An Efficient Exhaustive Search for the Discretizable Distance Geometry Problem with Interval Data

A. Mucherino*, J-H. Lin†

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

‡Research Center for Applied Sciences, Academia Sinica, Taipei, Taiwan.
jhlin@gate.sinica.edu.tw

***Abstract*—The Distance Geometry Problem (DGP) asks whether a simple weighted undirected graph can be realized in a given space (generally Euclidean) so that a given set of distance constraints (associated to the edges of the graph) is satisfied. The Discretizable DGP (DDGP) represents a subclass of instances where the search space can be reduced to a discrete domain having the structure of a tree. In the ideal case where all distances are precise, the tree is binary and one singleton, representing one possible position for a vertex of the graph, is associated to every tree node. When the distance information is however not precise, the uncertainty on the distance values implies that a three-dimensional region of the search space needs to be assigned to some nodes of the tree.**

By using a recently proposed coarse-grained representation for DDGP solutions, we extend in this work the branch-and-prune (BP) algorithm so that it can efficiently perform an exhaustive search of the search domain, even when the uncertainty on the distances is important. Instead of associating singletons to nodes, we consider a pair consisting of a box and of a most-likely position for the vertex in this box. Initial estimations of the vertex positions in every box can be subsequently refined by using local optimization.

The aim of this paper is two-fold: (i) we propose a new simple method for the computation of the three-dimensional boxes to be associated to the nodes of the search tree; (ii) we introduce the resolution parameter ρ, with the aim of controling the similarity between pairs of solutions in the solution set. Some initial computational experiments show that our algorithm extension, differently from previously proposed variants of the BP algorithm, is actually able to terminate the enumeration of the solution set by providing solutions that differ from one another accordingly to the given resolution parameter.

I. Introduction

Let $G = (V, E, d)$ be a simple weighted undirected graph, where vertices represent objects (whose nature depends on the application at hand), and the existence of an edge $\{u, v\}$ between two vertices u and v indicates that the distance between the two corresponding objects is known [18]. The weight $d(u, v)$ associated to the edge $\{u, v\}$ is in general a real-valued interval providing the lower and the upper bound on the distance values. However, the interval $d(u, v)$ can degenerate to one singleton, and in this situation only one approximation of the distance value is available.

Definition 1 *Given a simple weighted undirected graph $G = (V, E, d)$ and a positive integer K, the Distance Geometry Problem (DGP) asks whether a function*

$$x : v \in V \longrightarrow x_v \in \mathbb{R}^K$$

exists such that

$$\forall \{u, v\} \in E, \quad ||x_u - x_v|| \in d(u, v), \tag{1}$$

where $||\cdot||$ represents the Euclidean norm.

The function x is called a *realization* of the graph G. We say that a realization x that satisfies all constraints in equ. (1) is a *valid realization*.

The DGP is NP-hard [26], and has several different applications, including: (i) protein structure determination [7] (this is the application we will consider in our experiments in Section IV); (ii) sensor network localization [27]; (iii) multi-dimensional scaling [13]; (iv) clock synchronization [8]; (v) motion adaptation [25]; and others.

We give the following definition of a discretizable subclass of DGP instances [23]. Let E' be the subset of the edge set E such that the weight associated to the edges are degenerate intervals.

Definition 2 *A simple weighted undirected graph G represents an instance of the Discretizable DGP (DDGP) if and only if there exists a vertex ordering on V such that the following two assumptions are satisfied:*

- **(a)** *$G[\{1, 2, \ldots, K\}]$ is a clique whose edges are in E';*
- **(b)** *$\forall v \in \{K + 1, \ldots, |V|\}$, there exist $u_1, u_2, \ldots, u_K \in V$ such that*
  - **(b.1)** *$u_1 < v$, $u_2 < v$, ..., $u_K < v$;*
  - **(b.2)** *$\{\{u_1, v\}, \{u_2, v\}, \ldots, \{u_{K-1}, v\}\} \subset E'$, $\{u_K, v\} \in E$;*
  - **(b.3)** *$\mathcal{V}_S(u_1, u_2, \ldots, u_K) > 0$ (if $K > 1$),*

where $G[\cdot]$ is the subgraph induced by a subset of vertices of V, and $\mathcal{V}_S(\cdot)$ is the volume of the simplex generated by a valid realization of the vertices $u_1, u_2, \ldots, u_K$.

In the following, we will refer to assumptions **(a)** and **(b)** as the *discretization assumptions*. Such assumptions can be verified only if a vertex ordering is associated to V [10], which is generally referred to as a *discretization order* when the two assumptions above are satisfied.

