Journal of Bioinformatics and Computational Biology

Exploiting symmetry properties of the Discretizable Molecular Distance Geometry Problem

Antonio Mucherino
IRISA, University of Rennes 1, Rennes, France
antonio.mucherino@irisa.fr

Carlile Lavor
IMECC-UNICAMP, Campinas-SP, Brazil.
clavor@ime.unicamp.br

Leo Liberti
LIX, École Polytechnique, Palaiseau, France.
liberti@lix.polytechnique.fr

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The Discretizable Molecular Distance Geometry Problem (DMDGP) consists in a subset of instances of the distance geometry problem for which some assumptions allowing for discretization are satisfied. The search domain for the DMDGP is a binary tree that can be efficiently explored by employing a Branch & Prune (BP) algorithm. We showed in recent works that this binary tree may contain several symmetries, which are directly related to the total number of solutions of DMDGP instances. In this paper, we study the possibility of exploiting these symmetries for speeding up the solution of DMDGPs, and propose an extension of the BP algorithm that we named symmetry-driven BP (symBP). Computational experiments on artificial and protein instances are presented.

1. Introduction

We consider the Molecular Distance Geometry Problem[3,24] (MDGP), which consists in finding the three-dimensional conformation of a molecule by exploiting some information about the relative distances between pairs of its atoms. In particular, we are working on a discretization for this problem that allows us to transform the search space from a continuous to a discrete domain. We refer to the class of problems that can be discretized as the Discretizable MDGP (DMDGP). Both the MDGP and the DMDGP are NP-hard problems[8,23]. In the field of biology, instances of the MDGP/DMDGP can be obtained by Nuclear Magnetic Resonance (NMR) experiments, which provide estimates of distances between pairs of atoms of a given molecule. In this context, we are particularly interested in protein conformations. The scientific community is very active on this topic[2,4,17,25]; some recent surveys

are available[7,9,14].

Instances of the MDGP and of the DMDGP can be represented by weighted undirected graphs $G = (V, E, d)$. Each vertex in V corresponds to an atom of the molecule, whereas each edge in E indicates that the distance between the two corresponding vertices (atoms) is known. The weight associated to each edge is the numerical value of the distance. The DMDGP class contains instances that satisfy the two following assumptions. There exists an ordering of $V = \{1, \ldots, n\}$ such that

(1) $\forall v \in \{4, \ldots, n\}\ \forall j, k \in \{v-3, \ldots, v\} \quad \{j, k\} \in E$,
(2) $\forall v = 2, \ldots, n-1, \quad d(v-1, v+1) < d(v-1, v) + d(v, v+1)$.

The idea behind the discretization is the following. By the first assumption, for each atom $v > 3$, the distances between v and the three immediate preceding atoms $v-3$, $v-2$ and $v-1$ are known. If we suppose that these atoms are already placed somewhere in space, then 3 spheres centered in the Cartesian coordinates x_{v-3}, x_{v-2} and x_{v-1}, respectively, and having as radii the distances $d(v-3, v)$, $d(v-2, v)$ and $d(v-1, v)$, can be defined. Their intersection consists of the set of atomic positions for x_v satisfying these distances, which, by our assumptions, contains at most 2 points with probability one. This suggests a recursive procedure which defines a binary tree of atomic coordinates containing all possible positions for each vertex v on the v^{th} layer of the tree. Each solution can therefore be represented as a path from the root to a leaf node at level n. Each of such paths can be seen as a list of choices (left branch, right branch) on each layer of the tree. Therefore, each solution can also be represented by a binary vector of length n.

In order to efficiently explore this binary tree with the aim of finding solutions to the DMDGP, we employ the Branch & Prune (BP) algorithm[13]. The basic idea is to exploit the information regarding the distances considered in our assumptions for building the binary tree, and to use additional information regarding other available distances for checking the feasibility of the computed atomic positions. Distances in DMDGP instances can therefore be divided in two subgroups: the *discretization distances*, which are used for building the binary tree, and the *pruning distances*. When a position is not feasible, it makes the corresponding branch infeasible as well, and hence such a branch can be removed from the tree. This pruning phase in the BP algorithm allows to focus the search on the feasible parts of the binary tree.

One important feature of the BP algorithm is that it is potentially able to enumerate all solutions for any given DMDGP instance. Differently from heuristics and deterministic methods based on a continuous representation of the problem[7,9,14], this algorithm is based on a binary tree search, whose exploration can identify the complete set of solutions for each instance. This feature has important biological consequences. Most importantly, NMR data could bring to the definition of different protein conformations, that all satisfy the distance constraints, but only one of them may be the energically stable conformation. As a consequence, finding only one solution to the problem does not imply that the actual molecular conformation

has been identified.

