Optimal Partial Discretization Orders for Discretizable Distance Geometry

Douglas Gonçalves, Antonio Mucherino

Optimal partial discretization orders for discretizable distance geometry

Douglas S. Gonçalves[1], Antonio Mucherino[2]

Departamento de Matemática, Universidade Federal de Santa Catarina, Brazil.
Email: douglas.goncalves@ufsc.br
IRISA, University of Rennes 1, Rennes, France.
Email: antonio.mucherino@irisa.fr

Abstract. The Distance Geometry Problem (DGP) asks whether a simple weighted undirected graph $G = (V, E, d)$ can be embedded in a given space so that the weights of the edges of G, when available, are the same as the distances between pairs of embedded vertices. The DGP can be discretized when some particular assumptions are satisfied, which are strongly dependent on the vertex ordering assigned to G. In this work, we focus on the problem of identifying optimal partial discretization orders for the DGP. The solutions to this problem are in fact vertex orders that allow for the discretization of the DGP. Moreover, these partial orders are optimal in the sense that they optimize, at each rank, a given set of objectives aimed to improve the structure of the search space after the discretization. This ordering problem is tackled from a theoretical point of view, and some practical experiences on sets of artificially generated instances, as well as on real-life instances, are provided.

1 Introduction

Let $G = (V, E, d)$ be a simple weighted undirected graph. The Distance Geometry Problem (DGP) [20, 29, 30] asks whether G can be embedded in an Euclidean space having dimension $K > 0$ so that all distances d_{uv} (the edge weights), with $(u, v) \in E$, are satisfied (the acronym DGP$_K$ is also employed when it is important to point out the dimension K). When it exists, the embedding $\sigma : V \longrightarrow \mathbb{R}^K$ provides coordinates for every vertex of G, in a K-dimensional space, that satisfy the distance information associated to the edges of G.

The DGP$_K$ is NP-Hard [31] and has several interesting applications. In dimension $K = 1$, the clock synchronization problem is the one of identifying the time of each clock in a given network by exploiting their own offset with respect to a predefined clock, which is used as a reference [13, 36]. In dimension $K = 2$, the sensor network localization problem is the one of positioning in the space the sensors of a given network by using the available relative distances. Such distances can be estimated by measuring the power for a 2-way communication between pairs of sensors [5, 6, 12, 35]. In dimension $K = 3$, molecular conformations can be obtained by exploiting information about distances between atom pairs that can be either derived from the chemical structure of the molecule, or estimated by experiments of Nuclear Magnetic Resonance (NMR) [2, 23]. This problem is also referred to as Molecular DGP (MDGP) [1, 11, 34, 32].

The DGP$_K$ can be discretized when some particular assumptions are satisfied [21]. With the discretization, the search domain of the DGP$_K$ can be reduced from a continuous to a discrete domain having the structure of a tree [17, 28]. The discretization assumptions are strongly based on the ordering assigned to the vertices of graph G which is employed for representing instances of the problem [16]. Vertex orders that allow for the discretization of DGP$_K$ are called *discretization orders*.

In previous works, discretization orders were either handcrafted [10, 18], or searched over pseudo de Bruijn graphs constructed from sets of cliques of G [25], or obtained by means of a greedy algorithm [16, 24], or even found by formulating a constraint programming problem whose solution was attempted by an Answer Set Programming (ASP) solver [14]. In the last cited paper, the idea to search for discretization orders that can satisfy some additional assumptions, which may have an impact on the search domain and make it "easier" to explore, was firstly proposed.

In this work, we formalize the problem of finding optimal partial discretization orders. The discretization assumptions lead to the definition of a partial order for G that allows for the discretization, and from which several total orders can be derived. Our interest is in selecting partial orders that are able to optimize the structure of the search tree. To this purpose, we consider a set of objectives that are defined over subsets of vertices sharing the same rank in the partial order. Such objectives, with a predefined priority level, define a sequence of filters, that can be viewed as multi-level optimization subproblems having finite and relatively small search domains. We propose an algorithm for finding optimal partial discretization orders by solving this sequence of simple subproblems, which has polynomial complexity.

The rest of the paper is organized as follows. In Section 2, we will give some preliminary definitions and briefly introduce the problem of finding partial discretization orders. In Section 3, we will describe an algorithm for the identification of partial orders for G that satisfy the discretization assumptions. In Section 4, we will discuss on how to identify, among such partial orders, the ones that optimize a given set of objectives, with predefined priority levels, over certain subsets of vertices. In Section 5, we will present four objectives to be optimized during the search for optimal partial orders. Section 6 will provide some computational experiments, on artificial and protein-like instances in dimension $K = 3$. Conclusions are given in Section 7.

2 Definitions and preliminaries

Let $G = (V, E, d)$ be a simple weighted undirected graph representing an instance of the DGP$_K$, with $K > 0$. Let $d : E \longrightarrow d_{uv} \equiv [\underline{d}_{uv}, \bar{d}_{uv}] \subset \mathbb{R}_+$ be the weight function, that associates edges to real-valued nonnegative intervals. Since edge weights represent distances, whenever the bounds are non-degenerate, $\underline{d}_{uv} < \bar{d}_{uv}$, we say that d_{uv} is an interval distance. If the interval is degenerate ($d_{uv} = \underline{d}_{uv} = \bar{d}_{uv}$), we say that the distance d_{uv} is exact. Let E' be the subset of E related to the exact distances.

A partition of V is a family of disjoint subsets of V whose union results in V. A partition equipped with a total order on its sets is called an ordered partition [33].

Definition 2.1. *An ordered partition of length p on V, denoted by $r = (r_1, \ldots, r_p)$, is a partition of V in subsets r_i, ranked by $1, 2, \ldots, p$, such that*

1. $\bigcup_{i=1}^{p} r_i = V$, where $r_i \subset V, \forall i \leq p$;
2. $r_i \cap r_j = \emptyset$, $\forall i, j \leq p$, with $i \neq j$.

Let us denote the length p of an ordered partition r by $|r| = p$. The indices $i \leq |r|$ are the *ranks* of these subsets where the vertices of V belong to.

The first condition in Def. 2.1 ensures that every vertex $v \in V$ belongs to at least one subset r_i, with $i \leq |r|$. The second condition avoids the possibility of vertex repetitions, i.e. of orders where the same vertex can appear in more than one set r_i.

Notice that an ordered partition of V induces a partial order on the vertices of V. The *order length* of such partial order is $|r|$, that is, the number of ranks in the partial order. Henceforth, we will use the term *partial order* to refer to the ordered partition related to Def. 2.1

Let $|r_i|$ denote the cardinality of a subset of V.

Proposition 2.2. *An ordered partition $r = (r_1, \ldots, r_{|r|})$ defines a total order on V if and only if $|r_i| = 1$ for each $i \leq |r| = |V|$.*

Proof. Immediate consequence of Def. 2.1 □

During the solution of an instance of the DGP$_K$, when a certain partial order is assigned to the vertices of G, the embedding of these vertices can follow this given ordering. In this way, vertices u can serve as a reference for other vertices v having a greater rank. In fact, if a position for u is already available, it can be used for finding suitable positions for v. In case the distance d_{uv} between the vertex u and the vertex v is known, we can define the sphere centered in u and having radius d_{uv}, which contains the set of feasible positions for v that are compatible with d_{uv}. In this context, we will say that the vertex u is a *reference vertex*, and that the distance d_{uv} is a *reference distance*.

