Intl. Trans. in Op. Res. 15 (2008) 1–17

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

A Branch-and-Prune algorithm for the Molecular Distance Geometry Problem

Leo Liberti[a], Carlile Lavor[b] and Nelson Maculan[c]

[a]*LIX, Ecole Polytechnique, F-91128 Palaiseau, France,*
[b]*Departamento de Matemática Aplicada (IMECC-UNICAMP), Universidade Estadual de Campinas, CP 6065, 13081-970, Campinas-SP, Brazil,*
[c]*COPPE, Universidade Federal do Rio de Janeiro (UFRJ) C.P. 68511, Rio de Janerio 21945-970, Brazil*
E-mails: liberti@lix.polytechnique.fr [Liberti]; clavor@ime.unicam.br [Lavor]; maculan@cos.ufrj.br [Maculan]

Received 25 April 2007; received in revised form 19 October 2007; accepted 27 October 2007

Abstract

The Molecular Distance Geometry Problem consists in finding the positions in $\mathbb{R}^3$ of the atoms of a molecule, given some of the inter-atomic distances. We show that under an additional requirement on the given distances this can be transformed to a combinatorial problem. We propose a Branch-and-Prune algorithm for the solution of this problem and report on very promising computational results.

Keywords: Molecular Distance Geometry Problem; Branch-and-Prune algorithm

1. Introduction

We present a discrete formulation and a very fast and accurate solution method for a subclass of instances of the Molecular Distance Geometry Problem (MDGP) (Moré and Wu, 1997, 1999; Crippen and Havel, 1988; Dong and Wu, 2002; Hendrickson, 1995). The MDGP is related to the determination of the tridimensional structure of a molecule based on knowledge of some distances between pairs of atoms. The tridimensional structure is very important because it is associated to the physical and chemical properties of the molecule.

The MDGP can be seen as finding a distance-preserving immersion in $\mathbb{R}^3$ of a given undirected weighted graph $G = (V, E, d)$, so it can be very naturally cast as a continuous search problem. Under three additional assumptions that are satisfied by most proteins (a very interesting and rich class of molecules), we transform the MDGP to a discrete search problem. The assumptions are:

1. covalent bond lengths and angles are known;
2. the molecule has the shape of a protein backbone, i.e. it is a sequence of n atoms such that there is a covalent bond between every pair of consecutive atoms;

3. all distances between atoms separated by three covalent bonds are known;
4. no bond angle is equal to $k\pi$, for $k \in \mathbb{Z}$.

Naturally, distances between atoms separated by two covalent bonds can be easily calculated from the covalent bond lengths and bond angles. We remark that if Assumption 1 above is strengthened to require knowledge of the distances between atoms separated by *four* covalent bonds, then the problem becomes polynomially solvable (Dong and Wu, 2002; Eren et al., 2004). Further considerations on the complexity of the MDGP problem under assumption 1 are given in Lavor et al. (2006).

It should be noted that we refer to the MDGP as a precisely formalized decision problem, and not as a practical chemical problem. We therefore make three assumptions that in real life may be easily challenged: (a) a subset of exact (as opposed to approximate) distances is given as part of the input; (b) no measurement errors occur; (c) the optimal three-dimensional (3D) embedding of the graph is not influenced by a potential energy minimization term on the objective function. We refer to Klepeis et al. (1999) for a more realistic formulation.

In Section 2, we show a discrete formulation for the problem. In Section 3, we describe the Branch-and-Prune algorithm, which will be applied to the solution of the MDGP. The computational results are discussed in Section 4. Section 5 concludes the paper.

2. Molecular Distance Geometry Problem

Formally, the MDGP can be defined as the problem of finding Cartesian coordinates $x_1, \ldots, x_n \in \mathbb{R}^3$ of the atoms of a molecule such that for all $(i,j) \in S$,

$$||x_i - x_j|| = d_{ij},$$

where S is the set of pairs of atoms (i,j) whose Euclidean distances d_{ij} are known. If all distances are given, the problem can be solved in linear time (Dong and Wu, 2002). If there is an order on the atoms such that the given distances form cliques on each set of five contiguous atoms, the problem is polynomially solvable (Eren et al., 2004). In general, however, the problem is NP-hard (Saxe, 1979).

The MDGP is usually formulated as a continuous least-squares minimization problem, where the objective function is as follows:

$$f(x_1, \ldots, x_n) = \sum_{(i,j) \in S} (||x_i - x_j||^2 - d_{ij}^2)^2. \tag{1}$$

Obviously, $(x_1^*, \ldots, x_n^*)$ solve the problem if and only if $f(x_1^*, \ldots, x_n^*) = 0$.

Note that, as stated above, the MDGP bears no connection whatsoever with molecules. In fact, the MDGP appears in such diverse application fields as 3D graph drawing (Cruz and Twarog, 1996) and network design (Eren et al., 2004). Our assumption that all distances between atoms separated by one, two, and three covalent bonds are known can be expressed as an additional condition on the set S of distances, namely that S can be partitioned into two disjoint sets E, F

of distances where

$$E = \{(i, i+1) \mid 1 \leqslant i \leqslant n-1\} \cup \{(i, i+2) \mid 1 \leqslant i \leqslant n-2\} \cup \{(i, i+3) \mid 1 \leqslant i \leqslant n-3\},$$

and

$$F = \{(i,j) \mid j - i \geqslant 4\}.$$

We also assume that for all pairs of atoms in F, the distances are shorter than a given cut-off value Δ (usually this is taken to be 5 Å using, for example, NMR analysis; Creighton, 1993; Schlick, 2002), that is, $d_{ij} \leqslant \Delta \ \forall (i,j) \in F$.

As we shall show in Section 2.1, for each group of four consecutive atoms, if we know all the distances between them and fix the first three, with probability 1 (see Section 2.2) the fourth atom can only have two possible symmetric placements. This allows us to give a discrete formulation for the considered problem.