Assumption **(a)** allows us to fix a coordinate space where to construct DDGP solutions while making sure that none of them can be obtained from other solutions by applying translations or rotations. Assumption **(b)** ensures that every vertex v has at least K *reference vertices* u_i, with $1 \leq i \leq K$, such that the corresponding *reference distance* to v is known. Since it is supposed (see Assumption **(b.2)**) that only one distance is represented by a non-degenerate interval, the possible positions for v wrt its reference vertices can be obtained by intersecting $K-1$ spheres and one spherical shell, which gives at most two arcs [15]. These assumptions make it possible to reduce the DDGP search space to a discrete domain having the structure of a tree where (possibly degenerate) arcs are associated to its nodes.

In order to simplify the notations, and in accordance with the application that is considered in Section IV, we will suppose in the following that the dimension K is set to 3.

The branch-and-prune (BP) algorithm [17] can be employed for exploring the search tree obtained with the discretization. In a recent work, we have integrated the BP algorithm with a coarse-grained representation [24]. This representation allows us to deal in an efficient way with the uncertainty of the available distance values, which can have an important impact on the lengths of the arcs obtained with the intersections. Differently from [15], where sample points are selected from the arcs, the coarse-grained representation better deals with uncertainty by assigning to every node of the search tree not only a suitable position x_v for the corresponding vertex v, but also a three-dimensional region B_v where v is allowed to take its positions. While the initial estimation for x_v can be very rough, the region B_v contains all its feasible positions and can therefore be explored for *refining* the position x_v while searching in a relatively small neighborhood of the search domain. In our first studies, this three-dimensional region is represented by a box inscribing the arcs obtained with the intersections.

This work is a step ahead in the development of an implementation of the BP algorithm that is based on the coarse-grained representation. Our new implementation is the first one that is actually able to enumerate the entire solution set of DDGP instances containing approximated distances (see Section IV). To this final purpose, we propose the integration in the algorithm of the following two features:

- a simple strategy for the definition of the boxes inscribing the arcs obtained with the intersections of the spheres (degenerate intervals) and spherical shell (one non-degenerate interval);
- the introduction of the resolution parameter, which allows to neglect "on-the-fly" all solutions that are *too similar* to solutions that were already computed.

The rest of the paper is organized in three main sections. Section II will be focused on our implementation of the BP algorithm: we will describe the coarse-grained representation, as well as our new method to compute the boxes inscribing the arcs obtained with the discretization process. Section III will introduce the resolution parameter and discuss its impact on the execution of the BP algorithm. Finally, Section IV will present some experiments on DDGP instances of the protein structure determination problem, while Section V will conclude the paper.

II. An extended BP algorithm

We have recently proposed the use of a coarse-grained representation of the DDGP search space in [24]. In the present work, we will extend this approach by introducing some new features in the BP algorithm, so that a *complete* enumeration of the search space will in fact be possible, even in presence of interval distances. This was the main objective of various previous publications (see for example [6]), but it was not completely attained.

In the discussion below, we will focus on the following main points. A general sketch of the BP algorithmic framework will be given in Section II-A, while the coarse-grained representation will be detailed in Section II-B. Then, Section II-C will discuss on how arcs of vertex positions can be computed by exploiting the available distance information, and Section II-D will present our method for the definition of boxes inscribing the arcs.

A. The BP algorithm

The BP algorithm was formally introduced in [17], and its basic idea is to perform a systematic exploration of the DDGP search tree. This search domain can be explored starting from its top, where the first vertex belonging to the initial clique is placed. Subsequently, all other vertices in the initial clique can be placed in their unique positions [14], and then the search can actually start with the vertex having rank 4 in the associated discretization order. At each step, the candidate positions for the current vertex v are computed, and the search is branched. This phase of the BP algorithm is named *branching phase*. Depending on the available distance information, represented by one approximated value, or rather by a real-valued interval, the set of candidate positions may either contain two singletons, or two disjoint arcs, respectively. Therefore, an arc is in general associated to every tree node, which can be in some cases degenerate. The distances that are used during the branching phase are called "discretization distances".