In order to verify the number of references related to the vertices in each subset r_i for a given order r, we introduce the two following subsets of edges:

$$\Lambda^i_\alpha(v) = \{(u,v) \in E \mid \exists j < i,\ u \in r_j\},$$

$$\Lambda^i_\beta(v) = \{(v,u) \in E \mid \exists j \geq i,\ u \in r_j\}.$$

Let r_i be the set of vertices having rank i in the partial order r. Let v be one of such vertices. The cardinality of the set $\Lambda^i_\alpha(v)$ corresponds to the number of reference distances for v. We consider the following counter:

$$\alpha(r_i) = \min_{v \in r_i} |\Lambda^i_\alpha(v)|.$$

This counter can be used for verifying whether all vertices in r_i have enough distances for performing the discretization. Similarly, the counter

$$\beta(r_i) = \max_{v \in r_i} |\Lambda^i_\beta(v)|$$

can be used for counting the maximal number of times a vertex in r_i can play the role of reference vertex for other vertices having a greater rank.

Since the edge set of G can be divided in two parts, one containing only exact distances (E'), and the other containing interval distances ($E \setminus E'$), it is necessary to define counters that are similar to $\alpha(r_i)$ and $\beta(r_i)$, but where only exact distances are considered:

$$\alpha_{ex}(r_i) = \min_{v \in r_i} |\Lambda^i_\alpha(v) \cap E'|,$$

$$\beta_{ex}(r_i) = \max_{v \in r_i} |\Lambda^i_\beta(v) \cap E'|.$$

By considering the counters defined above, we give the following definition of partial discretization order.

Definition 2.3. *A partial discretization order in dimension K is a partial order induced by an ordered partition $r = (r_1, r_2, \ldots, r_{|r|})$ of V such that:*

(a) *$G[r_1] \equiv (C, E_C)$ is a clique with $|C| = |r_1| = K$ and $E_C \subset E'$;*
(b) *$\forall i \in \{2, \ldots, |r|\}$, $\alpha(r_i) \geq K$ and $\alpha_{ex}(r_i) \geq K - 1$.*

where $G[\cdot]$ is the subgraph induced by a subset of vertices.

The search domain of DGP_K instances satisfying assumptions (a) and (b) can be discretized. Assumption (a) allows in fact to embed in one unique position the first K vertices of the partial order r, avoiding this way to consider congruent solutions that can be obtained by rotations and translations [17]. The first K vertices are assigned to the subset r_1 and their relative ordering is not important for the discretization. Assumption (b) ensures that, for every $i > 2$ and for every vertex $v \in r_i$, at least K reference distances exist for v, and that at least $K - 1$ distances are exact. Since the values of $\alpha(r_i)$ and $\alpha_{ex}(r_i)$ are minima over the set of vertices in r_i, the fact that assumption (b) is satisfied by the minima implies that it is satisfied as well for all vertices $v \in r_i$.

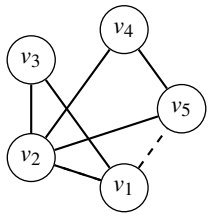

Fig. 1. A graph representing a DGP_2 instance that satisfies the discretization assumptions. Straight edges correspond to exact distances whereas dashed edges correspond to interval ones.

Figure 1 gives an example of a graph which admits a partial discretization order for $K = 2$. It is easy to verify that $r = (r_1, r_2, r_3)$, where $r_1 = \{v_1, v_2\}$, $r_2 = \{v_3, v_5\}$ and $r_3 = \{v_4\}$, is a partial order which fulfills assumptions (a) and (b).

Given a partial discretization order, any total order can be extracted and used for the discretization process. By considering a preselected total order, the discretization of the DGP_K can be performed as follows [20]. Once the first K vertices in the ordering

are placed in their unique positions, the distances that are ensured by assumption (b) are exploited for defining spheres (when the distance is exact) or spherical shells (when an interval represents the distance) that are centered in the reference vertices. Their intersection gives 2 points, with probability 1, when all K distances are exact [22]. If one of such distances is represented by an interval, leading therefore to the definition of one spherical shell, the intersection gives two arcs [18]. Arcs are unidimensional objects that can be discretized by selecting sample positions. The parameter D will refer to the number of sample positions that are extracted from each arc.

The discretization allows to define a discrete search domain for the DGP_K which has the structure of a tree. Layer by layer, candidate vertex positions for the same vertex appear in the tree. For every vertex position on layer $i-1$ of the tree, there are new branches, rooted at this position, and leading to the possible positions for the vertex having rank i. The number of branches depends upon the presence (or not) of a spherical shell in the above-mentioned intersection. The presence of interval distances can potentially increase the width of the search domain.

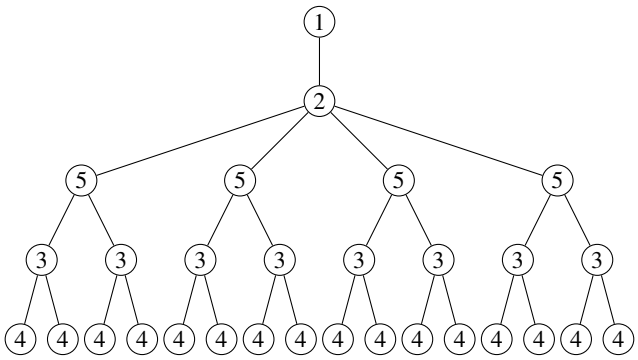

Fig. 2. Search tree related to the instance of Figure 1 for the total order $r = (v_1, v_2, v_5, v_3, v_4)$.

Recall that $r = (\{v_1, v_2\}, \{v_3, v_5\}, \{v_4\})$ is a possible partial discretization order for the DGP_2 instance represented in Figure 1. Consider $(v_1, v_2, v_5, v_3, v_4)$ as a total order extracted from this partial one. Taking $D = 2$, the search domain after the discretization is represented in Fig. 2. Notice that the number of nodes doubles from one layer to another when the involved distances are exact whereas it increases by a factor of $2D$ when an interval distance is used as reference.

The Branch & Prune (BP) is a general framework for the solution of DGP_K for which the search domain can be discretized [18, 19]. The main idea is to construct recursively the search tree, and to check the feasibility of candidate vertex positions as soon as they are computed. In fact, while assumption (b) ensures that the number of reference distances is at least K, more than K references could be actually available. In this case, a subset of K available distances (where at most one is represented by an interval) can be used for the discretization (this subset contains the so-called *discretization distances*); the remaining available distances form instead a subset of distances that can be

Algorithm 1 An algorithm for finding minimal-rank partial discretization orders

find minimal-rank partial discretization orders *in:* G, K *out:* r
// initial clique
choose a K-clique (C, E_C) of G such that $E_C \subset E'$
set $r_1 = C$
set $A = V \setminus C$
set $i = 2$
// constructing the rest of the order
while $(A \neq \emptyset)$ **do**
  r_i = A;
  r_i = `filter`(r_i);
  if $(r_i = \emptyset)$ **then**
    break: no possible orders; **choose** another initial clique
  else
    r_i = `optimize`(r_i);
    $A = A \setminus r_i$; $i = i + 1$;
  end if
end while

exploited for verifying the feasibility of computed candidate solutions (these distances are generally called *pruning distances*).