2.1. Discrete formulation

Consider a molecule as being a sequence of n atoms with Cartesian coordinates given by $x_1, \dots, x_n \in \mathbb{R}^3$ and such that there is a covalent bond between every pair of atoms $(i, i+1)$, for $i = 1, \dots, n-1$. The *bond length* r_i is the Euclidean distance between atoms $i-1$ and i (i.e. $r_i = d_{i-1,i}$ for all $i = 2, \dots, n$). The *bond angle* $\theta_i \in [0, \pi]$ is the angle between the segments joining atoms $i-2$, $i-1$ and $i-1$, i (for all $i = 3, \dots, n$). The *torsion angle* $\omega_i \in [0, 2\pi]$ is the angle between the normals through the planes defined by the atoms $i-3$, $i-2$, $i-1$ and $i-2$, $i-1$, i (for all $i = 4, \dots, n$) (see Fig. 1).

In most molecular conformation calculations, all covalent bond lengths and bond angles are assumed to be known *a priori* (Phillips et al., 1996). Thus, the first three atoms in the sequence can be fixed and the fourth atom is determined by the torsion angle ω_4. The fifth atom can be determined by the torsion angles ω_4 and ω_5, and so on. So, given all bond lengths $r_2, r_3, \dots, r_n$,

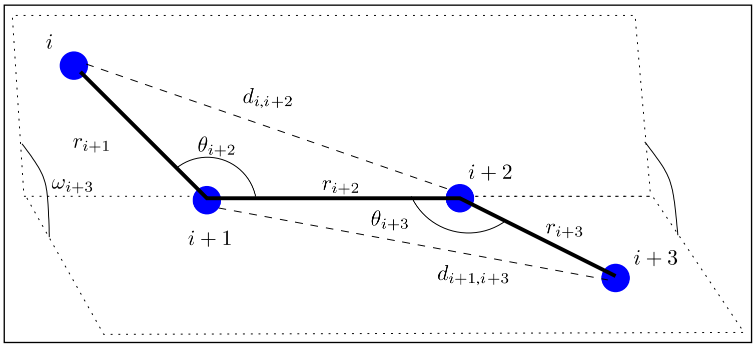

Fig. 1. Definitions of bond lengths, bond angles, and torsion angles.

bond angles $\theta_3, \theta_4, \ldots, \theta_n$, and torsion angles $\omega_4, \omega_5, \ldots, \omega_n$ of a molecule with n atoms, the Cartesian coordinates $x_i = (x_{i1}, x_{i2}, x_{i3})$ for each atom i in the molecule can be obtained using the following formulae (Phillips et al., 1996):

$$\begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ 1 \end{bmatrix} = B_1 B_2 \cdots B_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \forall\, i = 1, \ldots, n, \tag{2}$$

where

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -1 & 0 & 0 & -r_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_3 = \begin{bmatrix} -\cos\theta_3 & -\sin\theta_3 & 0 & -r_3\cos\theta_3 \\ \sin\theta_3 & -\cos\theta_3 & 0 & r_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and

$$B_i = \begin{bmatrix} -\cos\theta_i & -\sin\theta_i & 0 & -r_i\cos\theta_i \\ \sin\theta_i\cos\omega_i & -\cos\theta_i\cos\omega_i & -\sin\omega_i & r_i\sin\theta_i\cos\omega_i \\ \sin\theta_i\sin\omega_i & -\cos\theta_i\sin\omega_i & \cos\omega_i & r_i\sin\theta_i\sin\omega_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

for $i = 4, \ldots, n$. We call B_i the *torsion matrices* and denote by $C_i = \prod_{j \leqslant i} B_j$ the *cumulative torsion matrices*. For every four consecutive atoms x_i, x_{i+1}, x_{i+2}, x_{i+3} we can express the cosine of the torsion angle ω_{i+3} in terms of the distances r_{i+1}, $d_{i+1,i+3}$, $d_{i,i+3}$ and the bond angle θ_{i+2}, θ_{i+3} by using the cosine law for torsion angles (Pogorelov, 1987, p. 278), as follows:

$$\cos\omega_{i+3} = \frac{r_{i+1}^2 + d_{i+1,i+3}^2 - 2r_{i+1}d_{i+1,i+3}\cos\theta_{i+2}\cos\theta_{i+3} - d_{i,i+3}^2}{2r_{i+1}d_{i+1,i+3}\sin\theta_{i+2}\sin\theta_{i+3}}. \tag{4}$$

Hence, if we know all the bond lengths (r_i), bond angles (θ_i), and distances between atoms separated by three covalent bonds ($d_{i,\,i+3}$), we can calculate the cosine of the torsion angles defined by the atoms i, $i+1$, $i+2$, $i+3$ for $i = 1, \ldots, n-3$. We note in passing that in order for (4) to hold, we obviously need the denominator to be nonzero.

Using the bond lengths r_2, r_3 and the bond angle θ_3, we can determine the torsion matrices B_2 and B_3 and obtain

$$x_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$x_2 = \begin{pmatrix} -r_2 \\ 0 \\ 0 \end{pmatrix},$$

$$x_3 = \begin{pmatrix} r_3\cos\theta_3 - r_2 \\ r_3\sin\theta_3 \\ 0 \end{pmatrix},$$

fixing the first three atoms of the molecule. Because we also know the distance d_{14}, by (4) we can obtain the value $\cos\omega_4$. Thus, the sine of the torsion angle ω_4 can have only two possible values: $\sin\omega 4 = \pm\sqrt{1-(\cos\omega_4)^2}$. Consequently, we obtain only two possible positions x_4, x'_4 for the fourth atom:

$$x_4 = \begin{bmatrix} -r_2 + r_3\cos\theta_3 - r_4\cos\theta_3\cos\theta_4 + r_4\sin\theta_3\sin\theta_4\cos\omega_{14} \\ r_3\sin\theta_3 - r_4\sin\theta_3\cos\theta_4 - r_4\cos\theta_3\sin\theta_4\cos\omega_4 \\ -r_4\sin\theta_4\sqrt{1-(\cos\omega_{14})^2} \end{bmatrix},$$