We developed a software tool called `MD-jeep` which implements the BP algorithm[21]. Computational experiences which we performed during these years and published in various papers showed that the BP algorithm is very efficient in solving this combinatorial problem, in terms of CPU time and quality of the solutions, in comparison to other existing algorithms[6,8,10,11,12,20].

Our computational experiments, moreover, also allowed us to make some important observations. Even if the BP algorithm is based on a binary tree search that, in the worst case, can have an exponential number of leaf nodes, the CPU time appears to be linear in n. More recently, we noticed that this behavior is due to particular properties of protein instances, for which we can prove that the binary tree has a bounded width[15]. As a consequence, the branches of the tree cannot drastically increase in number, and BP executions are very efficient, specially when only one solution is required. If, instead, all solutions need to be computed (and the instance at hand has several solutions), then the execution of BP can be expensive and the computational time can be strongly dependent on the number of solutions.

One way for enhancing the performances of BP is to exploit the information regarding all symmetries that may appear in DMDGP instances having several solutions[16]. Since the beginning of our work on the DMDGP, we noticed that BP trees are symmetric, so that exploring half of the tree is enough for recontructing the solutions contained in the whole tree[8]. Immediate consequence of this result is the fact that all DMDGP instances have an even number of solutions. However, our computational experiments showed a stronger law: all DMDGP instances we considered have a number of solutions which is equal to 2^s, with $s > 0$. While we conjectured the relationship between this law and the presence of other symmetries in BP trees, we could not prove formally this result for a long time. It is therefore a very recent theoretical result the one we are going to exploit in this paper[16].

We present a variant of the BP algorithm which is able to exploit the presence of symmetries in BP trees, to which we will refer as symBP (symmetry-driven BP). Once the first solution is found by employing the basic BP approach, indeed, all other solutions can be computed by applying symmetry rules. Moreover, symmetries allow us to know in advance which branches of the binary tree are feasible or not, so that the search can be efficiently driven toward feasible branches only. For this reason, symBP can manage problems with several solutions more efficiently than the basic BP.

The rest of the paper is organized as follows. In Section 2, we discuss the symmetry properties of the DMDGP, while, in Section 3, we present the symBP algorithm. Computational experiments on protein instances and comparisons to the basic BP are presented in Section 4. Conclusions are given in Section 5.

2. Symmetry properties of the DMDGP

Let us consider an instance of the DMDGP represented by the graph $G = (V, E, d)$. As explained in the Introduction, part of the known distances (represented by the edges in E) are used for generating the candidate atomic positions, for a given atom, by intersecting 3 spheres, while the others are exploited for verifying the feasibility of such candidates. Let us therefore split E in two disjoint subsets: E_d, which contains all distances used for performing the discretization, and E_p, which contains all distances used for pruning purposes.

There can be two extreme situations. If E_p is empty, then the feasibility of any of the computed atomic positions cannot be verified, and all of them are therefore considered. In such a case, pruning never occurs, and a full binary tree is constructed during the execution of the BP algorithm. As a consequence, the considered instance has 2^{n-3} solutions, which correspond to the 2^{n-3} leaf nodes of the BP tree ($n = |V|$). In this situation, the execution of the BP algorithm is exponential, and it provides as solution the full search space, because there is no information to be used for discarding any of the solutions.

Let us suppose now that E_p is not empty. The presence of a pruning distance allows us to select some of the candidate atomic positions, and hence to select a subset of branches on which we can focus our search. If there is a distance $(u, v) \in E_p$, with $u \notin \{v-3, v-2, v-1\}$, then it can be easily proved that there is no branching on the tree at layer v. By the first assumption of the DMDGP, indeed, $\{v-3, v\}$, $\{v-2, v\}$ and $\{v-1, v\}$ belong to E_d. Moreover, if $\{u, v\} \in E_p$, the possible positions for v can be found by intersecting 4 spheres (not only 3) which are centered in u, $v-3$, $v-2$ and $v-1$ and having radii $d(u, v)$, $d(v-3, v)$, $d(v-2, v)$ and $d(v-1, v)$, respectively. Supposing that the 4 distances are compatible, this intersection can give one point only with probability one. Therefore, only one of the two positions generated by applying the discretization process is actually feasible. As a consequence, for each branch on the layer $v-1$ of the tree, there is only one branch on the layer v. The total number of branches on the two layers $v-1$ and v is thus the same.

The second extreme situation is therefore the following. Let us suppose that, for each layer $v > 4$ of the binary tree, there is a pruning distance $\{u, v\} \in E_p$. In such a case, branching is only allowed at layer 4, because, for all other atoms v, we do have a pruning distance. There are only 2 leaf nodes, and so the total number of solutions is 2, which is an even number, as well as a power of 2. The only two feasible branches of the tree are also symmetric[8]. We will refer to the symmetry at layer 4 of the binary tree as *first symmetry*. This symmetry is present in all DMDGP instances.