3 Searching for partial orders

In order to find partial orders for the vertices of G which are discretization orders, we consider an extension of the algorithm that was initially proposed in [16], successively tailored to interval distances in [24], and more recently modified for dealing with partial orders in [26]. Alg. 1 is a sketch of the algorithm that we consider in this work. The basic idea for this algorithm is to find a K-clique from G and, starting from this K-clique, to attempt the construction of a partial discretization order, rank by rank. Alg. 1 makes use of the function `filter`, which is able to select, for every rank i, all vertices for which the discretization assumptions (a) and (b) in Def. 2.3 are satisfied (see Alg. 2). Only vertices that do not appear already in the ordering (i.e. having rank $j < i$) are passed to the function `filter`. Notice that, for a given fixed initial clique, there may exist no discretization orders for G. However, as proved in [26], this algorithm is able to identify partial discretization orders, when they exist. In case no orders can be found

Algorithm 2 The function `filter`

function `filter`(r_i)
while $(\alpha(r_i) < K$ and $r_i \neq \emptyset)$ **do**
  $r_i = r_i \setminus \{u\}$, where $u = \arg\min_{v \in r_i} |\Lambda^i_\alpha(v)|$
end while
while $(\alpha_{ex}(r_i) < K - 1$ and $r_i \neq \emptyset)$ **do**
  $r_i = r_i \setminus \{u\}$, where $u = \arg\min_{v \in r_i} |\Lambda^i_\alpha(v) \cap E'|$
end while

for all possible choices of initial cliques, then we can state that no discretization orders exist for G. Finally, at line 14, Alg. 1 invokes the function `optimize`. We will suppose in this section that it returns the set r_i in output without any change. We will use this function in Section 4 for optimizing, at each rank, a given set of objectives.

Let r be a partial order obtained by applying Alg. 1. Apart from the initial clique, every r_i contains the vertices of V that are eligible to be the next vertex to be embedded during the search of solutions for the corresponding DGP_K in a BP framework. For all these candidate vertices, in fact, enough reference vertices are contained in the sets r_j, with $j < i$, to which candidate positions are already supposed to be assigned. The internal order for the vertices in r_i is irrelevant for the discretization. As a consequence, several different total orders can be constructed from the partial order r, by simply selecting a different permutation of the internal orders for every set r_i containing more than one vertex.

It is important to remark that other partial discretization orders can be deduced from r. The fact that a certain vertex v belongs to r_i suggests that it can be included in no r_j's with $j < i$, because the discretization assumptions would not be satisfied otherwise. However, this vertex v may be placed in sets r_j with $j > i$, as far as v is not a *necessary* reference in between. Thus, partial orders that differ from the original r can be constructed by moving vertices from their original subset r_i to further suitable subsets r_j. Additionally, other partial discretization orders, with a greater order length, can be deduced from r by splitting a subset r_i in smaller subsets and by shifting its successors.

The following proposition characterizes the partial discretization orders that can be obtained by Alg. 1.

Proposition 3.1. *Let G be a simple weighted undirected graph representing an instance of the DGP_K, and let C be a K-clique of G. When it exists, the partial discretization order r for G obtained by Alg. 1, having C as initial clique, has the following properties:*

1. *for $i = 2, \ldots, |r|$, if $v \in r_i$, then $\alpha(\hat{r}_j) < K$ or $\alpha_{ex}(\hat{r}_j) < K-1$, where $\hat{r}_j = r_j \cup \{v\}$ and $j < i$;*
2. *the order length $|r|$ is minimal for C.*

Proof. Given the initial clique C, Alg. 1 assigns the rank set $r_1 = C$. Then, for each $i \geq 2$, the subset r_i is built by selecting, among the set of remaining vertices A, the ones that are allowed to be in r_i. This selection is performed by the routine `filter` which verifies, for every $v \in A$, whether v has enough references in rank sets r_j, with $j < i$. If `filter` returns an empty set, it means that there is no suitable $v \in A$ for the rank r_i, which implies, in its turn, that there is no partial order for the initial clique C satisfying assumptions (a) and (b) of Def. 2.3.

For a given r_i, if $v \in A$ is not returned by `filter`, this is because $\alpha(\hat{r}_i) < K$ or $\alpha_{ex}(\hat{r}_i) < K-1$, where $\hat{r}_i = r_i \cup \{v\}$, that is, v does not have enough references in previous subsets r_j ($j < i$). Thus, if $v \in r_i$, then $\alpha(\hat{r}_j) < K$ or $\alpha_{ex}(\hat{r}_j) < K-1$, where $\hat{r}_j = r_j \cup \{v\}$: in other words, r_i is the first possible rank for v.

Finally, since Alg. 1 assigns vertices $v \in A$ to ranks r_i as soon as possible, the number of rank subsets $|r|$, i.e. the partial order length, is minimal for that initial clique C (no consecutive subsets can be merged). □

Henceforth, the partial discretization order found by Alg. 1 will be called *initial partial order*.

4 Searching for optimal partial orders

A vertex order for G is a discretization order if it allows for discretizing the DGP_K that G represents. The discretization is possible when assumptions (a) and (b) in Def. 2.3 are satisfied. In Section 3, we presented a very simple and yet efficient algorithm for the identification of such orders. Alg. 1, with the function `filter` in Alg. 2, gives as a result one partial order, from which total orders can be constructed. In fact, several different orders may be employed, in the practice, for discretizing DGP_K instances. In this section, our aim is to develop a method for the identification of orders that do not only allow for the discretization, but that are also able to optimize some given criteria. The main idea is to select orders that can have an impact on the search tree, for its exploration to become more efficient.

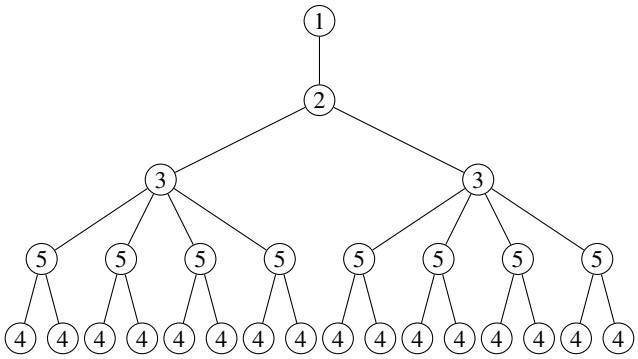

Fig. 3. Search tree related to the instance of Figure 1 for the total order $r = (v_1, v_2, v_3, v_5, v_4)$.

For example, the search tree of Figure 2, related to the instance represented in Fig. 1, was built on the basis of the total order $(v_1, v_2, v_5, v_3, v_4)$ that was extracted from the partial discretization order $r = (\{v_1, v_2\}, \{v_3, v_5\}, \{v_4\})$. Let us consider an alternative partial discretization order $r = (\{v_1, v_2\}, \{v_3\}, \{v_5\}, \{v_4\})$ and the total order $(v_1, v_2, v_3, v_5, v_4)$ extracted from it. Notice that the latter partial order forces v_3 to come before v_5. The former and the latter total orders define two different search trees, represented in Figures 2 and 3, respectively. Although in this case the tree width is constant, we notice a reduction in the number of nodes for the second search tree (Fig. 3). In these examples, we considered $D = 2$, but the difference could be more significant for larger values of D and for graphs with more vertices.

Once an initial clique has been defined (see Alg. 1), for every vertex v that does not belong to this clique, the dependency of v on other vertices can be identified. This information is given by the rank assigned to v, because rank i simply implies that all vertices having a lower rank need to appear before v in all discretization orders (Prop. 3.1).