$$x'_4 = \begin{bmatrix} -r_2 + r_3\cos\theta_3 - r_4\cos\theta_3\cos\theta_4 + r_4\sin\theta_3\sin\theta_4\cos\omega_4 \\ r_3\sin\theta_3 - r_4\sin\theta_3\cos\theta_4 - r_4\cos\theta_3\sin\theta_4\cos\omega_4 \\ r_4\sin\theta_4\sqrt{1-(\cos\omega_4)^2} \end{bmatrix},$$

along with the respective torsion matrices B_4, B'_4 such that

$$(x_4, 1)^\top = C_3 B_4 (0,0,0,1)^\top,$$

$$(x'_4, 1)^\top = C_3 B'_4 (0,0,0,1)^\top,$$

where C_3 is a cumulative torsion matrix. This dichotomy, shown pictorially in Fig. 2, is the basic reason why this problem can be formulated combinatorially.

For the fifth atom, we will obtain four possible positions, one for each combination of $\pm\sqrt{1-(\cos\omega_4)^2}$ and $\pm\sqrt{1-(\cos\omega_5)^2}$. By an easy induction argument, we can see that for the ith atom we obtain 2^{i-3} possible positions. Hence, for a molecule shaped as a sequence (a linear chain) of n atoms, we get 2^{n-3} possible sequences of torsion angles $\omega_4, \omega_5, \ldots, \omega_n$, each defining a different tridimensional structure. Using the matrices B_i defined above, we can convert a sequence

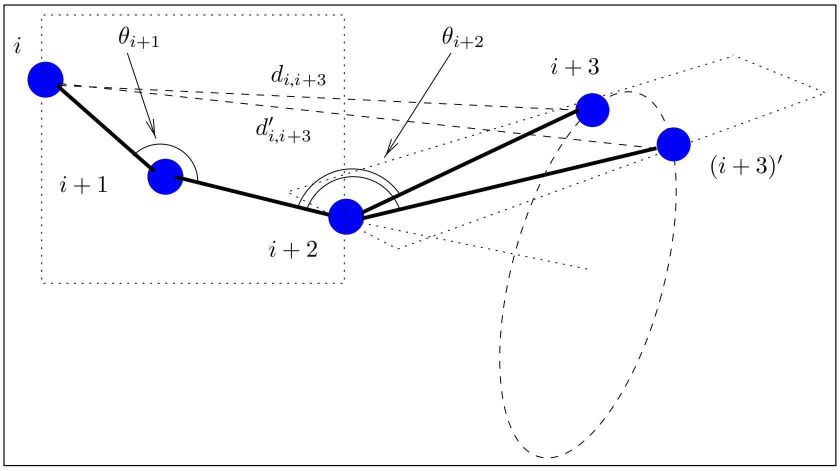

Fig. 2. Discretization of the problem. The atom i+3 can only be in the two shown positions in order to be feasible with the distance $d_{i,i+3}$.

of torsion angles into Cartesian coordinates $x_1, \ldots, x_n \in \mathbb{R}^3$. Thus, this problem has a finite search space. To test a candidate solution we simply use the function f defined in (1); the candidate solution $(x_1, \ldots, x_n)$ will be a valid solution if and only if $f(x_1, \ldots, x_n) = 0$.

The discussion above can be summarized in the following theorem.

Theorem 2.1. *Consider a sequence M of n atoms such that:*

(i) *atom i is covalently bonded to atom i+1 for all* $i \leqslant n-1$;
(ii) *all bond angles and bond lengths are known*;
(iii) *no bond angle is a multiple of* π;
(iv) *all distances between atom i and i+3 are known, for all* $1 \leqslant i \leqslant n-3$.

Then there is a finite number of distinct immersions $p{:}M \rightarrow \mathbb{R}^3$ *such that:*

(a) $p(1) = (0,0,0)$, $p(2)_1 = 0$, $p(2)_2 = 0$, $p(3)_1 = 0$ *(where* $p(i)_k$ *is the kth coordinate of* $p(i)$ *for* $k \leqslant 3$, $i \leqslant n$*);*
(b) *for all atoms i,j with known atomic distance* d_{ij} *we have*:

$$||p(i) - p(j)|| = d_{ij}.$$

We remark that the idea that three distances suffice to determine at most two possible positions of the fourth atom, which is the basis on which our algorithm rests, was also very recently used in Wu et al. (in press) in the framework of the geometric build-up algorithm (Dong and Wu, 2002).

2.2. Undiscretizable instances

As has been remarked, the instances of the considered problem have a finite number of valid solutions with probability 1. The only case where an instance is not susceptible of a discrete formulation is when there is a subsequence of three consecutive atoms i, i+1, i+2, where the bond

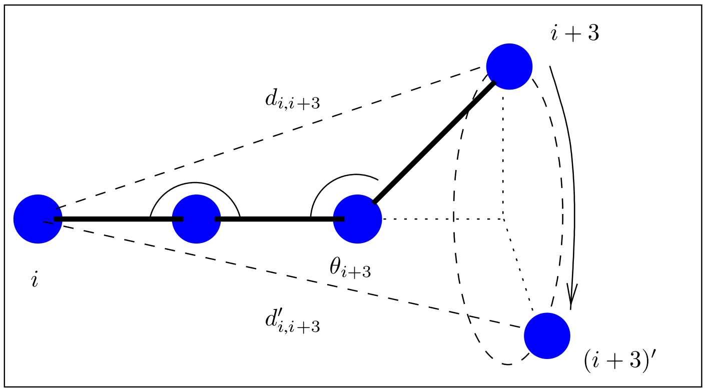

Fig. 3. An instance that cannot be discretized. The $i+3$-rd atom can be on any position on the circle shown without affecting the feasibility of the distance $d_{i,i+3}$.

angle θ_{i+2} is $k\pi$ for $k \in \mathbb{Z}$: because ω_{i+3} is an angle between two normal vectors to given planes, ω_{i+3} is undefined when at least one of the planes is undefined, i.e. if the two vectors defining the plane are collinear. In other words, if the bond angle θ_{i+2} is a multiple of π, we have the situation depicted in Fig. 3, where $d_{i,i+3}$ is feasible for every position of atom $i+3$ on the circle shown in the figure. Because the set $\{\pi\}$ has measure 0 in $[0, 2\pi]$, the probability that any given instance is discretizable is 1. In any case the 'undiscretizable cases' do not often occur in practice.