The most interesting situation is naturally the one in which E_p is not empty and pruning distances are not available on each layer. At a given layer v, branching is not possible if there is a pruning distance $\{u, v\} \in E_p$ ($u \notin \{v-3, v-2, v-1\}$). If there is no pruning distance, however, the number of branches on this layer is

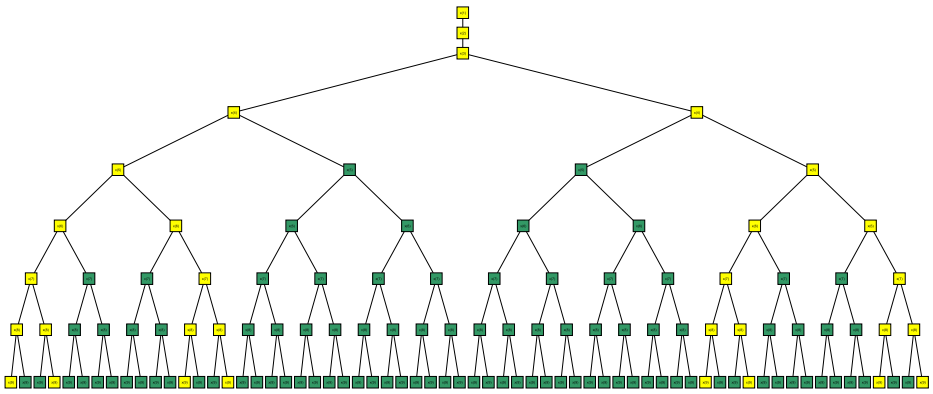

Fig. 1. A small binary tree related to an instance containing 9 atoms. $B = \{4, 6, 8\}$.

duplicated with respect to the number of branches on the previous layer $v-1$. It is important to remark that this phenomenon does not imply that all such branches are feasible. For each branch at layer $v-1$, there are two child branches at layer v, but one of the two (or even both) can be pruned later on at a futher layer if there is a pruning distance[16]. A pruning distance does not have to directly concern v for making a branch rooted at v infeasible: a distance between two atoms u and w such that $u+3 < v < w$ is indeed sufficient. In other words, there is a feasible branching on layer v (i.e. a branching defining two branches of length $|V|$) in the only case in which there is no pruning distance $\{u, w\}$ *passing over* the layer v. More formally, the layers of the binary tree where there is feasible branching are the ones contained in the set:

$$B = \{v \in V : \nexists (u, w) \text{ s.t. } u + 3 < v \leq w\}\,.$$

Since $v = 4$ is always contained in B, pruning can never occur at this layer, and this reflects the presence of the first symmetry in all BP trees. In general, for each $v \in B$, there is a duplication of feasible branches of the tree. Let us suppose for a moment that our molecule starts with the atom $v-3$ and that we consider the corresponding BP tree T'. By construction, T' is a subtree of the tree T which corresponds to the full molecule. Since all trees have the first symmetry, T' has a symmetry at layer v. This intuitively proves that there is also a symmetry at layer v of T. We recently published a formal proof of this result[16]. The same idea cannot be applied to $v \notin B$, because the presence of the pruning distance for v implies the possibility to prune at least one of the two newly generated branches.

Fig. 1 shows the BP tree for a small instance formed by 9 atoms. There are 3 symmetries in this tree. The first one is given by the first symmetry, at layer 4. As in all DMDGP instances, two symmetric branches start at layer 4. Then, other two symmetries are also present in this particular tree. One is at layer 6: 4 symmetric branches are rooted at positions belonging to layer 6. Similarly, 8

symmetric branches start from atomic positions belonging to layer 8, where another symmetry is present. In this case, therefore, $B = \{4, 6, 8\}$. The total number of solutions for this instance is $2^{|B|} = 8$.

As mentioned in the Introduction, solutions for this instance can be represented in different ways. First of all, if we consider Fig. 1, we can represent each solution as a path that goes from the root position x_1 to one of the feasible leaf nodes. Since there are, in this particular case, 8 solutions in total, there are 8 feasible paths that can be identified on the binary tree.

Another efficient way to represent solutions is through a vector of binary variables. On the generic layer of the tree, each path/solution can either pick the left or the right branch. This information can be coded in a binary variable (0/1), so that a binary vector of length $n = |V|$ can represent a solution to the DMDGP. It is important to remark that solutions sharing symmetric branches of the tree have symmetric local binary representations. Let us consider for example the solution in Fig. 1 corresponding to the second leaf node (from left to right). The binary vector corresponding to this solution is

$$\mathbf{s}_2 = (0, 0, 0, 0, 0, 0, 0, 1, 1),$$

where we suppose that 0 represents the choice *left*, and 1 represents *right* (the first three zeros are associated to the first three fixed atoms of the molecule). Since there is a symmetry at layer 6, another solution to the problem can be easily computed by repeating all choices from the root node until the layer 5, and by inverting all other choices. On the binary vector, repeating means copying, and inverting means flipping. So, another solution to the problem is

$$\mathbf{s}_3 = (0, 0, 0, 0, 0, 1, 1, 0, 0).$$

This solution corresponds to the third leaf node in Fig. 1.