Alg. 1 is able to provide us with an initial partial order. Every other partial order compatible with this initial order can be obtained by moving vertices to further suitable ranks, and/or by splitting sets r_i in smaller sets.

Prop. 3.1 shows that the initial partial order is such that every vertex v is always included in the lowest-rank subset r_i where it satisfies the discretization assumptions. Moving a vertex v to further ranks can transform the initial order in another partial one having the same length. The specialization of these orders, achieved by splitting subsets r_i in smaller ones, can produce orderings having a greater length. Both operations (moving vertices and splitting sets r_i) are consequences of *optimizing* some objectives over single sets r_i. We introduce the following objectives over partial orders r.

Definition 4.1. *An objective over a rank r_i is a function $f_\ell : S \setminus \{\emptyset\} \longrightarrow \mathbb{R}$, where S is the power set of r_i, that associates subsets of r_i to a real number. We say that an objective f_ℓ over r_i is simple when there exists a unique subset of r_i for which f_ℓ is maximized.*

In this work we consider only *simple* objectives over each rank r_i. Notice that Def. 4.1 restricts the sort of objetive functions that can be used. For instance, let $x \subset r_i$ and consider the following objective over r_i:

$$f(x) = 1/|x|.$$

It is clear that f is maximized when x is a singleton, but any subset $\{v\} \in r_i$ achieves the maximum and thus there is no uniqueness of the maximizer.

We also consider that more than one objective over each r_i can be used. In this case, we will assign priorities to those objectives, following a previous idea first employed in [14]. Henceforth, the subscript $\ell \in \mathbb{N}$ will be the label associated to every objective, which gives the priority order for the objective. We assume, without losing generality, that all objectives are to be maximized, as any objective f_ℓ can be minimized by maximizing $-f_\ell$.

Definition 4.2. *Given a set of $M > 0$ objectives f_ℓ, with priority levels $\ell \in \{1, 2, \ldots, M\}$, an optimal partial discretization order is a partial discretization order where every r_i, with $i > K$, is solution of the multi-level optimization problem*

$$\begin{aligned} &\max_{x \subset x_M} f_M(x) \\ &\quad \text{s.t. } x_M = \arg\max_{x \subset x_{M-1}} f_{M-1}(x) \\ &\qquad \text{s.t} \ldots \\ &\qquad\quad \text{s.t. } x_2 = \arg\max_{x \subset x_1} f_1(x), \end{aligned} \tag{1}$$

where x_1 is the initial set of vertices that admit rank i.

Multi-level optimization is a class of generally very hard optimization problems [3, 9]. The increase in hardness is commonly due to the considered objectives, that, at different levels of the optimization problem, may have contrasting effects during the optimization process. In our case, the situation is much simpler, because the multi-level problem is defined over a discrete set x_1, containing all the vertices that are candidates

Algorithm 3 The function `optimize`

1: function `optimize`(r_i)
2: **for** (each objective f_ℓ, with $\ell = 1,\ldots,M$) **do**
3: $r_i = \{v \in r_i : f_\ell \text{ is optimized}\}$
4: **end for**

for being placed in the subset r_i. In order to solve our simple multi-level optimization problem, the objectives f_ℓ can be optimized one by one, by taking into consideration their priority levels. Alg. 3 is a sketch of a function devoted to this task, to be invoked at line 14 of Alg. 1.

Alg. 1 can be divided into two main parts. First, a K-clique C of G is identified, which will play the role of the initial clique in the order. Then, an iterative procedure starts for the construction of the rest of the ordering. At iteration i, from the set A of all vertices that do not appear in any r_j, with $j < i$, the subset r_i needs to be defined. To this purpose, the set is first of all filtered by invoking the function `filter`, where all vertices satisfying the discretization assumptions (a) and (b) are selected. We remark that the function `filter` may reduce the initial set to an empty set: in this case, there are no possible discretization orders having C as an initial clique. After, the set is filtered for a second time by invoking the function `optimize`. This function removes from the set r_i all vertices that do not optimize the objectives, in their priority levels. Notice that the function `optimize` cannot reduce a non-empty set to an empty one.

Theorem 4.3. *Let $G = (V,E,d)$ be a simple weighted undirected graph representing an instance of the DGP$_K$. When they exist, Alg. 1, used in conjunction with Alg. 2 and Alg. 3, is able to construct partial discretization orders for G, which are optimal.*

Proof. The correctness of Alg. 1, using the routine `filter` (Alg. 2), was proved in Prop. 3.1. Since `optimize` (Alg. 3) is an additional filter applied to the initial r_i (the set of vertices allowed in rank i) that always returns a non-empty subset of r_i, the vertices returned by `optimize` still satisfy the discretization assumptions. The vertices that were in r_i discarded by `optimize` will be considered again for further ranks, and since they were eligible for rank i, they will be eligeble for ranks $j > i$. Therefore, we end up with a partial discretization order. The optimality of each rank subset r_i follows from the definition of `optimize` and the fact that we are considering simple objectives. □

One consequence of the optimization of the objectives is that it tends to increase the number of ranks (i.e. the number of sets r_i).

After the optimization of the set of objectives, the number of total orders that can be extracted from the optimal partial one tends to decrease. In other words, total orders that can be obtained from an optimal partial order are also compatible with partial orders that are not optimized (such as the partial orders that Alg. 1 can find).

The next result states the complexity of Alg. 1 (which uses Alg. 2) using the additional filter Alg. 3.

Theorem 4.4. *Alg. 1, used in conjunction with Alg. 2 and Alg. 3, has polynomial complexity.*

Proof. Line 3 of Alg. 1 requires a search for a K-clique in G, where the dimension K is a priori known. The task of enumerating all these K-cliques can be performed in $O(n^K)$, where $n = |V|$. Every K-clique can potentially contain the first K vertices of a partial discretization order. For each of such cliques, the construction of the rest of the partial order has the complexity of the function `filter`, plus the complexity of the function `optimize`. The complexity of the function `filter` is linear: $O(n)$. The function `optimize` contains a `for` loop which, at each iteration, considers a different objective f_ℓ and performs its optimization on a discrete subset r_i. By hypothesis, the objectives f_ℓ over r_i are simple, which means there exists a unique subset of vertices in r_i achiving the optimum. Therefore, a simple loop over the vertices in r_i is sufficient for identifying the ones that do not *optimize* the objective f_ℓ. The complexity of the function `optimize` is therefore $O(n \times M)$, where M is the number of objectives. As a consequence, the overall complexity of Alg. 1 is polynomial. □

It is worth mentioning that the complexity of the problem of finding a partial discretization order would become exponential if the objectives f_ℓ were *not simple* (see Def. 4.1).

5 Objectives to be optimized

Recall that partial discretization orders have an impact on the strucuture of the search trees of DGP$_K$ instances (see examples in Fig. 2 and 3).

In this section we present some possible simple objectives to be optimized during the search for discretization orders. In the computational experiments in Section 6, subsets of objectives will be considered, with different priority levels. Therefore, Latin letters will be used in the following for the objectives' subscripts, as their priority levels are not pre-defined. These objectives are conceived with the aim of improving the structure of the search tree, and in particular for reducing the number of nodes it contains. We point out, however, that the optimization of such objectives, for particular instances, may result in worse tree structures, even if the experiments in Section 6 will show that this is unlikely to happen.