Pruning devices can be employed for discovering infeasible vertex positions. In BP, the main pruning device verifies whether available distances, that are not used for the discretization, are satisfied by candidate vertex positions or not. As soon as a vertex position is found to be infeasible, then the corresponding branch can be pruned and the search can be backtracked [16]. This phase of the BP algorithm is named

Algorithm 1 The BP algorithm's main framework

```
1: BP(v, G)
2: if (v > |V|) then
3:    // one solution is found
4:    print current conformation;
5: else
6:    // coordinate computation
7:    if (one discretization distance belongs to E \ E') then
8:       compute two candidate arcs
9:       add them to the list L
10:   else
11:      compute two candidate positions y^1 and y^2
12:      add them to the list L
13:   end if
14:   for i = 1, ..., |L| do
15:      if (L(i) is an arc) then
16:         choose sample x_v from the arc L(i)
17:      else
18:         set x_v = y^i
19:      end if
20:      // verifying the feasibility of the computed positions
21:      if (x_v is feasible) then
22:         BP(v + 1, G);
23:      end if
24:   end for
25: end if
```

pruning phase, and the used distances are called "pruning distances".

Algorithm 1 is a sketch of the main framework for the BP algorithm. In the BP call, $v \in V$ is the current vertex to be positioned and G is the simple weighted undirected graph representing a DDGP instance. Once the initial clique has been realized, the BP algorithm can be invoked recursively, starting from the vertex v having rank 4. As mentioned above, a lot of research has been conducted in recent years to find the best way to implement the line 16 of the algorithm. In [15], for example, a predefined number D of sample points are extracted from the generated arcs (the parameter D is the "discretization factor"). However, this strategy for a *lossy discretization* of the arcs has an important impact on the quality of the solutions, with a consequent amplification of error propagation along the search tree [12].

In our current implementation of the BP algorithm, we do not discretize the arcs, but we rather consider the coarse-grained representation presented in Section II-B. Only one vertex position is associated to every node of the tree, but this position is not fixed in one unique position. If necessary, it can rather be refined subsequently when deeper layers of the tree are reached, by exploring other possible positions inside the box that is associated to the node. It is important to remark that, when the generated arcs are larger, the corresponding box becomes bigger, and it might include positions that are feasible with more than one solution.

B. A coarse-grained representation for BP

Previous attempts to improve the efficiency of the BP algorithm (see for example [1], [11]) were based on the idea to avoid branching over subsets of positions from the arcs that may be found to be infeasible at the current layer *before* starting the branching phase. While some improvements were observed, these BP variants are however not able to consider distances that appear subsequently at further tree layers.

This is the main motivation for a coarse-grained representation of DDGP solutions. Instead of fixing, on every branch of the tree, all vertices in unique positions, the idea is to rather associate a small *region* of the search space to every vertex, together with a most-likely position. The shape of the region can be chosen on the basis of the methods that are then used for their manipulation.

In our coarse-grained representation, we use the following function:

$$z : v \in V \longrightarrow (x_v, B_v) \in \mathbb{R}^3 \times \mathbb{R}^6,$$

where B_v is a box defined in the Cartesian system given by the initial clique (see Section II-A). We point out that B_v has 6 dimensions (in dimension $K = 3$, the position of one vertex of the box, plus the corresponding depth, length and height values are necessary for its unique determination). When a new vertex position x_v is generated for the current vertex v, the function z does not only allow to assign a position x_v to v, but also to keep track of the feasible region where it belongs to. On further layers, in fact, the position x_v may not be feasible wrt some other distances, and it could therefore be *refined* in order to ensure global feasibility. This can be done, for example, by employing solvers for local optimization. The position x_v is naturally constrained to stay in the original box B_v for two main reasons. Firstly, the (continuous) search space of the local solver is in this way bounded; secondly, the situation where the local solver can move to solutions belonging to other tree branches is avoided.

We motivate the choice of employing a local solver with the fact that, at every layer of the tree where an infeasibility is discovered, there are only a few distances that are not satisfied, and the actual search space consists of the set product of all boxes B_v. This makes the corresponding subproblem to solve easier to tackle. Naturally, an important point concerning the use of a local solver is also its fast convergence: in fact, when attempting the solution of harder instances, we expect the local solver to be invoked at almost all recursive BP calls.

In this work, we will use a Spectral Projected Gradient (SPG) method [3], [22], [24] for this refinement step. When the BP algorithm reaches a leaf node, a valid realization x can be extracted from z by simply extracting the set of positions x_v, for every $v \in V$.