3. The algorithm

In this section we shall present a Branch-and-Prune (BP) algorithm designed for solving the considered problem. The approach is very simple and mimics the structure of the problem closely: at each step we can place the ith atom in two possible positions x_i, x'_i. We then branch the search and prune away the infeasible branches. More precisely, for each position we check feasibility with all distance pairs $(j,i) \in F$ by checking that $(||x_j - x_i||^2 - d^2_{j,i})^2 < \varepsilon$, where $\varepsilon > 0$ is a given tolerance. There are four possible outcomes:

1. both x_i, x'_i are feasible: in this case we store both positions and explore both branches in a depth-first fashion;
2. only x_i is feasible: we only store the feasible position x_i and prune the infeasible branch x'_i;
3. only x'_i is feasible: we only store the feasible position x'_i and prune the infeasible branch x_i;
4. neither position is feasible: we prune both branches and backtrack the search.

Notice that this algorithm, as described, will find all solutions to the problem. If we are only interested in one, we can stop the search as soon as we have placed the last atom in a feasible position.

Let T be a graph representation of the search tree. Initially, T is initialized to $1 \rightarrow 2 \rightarrow 3$ because the first three atoms can be fixed to feasible positions x_1, x_2, x_3 as explained earlier. By the current rank of the search tree we mean the index of the atom being placed at the current node. At each search tree node of rank i we store:

- the position $x_i \in \mathbb{R}^3$ of the ith atom;
- the cumulative product $C_i = \prod_{j=1}^{i} B_j$ of the torsion matrices;

- a pointer to the parent node $P(i)$;
- pointers to the subnodes $L(i), R(i)$ (initialized to a dummy value PRUNED if infeasible).

Notice that the edge structure of the graph T is encoded in the operators $P(), L(), R()$ defined at each node. The recursive procedure at rank $i-1$ is given in Algorithm 1. Let $y = (0,0,0,1)^\top$, $\varepsilon > 0$ a given tolerance and v a node with rank $i-1$ in the search tree T.

3.1. Detailed example

We now discuss the application of Algorithm 1 to a simple example (artificially generated as explained in Lavor (2006), see also Section 4.2).

The instance in question (called lavor11_7), with all bond lengths set to 1.526 Å and bond angles set to 1.91 radians, has 11 atoms with distances in F given by

$$\begin{aligned}
\delta(2) &= \{9\}, d_2^F = \{3.387634917\},\\
\delta(3) &= \{8,9,10\}, d_3^F = \{3.96678038, 3.003368265, 3.796280236\},\\
\delta(4) &= \{8,9,10\}, d_4^F = \{2.60830758, 2.102385055, 3.159309539\},\\
\delta(5) &= \{9,10\}, d_5^F = \{2.689078459, 3.132251169\},\\
\delta(6) &= \{10\}, d_6^F = \{3.557526815\},\\
\delta(7) &= \{11\}, d_7^F = \{3.228657023\},
\end{aligned}$$

where $\delta(i)$ indicates the atoms j such that $d_{ij} \leqslant 4$ Å (the cut-off value). The distances in E are of course $\delta(i) = \{i+1, i+2, i+3\}$ for all $i \leqslant n-3$, $\delta(n-2) = \{n-1, n\}$, $\delta(n-1) = \{n\}$. The vector of the distances in E is:

$$\begin{aligned}
d^E = (&1.526,\ 2.491389536,\ 3.83929637,\\
&1.526,\ 2.491389536,\ 3.831422399,\\
&1.526,\ 2.491389536,\ 3.835602674,\\
&1.526,\ 2.491389535,\ 3.030585263,\\
&1.526,\ 2.491389534,\ 2.899348439,\\
&1.526,\ 2.491389535,\ 3.086914764,\\
&1.526,\ 2.491389536,\ 2.788611167,\\
&1.526,\ 2.491389536,\ 2.888815709,\\
&1.526, 2.491389537,\\
&1.526),
\end{aligned}$$

where the ith line contains the distances among atoms i and $i+1$, $i+2$, $i+3$. Of course, the last two lines contain the distances among the atom $n-2$ and atoms $n-1$ and n, and the distance between the atom $n-1$ and n, respectively.

Algorithm 1. BP algorithm

```
0: BranchAndPrune($T$, $v$, $i$)
if ($i \leqslant n-1$) then
  COMPUTE THE POSSIBLE PLACEMENTS FOR $i$TH ATOM:
  calculate the torsion matrices $B_i, B_i'$ via Eq. (3);
  retrieve the cumulative torsion matrix $C_{i-1}$ from the parent node $P(v)$;
  compute $C_i = C_{i-1}$, $B_i$, $C_i' = C_{i-1}B_I'$ and $x_i, x_i'$ from $C_iy, C'y$;
  let $\lambda = 1$, $\rho = 1$;
  CHECK FEASIBILITY:
  for all $(j,i) \in F$ do
    let $\delta_{ji} = (||x_j - x_i||^2 - d_{ji}^2)^2$ and $\delta'_{ji} = (||x_j - x'_i||^2 - d_{ji}^2)^2$;
    if ($\delta_{ji} > \varepsilon$) then
      $\lambda = 0$;
    end if
    if ($\delta_{ji}' > \varepsilon$) then
      $\rho = 0$;
    end if
  end for
CREATE SUBNODES AS REQUIRED:
if ($\lambda = 1$) then
  create a node $z$, store $C_i$ and $x_i$ in $z$, let $P(z) = v$ and $L(v) = z$;
  set $T \leftarrow T \cup \{Z\}$;
  BranchAndPrune($T$, $z$, $i+1$);
else
  set $L(v) =$ PRUNED;
End if
if ($\rho = 1$) then
  create a node z', store $C_i$ and $x_i$ in $z'$, let $P(z) = v$ and $R(v) = z'$;
  set $T \leftarrow T \cup \{Z'\}$;
  BranchAndPrune($T$, $z'$, $i+1$);
else
  set $R(v) =$ PRUNED;
End if
else
  RANK $n$ REACHED, A SOLUTION WAS FOUND:
  solution stored in parent nodes ranked $n$ to 1, output by back-traversal;
end if
```

