On the Discretization of Distance Geometry: Theory, Algorithms and Applications

Antonio Mucherino

Mémoire d'Habilitation à Diriger des Recherches (HDR)

On the Discretization of Distance Geometry: Theory, Algorithms and Applications

Candidate: Antonio Mucherino
IRISA, University of Rennes 1

July 17^{th}, 2018

Rapporteur
Zhijun Wu
Iowa State University, USA

Examinateur
Jung-Hsin Lin
Academia Sinica, Taiwan

Rapporteur et Examinateur
Michel Petitjean
Université Paris 7, France

Rapporteur et Examinateur
Rosa Figueiredo
Université d'Avignon, France

Examinateur
Franck Multon
Université de Rennes 2, France

Président du jury
Kadi Bouatouch (Emeritus)
Université de Rennes 1, France

To Mila, Anna and Jakob

Acknowledgments

First of all I have to thank Leo Liberti, not only for having offered me the possibility to have a postdoc contract in 2008 at the École Polytechnique under his supervision, but most of all for having proposed me at that time to work on *distance geometry* with him. A special thank to Carlile Lavor (UNICAMP, Campinas-SP, Brazil) and Nelson Maculan (UFRJ, Rio de Janeiro, Brazil, now Emeritus professor) for all these years of joint collaboration, which have started during that initial period at the École Polytechnique. Since then, our collaboration network has increased with a certain number of students that we have co-supervised. Douglas S. Gonçalves is an important example: first PhD student of Carlile Lavor, then postdoctoral fellow under my supervision at IRISA, and now currently assistant professor in Brazil, and still very active in the distance geometry community.

I wish to thank Thérèse Malliavin and Michael Nilges (Institut Pasteur, Paris) for their fundamental help with the biological aspects concerning one of the main applications of distance geometry. Our very first discussions have started during my postdoctoral contract at the École Polytechnique, and our collaboration is nowadays still going on and becoming more and more concrete thanks to a some recently funded projects.

Many thanks to Franck Multon and to the other members of my hosting research team at IRISA: the MimeTIC team. All the recent results about dynamical distance geometry are due to the discussions I had during the last two years with people of the team (mainly with Franck Multon, Université de Rennes 2; Ludovic Hoyet, INRIA Rennes; Anne-Hélène Olivier, Université de Rennes 2), as well as with Julien Pettré (INRIA Rennes) and Paolo Robuffo Giordano (CNRS/IRISA, Rennes).

I am very grateful to the colleagues that reviewed this monograph: Zhijun Wu (Iowa State University, USA), Michel Petitjean (Université Paris 7) and Rosa Figueiredo (Université d'Avignon).

And many thanks to all members of the committee!

Preface

The adventure that is summarized in this HDR monograph began in Palaiseau (France) in 2008, when I started my postdoctoral fellowship at the École Polytechnique, in Leo Liberti's team at LIX. I remember I read the first paper about distance geometry, co-authored by Leo Liberti, Carlile Lavor and Nelson Maculan [95], during the attendance of my first French workshop. The beginning of my postdoc in Palaiseau was in fact also the beginning of my efforts to learn the French language. At that time, hearing spoken French was for me like listening to a fluid and nonrhythmic sound, totally incomprehensible. Probably this was the main reason why I could enter deeply in my reading during the presentations given in French (fortunately for me, some of them were given in English).

I remember my reading was fluid, and some simple ideas for future works began to shape into my mind. The whole thing sounded quite simple actually: I could have not imagined that, after about 10 years of research in the field, many results (and much more complex than those initial insights) were going to be found until today.

Differently from the traditional Euclidean Geometry, Distance Geometry (DG) is solely based on the concept of distance. The main problem arising in this field is the one of constructing a geometrical figure by using a set of *sides* having known length, whose vertices are labeled and need to match with the other side's vertices when in contact in the figure. This problem appears in the scientific literature under different guises, and it is generally referred to as the DG Problem (DGP) [96].

In recent years, DG has been attracting more and more interest from the scientific community. There is in fact an increasing number of books and journal special issues on this topic that are appearing in the scientific literature. Recent examples are [84, 94, 118, 128], whereas other collections are currently still under production (as for example [122]).

Distances are part of our everyday life, and this can be a motivation of this interest. A newborn is able to recognize, a few days after birth, whether the mother is approaching or leaving, and to adapt the reactions accordingly. Because of this importance, humankind has always tried to do the best to come to measure distances with the necessary precision. One nice historical example is Leornardo's Odometer,

which consists in a mobile machine equipped with a wheel connected to a gear designed for letting regularly fall stones in a container: the number of stones in the container at the end of the trip gives therefore an approximation of the distance traveled by the machine.

Trigonometric rules were then used to obtain the first approximations of the distance between earth and the sun (see for example the monograph [36] dated 1865). In another ancient book, dated 1945, Colonel Mathieu opened his textbook on Topography for military schools [107] with a list of methods for measuring distances that do not need special devices. An example is the horse step (by paying attention to adapt the step length to *walking*, *trotting* and *galloping*); another is the comparison of apparent dimensions, where the main principle is that the apparent size of two objects having actually the same size is inversely proportional to the distances between the objects and the observer. The list can continue with other methods that are based, for example, on the laws of propagation of sounds.

In our contemporary era, the measure of distances has become very precise, and many applications in DG are fed with data coming from specialized techniques. An example is given by machines performing Nuclear Magnetic Resonance (NMR), which are able to *look* inside the chemical structure of molecules, and to measure some distances between the atoms that form the molecule. Another typical example is given by sensor networks. Sensors are capable to exchange information with some other sensors belonging to a common network, and the battery power for a two-way communication provides an estimation of the relative distances between pairs of sensors. More modern sensors are not only able to estimate the distance between themselves, but also the distance to a near solid object (as for example the sensors installed behind our more recent cars).

This monograph is written in a didactic style, and it covers the main lines of the research performed during the last 10 years. The scientific results where I have mostly been involved, and that mostly fascinated me, are collected in this monograph. The reader interested in more details is pointed to some of my previous publications. To avoid losing the focus on the research work, I decided not to include in the monograph details about event organizations, student supervisions, reviewing work, etc. Such an information can be found in my curriculum vitæ, which is available online[1] and is regularly updated.

A large part of my HDR work consisted in reading my previous publications, and to summarize the main results at the light of the new developments in the field. Part of the text is essentially original. The introduction to the DGP in Section 1.1 is new, as well as the part extending the discretizability theory to DGP instances related to Def. 1.1.3. Prop. 2.3.2 was previously published in a journal paper [124], but the proof was limited to the three-dimensional case, and only for sets of exact distances. Even if I am the author of this monograph, and therefore I take the full responsibility for its content, most of the results that are here presented were obtained while working in collaboration with other colleagues. For this reason, in

[1] `http://www.antoniomucherino.it/en/curriculum.php`

my entire discussion in this monograph, I will employ the plural form.