C. Computing arcs of vertex positions

When the discretization assumptions are satisfied, the possible positions for a given vertex v can be computed by exploiting the set of discretization distances, together with the positions, along the same tree branch, of the corresponding

reference vertices. Let u_3, u_2, and u_1 be the three reference vertices for the vertex $v \in V$. Since all reference vertices precede v in the given discretization order, one position for every u_i is assigned to the current tree branch, and distances between pairs of reference vertices u_i may be computed (if not already available). Therefore, the subgraph $G[\{u_3, u_2, u_1, v\}]$, when completed with missing distances between reference vertices, always induces a clique. As shown in [14], the distance information in the clique can be exploited to represent the possible positions for v in terms of torsion angles ω. More precisely, once the distances $d(u_3, u_2)$, $d(u_2, u_1)$, $d(u_1, v)$ are fixed, as well as the angles θ_{u_2} and θ_{u_1} formed, respectively, by the triplets (u_3, u_2, u_1) and (u_2, u_1, v), then the distance $d(u_3, v)$ corresponds to two possible values for the torsion angle ω formed by the quadruplet (u_3, u_2, u_1, v) [19].

Let us initially suppose that the distance $d(u_3, v)$ is exact: at most two distinct values ω^+ and ω^- can be computed for the torsion angle (it might happen that they coincide). The two corresponding positions for the vertex v can therefore be computed with the following formula:

$$\chi(v, \omega) = x(u_1) + Uw(v, \omega), \tag{2}$$

where the matrix U is a rotation matrix (see [9]) and

$$w(v, \omega) = \begin{bmatrix} -d(u_1, v)\cos\theta_{u_1}, \\ d(u_1, v)\sin\theta_{u_1}\cos\omega, \\ d(u_1, v)\sin\theta_{u_1}\sin\omega. \end{bmatrix}. \tag{3}$$

When the distance $d(u_3, v)$ is represented by an interval, two intervals of ω can be defined [11]. From a geometrical point of view, the interval distance $d(u_3, v)$ corresponds to two arcs that can be identified over the circle obtained by intersecting two spheres centered in u_1 and u_2 and having as radius, respectively, the corresponding reference distances. We generate a three-dimensional box inscribing every obtained arc, so that it can then be associated to the tree node, together with one position taken from the arc. This chosen position is the one that will be used at further layers when defining spheres or spherical shells for the intersections; however, this position may need to be refined when infeasibilities are detected by the pruning devices. The refinement step is supposed to keep the given position x_v inside the predetermined box B_v.

D. Computing the boxes

The boxes are defined by using the minimal and maximal possible coordinates associated to a given interval for ω. To perform this calculation, we remark that the three components of $\chi(v, \omega)$ (see equ. (2) and equ. (3)) can be rewritten (we explicitly write only the first component χ') as:

$$\chi'(v, \omega) = A(v)\cos\omega + B(v)\sin\omega + C(v),$$

where $C(v)$ is an additive term, and $A(v)$ and $B(v)$ are two multiplicative terms. For fixed $v \in V$, since $C(v)$ is an additive constant, the determination of the minimal and maximal values for the first component of $\chi(v, \omega)$ can be focused on the optimization of the remaining terms. By following [11], we remark that

$$A(v)\cos\omega + B(v)\sin\omega = R(v)\cos(\omega - \alpha),$$

where the pair $(R(v), \alpha)$ corresponds to the polar coordinates of $(A(v), B(v))$ in dimension 2:

$$\begin{cases} A(v) = R(v)\cos\alpha, \\ B(v) = R(v)\sin\alpha. \end{cases}$$

As a consequence, the problem of finding the minimal and maximal values for $\chi'(v, \omega)$ reduces to the one of finding the optimal values for the cosine function. Notice that the value of α can be computed as

$$\alpha = \arctan\left(\frac{B(v)}{A(v)}\right)$$

and that the same strategy can be used for finding the minimal and maximal values for the other components of $\chi(v, \omega)$ over the given interval for ω. A graphical representation of the obtained boxes is given in Fig. 1.

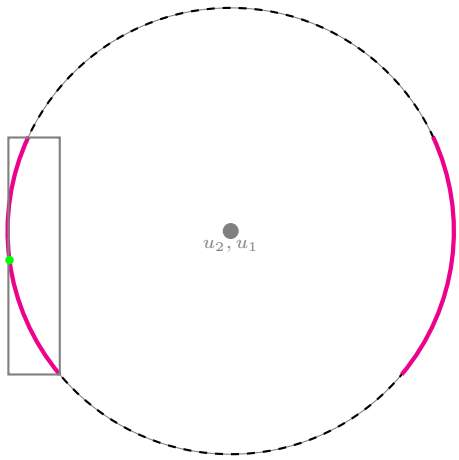

Fig. 1. In dashed line, the circle obtained by intersecting the two spheres centered in u_1 and u_2. In purple, the arcs on the circle obtained by intersecting this circle with the spherical shell centered in u_3. For one arc, we show the box inscribing the arc in gray, together with the arc central position in green.