This procedure can be exploited for speeding up the solution to DMDGPs, where the generation of any solution to the problem can be performed by exploiting one precomputed solution and the information concerning the set B (and the corresponding symmetries). The set B is able to provide a priori information on the quantity and on the location of the symmetries in BP trees. If the current layer is related to an atom $v \in B$, then, for each x_{v-1} on the previous layer, both the newly generated atomic positions for x_v are feasible. If $v \notin B$, instead, only one of the two positions can be part of a branch leading to a solution. The other position is either infeasible or it defines a branch that will be pruned later on at a further layer v, in correspondence with a pruning distance whose graph edge $\{u, w\}$ is such that $u + 3 < v \leq w$. Therefore, we can exploit such information for performing the selection of the branches that actually define a solution to the problem. When $v \notin B$ (only one position is feasible), it is not known which of the two branches is the correct one. This is the reason why at least one solution must be obtained before having the possibility of exploiting the symmetries for computing all the others.

Let us suppose that the solution $\mathbf{s}_1$ to the DMDGP has already been computed (by applying the BP algorithm, for example), and that it is the leftmost feasible branch in Fig. 1:

$$\mathbf{s}_1 = (0, 0, 0, 0, 0, 0, 0, 0, 0).$$

Recall that $B = \{4, 6, 8\}$ in this example. Let us consider the last symmetry in this tree, which is present at layer 8. By applying the procedure detailed above (we copy the binary variables from 0 to 7 and we flip all the others), we can obtain the solution:

$$\mathbf{s}_2 = (0, 0, 0, 0, 0, 0, 0, 1, 1).$$

At this point we consider the last but one symmetry on layer 6, and, by applying the same procedure to both solutions $\mathbf{s}_1$ and $\mathbf{s}_2$, we obtain:

$$\mathbf{s}_3 = (0, 0, 0, 0, 0, 1, 1, 0, 0), \quad \mathbf{s}_4 = (0, 0, 0, 0, 0, 1, 1, 1, 1),$$

where $\mathbf{s}_3$ is symmetric to $\mathbf{s}_2$, and $\mathbf{s}_4$ is symmetric to $\mathbf{s}_1$. Finally, by considering the first symmetry on layer 4, we obtain the remaining solutions:

$$\mathbf{s}_5 = (0, 0, 0, 1, 1, 0, 0, 0, 0), \quad \mathbf{s}_6 = (0, 0, 0, 1, 1, 0, 0, 1, 1),$$
$$\mathbf{s}_7 = (0, 0, 0, 1, 1, 1, 1, 0, 0), \quad \mathbf{s}_8 = (0, 0, 0, 1, 1, 1, 1, 1, 1).$$

During this phase, pruning distances in E_p do not need to be verified, because all branches constructed by symmetry are feasible.

3. The symmetry-driven BP (symBP)

Algorithm 1 is a sketch of the BP algorithm[13], which is at the basis of the symBP algorithm that is the focus of this paper. In the BP call, v is the current vertex to be positioned in space, $n = |V|$ and d represents the full set of known distances.

Once the positions for the first three atoms have been fixed by using the available information, BP can be invoked by setting $v = 4$, because the search on the tree actually starts on this tree layer. Then, BP recursively calls itself for a complete exploration of the binary tree. In each call, the two possible positions for the vertex v, x_v^0 and x_v^1, are computed, and their feasibility is verified. If, for example, x_v^0 is feasible, then this position could be part of a solution, and therefore the branch of the binary tree rooted at x_v^0 needs to be explored. In this case, the algorithm invokes itself for computing the possible positions for the next vertex. Instead, if x_v^0 is not feasible, then the current branch does not contain any solution and must be pruned. In this case, the algorithm does not invoke itself. BP behaves similarly when the feasibility of x_v^1 is verified.

The computation of the two positions x_v^0 and x_v^1 can be performed by computing the intersection among three spheres. Once obtained, their feasibility is checked by applying the Direct Distance Feasibility (DDF) pruning device, which is based on

Algorithm 1 The BP algorithm.

```
1: BP(v, n, d)
2: compute x_v^0;
3: if (x_v^0 is feasible) then
4:    if (v = n) then
5:       let nsols = nsols + 1;
6:    else
7:       BP(v + 1, n, d);
8:    end if
9: end if
10: compute x_v^1;
11: if (x_v^1 is feasible) then
12:    if (v = n) then
13:       let nsols = nsols + 1;
14:    else
15:       BP(v + 1, n, d);
16:    end if
17: end if
```

the idea of verifying whether the pruning distances are satisfied (for a certain tolerance). We remark that other pruning devices can be added to the BP algorithm[20].