The monograph is composed of four chapters. Chapter 1 will give an overview on DG, present some variants of its formal definition, review some solution methods and discuss some of its classical applications. Chapter 2 will be devoted to a discretization procedure for the DGP that allows to reduce the DGP search domain to a discrete set having the structure of a tree. The algorithm we developed for this particular class of DGPs, the Branch-and-Prune (BP) algorithm, will be described in many details, from how to compute position coordinates, to the different ways to verify the feasibility of such positions during the search. A study on the symmetries of the search tree will also be presented in the same chapter.

Chapter 3 will focus on the so-called *discretization orders*. DGP instances belonging to the discretizable class are such that the objects (atoms, sensors, etc) they are formed with admit a special order that basically guarantees that every object has a sufficient number of predecessors to be used as a reference for computing a discrete set of positions. A detailed review of methods for finding different kinds of discretization orders will be given, for the general case, as well as for some particular applications.

At the end of every chapter, the last section will be devoted to an open problem that is mostly related to the content of that chapter. Finally, Chapter 4 will completely be devoted to a project on novel DG applications, with a particular emphasis on dynamical applications.

The present monograph was written during different separated working sessions, and in different places. The very first draft, with the preliminary project for the table of contents, was written in Rennes, during the summer 2017, while my kids were visiting the grandparents in Italy. After this initial preparatory work, where I could be focused on the task for a few consecutive days, the rest of the work was performed between two teaching days, or among the various duties of a researcher life. In a few occasions (too few!), I was able to work in some quite inspiring places. For example, Section 1.1, the one that introduces the DGP and gives its different variants, was written in Grünheide (Germany), in a beautiful place where I could stare at an endless nature developing from the window in front of my working table. Similar situation, but with a different kind of view, was in Florianópolis (Santa Catarina, Brazil), but with chair and table on my room balcony. The palm trees of the hotel[2] helped me finalize Chapter 4 and the parts related to graph rigidity.

Another nice working place I had found for my HDR work was in Prague, in a bookstore[3] with cafeteria located a few walking minutes from the historical center. My coffee table had a very inspiring view over the bookshelves. However, on that day, I did the mistake to check my emails before starting with the work: all inspiration disappeared and I was finally caught by a certain number of emails that required a quick answer. I paid attention not to repeat the same mistake when I was in Taipei (Taiwan). It was the Dragon Boat holiday, and I was back from a trip to the river to

[2]Hotel Quinta da Bica D'Agua, rua Cap. Romualdo de Barros 641, Carvoeira (Florianpolis-SC), 88040-600, Brazil.

[3]globe bookstore & café, Pštrossova 6, 110 00 Prague 1, Czech Republic.

watch some boat competitions with some colleagues; I had never experienced such a humid and hot weather before (not for so long, maybe). Back to my hotel room, the first need was to have shower, and then air-conditioning, laptop, and final reading of the entire text.

After all, a lot of merit for this monograph to exist goes to my family, which accepted to be left aside during the several necessary moments of isolation, either in Rennes, where we live since 2012, or during my professional travels.

This monograph is dedicated to them.

July 17^{th}, 2018
Rennes, France *Antonio Mucherino*

Contents

Chapter 1

Distance Geometry

1.1 Formal Definitions

Several scientific papers on Distance Geometry (DG) have been appearing in the scientific literature over the last years. The authors of such papers belong to different scientific communities (see Section 1.2 for a quick survey). Therefore, from paper to paper, the given definitions can differ, and they are also formalized in different ways. What follows below is an attempt to introduce some general definitions, and to characterize some particular DG cases that have been considered in these scientific publications, including some very recent ones.

Poorly speaking, the Distance Geometry Problem (DGP) consists in verifying whether the positions for the objects in a given set can be obtained by exploiting a given inter-object distance list $\mathcal{D}$. Let $\mathbf{d}$ be a distance in the list $\mathcal{D}$. Let us suppose that the number of involved objects is n, where each object can be identified with a numerical label ranging from 1 to n. Let V be the set containing such labels.

The generic element $\mathbf{d}$ of the list $\mathcal{D}$ is a closed subset (an interval) of $\mathbb{R}_+$, indicating the real nonnegative values that the considered distance can take. Alternatively, these distances can be represented by pairs of lower and upper bounds on the values that they can have. The corresponding real interval can be degenerate for representing distances to which only one numerical value is associated. Such a unique real value can either be an exact value, in the sense that it can fully be trusted, or rather an approximation. It follows immediately that

$$|\mathcal{D}| \leq \frac{n(n-1)}{2}, \tag{1.1}$$

where $|\cdot|$ denotes the cardinality of a set.

Definition 1.1.1 *Given a set of objects V, a distance list $\mathcal{D}$ and a positive integer K, the (unassigned) DGP asks whether a function*

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

exists such that

$$\forall \mathbf{d} \in \mathcal{D} \quad \exists u, v \in V \quad (\text{with } u \neq v) \quad : \quad ||x_u - x_v|| \in \mathbf{d}, \tag{1.2}$$

where $||\cdot||$ *is the Euclidean norm.*

By definition, the DGP is a decision problem: it asks whether the function x exists so that the constraints (1.2) are satisfied. The function x is also named *realization* of the distance list $\mathcal{D}$ in dimension $K > 0$. The existence of a realization x does not deny the existence of (many) others. The function x associates a position x_v to every object of V in the Euclidean space having dimension K, so that the value $||x_u - x_v||$ is the actual value that the distance takes in the realization x. It is important to remark that the DGP can also be defined for non-Euclidean spaces, but we prefer here this more restricted definition because coherent with the applications considered in this monograph. For a wide discussion on non-Euclidean distances, the reader is referred to [38].

NP-completeness of the DGP was proved for one special case (see Def. 1.1.2) in dimension $K = 1$, and NP-hardness was proved for $K > 1$ [142]. By inclusion, we can state that the DGP is in general NP-hard. In simple words, no algorithm having polynomial complexity will ever be discovered for solving general instances of the DGP.

This complexity result is intuitively suggested by the exponential number of *combinations* of the distances in $\mathcal{D}$ that can be considered for constructing a realization that is solution to the DGP. Every distance $\mathbf{d}$ between u and $v \in V$ defines a spherical shell in $\mathbb{R}^K$, where the object u is fixed in its center, and the possible positions for v are constrained to lie between the surface of the sphere having the distance lower bound as its radius, and the sphere having the distance upper bound as its radius. Intuitively, the exponential complexity of the problem does not only come from the number of ways to combine the spheres in order to have non-empty intersections when the same object is concerned, but also from the (generally continuous) set of possible ways for the spheres to slide one into another (while keeping a non-empty intersection).