III. Resolution parameter

At every recursive call of the BP algorithm (see Section II-A), the intersection of two spheres with one spherical shell produces two arcs [11]. The coarse-grained representation (see Section II-B) replaces every arc with a box inscribing the arc (see Section II-D) and a most-likely position, which is initially set at the arc central point. Every generated pair (x_v, B_v) consisting of a position and a box for the vertex v can be then assigned to the nodes of the search tree.

The *resolution* parameter ρ is integrated in the BP algorithm for controling the size of the boxes associated to the tree nodes. This is done at two different levels:

1) If the length of the current arc is larger than the resolution parameter ρ, then the arc is split in a sufficient number of equally-long sub-arcs whose length is smaller

than ρ. Naturally, this strategy implies that the search domain is not binary anymore at all layers, for many vertices may need more than two arcs for satisfying the required resolution. On the one hand, this procedure increases the complexity of the search; on the other hand, it allows to assign smaller boxes to the tree nodes, so that the search domain of the local optimizer becomes smaller as well.

2) If at least one solution has already been found, and the BP algorithm is currently exploring alternative nodes at a given layer v, then only positions x_v whose Euclidean distance from the position of v in the previously found solution is larger than ρ are considered. In fact, when this distance is smaller than ρ, the previous and the current solution are "too" similar, and the current one can therefore be discarded. Notice that, even if this Euclidean distance may be larger than ρ when the nodes are generated, the local optimizer may modify the positions x_v so that it subsequently becomes smaller than ρ. In such a case, the previous and the current solution have the tendency to converge to the same conformation, and thus the current branch can be discarded.

To sum up, the resolution parameter ρ does not only influence the branching phase of the BP algorithm, but it rather performs two kinds of verification during the execution of the algorithm ensuring that all generated solutions differ from one another in accordance with the chosen resolution parameter.

IV. Computational experiments

We present in this section some initial computational experiments on a set of artificially generated instances. All codes were written in C programming language and all experiments were carried out on an Intel Core i7 2.30GHz with 8GB RAM, running Linux. The codes have been compiled by the GNU C compiler v.4.9.2 with the `-O3` flag.

Before showing our computational experiments, we will briefly present the considered application in structural biology concerning the determination of protein structures (see Section IV-A), and we will explain how we generated our instances (see Section IV-B). Section IV-C will present the experiments.

A. Protein structure determination

One of the classical applications of the DGP arises in structural biology [7]. Distances between atom pairs in a given molecule can be estimated through experiments of Nuclear Magnetic Resonance (NMR), so that the possible conformations of the molecule in the three-dimensional space can be identified by solving an instance of the DGP. This application is of relevant interest, especially when dealing with proteins, because the identification of protein conformations can give insights on the dynamics of such molecules, and therefore on their function.

It was proved that protein instances of the DGP belong to the subclass of the DDGP [14], [23]. In many papers cited above (see for example [15], [6], [12]), protein instances are used to perform the experiments. However, as already pointed out in the Introduction, none of such previous works were able to perform an exhaustive search on the domain of the considered instances. Our experiments will show that the BP algorithm, integrated with the new features introduced in this work, is actually able to perform this exhaustive search.

protein	$\|V\|$	$\|E\|$	$\|E'\|$	ρ	#sols	best MDE	time
2jmy	77	428	219	0.5	6	1.73e-05	1m 38s
				1.0	3	1.90e-05	54s
				2.0	2	2.40e-05	51s
2kxa	121	700	367	2.0	2	3.14e-05	45m 28s
				3.0	1	9.94e-05	7m 31s
2ksl	254	1388	684	2.0	2	2.42e-05	16m 55s
				2.0	1	3.47e-05	4m 5s

TABLE I
SOME EXPERIMENTS ON PROTEIN INSTANCES RESEMBLING NMR DATA.

B. Generation of the instances

We selected the protein conformations that were considered in the experiments presented in [6] and [24]. We do not use real NMR data, but we rather generate our protein instances from known models of the selected proteins. The three considered proteins, having codes `2jmy`, `2kxa` and `2ksl` in the Protein Data Bank (PDB) [2], have been experimentally determined by NMR experiments, and, as it is usually the case, more than one model for each protein was deposited. In our instance generation, we have simply considered the first model that appears in the corresponding PDB file.

Our instances are generated in a way to resemble NMR data. From the initial conformation model, we compute all interatomic distances, and we include in our instance the following distances:

- distances between bonded atoms (only one real value approximated to 3 decimal digits);
- distances between atoms bonded to a common atom (only one real value approximated to 3 decimal digits);
- distances between the first and the last atom forming a torsion angle (distances represented by an interval);
- distances between hydrogen atoms that are shorter than 5Å (distances represented by an interval).