We present three objectives that are suitable for all classes of DGP$_K$s. They are mostly based on properties of reference distances for the vertices belonging to the subsets obtained at line 10 of Alg. 1. Recall that the function `filter` (see Alg. 2) allows to identify all vertices, that do not appear yet in order, for which the discretization assumptions (a) and (b) in Def. 2.3 are satisfied. For every selected vertex v, therefore, there exist *at least* K reference distances, where only one distance can be represented by an interval, while all remaining $K-1$ distances are exact. It is also important to verify, given a certain distance d_{uv}, the difference in ranks for the two vertices u and v.

The fourth objective that we present in this work was particularly conceived for the class of DGP$_3$ related to molecular conformations.

5.1 Early use of exact distances

The counter $\alpha_{ex}(r_i)$ (see Section 2) is able to provide the minimum number of exact reference distances over the vertices in r_i. By applying Alg. 2, it is possible to select

the vertices for which the discretization assumptions are satisfied, and in particular the vertices for which the value of $\alpha_{ex}(r_i)$ is at least $K-1$. We are interested moreover to maximize the value of $\alpha_{ex}(r_i)$ at every rank. In fact, if the number of exact distances is K or more, the Euclidean objects to be intersected for computing candidate solutions (see Section 2) are K or more (hyper-)spheres, which, with probability 1, consists of 2 or even 1 point only [22].

Our objective $f_A(r_i)$ is therefore $\alpha_{ex}(r_i)$. The early use of exact distances employed for generating the search tree implies a reduction of the tree width, as well as of the total number of nodes composing the tree. In order to maximize the objective $f_A(r_i)$, vertices that make the value of $\alpha_{ex}(r_i)$ decrease can be unequivocally identified and moved to further ranks.

5.2 Early pruning

The counter $\alpha(r_i)$ is similar to $\alpha_{ex}(r_i)$ (see Section 2), but it considers all kinds of distances (either exact or represented by an interval). When the value of $\alpha(r_i)$ is maximized, not only distances necessary for the discretization are available (as ensured by Alg. 2), but additional distances, that can be exploited for pruning purposes, may also be available.

Thus, our objective $f_B(r_i)$ corresponds to $\alpha(r_i)$. In fact, if pruning occurs at upper layers of the tree, its width may be reduced. This objective was already considered in previous publications, first in [16], and then in [14].

5.3 Minimization of the rank difference in pruning distances

For a given discretization order, if there are reference distances $(v,w) \in E$, with $v \in r_i$ and $w \in r_k$ such that $k >> i$, then there are several nodes in the subtree rooted at v and having w as a leaf. In this situation, the search on this subtree can be rather expensive, because candidate positions for v can be discovered to be infeasible only when reaching the subtree layer corresponding to w (in the situation where no position for w can be found such that d_{vw} is satisfied). In order to reduce the impact of this kind of pruning distances on the search, we consider the following objective:

$$f_C(r_i) = \max_{v \in r_i} \{ i - j \,|\, \exists j < i, u \in r_j, (u,v) \in E \},$$

which provides the maximal difference between the rank i and the ranks containing the reference vertices of every $v \in r_i$.

When this objective is maximized, only the vertices v that make the maximal rank difference as large as possible are identified. When optimized, this objective allows to select, as soon as they are available, the vertices related to the farthest reference distances, in terms of ranks. If not selected at the current rank i, in fact, these vertices would make the rank difference for the reference distances larger, and the impact of these distances on the search would be amplified.

5.4 Early use of distances between pairs of hydrogens

When dealing with molecules, the distance information can come from two different sources. First of all, the chemical structure of the molecule (which is generally known as in the case of protein conformations) can provide exact values for some distances [23]. Secondly, experiments of Nuclear Magnetic Resonance (NMR) [2] can provide approximations of some short-range distances between pairs of hydrogen atoms. Therefore, the use of this kind of distances at upper layers of the search tree can help reducing the tree width, because these distances are likely to be represented by tighter intervals (not only based on chemical or geometrical constraints). Let $\mathbb{H}$ be the subset of E such that all edges are related to hydrogen pairs:

$$\mathbb{H} = \{(u,v) \in E : u \text{ and } v \text{ are hydrogens}\}.$$

Let us consider the following counter of edges related to hydrogens, defined for every set r_i (i.e. for every rank in the order):

$$\alpha_{\mathbb{H}}(r_i) = \min_{v \in r_i} |\Lambda^i_\alpha(v) \cap \mathbb{H}|.$$

Our objective $f_D(r_i)$ is therefore $\alpha_{\mathbb{H}}(r_i)$.

6 Computational experiments

We present in this section a set of computational experiments where we compute optimal partial discretization orders for artificially generated instances, as well as for instances related to protein conformations. In Section 6.1, we will present a simple method for estimating an upper bound on the number of nodes that form our search trees. In fact, our objectives are conceived with the aim of reducing the number of nodes of these trees, for making their exploration more efficient. Then, in Section 6.2, we will discuss in detail one small handcrafted instance, with the aim of showing the impact of every objective on the found partial orders. The following two sections will present the experiments. Section 6.3 will firstly describe the method we employ for generating our set of instances, and then will present and discuss the computational experiments on this set. Section 6.4 will be devoted to the experiments related to protein conformations.

6.1 Estimating the number of nodes in the search tree

The four objectives presented in Section 5 are conceived with the aim of improving the structure of the search tree, mostly by decreasing the number of nodes it contains. In this section, therefore, we propose a simple algorithm for estimating the number of nodes in the search tree implied by a partial discretization order. This estimation is based on the discretization and on the BP framework (see Section 2). When the set of feasible positions for a given vertex v are computed by using an interval distance (all others are exact), then this set consists of two arcs. Such arcs can be discretized by extracting sample positions. In a BP framework, the parameter D corresponds to the number of sample positions that are taken from each arc.

Algorithm 4 An algorithm for estimating the number of nodes of the search tree

1: **Number of nodes estimation** *in: r, D, K out: nodes*
2: $prev\_width = 1$, $nodes = K$;
3: **for** $(i = K+1, |r|)$ **do**
4: $L = \emptyset$;
5: **for** $(v \in r_i)$ **do**
6: **if** $(\alpha_{ex}(v) \geq K+1)$ **then**
7: $b = 1$;
8: **else**
9: **if** $(\alpha_{ex}(v) \geq K)$ **then**
10: $b = 2$;
11: **else**
12: $b = \max\{2, (2*D)/[(\alpha(v) - K) + \alpha_{\mathbb{H}}(v)]\}$;
13: **end if**
14: **end if**
15: $L = L \cup \{b\}$;
16: **end for**
17: **for** $(b \in L$, in decreasing order$)$ **do**
18: $prev\_width = prev\_width \times b$;
19: $nodes = nodes + prev\_width$;
20: **end for**
21: **end for**

Alg. 4 is a sketch of the algorithm that we use for estimating the number of nodes in the search tree. It takes in input a partial order r and the parameter D, and it outputs the estimated number of nodes in the corresponding search tree. The algorithm performs a worst-case estimation, by providing an upper bound on the number of nodes for all possible trees obtained from total orders that are compatible with the input partial order r. In fact, for every subset r_i, we consider the "worst" ordering for its vertices, i.e. the ordering that gives rise to the maximal local amplification of the tree width. In Alg. 4, this is implemented by using the set L which contains, for all r_i, the amplification factor b from the previous layer to the current layer, for every vertex $v \in r_i$. Then, the values in L are reordered in decreasing order, for the vertex internal ordering in r_i to be the worst possible ordering.