The set of constraints in Def. 1.1.1 implicitly defines an assignment function

$$\kappa : \mathbf{d} \in \mathcal{D} \longrightarrow \{u, v\} \in V \times V$$

that assigns a pair of objects u and v of V to every distance of $\mathcal{D}$. We suppose that this function is injective, so that each distance $\mathbf{d}$ is assigned to a unique pair. When the distance list $\mathcal{D}$ contains more than once the same distance, then the assignment function is evidently not unique.

In some applications, the assignment function κ is part of the input, so that the constraints in Def. 1.1.1 can be rewritten as:

$$\forall \{u, v\} \in \kappa(\mathcal{D}), \quad ||x_u - x_v|| \in \kappa^{-1}(u, v).$$

On the one hand, a previous knowledge on the assignment function allows to decrease the problem complexity (even if it can remain NP-hard). On the other hand, more

than one assignment function may be suitable to the instance at hand, and fixing one function or another may have important consequences on the solution of the instance. In fact, separating the definition of κ from finding a solution x may lead to assignment functions that may be incompatible with the instance: there may exist no function x that satisfies at the same time the assignment κ and the distances in the list $\mathcal{D}$. In this case, the DGP instance with pre-assigned κ is naturally infeasible. Moreover, the choice of one suitable function κ can prevent obtaining solutions to the problem which are compatible with a different assignment. Nevertheless, only a small part of the current scientific literature (see for example [16, 43]) is concerned with the DGP unassigned case, whereas the situation in which the function κ is a priori given is more generally taken into consideration.

When an assignment function is available, a DGP instance can be represented by a simple weighted undirected graph $G = (V, E, d)$. In fact, the edge set E can be simply derived as the image $\kappa(\mathcal{D})$ of $\mathcal{D}$ through the function κ, and it indicates whether the distance information is available for a given pairs of vertices. In the following, we will use the function δ to indicate the inverse of the assignment function κ. The function δ assigns, to every edge in E, a distance $\mathbf{d}$ of the distance list $\mathcal{D}$. The role of the weight function d of the graph G can therefore be played by δ:

$$d : \{u, v\} \in E \longrightarrow \delta(u, v) \in \mathcal{D}.$$

We can give the following alternative definition of DGP.

Definition 1.1.2 *Given a simple weighted undirected graph $G = (V, E, d)$ and a positive integer K, the (assigned) DGP asks whether a function*

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

exists such that

$$\forall \{u, v\} \in E, \quad ||x_u - x_v|| \in \delta(u, v).$$

In this context, the realization x is also called an *embedding* of the graph G. We say that a realization x that satisfies all constraints in Def. 1.1.2 is a *valid* realization. This is the most common definition of DGP that is given in the scientific literature (see for example [96]).

The constraints in Def. 1.1.2 can naturally be specialized to particular cases, on the basis of special features the distances may have. When the distance $\delta(u, v)$ is exact, the symbol "$\in$" can be replaced by an equality. When the distance $\delta(u, v)$ is instead represented by an interval $[\underline{\delta}(u, v), \bar{\delta}(u, v)]$, then the generic constraint can be replaced by two inequalities:

$$\underline{\delta}(u, v) \leq ||x_u - x_v|| \leq \bar{\delta}(u, v).$$

A more difficult case is the one where only one real value is given for the distance, but it represents an estimation, rather than an exact value that can be fully trusted. In this situation, it is likely that a DGP instance consisting of constraints where "$\in$"

was substituted with "=" admits no solutions. Since the magnitude of the errors that the available approximated distances may bring in the realization is not a priori known, replacing such distances with ad-hoc intervals is not an advisable procedure in general.

For this last situation described above, therefore, we propose an alternative definition of the DGP. Let us suppose that the weight function d associates two elements to every edge $\{u, v\} \in E$: the distance $\delta(u, v)$ between the two vertices, as well as a nonnegative real value $\pi(u, v)$ indicating the *importance* of such a distance. We also refer to $\pi(u, v)$ as the "priority" of $\delta(u, v)$. In the following, we will suppose that the values for the priorities $\pi(u, v)$ are normalized between 0 and 1, and that priority 0 is the less important.

Let X be the $K \times |V|$ matrix containing the positions of every x_v, column by column.

Definition 1.1.3 *Given a simple weighted undirected graph $G = (V, E, d)$, with*

$$d : \{u, v\} \in E \longrightarrow (\delta, \pi) \in \mathcal{D} \times [0, 1],$$

and a positive integer K, the (priority-based) DGP is the problem of finding a function

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

that minimizes the penalty function

$$\sigma(X) = \sum_{\{u,v\} \in E} \pi(u, v) \left(||x_u - x_v|| - \delta(u, v) \right)^2 . \tag{1.3}$$

Notice that the function in equ. (1.3) is only one of the possible examples of penalty function that can be associated to a DGP. An interesting property of $\sigma(X)$ is that it is differentiable at X when $\|x_u - x_v\| > 0$ for all $\{u, v\} \in E$ such that $\pi(u, v)\delta(u, v) > 0$ [35]. The gradient of $\sigma(X)$ can be written as follows:

$$\nabla\sigma(X) = 2 \left(WX - B(X)X \right),$$

where the generic element of the matrix W (notice the use of a more compact notation for $\delta(u, v)$, $\pi(u, v)$, etc) is:

$$w_{u,v} = \begin{cases} -\pi_{u,v}, & \text{if } u \neq v, \\ \displaystyle\sum_{w \neq u} \pi_{u,w} & \text{otherwise.} \end{cases}$$

Moreover, the generic element of the matrix $B(X)$ is

$$b_{u,v}(X) = \begin{cases} -\dfrac{\pi_{u,v}\delta_{u,v}}{\|x_u - x_v\|}, & \text{if } u \neq v \text{ and } \|x_u - x_v\| > 0, \\ 0, & \text{if } u \neq v \text{ and } \|x_u - x_v\| = 0, \\ \displaystyle -\sum_{w \neq u} b_{u,w}(X), & \text{otherwise.} \end{cases}$$

When all priorities $\pi_{u,v}$ are set to 1 (all distances have the same importance) and all distances are exact (they are not estimations), then the problem defined in Def. 1.1.3 is equivalent to the one in Def. 1.1.2. In fact, in these hypotheses, the decision problem in Def. 1.1.2 can be reformulated as an optimization problem where a penalty function such as $\sigma(x)$, with all its priority levels $\pi_{u,v}$ set to 1, needs to be minimized.