Algorithm 2 is a sketch of symBP. In the symBP call, there are the input arguments necessary to BP, as well as some new ones. First of all, symBP also needs the set B containing the BP tree symmetries. Moreover, the parameter $nsols$ is used for monitoring the number of found solutions, whereas $prev$ is a vector of binary variables (0/1), which contains the last found solution in binary format.

At the beginning, symBP checks if the last atom (x_n) of the molecule has been placed during the previous call. In such a case, $nsols$ is updated (a solution has just been found) and all the branches rooted at $v \notin B$ and not defining a solution are removed from the tree. If at least one solution is known, indeed, the 0/1 choices are already available for each $v \notin B$. Since, on each layer, one of the choices brought to the identification of a solution, the other one needs to be discarded. Only partial branches rooted at $v \in B$ are kept, i.e. we only consider pairs of symmetric branches.

As explained in Section 2, the symmetric branches of the tree have their roots on the layers $v - 1$ such that $v \in B$. Therefore, if $v \in B$, the two atomic positions obtained by intersecting the three spheres are both feasible (the feasibility cannot be checked because there are no pruning distances at this level). At each call of the symBP algorithm, it is verified whether v belongs to B, and, in such a case, the algorithm simply computes the two positions x_v^0 and x_v^1, increases v and invokes itself twice.

If instead $v \notin B$, it is important to distinguish between two situations. If no solutions have been found so far ($nsols = 0$), then the information regarding the

Algorithm 2 The symBP algorithm.

```
1: symBP(v, n, d, B, nsols, prev)
2: if (v > n) then
3:    let nsols = nsols + 1;
4:    remove all branches such that:
4:          · their root is u ∉ B;
4:          · their leaf nodes v_f < n.
5:    return nsols;
6: end if
7: if (v ∈ B) then
8:    compute x_v^0;
9:    let prev(v) = 0;
10:   symBP(v + 1, n, d, B, nsols, prev);
11:   compute x_v^1;
12:   let prev(v) = 1;
13:   symBP(v + 1, n, d, B, nsols, prev);
14: end if
15: if (v ∉ B and nsols = 0) then
16:   compute x_v^0;
17:   if (x_v^0 is feasible) then
18:      let prev(v) = 0;
19:      symBP(v + 1, n, d, B, nsols, prev);
20:   end if
21:   if (nsols = 0) then
22:      compute x_v^1;
23:      if (x_v^1 is feasible) then
24:         let prev(v) = 1;
25:         symBP(v + 1, n, d, B, nsols, prev);
26:      end if
27:   end if
28: end if
29: if (v ∉ B and nsols > 0) then
30:   compute x_v^{¬prev(v)};
31:   let prev(v) = ¬prev(v);
32:   symBP(v + 1, n, d, B, nsols, prev);
33: end if
34: return nsols;
```

symmetries of the problem cannot be exploited yet, and the basic BP procedure must be performed. During this phase, the binary vector *prev* is kept updated so that it will contain the first found solution coded in binary format. Note that, if a solution is found when considering the position x_v^0 of v, it is useless to consider the

position x_v^1 (recall $v \notin B$).

Finally, if $v \notin B$ and at least one solution has already been found, symBP simply reconstructs such solutions by exploiting the symmetries on the tree and the last computed solution coded in binary format. There is no branching anymore, because only $x_v^{\neg prev(v)}$ can be feasible if $x_v^{prev(v)}$ was feasible for the previous solution. There is no pruning neither, because all generated positions are feasible. This way, the computational cost needed for computing all solutions (exception made for the first one) is drastically decreased.

We point out that the symBP algorithm is intrinsically parallel. Let us suppose that an instance of the DMDGP is divided in as many subinstances as the number p of processors which are available on a parallel computer. Each subinstance contains a group a consecutive atoms of the original instance, as well as the corresponding distances. With this setup, local solutions to the problem can be obtained by solving the p subinstances in parallel by using the basic BP[19]. After this parallel phase, the local solutions can be exchanged among the processors and the final set of solutions can therefore be generated by combining all of them.

The parallelization of symBP by using the same strategy is potentially able to improve the speed-up. On each processor, indeed, the complete enumeration of the solutions of the subinstances is not necessary: once one solution is found, the others can be obtained by applying the symmetry rules reported above. This allows to drastically reduce the computational time for the parallel solution of each subinstance. Moreover, once the first complete solution is obtained by combining the local ones, the final set of solutions can be computed by using the first complete solution and the information on the set B related to the original instance.