As can be seen from the BP tree given in Fig. 4 (this is actually the output of Algorithm 1 on the given instance), this instance has four solutions: the leaf nodes at rank 11—the rank is given by the number of the leftmost node in each row. Notice that the earliest node when some pruning occurs is at rank 7, i.e. no pruning occurs before the placement of the 8th atom. This happens because there are no distances $(j,k) \in F$ with $k < 8$, so each position for atoms with index $i < 8$ is

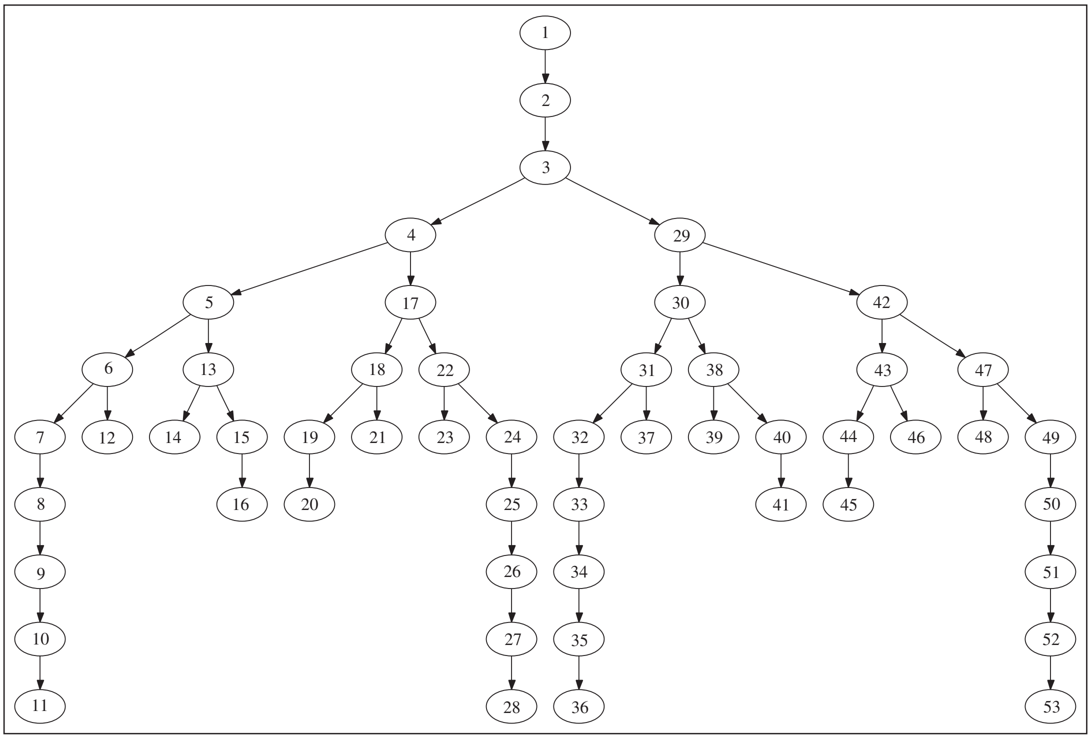

Fig. 4. The BP tree of the instance of Section 1.

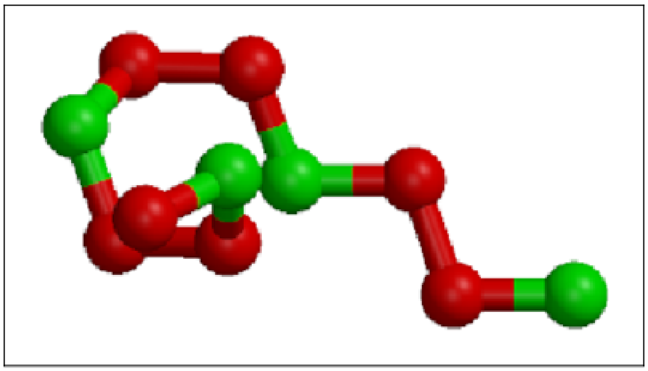

Fig. 5. One of the possible solutions of the lavor11_7 instance.

feasible (by construction of x_i, x'_i) with the distances in E. The only symmetry-breaking distances are in fact those in F. Again, there is pruning at ranks 8, 9, and 10, i.e. during the placement of atoms 9, 10, and 11, because there are distances $(j, k) \in F$ with $k = 9, 10, 11$. One of the solutions is shown in Fig. 5.

4. Computational experiments

In order to test the viability of the proposed method, we tested a class of randomly generated MDGP instances described in Lavor (2006). We present comparative results of BP and another existing MDGP software called dgsol (Moré and Wu, 1999). It turns out that BP is superior to dgsol for speed and solution accuracy, and inferior as regards memory requirements and running time reliability.

4.1. Software testbeds

The software code dgsol (Moré and Wu, 1999) (version 1.3) can be freely downloaded from http://www.mcs.anl.gov/more/dgsol/. The algorithm implemented by the dgsol code is very different from ours. First, it targets a more general problem class: the Molecular Distance Geometry Problem with Distance Bounds. In this problem, lower and upper bounds to atomic distances are known, rather than the exact atomic distances. Because these are usually estimated through NMR techniques, it is realistic to assume that there is an experimental error in the measurements (our approach does not consider this issue yet). Secondly, dgsol needs to make no assumption whatsoever about the distances of triplets and quadruplets of consecutive atoms being known. Thirdly, dgsol is based on a continuous smoothing of the original problem to a form that has fewer local minima. An ordinary NLP optimization method is then applied to the modified problem, and the optimum is traced back to the original problem. This is a fully continuous optimization algorithm, whereas BP is a discrete method.