In order to define the interval distances, we create an interval of range 0.1Å for the distances related to torsion angles, and an interval of range 0.5Å for distances related to hydrogens, and we place the true distance randomly inside such an interval. The atoms are sorted accordingly to the order proposed in [21], which ensures the discretizability of the instance.

C. Some initial experiments

We present some initial experiments performed by considering the instances generated as detailed in the previous section. Table I summarizes our experiments: the information about the graph representing the DDGP instance is given together with the chosen resolution ρ. Moreover, for every experiment, the

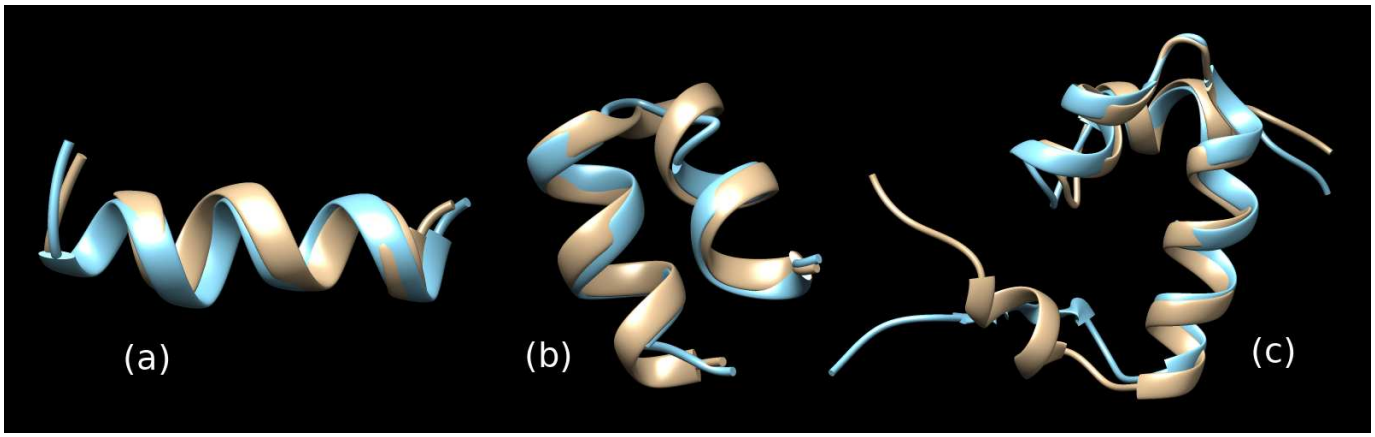

Fig. 2. For every considered protein instance (see Table I), we propose the alignment between the protein model (in gold) that we considered for the generation of the instances, and the best matching solution found by our BP algorithm implementation (in blue). (a) is the alignment obtained for `2jmy`; (b) for `2kxa`; (c) for `2ksl`. The BP solution was subject to energy minimization before the alignment.

total number (#sols) of found solutions is reported, together with the best MDE value:

$$\mathrm{MDE}(x) = \frac{1}{|E|} \sum_{\{u,v\} \in E} \frac{\Delta(||x_u - x_v||, d(u,v))}{d(u,v)}, \quad (4)$$

and with the time in minutes (m) and seconds (s). We point out that the given number #sols of solutions is the value of our solution counter at the end of the *complete* enumeration of the search tree, with the specified resolution parameter ρ. The function $\Delta(y, I)$ in equ. (4) computes the distance on a line between the real value y and the real-valued interval I.

SPG is invoked as a local optimization solver for performing the refinement step (see Section II-B), with the same general settings used in [22]. SPG can terminate because of different criteria: either when the objective function value becomes smaller than 10^{-6}, or when the norm of the search direction becomes smaller than 10^{-6}, or when it reaches the maximum number of allowed interactions, which is set to 10000 in our experiments. Notice that the objective function optimized by SPG is not the MDE function above (MDE is in fact not differentiable in its entire definition domain): more information about SPG and its implementation can be found in [22], [25].

It is easy to see that the newly introduced resolution parameter is able to control the cardinality of the final solution set. The more its value is large, the less are the found solutions (i.e. more solutions are discarded because considered to be too similar to previously obtained solutions). The resolution parameter also influences the total computational time, because it allows to skip all branches potentially leading to similar solutions.