1.2 A Quick Review

This section collects some methods and algorithms for the DGP. It does not provide an exhaustive list. The methods can refer to any of the definitions for the DGP given in Section 1.1. Most methods included in the following list were described in the chapters of a collection on the DGP [128] dated 2013.

1.2.1 Principal Component Analysis

Principal Component Analysis (PCA) [49, 76, 78] is a classical method for linear dimensionality reduction, that was proposed by Karl Pearson in 1901 [138]. The basic idea is to represent an original dataset X containing n-dimensional samples by using another dataset Y where the dimension of the samples is $m < n$. To perform this task, PCA applies a linear transformation to the sample components, that is subsequently used to define Y. The coefficients of the linear transformation are obtained by minimizing the variance in the data samples: by doing so, the variance of some sample components can drop to 0, or be very near 0, and they can therefore be neglected when defining Y. The remaining components are therefore named "principal components". There exists a closed-form solution for the optimization problem related to the minimization of the variance, which turns out to be equivalent to finding the eigenvalues of the variance matrix. It is important to remark that the matrix related to the variance of the data points X can be represented in terms of Gram matrix.

The original application of PCA is dimensionality reduction. It can also be employed as a linear solution method for a subclass of DGPs (see Def. 1.1.2). Supposing that the number of distances in $\mathcal{D}$ is such that the corresponding distance matrix is dense (in other words, equ. (1.1) is satisfied with "="), then it is possible to construct the corresponding Gram matrix. As a consequence, the same PCA method briefly described above can be applied to a dense distance matrix, and not only to an initial set of points. Given a target dimension $K > 0$, PCA can select the first K principal components to construct, by linear projection, a realization Y best satisfying the available distances.

1.2.2 Nonlinear Programming solution methods

This class of methods for the DGP has emerged in the last decades, and is based on Nonlinear Programming (NLP) solution techniques. As shown in Section 1.1,

the DGP has a natural formulation as a decision problem (see Defs. 1.1.1 and 1.1.2), except when the error carried by approximated distances cannot be a priori estimated (see Def. 1.1.3). In all situations, however, the DGP can in fact be (re)formulated as a global unconstrained optimization problem where a penalty function is used for measuring the violation of the distance constraints. Different kinds of penalty functions were defined for the DGP: one example is the function σ in Def. 1.1.3, where the priority levels associated to the distances are also taken into consideration.

The same penalty function, but without the coefficients representing the priority levels, was used for example in [100]. This penalty function is differentiable on realizations x where there are no pairs of vertices that are placed in the same position. In the hypothesis all available distances are strictly positive, it was proved that the local minima of the penalty function are realizations where every x_u differs from x_v, when $u \neq v$ [34]. This makes it possible to use NLP techniques based on a sequence of local optimizations to find realizations for the DGP. Another well-known example in this context is the DGSOL algorithm [111, 112], which employs a homotopy method based on locally solving progressively finer Gaussian smoothings of the original problem.

1.2.3 Euclidean Matrix Completion problem

The EMBED algorithm [30] is an important method of the class of methods based on Euclidean Distance Matrix Completion [6, 39]. The graph G in Def. 1.1.2 is here replaced by a distance matrix containing the available distance bounds (distances are represented by suitable intervals) and having some missing entries. An order is associated to the vertices of G for obtaining the matrix representation, but the chosen order is not important for the solution method. The EMBED algorithm aims to fill in the missing distance bounds of the distance matrix by constraint propagation of triangle and tetrangle inequalities. Thereafter, a candidate distance matrix (named dissimilarity matrix) is sampled from the obtained interval distance matrix, and vertex coordinates are obtained by matrix decomposition [31, 145]. Since the dissimilarity matrix is not guaranteed to be a Euclidean Distance Matrix (EDM[1]), some of the original constraints might be violated. The last phase of the algorithm consists therefore in minimizing the constraint violation by local minimization, using the obtained preliminary solution as an initial point.

1.2.4 Semidefinite Programming solution methods

This class of methods is based on solving a Semidefinite Programming (SDP) relaxation of the DGP [19, 79, 106]. SDP can be used in the context of global optimization for finding approximated solutions to convex relaxations of the original problem. This approach was proved to provide quite good results when working with the DGP formulations; it was noticed, however, that the quality of such results tend to de-

[1] An EDM is a symmetric matrix containing a full set of distances defining a DGP, such as the one in Def. 1.1.2, which admits solutions when the symbol "$\in$" is substituted with "$=$".

crease when the distance information is very scarce [45]. For this reason, more recent works on SDP solutions methods for the DGP exploit additional information from the particular application at hand (see Section 1.3) in an attempt to improve the obtained results. For example, the work in [7, 79] exploits the cliques in the graph G to reduce the size of the SDP formulation. This technique has some common points with the one based on the discretization process detailed in Chapter 2.

1.2.5 The Build-Up algorithm

This class of methods is centered around the so-called Build-Up algorithm [40, 41, 156]. These methods are based on the ancient idea of triangulation, used by humankind ever since navigation existed. In the DG context, where a vertex position is determined by the distances to it rather than the angles they subtend, this is known as "trilateration". Build-Up algorithms in dimension $K = 3$ attempt to place an unknown vertex v by identifying at least four other vertices with known positions, and having known distances to v. It is important to remark, however, that the requirement of having four reference positions for every vertex to be placed can be too strong in several applications (one example is given by the classical application in structural biology, where the chemical structure of the molecules can guarantee the existence of 2 or 3 distances per atom, but very rarely more than 3 [105]). There exist however extensions of the Build-Up algorithm that are able to deal with missing reference positions [159], as well as with a certain level of uncertainty [148].

Even though, in the first versions of this algorithm, the assumptions for its actual use may have appeared to be too strong for the applications, those initial studies inspired further research lines in the context of the Build-Up algorithm, as well as the work presented in Chapter 2.

1.2.6 Stochastic Proximity Embedding

The Stochastic Proximity Embedding (SPE) is a heuristic for the DGP [2]. The idea behind this heuristic is very simple: given an initial starting realization x, at each step of the algorithm one selected distance $||x_u - x_v||$ is corrected if it does not correspond to $\delta_{u,v}$. This is different from the general approach in global optimization (see previous sections), where all positions x_v are modified at the same time during an iteration of the algorithm. During the correction procedure, SPE makes use of a parameter λ, which is called *learning rate*, as its use reminds of methods associated to neural networks [37]. Some successful experiences with this heuristic are reported in [3].

1.2.7 Methods for the unassigned DGP

As already remarked in Section 1.1, there are very few works in the scientific literature that deal with the *unassigned* DGP (see Def. 1.1.1). There are two main algorithms that have been proposed to solve this class of DGPs. The TRIBOND algorithm [63] works with instances consisting of a complete list of exact distances,

where the main step consists in finding a *core* formed by a $(K+2)$-clique, where K is the dimension of the Euclidean space. Subsequently, the rest of the realization is constructed from the initial core by identifying additional cliques in G. The LIGA algorithm [43] is an example of heuristic for the unassigned DGP, which is also able to work with instances containing approximated distances.