It turns out that the main advantages of BP over dgsol are:

1. tractability of larger instances;
2. higher solution accuracy;
3. BP can potentially find *all* feasible solutions, not just one.

By contrast, the main advantages of dgsol over BP are:

1. it targets a larger class of problems;
2. its running time seems to increase very slowly (and regularly) as a function of the number of atoms in the molecule, at least when the set of given distances is comparatively small;
3. the amount of memory needed to complete the search is negligible.

The BP algorithm behaves very unpredictably with respect to the amount of memory needed, sometimes requiring over 1 GB RAM for relatively small molecules (40 atoms), sometimes solving 1000-atom instances in a few seconds requiring very little memory.

4.2. Lavor instances

These instances, described in Lavor (2006), are based on the model proposed by Phillips et al. (1996), whereby a molecule is represented as a linear chain of atoms. Bond lengths and angles are kept fixed, and a set of likely torsion angles is generated randomly. Depending on the initial choice of bond lengths and angles, the Lavor instances give rather more realistic models of proteins than other randomly generated instances do (see for example the instances described in Moré and Wu, 1999). Figure 5 gives an example of a Lavor instance. In the numerical tables, we labeled the Lavor instances by lavor n–m, where n is the number of atoms in the molecule and m is an instance ID (because there is a random element of choice in the generation of the Lavor instances, many different instances can be generated having the same atomic size).

We generated 10 different Lavor instances for each size $n = 10, \ldots, 70$ ('small set'), and four different Lavor instances for each size n in $\{100i | 1 \leqslant i \leqslant 10\}$ ('large set').

4.3. Hardware and memory considerations

All tests have been carried out on an Intel Pentium IV 2.66 GHz with 1 GB RAM, running Linux. The code implementing the BP algorithm has been compiled by the GNU C++ compiler v.3.2 with the -02 flag. As mentioned above, BP can be very memory demanding. We deliberately took the choice of employing a low-end PC with just 1 GB RAM to show just how powerful this technique can be even with modest hardware.

Table 1
Computational results for a sample of small and large Lavor instances

Instance			dgsol		BP-One		BP-All	
Name	n	$\|S\|$	CPU	LDE	CPU	LDE	CPU	LDE
lavor10_0	10	33	0.02	1.57E – 5	0.00	5.36E – 10	0.00	4
lavor15_0	15	57	0.10	4.04E – 5	0.00	2.84E – 09	0.00	16
lavor20_0	20	105	0.14	2.77E – 5	0.00	6013E – 09	0.00	8
lavor25_0	25	131	0.84	1.18E – 4	0.00	1.38E – 09	0.00	8
lavor30_0	30	169	0.40	1.75E – 5	0.00	1.23E – 09	0.00	2
lavor35_0	35	171	0.81	9.33E – 5	0.00	1.52E – 09	0.00	64
lavor40_0	40	295	2.84	0.096	0.00	2.87E – 09	0.00	2
lavor45_0	45	239	3.33	0.170	0.00	6.92E – 09	0.00	2
lavor50_0	50	271	3.45	0.696	0.00	3.96E – 08	0.46	4096
lavor55_0	55	551	5.80	0.257	0.00	2.66E – 09	0.00	64
lavor60_0	60	377	5.15	0.049	0.00	3.51E – 09	0.00	64
lavor65_0	65	267	2.61	0.065	0.00	7.76E – 10	–	–
lavor70_0	70	431	8.73	0.107	0.02	1.64E – 09	–	–
lavor100_0	100	605	6.95	0.167	2.26	4.01E – 09	–	–
lavor200_0	200	1844	63.52	0.395	0.00	5.66E – 08	–	–
lavor300_0	300	2505	100.99	0.261	0.03	1.56E – 08	–	–
lavor400_0	400	2600	182.21	0.767	0.01	3.35E – 09	–	–
lavor500_0	500	4577	329.29	0.830	0.27	4.69E – 07	–	–
lavor600_0	600	5473	299.76	0.700	0.01	4.94E – 08	–	–
lavor700_0	700	4188	281.34	0.569	0.16	1.83E – 06	–	–
lavor800_0	800	6850	570.20	0.528	3.34	3.37E – 06	–	–
lavor900_0	900	7965	550.26	0.549	3.08	5.62E – 06	–	–
lavor1000_0	1000	8229	844.52	0.695	0.81	2.04E – 06	–	–

Missing values are due to excessive memory requirements (> 1 GB RAM).

The BP algorithm is in general very fast, because all it does is test feasibility with the known distances at each branched node. However, exploring the search space may require a lot of memory, especially if no pruning occurs early in the run. Consequently, when the physical RAM of the test machine is exhausted, and the operating system starts swapping to disk, the total CPU elapsed time size becomes unmanageable. Thus, it was decided to kill all jobs requiring more than 1 GB RAM. In particular, we solved almost all the Lavor instances in the 'small set' and found one solution for each of the Lavor instances in the 'large set'.

4.4. Comparative results

The full results table for the complete test suite includes 655 instances and spans 14 pages: thus, only a sample will be presented in detail. The averages, however, are taken with respect to the whole suite. The ε parameter of Algorithm 1 was set to 1×10^{-3} for all tests.

Table 1 contains detailed results for the sample. The instances are described by their name, their atomic size n and the number of given distances $|S|$. Note that in order to use dgsol, the lower and upper bounds to these distances were set to $d_{ij} \pm 5 \times 10^{-4}$. Other than this, dgsol was used with

Table 2
Average statistics for Lavor instances (over 10 instances for the set of small instances and over 4 for the set of large instances).