4. Computational results

We compare in this section the basic BP algorithm[13] to the symBP algorithm described in Section 3. All codes were written in C programming language and all the experiments were carried out on an Intel(R) Xeon(TM) CPU 3.40GHz with 4GB RAM, running Linux. The codes have been compiled by the GNU C compiler v.4.1.1 with the `-O3` flag.

We recently proved that the BP algorithm (and so symBP as well) can be applied for solving instances of the problem containing real NMR data[20], but the theory of the symmetries that we developed so far only concerns instances containing exact distances. Therefore, we consider two sets of DMDGP instances: Lavor instances and artificial protein instances.

Lavor instances are artificially generated instances that only simulate the general structure of proteins. However, they may contain several solutions, representing the ideal instances for stressing the differences between the two compared algorithms. Moreover, we also consider another set of instances (protein instances), which are generated from known protein conformations. For this set of instances, distances are computed from known atomic coordinates and used for generating instances simu-

instance			BP			symBP		
name	n	$\|E\|$	#Sol	#DDF	time	#Sol	#DDF	time
`lavor_10`	10	29	4	44	0.00	4	2	0.00
`lavor_20`	20	77	64	86	0.00	64	7	0.00
`lavor_30`	30	161	64	592	0.01	64	18	0.00
`lavor_40`	40	194	2048	30720	0.05	2048	19	0.01
`lavor_50`	50	203	1024	46728	0.07	1024	49	0.01
`lavor_60`	60	357	256	71352	0.12	256	89	0.01
`lavor_70`	70	591	16	232	0.00	16	67	0.00
`lavor_100`	100	605	2	4460924	7.55	2	815010	1.37
`lavor_200`	200	1844	32	35298	0.09	32	394	0.01
`lavor_300`	300	2505	4	38364	0.07	4	9378	0.03

Table 1. Some experiments with BP and symBP on a subset of Lavor instances.

lating realistic data (the only non-realistic assumption is given by the precision of the distances). Protein instances generally correspond to BP trees having a bounded width[15], so that they do not usually admit as many symmetries as Lavor instances. However, the interest in solving protein instances by symBP still remains, because the experiments here presented show that its performances can still be improved when interval data will be considered.

We begin our comparison by using the so-called Lavor instances[5,22]. In the generation of this set of instances, a molecule is simply 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, Lavor instances give rather more realistic models of proteins than other randomly generated instances do. We labeled the Lavor instances by `lavor`n, where n is the number of atoms in the molecule. Only one random instance is considered for each size.

Table 1 shows some experiments with a subset of 10 Lavor instances. Each instance is described by its own name, the number n of atoms forming the molecule, and the number of known distances, which corresponds to the cardinality of the edge set E. For both algorithms, we monitor the number #Sol of found solutions (which should be method-invariant), the number #DDF of times the pruning device DDF prunes away infeasible branches, and the CPU time (in seconds).

As explained in details in Section 3, once the first solution is found, symBP is able to avoid recursive calls to itself that would produce an infeasible atomic position. Therefore, one way to evaluate the performances of this algorithm w.r.t. the performances of the basic BP is to compare the corresponding values for #DDF. These experiments show that the symBP algorithm is actually able to find the same set of solutions by applying the pruning device fewer times, because many of the

calls where DDF would have pruned are suppressed in advance. This reduction in the number of recursive calls is naturally reflected on the total CPU time. Other Lavor instances having a larger dimension can also be considered. Such instances (with $n \in \{400, \ldots, 700\}$) contain so many symmetries that both the basic BP and symBP are not able to obtain the full set of solutions in a reasonable amount of time[18].

We present in the following some experiments on a set of instances generated from real protein conformations. We generated some instances of the DMDGP by considering some protein conformations of different size downloaded from the PDB[1,8]. For each downloaded protein, we selected the backbone atoms of the molecule (the chain of atoms N, C_α and C, and the hydrogens of the backbone), and we computed the distances between all pairs of such atoms. In order to simulate distances obtained by experiments of NMR, we removed all distances shorter than a certain threshold δ. Moreover, since protein instances generally contain a few symmetries[15], we also removed a certain number of pruning distances (randomly chosen) in order to increase the total number of solutions for each instance. For all instances, the threshold δ is set to 7Å.

Table 2 presents some experiments on the generated set of protein instances. For all instances, there is an evident gain in CPU time when symBP is applied. In most cases, the BP trees contain 2 or 4 symmetries (4 or 16 solutions, respectively); there is also a particular case in which the symmetries are 14, so that the total number of solutions is 16384 (instance `1dv0`). Naturally, the CPU time generally depends upon the size of the instance (cardinality of the vertex set V, as well as cardinality of the edge set E), but also upon the number of symmetries that are present in the binary trees. When there are more symmetries, the gain in CPU time when considering symBP is more evident.