1.2.8 Simulated Annealing based methods

The solution to DGPs arising in structural biology is typically attempted by employing software tools such as ARIA [101], CYANA [64] and UNIO [62], which are all based on the Simulated Annealing (SA) meta-heuristic [77]. These tools are particularly designed for working with structural biology problems, and they offer a certain number of features, aside from solving the associated DGPs. However, even though universally used in the structural biology community, it is well-known that SA-based methods are not able to provide a certificate of optimality. For this reason, alternative methods, having more rigid mathematical basis, are under currently development [105], within the structural biology community, and in collaboration with researchers in other fields.

1.3 Applications

DG has several and important applications. Over the last years, we mainly focused our attention on two applications: the protein structure determination problem (see Section 1.3.1), and the dimensionality reduction problem (see Section 1.3.2). These two applications are discussed in details in Sections 1.3.1 and 1.3.2. There are, however, many other applications that can match with the definitions for the DGP given in Section 1.1.

Another typical application of the DGP is the Sensor Network Localization Problem (SNLP) [19, 79, 155]. A simple example of sensor network is given by a set of mobile phones that communicate with a certain number of antennas, where it is of interest to find the positions of the mobile phones. This example is however obsolete because the GPS technology is nowadays able to give rather good approximations of sensor's positions. But there are still situations where the GPS technology cannot be exploited: examples are robot networks working under water [143], and networks where the sensors are so near that the error given by the GPS would be greater than the actual distance values.

The SNLP generally deals with a set of static antennas (called *anchors*), and mobile sensors. The power for a two-way communication between two sensors, or between one sensor and one antenna, can be used for estimating the distance between a sensor pair. Finding the overall set of coordinates for all the sensors in the network by exploiting this distance information is a DGP.

Other applications of the DGP include: the clock synchronization problem [48, 154, 158], the study of nanostructures [16, 43], and graph drawing [12, 69] and

coloring [33]. In Chapter 4, a discussion on more recent applications of the DGP will be given.

1.3.1 Protein Structure Determination

The vision of biology has been fundamentally modified during the last 50 years by the discovery of a large amount of molecular agents (biomolecules) performing important biological processes [105]. Just to mention some examples, molecular motors are the essential agents of movement in living organisms, the transcription factors regulate the genetic expressions, the enzymes are able to catalyze chemical reactions, and the ion channels help establishing and controlling the voltage gradient across the cell membrane. Transport proteins perform the function of moving other materials inside an organism.

A second discovery of the last years is given by the identification of the molecular structure for many of such biomolecules. Indeed, the structure of a molecule is essential in order to understand its function. The slightest modifications in this structure can drastically change the corresponding biomolecular function, as it is encountered, for example, for neurodegenerative diseases [74].

Proteins are important molecules that perform vital functions in the bodies of living beings. They are chains of smaller molecules named *amino acids*, whose order is a priori known (in other words, every amino acid is known with its rank/position in the chain). The *protein backbone* is defined by this chain, and basically contains, in sequence for each amino acid, a nitrogen N, a carbon named C_α and another carbon C, plus some additional atoms chemically bonded to them (two hydrogens and one oxygen are always present). Only 20 different amino acids can be involved in the protein synthesis. A group of atoms attached to the carbon C_α makes the 20 amino acids different from each other. Since this group of atoms looks like "hanging" from the protein backbone, it is said that it is the *side chain* of the amino acid. Due to the complexity of the problem of identifying protein conformations, many proposed methods focus on protein backbones, and the information about the side chains is either approximated or neglected. Once a suitable conformation for a protein backbone has been identified, there are methods that attempt the positioning of the side chains (see for example [21]).

Proteins display a hierarchical level of organization. Their *primary structures* consist of the sequence of amino acids composing the molecule. The *secondary structures* of proteins are relatively local arrangements of amino acids: in α-helices, the backbone is arranged as a helix, whereas in β-sheets, it forms a set of strands over a common plane. The *tertiary structure* is the global arrangement of the amino acids of monomeric proteins, for which there is a unique sequence of amino acids. For more complex proteins, the global three-dimensional structure is given by their *quaternary structure*, comprising the tertiary structures of the various chains of amino acids that compose the molecule. Fig. 1.1 shows two graphical representations of the human hemoglobin. On the left-hand side, an atomic representation of the protein, where carbons are in grey, nitrogens are in blue, and oxygens are in red (hydrogen atoms are

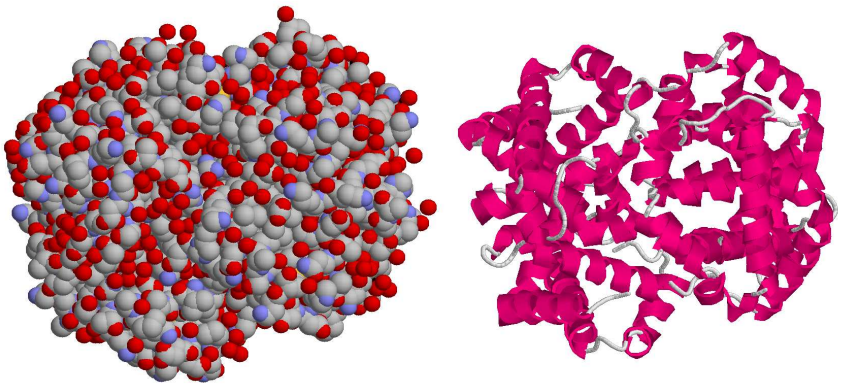


Figure 1.1: *Two graphical representations of the hemoglobin, the protein involved in the transfer of oxygen in the human body. PDB code:* `1a3n`*; figures generated by the freeware* `Rasmol`*.*

omitted). On the right-hand side, the typical representation which gives emphasis to the secondary structures of the molecule (in this case, only α-helices are present in the protein).

Because of the acknowledged importance of molecules in biology, and of the essential role of their molecular conformations, the scientific field called *structural biology* has experienced an enormous development in recent years. This research field originated from the application of powerful physical techniques to biological objects. The widespread application of structural biology methods has produced an astonishing molecular description of life. The study of protein conformations also helps to understand how they interact to each other and with other kinds of molecules. These studies can have an impact in the design of novel drugs [71, 150].

Due to this great interest, structures of biological molecules are nowadays deposited in public web databases. One of the most important databases is named Protein Data Bank (PDB[2]) [14], which reached the number of 140000 deposited molecules in 2018. About 10% of the molecules deposited on the PDB have been obtained by performing experiments of Nuclear Magnetic Resonance (NMR).