Instance	dgsol (average)		BP-One (average)		BP-All (average)	
n	CPU	LDE	CPU	LDE	CPU	#Sol
10	0.03	4.40E – 01	0.00	1.19E – 09	0.00	1.54E+01
15	0.08	1.96E – 02	0.00	1.23E – 09	0.00	3.72E+01
20	0.23	3.20E – 03	0.00	1.94E – 09	0.00	6.09E+01
25	0.56	1.58E – 02	0.00	1.58E – 09	0.02	1.14E+02
30	0.65	1.03E – 02	0.00	3.45E – 09	0.01	2.65E+02
35	1.10	5.43E – 02	0.00	2.84E – 09	0.10	3.35E+03
40	1.41	2.61E – 02	0.00	5.75E – 09	0.02	8.48E+02
45	2.13	5.80E – 02	0.00	6.25E – 09	0.12	2.48E+03
50	2.54	1.65E – 01	0.00	6.62E – 09	0.16	1.80E+03
55	4.10	7.29E – 02	0.00	5.53E – 09	0.03	4.28E+02
60	4.47	1.59E – 01	0.00	6.44E – 09	0.04	3.49E+02
65	4.64	1.16E – 01	0.00	8.37E – 09	1.21	3.80E+03
70	7.63	9.28E – 01	0.01	1.07E – 08	–	–
100	10.57	3.53E – 01	0.57	2.46E – 09	–	–
200	57.34	3.61E – 01	0.02	2.00E – 08	–	–
300	109.91	4.03E – 01	0.03	1.90E – 08	–	–
400	173.54	6.69E – 01	0.10	1.05E – 08	–	–
500	273.66	6.19E – 01	0.16	4.92E – 07	–	–
600	351.15	5.75E – 01	0.01	5.47E – 08	–	–
700	365.37	7.03E – 01	0.82	2.65E – 06	–	–
800	583.65	6.54E – 01	2.72	1.90E – 06	–	–
900	714.39	6.88E – 01	1.68	2.85E – 06	–	–
1000	787.30	6.88E – 01	0.41	1.45E – 06	–	–

all default parameter values. The results refer to three methods: dgsol, BP stopped after the first solution was found (BP-One), and BP run to completion (BP-All). For dgsol and BP-One, the user CPU time (in seconds) was reported, as well as the Largest Distance Error (LDE), defined as

$$\mathrm{LDE} = \frac{1}{|S|} \sum_{(i,j)\in S} \frac{|\,||x_i - x_j|| - d_{ij}|}{d_{ij}},$$

employed as a measure of solution accuracy (the lower, the better). We remark that we employed the LDE rather than the objective function value (1) as a solution quality measure because we wanted our results to be comparable with dgsol, which uses the LDE (see file dgerr.f in the dgsol 1.3 distribution http://www-unix.mcs.anl.gov/more/dgsol/). For the BP-All method, we reported

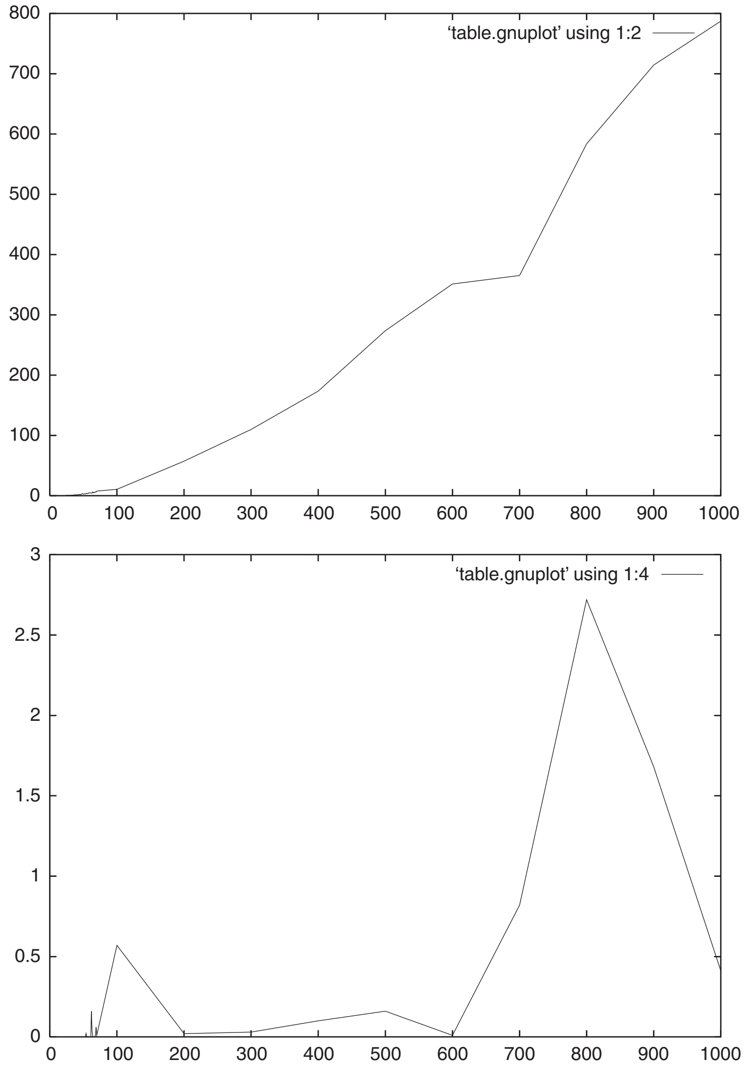

Fig. 6. Average user CPU time (plotted against molecular size) taken by dgsol (top) and BP-One (bottom).

the user CPU time and the number of solutions found (#Sol). Missing values are due to excessive memory requirements (over 1 GB RAM).

It is immediately noticeable that whereas dgsol always finds a solution, BP sometimes fails to find one within 1 GB RAM. It is instructive, however, to look at the solution accuracy (taken over the whole test suite): whereas dgsol ranges from 4.5×10^{-7} to 0.875 (excepting a couple of out-of-scale values clearly due to some numerical instability), BP scores a rather more impressive 4.74×10^{-11} to 5.62^{-6}. On average, the solution accuracy obtained by dgsol is 9.55×10^{-2} whereas BP averages 4.56×10^{-8}. Furthermore, all the instances in the Lavor 'large set' are solved by dgsol to a solution accuracy of order 10^{-1}: given that in BP pruning often occurs for feasibility differences of order 10^{-1} and even 10^{-2}, such a slack solution accuracy may mean that dgsol is not actually finding the correct solution.