5. Conclusions

We presented the symBP algorithm, an extension of the BP algorithm which is able to exploit a priori information about the symmetries in BP trees in order to speed up the solution of DMDGPs. Once one solution to the problem is found by applying the basic BP procedure, all the others can be obtained by using symmetry rules, which drastically decrease the computational cost of experiments related to instances where the number of solutions is large. Computational experiments on Lavor instances and protein instances showed the efficacy of this approach.

We are starting to work on DMDGPs in which some of the distances are represented by intervals[10,20], which is a more realistic situation, specially in the context of biology. When intervals are available, instead of exact distances, DMDGP instances are much more expensive to solve and symBP is potentially able to decrease the total computational cost. The extension of symBP to interval data is however not trivial. When a pruning distance is represented by an interval, there are subsets of atomic positions that are able to satisfy the constraints (not a single position

instance			BP			symBP		
name	n	$\|E\|$	#Sol	#DDF	time	#Sol	#DDF	time
1brv	128	623	128	3280	0.01	128	140	0.00
1a11	174	844	32	3152	0.00	32	64	0.00
1acw	201	1003	16	3456	0.01	16	93	0.01
1bbl	252	1228	16	34448	0.09	16	1293	0.02
1erp	259	1331	16	1053392	2.39	16	46326	0.11
1aqr	275	2500	4	1096	0.01	4	172	0.00
1k1v	278	1337	16	609488	1.94	16	32755	0.10
1h1j	300	1481	256	21264	0.09	256	235	0.02
1ed7	305	1520	16	4477376	19.44	16	158768	0.74
1jkz	315	3137	4	1260	0.01	4	116	0.00
1dv0	316	1621	16384	1741312	8.83	16384	231	2.04
1ahl	328	1586	16	5528480	23.84	16	181443	0.80
1brz	368	3831	4	1464	0.01	4	218	0.00
1ccq	406	2060	16	5381728	23.52	16	264723	1.19
1bqx	521	5170	4	2064	0.01	4	288	0.01
2hsy	715	7425	4	2872	0.03	4	423	0.02
1acz	724	7741	4	4668	0.04	4	823	0.02
1a2s	615	3349	16	1917840	12.44	16	33051	0.24
1ag4	694	7400	4	2768	0.03	4	383	0.02
1itm	892	9374	4	3576	0.04	4	546	0.02
1ngl	1206	11701	4	4828	0.07	4	704	0.04
1la3	1281	7027	16	1691360	22.68	16	98954	1.50
1a23	1287	13250	4	5168	0.09	4	766	0.05
1oy2	1293	12718	4	5192	0.08	4	741	0.05
2ron	1649	16566	4	22324	0.43	4	2779	0.12
1d8v	1787	17937	4	7440	0.19	4	1142	0.09
1ezo	2523	25040	4	17936	0.51	4	2608	0.23

Table 2. Experiments on protein instances.

only), and ad-hoc strategies need to be developed for exploiting the symmetries in a more general case. This extension, together with an efficient parallel version of the symBP, could allow us to solve very difficult biological problems in a short amount of time. This will be the focus of our research in the near future.

Acknowledgments

The authors wish to thank the anonymous referees. This work is partially supported by the ANR project "Bip:Bip". The authors would also like to thank the Brazilian research agencies FAPESP and CNPq, the French research agency CNRS and École

Polytechnique, for financial support.

References

1. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acid Research* **28**, 235–242, 2000.
2. P. Biswas, K.-C. Toh, Y. Ye, A Distributed SDP Approach for Large-Scale Noisy Anchor-Free Graph Realization with Applications to Molecular Conformation. *SIAM Journal on Scientific Computing* **30**, 1251–1277, 2008.
3. G. Crippen, T. Havel, Distance Geometry and Molecular Conformation, Wiley, New York, 1988.
4. N. Krislock. *Semidefinite Facial Reduction for Low-Rank Euclidean Distance Matrix Completion*. PhD thesis, University of Waterloo, 2010.
5. C. Lavor. On generating instances for the molecular distance geometry problem. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, pages 405–414. Springer, New York.
6. C. Lavor, J. Lee, A. Lee-St. John, L. Liberti, A. Mucherino, M. Sviridenko. Discretization Orders for Distance Geometry Problems, to appear in *Optimization Letters*, 2012.
7. C. Lavor, L. Liberti, and N. Maculan. Molecular distance geometry problem. In C. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*, pages 2305–2311. Springer, New York, second edition, 2009.
8. C. Lavor, L. Liberti, N. Maculan, and A. Mucherino. The discretizable molecular distance geometry problem. to appear in *Computational Optimization and Applications*, 2012.
9. C. Lavor, L. Liberti, N. Maculan, and A. Mucherino. Recent Advances on the Discretizable Molecular Distance Geometry Problem. *European Journal of Operational Research* **219**, 698–706, 2012.
10. C. Lavor, L. Liberti, and A. Mucherino. The interval Branch-and-Prune Algorithm for the Discretizable Molecular Distance Geometry Problem with Inexact Distances, to appear in *Journal of Global Optimization*, 2012.
11. C. Lavor, A. Mucherino, L. Liberti, and N. Maculan. Discrete approaches for solving molecular distance geometry problems using nmr data. *International Journal of Computational Biosciences* **1**, 88–94, 2011.
12. C. Lavor, A. Mucherino, L. Liberti, and N. Maculan. On the computation of protein backbones by using artificial backbones of hydrogens. *Journal of Global Optimization* **50**, 329–344, 2011.
13. L. Liberti, C. Lavor, and N. Maculan. A branch-and-prune algorithm for the molecular distance geometry problem. *International Transactions in Operational Research* **15**, 1–17, 2008.
14. L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research* **18**, 33–51, 2011.
15. L. Liberti, B. Masson, C. Lavor, and A. Mucherino. Branch-and-prune trees with bounded width. In *Proceedings of the* 10^{th} *Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW11)*, pages 189–193, Rome, Italy, 2011.
16. L. Liberti, B. Masson, J. Lee, C. Lavor, and A. Mucherino. On the number of solutions of the discretizable molecular distance geometry problem. In *Proceedings of the* 5^{th} *Annual International Conference on Combinatorial Optimization and Applications (COCOA11)*, volume 6831 of *Lecture Notes in Computer Science*, pages 322–342.