The NMR is one of the first and well-established techniques which are still nowadays used for studying biomolecular structures [8]. NMR is able to give very sensitive information on distances or angles between atoms, as well as to provide information on the internal dynamics of the molecule. The main output of an NMR experiment is a list of lower and upper bounds for some distances between pairs of hydrogen atoms. With some other distances that can be derived from the chemical composition of the molecule (its primary structure), the problem of determining its three-dimensional conformation can be formulated as a DGP where some distances are exact (e.g. chemical bonds) and others are represented by intervals (e.g. NMR distances). Let V be

[2] `http://www.rcsb.org`

the set of atoms forming the protein, and let E be the set of pairs in $V \times V$ such that it is possible, either by studying the chemical structure or by NMR, to obtain the distance between the two atoms in the pair. The corresponding graph G represents a DGP instance as in Def. 1.1.2.

We remark that there exist other experimental techniques that can give measurements of atomic distances. An example is given by fluorescence techniques (FRET, cellular imaging), or some hybrid methods, such as mass spectrometry coupled with cross-linking. It is important to remark that all these experimental techniques may provide a small percentage of wrong measurements. We will discuss this particular issue in Section 1.4 in the context of NMR experiments.

1.3.2 Dimensionality Reduction

Since one of the first pioneer papers on this topic [153] in 1952, dimensionality reduction received an increasing interest. Various surveys on this topic can be found in the scientific literature: a recent example is [73], published in 2013, where dimensionality reduction is defined as "the set of statistical techniques that are employed for reducing the dimensionality of a given set of data, with the aim of improving the visual appreciation of the underlying relational structures contained therein". The main idea is to attempt mapping the data into a Euclidean space, where *similar* items are represented by near points in the mapping, and *dissimilar* items are represented by points that are located proportionally further apart. By doing so, the complexity in the data is reduced, and the primary dimensions, along with the items differ, may be identified. Dimensionality reduction has a wide range of applications, in various scientific domains.

One of the best known methods for dimensionality reduction, especially in the field of mathematical statistics, is PCA (see Section 1.2.1). PCA basically consists of an orthogonal transformation that can project a high-dimensional data set into a new set of coordinate systems (principal axes) in the order of the variances. One of the most notable advantages of PCA is the preservation of the distance metric during the linear transformation. However, the essential feature of PCA, i.e. the use of linear transformation, also prevents PCA from discovering intrinsic non-linear degrees of freedom underlying complex natural phenomena.

One of the most prominent approaches in non-linear dimensionality reduction is the Isomap method proposed in [151]. Differently from PCA, the Isomap method has a better ability for successfully capturing the intrinsic global geometric features of the given set of points. It was noted, however, that the application of the Isomap method is able to provide good results only when transferring the data between two similar geometries [42].

In many applications of dimensionality reduction, an initial set X containing points represented in an initial Euclidean space is given. From the coordinates for the points in X, it is possible to compute all relative distances. With this information, we can define a simple weighted undirected graph $G = (V, E, d)$, where every vertex $v \in V$ represents a point of X, and an edge $\{u, v\} \in E$ represents the relative distance

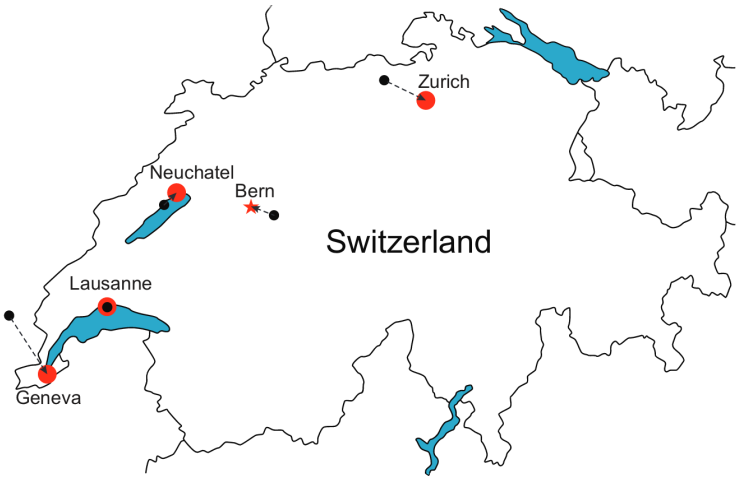


Figure 1.2: *A map of Switzerland obtained by dimensionality reduction. Image taken from [39].*

between the two corresponding points in X. All graphs G constructed in this way are complete, and all the distances associated to their edges are exact. These graphs represent instances of the DGP defined in Def. 1.1.3. The main interest is to reduce the data dimensionality, and to converge to small dimensions, such as 3, 2, or even 1, where the visualization of the data is possible.

A very simple yet nice example of dimensionality reduction is given in [73] and is concerned with the problem of drawing a small geographic map. All relative distances between the cities of Los Angeles, New York, Chicago and Dallas are given, and the aim is to find their correct locations on a two-dimensional map. When the information on the distances is exact, a very accurate map can be generated, i.e. a map for which the distances between points are proportional to the true distances between city pairs (modulo total translations, rotations and reflections). In general, however, solutions where the overall distance information is precisely verified may not exist, so that approximated realizations need to be searched.

Let us consider now an example given by M. Vetterli in one of his lectures (see Fig. 1.2). Suppose that the distances between city pairs is now represented by the time necessary to a train to travel from one city to another. Naturally, these distances may not satisfy the properties of a metric, because, for example, traveling from city A to city B may take more (or less) time than traveling from B to A. As Fig. 1.2 shows, this example takes place in Switzerland, and it is a quite easy example, because a good approximation of the city map can be obtained by applying PCA [39].

The Swissroll (see Fig. 1.3) is instead an example of dimensionality reduction problem that cannot be properly solved by PCA. In fact, if employed for reducing the dimensionality to 2, PCA would apply a linear transformation that would project all points on a plane crossing the Swissroll. The result obtained in this way would not contain any information about the original geometry of the Swissroll. The Isomap

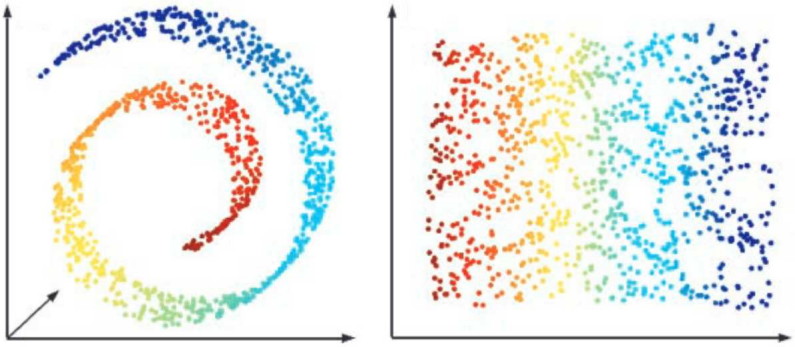


Figure 1.3: *The Swissroll example used for comparing the classical PCA and the Isomap method. Image taken from [151].*

method is instead better able to deal with this kind of problems, because it focuses on local geometry and gives low (or not at all) importance to distances that are far in the original geometry. The introduction of the concept of importance for a distance is coherent with Def. 1.1.3. The Isomap method is able to properly *unroll* the Swissroll.