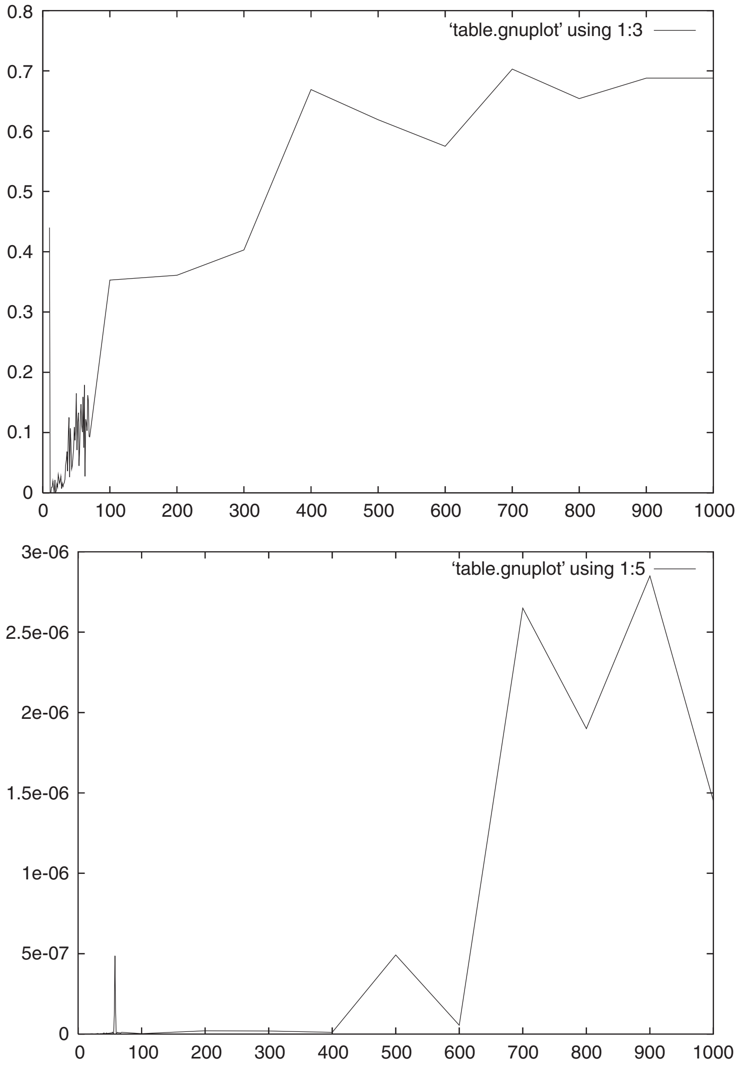

Fig. 7. Average accuracy (plotted against molecular size) attained by dgsol (top) and BP-One (bottom).

Table 2 reports the averages of the same parameters as in Table 1 taken over 10 Lavor instances in a sample of the 'small set' and over 4 Lavor instances in a sample of the 'large set'. It appears clear from these data that BP's strong points are indeed speed and accuracy. A graphical representation of the averages taken over the whole Lavor test set is shown in Fig. 6 (user CPU average taken to solve the instances in function of the molecular size by dgsol and BP-One) and Fig. 7 (average accuracy of the solution attained by dgsol and BP-One). We chose not to show the curves in the same plot because the huge scale difference on the ordinate axis 'pushed' the BP-One performance towards zero.

5. Final remarks

In this paper we presented a new discrete formulation for a particular subclass of the MDGP. We proposed a Branch-and-Prune algorithm and tested it against dgsol, an existing software for the MDGP. It appears that our method is faster and more accurate than dgsol by several orders of magnitude, albeit less predictable as concerns the running time, and requiring much greater memory.

Acknowledgments

The authors would like to thank FAPESP, FAPERJ, and CNPq for their support.

References

Creighton, T.E., 1993. *Proteins: Structures and Molecular Properties*. W.H. Freeman and Company, New York.

Crippen, G.M., Havel, T.F., 1988. *Distance Geometry and Molecular Conformation*. John Wiley & Sons, New York.

Cruz, I.F., Twarog, J.P., 1996. 3D Graph Drawing with Simulated Annealing. In: Brandenburg, F.-J. (ed.) *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany LNCS*, 1027. Springer, Berlin, pp. 162–165.

Dong, Q., Wu, Z., 2002. A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization* 22, 365–375.

Eren, T., Goldenberg, D.K., Whiteley, W., Yang, Y.R., Morse, A.S., Anderson, B.D.O., Belhumeur, P.N., 2004. Rigidity, computation and randomization in network localization. *IEEE InfoCom 2004 Proceedings*, pp. 2673–2684.

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

Klepeis, J.L., Floudas, C.A., Morikis, D., Lambris, J.D., 1999. Predicting peptide structures using NMR data and deterministic global optimization. *Journal of Computational Chemistry* 20, 13, 1354–1370.

Lavor, C., 2006. On generating instances for the Molecular Distance Geometry Problem. In: Liberti, L., Maculan, N. (eds), *Global Optimization: from Theory to Implementation*. Springer, Berlin, pp. 405–414.

Lavor, C., Liberti, L., Maculan, N., 2006. A Branch-and-Prune algorithm for the Molecular Distance Geometry Problem. *arXiv*:q-bio/0608012.

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

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

Phillips, A.T., Rosen, J.B., Walke, V.H., 1996. Molecular structure determination by convex underestimation of local energy minima. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 23, 181–198, American Mathematical Society, Providence.

Pogorelov, A., 1987. *Geometry*. Mir Publishers, Moscow.

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

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

Wu, D., Wu, Z., Yuan, Y., to appear. Rigid versus unique determination of protein structures with geometric buildup. *Optimization Letters*.