In order to verify how the BP algorithm is able to reconstruct the original protein models used to generate our instances, we have aligned the original structure with the obtained solutions. Before alignment, however, for the two compared structures to be in the same conditions, we optimized the internal energy of the BP structures. To perform such an optimization, the topology/parameter file and the coordinate file were prepared by the `tLEaP` module of the AMBER 16 program suite [5]. The `GBn` model of Mongan et al. [20] was used for the implicit solvent model; the Bondi radii set was also used [4]. 250 steps of steepest descent minimization were followed by 250 steps of conjugated gradient minimization. The MPI version of `pmemd` program of AMBER 16 suite was used for energy minimization.

Fig. 2 shows the obtained alignments. They show that the BP algorithm is actually able to *reconstruct* the original protein model that was used to generate our instances. We can remark moreover that the quality of the solutions, measured through the MDE function, is independent on the resolution parameter, and has a rather constant magnitude in all experiments. Its value indicates a very good quality for the found solutions (recall that the distances represented by only one approximated value are represented with 3 decimal digits).

V. Conclusion

We have presented an extended version of the BP algorithm which allows an efficient exploration of the search tree obtained with the discretization of the DGP. This extended version is in fact capable of enumerating exhaustively the search tree even when the distance information is given through real-valued intervals. A pair consisting of a three-dimensional box and a selected vertex position in the box is associated to every node of the tree, so that the selected position can be refined at further layers of the search tree when new distance information needs to be verified. The inclusion of a resolution parameter allows to generate boxes with controled sizes, and to perform the BP branching phase only when the new added branches lead to the generation of solutions that differ, in accordance with the resolution parameter, from solutions that were already computed.

One of the first objectives of our future works will consist in solving DGP instances containing NMR data that are obtained through the experimental technique. To this aim, the main challenge that we will need to face is given by the lower

precision of the distance information. In fact, the intervals related to distances derived from NMR experiments may correspond to ranges up to 3Å. Moreover, the possibility to skip the energy minimization step (that was performed in our computational experiments) will be studied by including more stringent distance constraints for important hydrogen bonding distances.

Finally, work will be performed for formalizing the concepts related to the introduction of the resolution parameter, which will require a complete understanding of the actual impact of this new parameter on the BP algorithm.

ACKNOWLEDGMENTS

Most of this work was performed during the visit of AM to Academia Sinica (April 2019), and of JHL to IRISA (May 2019), which were made possible thanks to our CNRS-MoST PRC project entitled "Rapid NMR Protein Structure Determination and Conformational Transition Sampling by a Novel Geometrical Approach" (years 2018–19).

REFERENCES

[1] R. Alves, C. Lavor, *Geometric Algebra to Model Uncertainties in the Discretizable Molecular Distance Geometry Problem*, Advances in Applied Clifford Algebra **27**, 439–452, 2017.

[2] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, P. Bourne, *The Protein Data Bank*, Nucleic Acids Research **28**, 235–242, 2000.

[3] E.G. Birgin, J.M. Martínez, M. Raydan, *Spectral Projected Gradient methods: Review and Perspectives*, Journal of Statistical Software **60**(i03), 21 pages, 2014.

[4] A. Bondi, *van der Waals Volumes and Radii*, Journal of Physical Chemistry **68**(3), 441–451, 1964.

[5] D.A. Case, R.M. Betz, D.S. Cerutti, T.E. Cheatham III, T.A. Darden, R.E. Duke, T.J. Giese, H. Gohlke, A.W. Goetz, N. Homeyer, S. Izadi, P. Janowski, J. Kaus, A. Kovalenko, T.S. Lee, S. LeGrand, P. Li, C. Lin, T. Luchko, R. Luo, B. Madej, D. Mermelstein, K.M. Merz, G. Monard, H. Nguyen, H.T. Nguyen, I. Omelyan, A. Onufriev, D.R. Roe, A. Roitberg, C. Sagui, C.L. Simmerling, W.M. Botello-Smith, J. Swails, R.C. Walker, J. Wang, R.M. Wolf, X. Wu, L. Xiao, P.A. Kollman, *AMBER 2016*, University of California, San Francisco, 2016.

[6] A. Cassioli, B. Bardiaux, G. Bouvier, A. Mucherino, R. Alves, L. Liberti, M. Nilges, C. Lavor, T.E. Malliavin, *An Algorithm to Enumerate all Possible Protein Conformations verifying a Set of Distance Restraints*, BMC Bioinformatics **16**:23, 15 pages, 2015.

[7] G.M. Crippen, T.F. Havel, *Distance Geometry and Molecular Conformation*, John Wiley & Sons, 1988.