It is important to remark that, even if dimensionality reduction and the DGP have been studied by separated communities, the two problems share many common points. In general, the DGP is more general and dimensionality reduction can be seen as a particular application. However, there are also studies where tools for dimensionality reduction have been tailored to the DGP [93]. The work on the discretization of the DGP (see Chapters 2) was also applied to dimensionality reduction [5, 61].

1.4 An Open Problem: the Unassigned DGP

As already discussed in Section 1.1, the most general definition of DGP is the one given in Def. 1.1.1, but most methods and algorithms developed for the DGP suppose that an assignment function $\kappa : \mathcal{D} \longrightarrow V \times V$ already exists. In Section 1.2.7, we have briefly reviewed two algorithms for the unassigned DGP.

In this section, we focus our attention on the assignment problem related to the application described in Section 1.3.1. From the chemical composition of a protein, and with the information derived from NMR experiments, it is possible to define instances of the DGP, whose solutions can provide the possible three-dimensional conformations for the protein that are compatible with the available distance information.

Constructing a graph G containing the overall distance information requires the definition, a priori, of the assignment function κ. The raw data obtained by NMR experiments basically consist in the so-called NMR spectra, that contains cross-

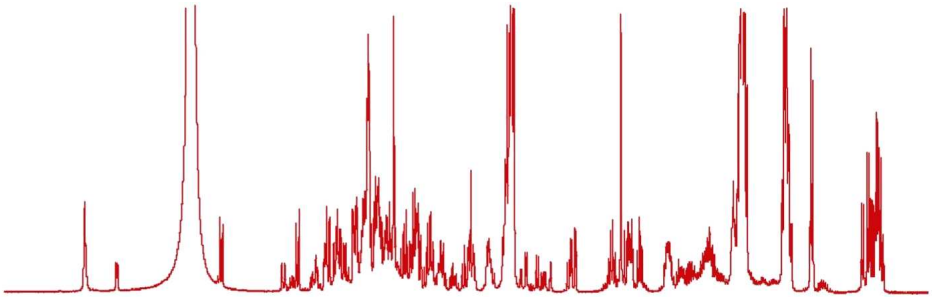


Figure 1.4: *An example of NMR spectra. Image taken from [92].*

peaks, which represent either covalent-bond linkage, or spatial relationships between non-bonded atoms [8]. Fig. 1.4 shows an NMR spectra, taken from [92]. The main problem is how to assign each cross-peak to a pair of atoms, without introducing errors that could spoil the final results. Typical sources of errors are false positive and negative cross-peaks: every cross-peak is generally interpreted in accordance with an intensity threshold, so that false positives can be generated when using too small threshold values, and genuine cross-peaks may be neglected when this threshold is instead too large. Finding the best threshold trade-off is naturally not an easy task. For a more detailed procedure for this assignment problem, the reader is referred to [113].

In future works, we plan to consider the problem of identifying the assignment function κ and the three-dimensional conformation x of the protein in one general method. This method would in fact solve a DGP instance as in Def. 1.1.1. The advantage in doing so is that possible errors in the assignments may be discovered on-the-fly by the part of the algorithm that is in charge of constructing the realization x. Valid starting points for our research in this direction are the algorithms in Section 1.2.7 for the unassigned DGP, together with other methods for performing an efficient assignment of the NMR spectra (see for example [157]).

Chapter 2

The Discretizable Distance Geometry Problem

2.1 The Discretization Assumptions

Let $G = (V, E, d)$ be a simple weighted undirected graph representing an instance of the DGP in dimension $K > 0$. We will consider the DGP as it is defined in Def. 1.1.2 and Def. 1.1.3. Let E' be the subset of E containing either exact distances (Def. 1.1.2) or distances with priority 1 (Def. 1.1.3). Conversely, the subset $E \setminus E'$ contains all distances represented by intervals (Def. 1.1.2), or having a priority lower than 1 (Def. 1.1.3). The *discretization assumptions*, i.e. the assumptions that allow to perform the discretization of the search space of DGP instances, are given below [86, 96, 97]. We suppose that the vertex order on V is given through the numerical label associated to the vertex: in this chapter, we will use the same symbol v for both the vertex and its rank in the vertex order, to which operators such as "$<$" and "$+$" can be applied. Vertex orders will be formally introduced in Chapter 3.

Definition 2.1.1 *A simple weighted undirected graph G represents an instance of the Discretizable DGP (DDGP) 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 K vertices $u_1, u_2, \ldots, u_K \in V$ such that*

- **(b.1)** *$u_1 < v$, $u_2 < v$, $\ldots$, $u_K < v$;*
- **(b.2)** *$\{\{u_1, v\}, \{u_2, v\}, \ldots, \{u_{K-1}, v\}\} \subset E'$ and $\{u_K, v\} \in E$;*
- **(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 rest of the discussion, we will refer to assumptions **(a)** and **(b)** as the "discretization assumptions". Notice that such assumptions can be verified only if a vertex ordering is associated to V. The problem of finding suitable vertex orderings for a given DGP instance will be widely discussed in Chapter 3.

Assumption **(a)** ensures the existence of an initial clique in G: the first K vertices in the ordering belong to this clique, where all edges are related to distances in E'. A unique realization for this clique can be computed (modulo total translations, rotations and reflections, see Section 2.5). We exploit this initial realization for fixing the Cartesian coordinate system where the solutions of the DDGP are constructed.