The amplification factor b, for every vertex v belonging to one of the sets r_i defining the partial order, is computed as follows. When more than K exact reference distances are available, then there is no branching in the tree and $b = 1$; when the number of exact reference distances is equal to K, then $b = 2$ (refer to Section 2). Otherwise, we compute an estimation of b by applying the formula at line 12 of Alg. 4. The number D of sample positions taken from the 2 arcs obtained during the discretization is divided by the number of distances that are available for pruning purposes (distances that are not necessary for the discretization), plus the number of distances that are related to hydrogen atoms (only for protein-like instances). In fact, as the number of pruning distances increases, there are more and more chances in the BP framework to tighten the arcs and reduce the number of feasible sample positions. Moreover, when working with proteins, the distances between pairs of hydrogens give important information about

	rank 1	rank 2	rank 3	rank 4	rank 5	rank 6	rank 7	number of nodes
NO	v_1, v_2, v_3	v_6	v_4, v_5	v_7, v_8	v_9	$\emptyset$	$\emptyset$	285
A	v_1, v_2, v_3	v_6	v_5	v_4, v_8	v_7	v_9	$\emptyset$	129
B	v_1, v_2, v_3	v_6	v_4, v_5	v_8	v_7	v_9	$\emptyset$	141
C	v_1, v_2, v_3	v_6	v_4, v_5	v_8	v_7	v_9	$\emptyset$	141
ABC	v_1, v_2, v_3	v_6	v_5	v_4	v_8	v_7	v_9	73

Table 1. The initial partial order and some optimal partial orders for the handcrafted instance in Fig. 4. D is fixed to 2. NO means that no objective was optimized. The objectives labeled with A, B or C were optimized, one per time, in the other (specialized) partial orders. ABC means that the three objectives were considered together and in that priority order.

the global fold of the molecule, which can help in pruning the search tree. This is the reason why we count these distances twice in our formula at line 12.

6.2 Analysis of a handcrafted instance

We consider in this section a very small instance that we handcrafted with the aim of showing how the set of objectives in Section 5 are able to modify the initial partial order obtained by Alg. 1. The instance is formed by 9 vertices connected with edges representing either exact distances (straight edges) or interval distances (dashed edges, see Fig. 4). The thick edges represent the initial clique C. Table 1 shows the initial partial

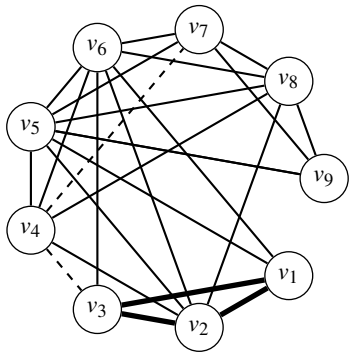

Fig. 4. A handcrafted instance.

order that it is possible to obtain from Alg. 1 without the optimization of any objective, as well as the optimal orders obtained with different objectives. The initial order has 5 ranks, with some vertices sharing the same rank with others (see first line in the table, "NO" means that no objectives are optimized). Notice that we included all the vertices of initial clique C in r_1, as the internal ordering of the vertices forming this clique is irrelevant for the discretization (and for the optimization of the objectives). When the objective f_A is considered, the total number of ranks grows to 6, and the vertex v_4, that

has less exact reference edges than v_5, is pushed at rank 4. The vertex v_7 (for which the discretization assumptions are already satisfied at rank 4), in its turn, is relegated to rank 5, because vertices v_4 and v_8 have more exact reference edges.

When the objective f_B is used, a different specialization in the ordering is introduced. In this particular case, the optimal order found when using f_B coincides with the one obtained when using f_C. We do not consider here the objective f_D because it is particularly related to protein conformations. Finally, when all objectives f_A, f_B and f_C are considered, in alphabetic priority order, we obtained a specialization of the initial order that is a total ordering with 7 ranks. The estimated number of nodes for the tree obtained from this total ordering is almost 4 times smaller than the number of nodes corresponding to the initial partial order.

6.3 Experiments on artificial instances

We report in this section some computational experiments on artificial generated instances. These instances are not related to any particular class of the DGP$_K$ and have been generated by using the following procedure. Let n be the size of the instance, and K its dimension. This procedure constructs a simple weighted undirected graph G, that is supposed to represent our instance, where no numerical values (the distances) are associated to its edges, but rather the binary information: "exact/interval" distance.

Let $V = \emptyset$ and $E = \emptyset$. At the beginning, K vertices forming the K-clique (C, E_C), with only exact edges, are generated. To each appended vertex v, we assign a real number $p(v) \in [0,1]$: this is the predefined probability for a vertex v to serve as a reference for further vertices. We set $V = V \cup C$, and $E = E \cup E_C$.

Once the initial clique has been defined, we iterate the following steps for every additional vertex to be added in the graph, i.e. $n-K$ times. In each iteration, we consider a new vertex v ($V = V \cup \{v\}$). The edge e between v and a vertex u in the current V is created with probability $p(u)/(c \times |v-u|)$, where c is a positive constant in the real interval $[0,1]$. The distance $|v-u|$ represents here the number of iterations of our procedure separating the generation of u with the generation of v. A random mechanism assigns the label "exact", or rather the label "interval", to every generated edge. Every time a new edge e is created, it is added to the edge set: $E = E \cup \{e\}$.

Notice that this procedure does not ensure that the final graph G will be discretizable. At each iteration of our procedure, therefore, we verify whether the discretization assumptions (a) and (b) are satisfied in the natural vertex ordering implied by the procedure. If not, we include additional edges to the current vertex v.

Table 2 shows some experiments where we consider instances generated by the procedure above, for different values of the size n (which corresponds to the cardinality $|V|$ of the vertex set). Other properties of the instances, such as $|E|$ and $|E'|$, are also reported in the table. The dimension of all instances is fixed to $K = 3$. We point out that the order of magnitude, in base 10, for the estimated number of nodes is here reported (see Section 6.1), and not the actual value. In parenthesis, the partial order length $|r|$ is also given. In all experiments, the initial clique C is fixed to the one obtained during the random procedure described above.

These experiments show that the optimization of the objectives makes actually decrease the order of magnitude of the size of the search tree, measured in terms of total

instance				tree nodes ($\|r\|$)						
name	$\|V\|$	$\|E\|$	$\|E'\|$	NO	A	B	C	BA	CBA	BAC
r100	100	728	245	42(9)	38(22)	28(52)	35(27)	27(62)	31(82)	28(91)
r200	200	889	426	99(23)	98(34)	91(88)	92(148)	91(102)	92(185)	88(191)
r300	300	1903	673	127(20)	127(33)	99(128)	114(163)	98(163)	112(271)	97(290)
r400	400	3363	921	164(17)	159(32)	122(184)	142(189)	122(212)	140(367)	121(386)
r500	500	3832	1120	209(19)	205(39)	152(215)	180(284)	152(257)	180(463)	150(490)
r600	600	6205	1434	236(17)	219(43)	176(280)	199(237)	176(326)	197(558)	174(590)

Table 2. Number of nodes (in order of magnitude) of the search trees related to the optimal partial orders found by Alg. 1 with different priority orders for the objectives; in parenthesis, we provide the length $|r|$ of the optimal partial order. For these instances, we fixed $D = 2$. The labels NO, A, B, C have the meaning as in Table 1.

nodes. The optimization of the objectives implies as well an increase in the ordering length. The objective that is able to give major benefits to the search tree is, for this set of instances, the objective f_B. This objective anticipates the use of distances through the layers of the search tree (see Section 5). The objective f_A, which differently from f_B anticipates only exact distances, is also able to improve the structure of the tree, but not as much as f_B does. Finally, the objective f_C, which minimizes the rank difference in pruning distances, has, if compared to f_A and f_B, an intermediary effect on the tree. We remark moreover that some orders obtained with some combinations of all objectives are *almost* total orders. This is due to the fact that, during the optimization of several objectives, the initial partial order becomes more and more specialized. One of such examples can be seen in Table 2 for the instance rand600, for the ordering obtained with the objectives f_B, f_A and f_C, in this priority order.