Springer, 2011.
17. J.J. Moré, Z. Wu, Distance Geometry Optimization for Protein Structures, *Journal of Global Optimization* **15**, 219–234, 1999.
18. A. Mucherino, C. Lavor, L. Liberti, A Symmetry-Driven BP Algorithm for the Discretizable Molecular Distance Geometry Problem, IEEE Conference Proceedings, Computational Structural Bioinformatics Workshop (CSBW11), International Conference on Bioinformatics & Biomedicine (BIBM11), Atlanta, GA, USA, 390–395, 2011.
19. A. Mucherino, C. Lavor, L. Liberti, E-G. Talbi, A Parallel Version of the Branch & Prune Algorithm for the Molecular Distance Geometry Problem, IEEE Conference Proceedings, ACS/IEEE International Conference on Computer Systems and Applications (AICCSA10), Hammamet, Tunisia, 1–6, 2010.
20. A. Mucherino, C. Lavor, T. Malliavin, L. Liberti, M. Nilges, and N. Maculan. Influence of pruning devices on the solution of molecular distance geometry problems. In P.M. Pardalos and S. Rebennack, editors, *Proceedings of the* 10^{th} *International Symposium on Experimental Algorithms (SEA11)*, volume 6630 of *Lecture Notes in Computer Science*, pages 206–217, Crete, Greece, 2011. Springer.
21. A. Mucherino, L. Liberti, and C. Lavor. `MD-jeep`: an implementation of a branch & prune algorithm for distance geometry problems. In K. Fukuda et al., editor, *Proceedings of the Third International Congress on Mathematical Software (ICMS10)*, volume 6327 of *Lectures Notes in Computer Science*, pages 186–197, Kobe, Japan, 2010. Springer.
22. A.T. Phillips, J.B. Rosen, and V.H. Walke. Molecular structure determination by convex underestimation of local energy minima. In P.M. Pardalos, D. Shalloway, and G. Xue, editors, *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding*, volume 23, pages 181–198. American Mathematical Society, 1996.
23. J.B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.
24. T. Schlick. *Molecular modelling and simulation: an interdisciplinary guide*. Springer, New York, 2002.
25. D. Wu, Z. Wu, An Updated Geometric Build-Up Algorithm for Solving the Molecular Distance Geometry Problem with Sparse Distance Data, *Journal of Global Optimization* **37**, 661–673, 2007.

Antonio Mucherino is currently an Assistant Professor at University of Rennes 1 (IRISA), Rennes, France. He obtained his Ph.D. degree in Computational Biology in 2005 at the Second University of Naples (SUN), Italy, and worked as postdoctoral fellow in USA (2 years) and France (3 years). The main research interests of Antonio Mucherino mainly include bioinformatics, data mining, combinatorial optimization and parallel computing.

Carlile Lavor is a mathematician and received his Ph.D. degree in Computer Science at Federal University of Rio de Janeiro, in 2001. In 2006, he obtained his Habilitation in Combinatorics at State University of Campinas, where he is currently an Associate Professor. His main research interests are distance geometry and applications, and quantum information.

Leo Liberti is currently an Associate Professor at LIX, Ecole Polytechnique. He obtained his Ph.D. degree in Global Optimization at Imperial College, London, in 2004, and worked as postdoctoral fellow at Politecnico di Milano, Italy, during 2004-2005. His main research interests are reformulations in mathematical programming, global and combinatorial optimization with applications to complex industrial systems and bioinformatics.