[8] N.M. Freris, S.R. Graham, P.R. Kumar, *Fundamental Limits on Synchronizing Clocks Over Networks*, IEEE Transactions on Automatic Control **56**(6), 1352–1364, 2010.

[9] D.S. Gonçalves, A. Mucherino, *Discretization Orders and Efficient Computation of Cartesian Coordinates for Distance Geometry*, Optimization Letters **8**(7), 2111–2125, 2014.

[10] D.S. Gonçalves, A. Mucherino, *Optimal Partial Discretization Orders for Discretizable Distance Geometry*, International Transactions in Operational Research **23**(5), 947–967, 2016.

[11] D.S. Gonçalves, A. Mucherino, C. Lavor, *An Adaptive Branching Scheme for the Branch & Prune Algorithm applied to Distance Geometry*, IEEE Conference Proceedings, Federated Conference on Computer Science and Information Systems (FedCSIS14), Workshop on Computational Optimization (WCO14), Warsaw, Poland, 463–469, 2014.

[12] D.S. Gonçalves, A. Mucherino, C. Lavor, L. Liberti, *Recent Advances on the Interval Distance Geometry Problem*, Journal of Global Optimization **69**(3), 525–545, 2017.

[13] M.C. Hout, M.H. Papesh, S.D. Goldinger, *Multidimensional Scaling*, Wiley Interdisciplinary Reviews: Cognitive Science **4**(1), 93–103, 2013.

[14] C. Lavor, L. Liberti, N. Maculan, A. Mucherino, *The Discretizable Molecular Distance Geometry Problem*, Computational Optimization and Applications **52**, 115–146, 2012.

[15] C. Lavor, L. Liberti, A. Mucherino, *The interval Branch-and-Prune Algorithm for the Discretizable Molecular Distance Geometry Problem with Inexact Distances*, Journal of Global Optimization **56**(3), 855–871, 2013.

[16] C. Lavor, L. Liberti, A. Mucherino, N. Maculan, *On a Discretizable Subclass of Instances of the Molecular Distance Geometry Problem*, ACM Conference Proceedings, 24^{th} Annual ACM Symposium on Applied Computing, Hawaii, USA, 804–805, 2009.

[17] L. Liberti, C. Lavor, N. Maculan, *A Branch-and-Prune Algorithm for the Molecular Distance Geometry Problem*, International Transactions in Operational Research **15**, 1–17, 2008.

[18] L. Liberti, C. Lavor, N. Maculan, A. Mucherino, *Euclidean Distance Geometry and Applications*, SIAM Review **56**(1), 3–69, 2014.

[19] L. Liberti, C. Lavor, A. Mucherino, N. Maculan, *Molecular Distance Geometry Methods: from Continuous to Discrete*, International Transactions in Operational Research **18**(1), 33–51, 2011.

[20] J. Mongan, C. Simmerling, J.A. McCammon, D.A. Case, A. Onufriev, *Generalized Born with a Simple, Robust Molecular Volume Correction*, Journal of Chemical Theory and Computation **3**, 156–169, 2007.

[21] A. Mucherino, *On the Identification of Discretization Orders for Distance Geometry with Intervals*, Lecture Notes in Computer Science **8085**, F. Nielsen and F. Barbaresco (Eds.), Proceedings of Geometric Science of Information (GSI13), Paris, France, 231–238, 2013.

[22] A. Mucherino, D.S. Gonçalves, *An Approach to Dynamical Distance Geometry*, Lecture Notes in Computer Science **10589**, F. Nielsen, F. Barbaresco (Eds.), Proceedings of Geometric Science of Information (GSI17), Paris, France, 821–829, 2017.

[23] A. Mucherino, C. Lavor, L. Liberti, *The Discretizable Distance Geometry Problem*, Optimization Letters **6**(8), 1671–1686, 2012.

[24] A. Mucherino, J-H. Lin, D.S. Gonçalves, *Coarse-Grained Representation for Discretizable Distance Geometry with Interval Data*, to appear in Lecture Notes in Computer Science, 2019.

[25] A. Mucherino, J. Omer, L. Hoyet, P. Robuffo Giordano, F. Multon, *An Application-based Characterization of Dynamical Distance Geometry Problems*, to appear in Optimization Letters, Springer, 2019.

[26] J. Saxe, *Embeddability of Weighted Graphs in k-Space is Strongly NP-hard*, Proceedings of 17^{th} Allerton Conference in Communications, Control and Computing, 480–489, 1979.

[27] J. Wang, R.K. Ghosh, S.K. Das, *A Survey on Sensor Localization*, Journal of Control Theory and Applications **8**(1), 2–11, 2010.