The decrease in magnitude of the tree size has a great impact on the execution of the BP framework. For the instance rand600, the reduction in the worst case tree size is of 10^{62} times the original number of nodes. It is important to remark that, for these artificial instances, the length of the initial partial orders is generally very small. This implies a great freedom for the vertices during the optimization process.

6.4 Experiments on protein-like instances

We consider in this section a set of protein-like instances. These instances have been obtained from a small set of protein conformations downloaded from the Protein Data Bank (PDB) [4]. From each PDB file, we extract the coordinates of all atoms forming their protein backbones, and we compute pairwise distances between such pairs of atoms. The distances related to bond lengths or defined by bond angles are considered as exact. The peptide plane gives rise to the definition of other exact distances in the protein backbones as well. Finally, distances between pairs of hydrogens are also included, if their relative distance is shorter than 5Å. This is done for simulating distances obtained by NMR experiments [23].

We remark that torsion angles among quadruplets of atoms in protein backbone may define additional distances, that are described by intervals representing the minimal

and maximal extension of these torsion angles. Although these interval distances were commonly used in previous works, we have not included them in our instances. If we had have done that, the results below would have significantly improved. However, these distances are not really able to help the search: they are not able to give a relevant contribution to the pruning phase of the BP framework. We considered therefore unfair to include these distances, because Alg. 4 would have counted them as any other pruning distance.

Table 3 shows computational experiments on a set of protein-like instances. In this case, the changes in magnitude are not so expressive. In fact, the partial orders that are initially obtained for every instance have longer lengths, if compared to the ones in Table 2. This is a consequence of the regular structure of protein backbone, which gives few freedom for the vertices(atoms) to change their rank in the optimal orders. It is possible to see more significant decreases in magnitude for proteins that are more compact, such as the proteins 2jwu and 1yez. For the former, the reduction in magnitude for the number of nodes is from 159 (this is the initial partial order) to 153 (when the objectives f_B, f_C and f_D are optimized, in this priority order). Notice that a reduction of 6 orders of magnitude corresponds to a decrease of one million times the total number of nodes in the search tree.

When working with proteins, the objective f_A does not provide any improvement. In fact, the initial partial order(where the initial clique C is formed by the first 3 atoms of the first amino acid in the protein) is already *almost* optimized for f_A. Only one change is introduced at the early ranks, which has as effect to delay two vertices from rank i to rank $i+1$, without modifying the total length of the order. Even if this modification decreases the value of the amplification factor b (see Alg. 4) at rank i, the added vertices at rank $i+1$ make this amplification larger at rank $i+1$. This phenomenon appears in all our protein-like instances: the length $|r|$ is constant, while the magnitude of the search tree size increases. This is the perfect example to point out that our objectives are conceived for attempting the improvement of the search tree structure, but, depending on the particular considered graph, they may actually fail. In other words, even if the objective f_A imposed in these orders the early use of exact distances, this did not help in reducing the order of magnitude for the total number of nodes. In the overall set of our experiments, however, we could observe this phenomenon only in this particular situation.

For these instances, the main improvement is observed for the objective f_B, the one that tries to anticipate the interval pruning distances. However, the resulting reduction in terms of number of nodes becomes relevant only for proteins presenting long-range pruning distances, i.e. distances between atoms that are close in space but far in the backbone chain.

7 Conclusions

Finding vertex orders for simple weighted undirected graphs G that allow for discretizing the underlying DGP_K is a fundamental pre-processing step for the solution of such problems by using the BP framework. In this work, we focused on the problem of find-

instance					tree nodes ($\|r\|$)						
name	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\mathbb{H}\|$	NO	A	B	C	D	CB	BC	BCD
1tor	77	374	230	130	29(40)	30(40)	29(45)	28(46)	28(46)	28(47)	28(47)	28(48)
1niz	107	519	324	184	41(55)	42(55)	40(66)	40(63)	40(63)	40(66)	40(66)	40(67)
2jmy	120	660	366	268	42(60)	43(60)	42(71)	42(68)	42(72)	42(73)	42(73)	42(74)
2kxa	177	973	546	384	61(95)	62(95)	61(110)	61(104)	61(108)	61(111)	61(111)	61(112)
2ppz	287	1522	886	578	102(144)	103(144)	100(170)	100(167)	100(172)	100(176)	100(176)	100(179)
2e2f	315	1716	971	698	115(164)	116(164)	111(198)	112(193)	112(194)	111(198)	111(198)	111(199)
2erl	324	1792	1015	715	113(160)	114(164)	110(194)	111(187)	111(193)	110(200)	110(200)	110(201)
2jws	446	2494	1385	1016	152(224)	154(224)	149(272)	149(264)	150(267)	149(277)	149(277)	149(278)
2jwu	448	2416	1391	976	159(224)	160(224)	153(275)	153(271)	154(267)	154(278)	153(278)	153(279)
2kiq	455	2452	1411	947	159(227)	160(227)	157(278)	157(259)	157(274)	157(282)	157(282)	157(284)
2low	497	2650	1544	1026	175(256)	176(256)	172(306)	172(298)	172(299)	172(311)	172(311)	172(312)
1yez	531	2813	1646	1108	190(272)	191(272)	183(327)	183(328)	184(328)	184(332)	183(332)	183(334)

Table 3. The number of nodes (in order of magnitude) for the search trees related to the optimal partial orders found by Alg. 1 and the length of each order (in parenthesis). Alg. 4 is employed for estimating the number of nodes. D is fixed to 8. The labels NO, A, B, C have the same meaning of Table 1.

ing *optimal* partial discretization orders, so that our orderings do not only allow for the discretization, but also for optimizing a given set of objectives at each rank.

To this aim, we have extended the algorithm that was previously presented in [16, 24] for dealing with partial discretization orders. Then, among the set of obtained partial orders, we identified a subset of (partial)orders that, rank by rank, are solutions of small multi-level optimization problems. During this optimization process, we considered four different objectives, conceived for reducing the size (in terms of nodes) of the search tree obtained with the discretization. We presented three objectives that are general for every instance of the DGP$_K$, and one objective that is designed for dealing with protein-like instances.

Our computational experiments showed that all objectives, when optimized on subsets of vertices related to an initial partial order, are able to improve the structure of the search tree. The anticipation of exact distances (objective f_A), as well as the anticipation of all kinds of distances (objective f_B), are generally able to reduce the total number of nodes composing the search tree. However, we noticed that objective f_A does not give any significant improvement when protein-like instances are considered. The objective f_C, that forces pruning distances to cross rank sets having the minimal rank difference, was also proved to be effective on our sets of instances. Finally, the anticipation of the distances between hydrogen atoms, encoded by the objective f_D when protein-like instances are concerned, has an impact on the tree that is similar to the one given by f_B, as the most available distances in protein instances are (pruning) distances between hydrogens. We tried experimentally to consider different combinations of such objectives, in different priority levels, in order to discover, for a certain class of instances, the one that generally leads to better results.

