge University Press, Cambridge.

Rinnooy-Kan, A.H.G. and Timmer, G.T. (1987a). Stochastic global optimization methods; part I: Clustering methods. *Mathematical Programming*, 39:27–56.

Rinnooy-Kan, A.H.G. and Timmer, G.T. (1987b). Stochastic global optimization methods; part II: Multilevel methods. *Mathematical Programming*, 39:57–78.

Ryoo, H.S. and Sahinidis, N.V. (1995). Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566.

Schoen, F. (1998). Random and quasi-random linkage methods in global optimization. *Journal of Global Optimization*, 13:445–454.

Schoen, F. (1999). Global optimization methods for high-dimensional problems. *European Journal of Operations Research*, 119:345–352.

Schoen, F. (2002). Two-phase methods for global optimization. In [Pardalos and Romeijn, 2002], pages 151–177.

Smith, E.M.B. (1996). *On the Optimal Design of Continuous Processes*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London.

Smith, E.M.B. and Pantelides, C.C. (1999). A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478.

Tawarmalani, M. and Sahinidis, N.V. (2002). Exact algorithms for global optimization of mixed-integer nonlinear programs. In [Pardalos and Romeijn, 2002], pages 1–63.