Additionally, the experiments show that a considerable reduction in the number of nodes of the search tree can be achieved for graphs with a reasonably dense set of edges, because such structure gives more freedom for optimizing the ranks in the initial partial order.

The conception of new objectives to be included in our multi-level optimization represents one research line that we will explore in our future works. The inclusion of side chains in our protein-like instances is also another interesting point, as they may be theoretically "detached" from the protein backbone in order to optimize our objectives. For the particular case of instances related to protein conformations, where the possible orders are constrained by their chemical structure, this work will have to be integrated with novel suitable strategies for an efficient reduction of the search tree size.

Acknowledgments

AM would like to thank the "Direction de la recherche et de l'innovation" (DRI) of University of Rennes 1 for financial support.

References

1. B. Alipanahi, N. Krislock, A. Ghodsi, H. Wolkowicz, L. Donaldson, M. Li, *Determining Protein Structures from NOESY Distance Constraints by Semidefinite Programming*, Journal of Computational Biology **20**(4), 296–310, 2013.

2. F.C.L. Almeida, A.H. Moraes, F. Gomes-Neto, *An Overview on Protein Structure Determination by NMR. Historical and Future Perspectives of the Use of Distance Geometry Methods*, In [29], 377–412, 2013.
3. G. Anandalingam, T.L. Friesz, *Hierarchical Optimization: An Introduction*, Annals of Operations Research **34**(1), 1–11, 1992.
4. 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.
5. P. Biswas, T. Lian, T. Wang, Y. Ye, *Semidefinite Programming based Algorithms for Sensor Network Localization*, ACM Transactions in Sensor Networks **2**, 188–220, 2006.
6. P. Biswas, Y. Ye, *A Distributed Method for Solving Semidefinite Programs Arising from Ad Hoc Wireless Sensor Network Localization*. In: "Multiscale Optimization Methods and Applications", W.W. Hager, S-J. Huang, P.M. Pardalos, O.A. Prokopyev (Eds.), Nonconvex Optimization and Its Applications series **82**, 69–84, 2006.
7. 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.
8. I.D. Coope, *Reliable Computation of the Points of Intersection of n Spheres in n-space*, ANZIAM Journal **42**, 461–477, 2000.
9. B. Colson, P. Marcotte, G. Savard, *An Overview of Bilevel Optimization*, Annals of Operations Research **153**, 235–256, 2007.
10. V. Costa, A. Mucherino, C. Lavor, A. Cassioli, L.M. Carvalho, N. Maculan, *Discretization Orders for Protein Side Chains*, Journal of Global Optimization **60**(2), 333–349, 2014.
11. G.M. Crippen, T.F. Havel, *Distance Geometry and Molecular Conformation*, John Wiley & Sons, 1988.
12. Y. Ding, N. Krislock, J. Qian, H. Wolkowicz, *Sensor Network Localization, Euclidean Distance Matrix Completions, and Graph Realization*, Optimization and Engineering **11**(1), 45–66, 2010.
13. 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.
14. D. Gonçalves, J. Nicolas, A. Mucherino, C. Lavor, *Finding Optimal Discretization Orders for Molecular Distance Geometry by Answer Set Programming*. In: "Studies in Computational Intelligence", S. Fidanova (Ed.), to appear, 2015.
15. D.S. Gonçalves, A. Mucherino, *Discretization Orders and Efficient Computation of Cartesian Coordinates for Distance Geometry*, Optimization Letters **8**(7), 2111–2125, 2014.
16. C. Lavor, J. Lee, A. Lee-St.John, L. Liberti, A. Mucherino, M. Sviridenko, *Discretization Orders for Distance Geometry Problems*, Optimization Letters **6**(4), 783–796, 2012.
17. C. Lavor, L. Liberti, N. Maculan, A. Mucherino, *The Discretizable Molecular Distance Geometry Problem*, Computational Optimization and Applications **52**, 115–146, 2012.
18. 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.
19. 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.
20. L. Liberti, C. Lavor, N. Maculan, A. Mucherino, *Euclidean Distance Geometry and Applications*, SIAM Review **56**(1), 3–69, 2014.
21. 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.
22. L. Liberti, B. Masson, J. Lee, C. Lavor, A. Mucherino, *On the Number of Realizations of Certain Henneberg Graphs arising in Protein Conformation*, Discrete Applied Mathematics **165**, 213–232, 2014.

23. T.E. Malliavin, A. Mucherino, M. Nilges, *Distance Geometry in Structural Biology: New Perspectives*. In [29], 329–350, 2013.
24. 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.
25. A. Mucherino, *A Pseudo de Bruijn Graph Representation for Discretization Orders for Distance Geometry*, Lecture Notes in Computer Science **9043**, Lecture Notes in Bioinformatics series, F. Ortuño and I. Rojas (Eds.), Proceedings of the 3^{rd} International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO15), Granada, Spain, 514–523, 2015.
26. A. Mucherino, *Optimal Discretization Orders for Distance Geometry: a Theoretical Standpoint*, Lecture Notes in Computer Science 9374, Proceedings of the 10^{th} International Conference on Large-Scale Scientific Computations (LSSC15), Sozopol, Bulgaria, June 2015.
27. A. Mucherino, C. Lavor, T. Malliavin, L. Liberti, M. Nilges, N. Maculan, *Influence of Pruning Devices on the Solution of Molecular Distance Geometry Problems*, Lecture Notes in Computer Science **6630**, P.M. Pardalos and S. Rebennack (Eds.), Proceedings of the 10^{th} International Symposium on Experimental Algorithms (SEA11), Crete, Greece, 206-217, 2011.
28. A. Mucherino, C. Lavor, L. Liberti, *The Discretizable Distance Geometry Problem*, Optimization Letters **6**(8), 1671–1686, 2012.
29. A. Mucherino, C. Lavor, L. Liberti, N. Maculan (Eds.), *Distance Geometry: Theory, Methods and Applications*, 410 pages, Springer, 2013.
30. A. Mucherino, R. de Freitas, C. Lavor (Eds.), *Distance Geometry and Applications*, Discrete Applied Mathematics 197, 1–144, 2015.
31. 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.
32. A. Sit, Z. Wu, *Solving a Generalized Distance Geometry Problem for Protein Structure Determination*, Bulletin of Mathematical Biology **73**, 2809–2836, 2011.
33. R. P. Stanley, *Enumerative Combinatorics*, Vol. 2, Cambridge Studies in Advanced Mathematics 62, Cambridge University Press, p. 297, 1997.
34. Z. Voller, Z. Wu, *Distance Geometry Methods for Protein Structure Determination*, In: [29], 329–350, 2013.
35. Z. Wang, S. Zheng, Y. Ye, S. Boyd, *Further Relaxations of the Semidefinite Programming Approach to Sensor Network Localization*, SIAM Journal on Optimization **19**(2), 655–673, 2008.
36. Y-C. Wu, Q. Chaudhari, E. Serpedin, *Clock Synchronization of Wireless Sensor Networks*, IEEE Signal Processing Magazine **28**(1), 124–138, 2011.