Assumption **(b)** is the one that in fact allows us to reduce the search space for the DGP instance to a discrete domain having the structure of a tree, where the positions of vertices are organized layer by layer. Since the K *reference vertices* $u_1, u_2, \ldots, u_K$ for the current vertex v precede v in the vertex ordering, it is supposed that, when positions for v are searched, possible positions are already available for its reference vertices. By exploiting the corresponding *reference distances*, K Euclidean objects can be defined, whose intersection gives the set of possible positions for the vertex v. When all considered distances belong to E', then the K objects are K spheres centered in the reference vertex u_i and having radius $\delta(u_i, v)$, and their intersection gives a set of cardinality 2 [124, 139] (with probability 1). If one of the reference distances belongs instead to $E \setminus E'$ (recall that $\{u_k, v\} \in E$), one of the K objects may be a spherical shell, and the intersection of $K-1$ spheres with one spherical shell gives two disjoint arcs with probability 1 [87]. An approximation of the *thickness* of the spherical shell needs to be estimated when only one value for the corresponding distance is given (see Def. 1.1.3). Given the distance value $\delta_{u,v}$ and the corresponding priority $\pi_{u,v}$, one possible way to define an interval for this distance that reflects the information about its importance is:

$$[\underline{\delta}_{u,v}, \bar{\delta}_{u,v}] = [\pi_{u,v}\delta_{u,v}, \delta_{u,v} + (1 - \pi_{u,v})\delta_{u,v}] \,.$$

This formula reflects the fact that priority 1 distances are exact, but it assigns tighter intervals to smaller distances and may therefore not be suitable with set of distances having too variable orders of magnitude.

All results reported above about the intersection of K spheres, or $K-1$ spheres and 1 spherical shell, are valid only if assumption **(b.3)** is satisfied. If the subset $\{u_1, u_2, \ldots, u_K\}$ induces a clique of G, then a unique valid realization (modulo total translations, rotations and reflections) can be computed (as for the initial clique in assumption **(a)**), so that the volume $\mathcal{V}_S$ can be computed for verification by exploiting this realization. However, in general, the set $\{u_1, u_2, \ldots, u_K\}$ may not induce a clique, so that the volume $\mathcal{V}_S$ could not be computed a priori. From a theoretical point of view, this volume can be equal to zero with probability 0 [87]. For this reason, when checking the discretization assumptions of DDGP instances related to real-life applications, we generally neglect assumption **(b.3)**. Also notice that, for $K = 1$, the simplex reduces to a singleton.

Fig. 2.1 shows the results of the sphere intersections for the case $K = 3$. In order to find candidate positions for the vertex v, we can exploit the information

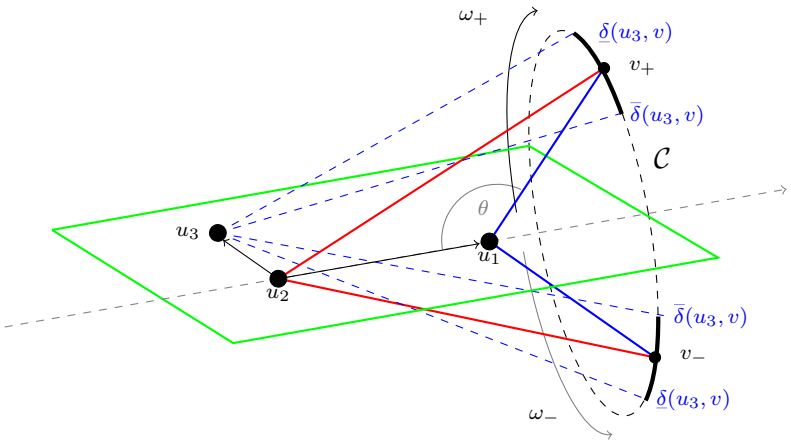


Figure 2.1: *Intersection of 2 spheres with 1 spherical shell in the Euclidean space of dimension* $K = 3$.

concerning 3 other vertices u_1, u_2 and u_3 that can play the role of reference vertex for v. It is supposed in fact that a possible position is already assigned to each of these three vertices, and that the three corresponding reference distances are available. The positions for the three reference vertices can be used as centers of spheres or spherical shells, having as a radius the corresponding reference distance. By Def. 2.1.1, the reference distances concerning u_1 and u_2 are related to edges of E', and the intersection of the two corresponding spheres gives (with probability 1) a circle $\mathcal{C}$, depicted in Fig. 2.1 in dashed lines. If also the distance related to $u_3 \in E'$, then the intersection of $\mathcal{C}$ with another sphere gives the two points v_- and v_+. If instead this third distance is represented by an interval, then the corresponding Euclidean object is a spherical shell, and its intersection with $\mathcal{C}$ provides two arcs, in general disjoint, over $\mathcal{C}$.

Let us initially suppose that our graph G is such that $E = E'$. Let v_{K+1} be the vertex of G having rank $K + 1$; similarly, let v_{K+2} be the vertex of G having rank $K + 2$ in the vertex ordering. By using as a reference the vertices of the initial clique, two possible positions for the vertex v_{K+1} can be computed. Thereafter, when we step at level $K + 2$, while some of the vertices from the initial clique can still be used as a reference, the vertex v_{K+1} can also be involved. In this case, two possible positions (and not only one) for one of the reference vertices (i.e. v_{K+1}) are available. As a consequence, not only 2 positions can be computed for v_{K+2}, but rather 4 positions (a new pair of positions for every possible position already computed for v_{K+1}).

When this procedure is iterated over all subsequent ranks, more and more vertex positions can be generated for the current vertex. The obtained search domain can formally be represented as a tree $\mathcal{T}$. The tree $\mathcal{T}$ is rooted at the node related to the first vertex of the initial clique. At every layer v, nodes contain vertex positions for v, and are connected through an edge to the nodes concerning its preceding vertex

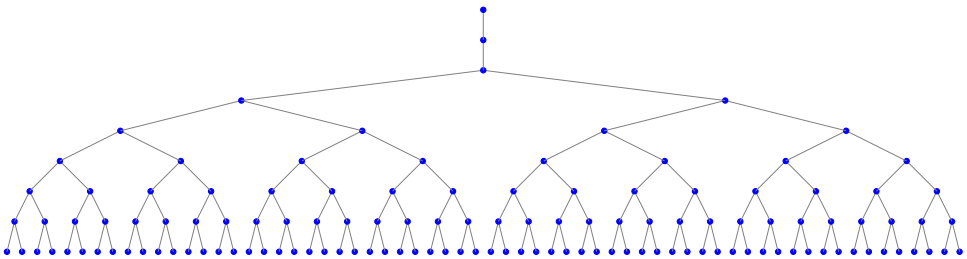


Figure 2.2: *A small binary tree related to a DDGP instance containing 9 vertices.*

$v - 1$. The tree has an initial branch starting from its root and going linearly over the other initial clique nodes (see Section 2.5); branching starts at level $K + 1$ (see Fig. 2.2).

A realization x can be seen as a path from the root of the tree $\mathcal{T}$ to one of its leaf nodes. It is important to remark, however, that only the distances that are used for generating the spheres to be intersected are satisfied by all such realizations. A path on $\mathcal{T}$ that satisfies the overall distance information can be selected by using as a criterion the satisfaction of the additional distances that may be available (see Section 2.7). In the following, we will refer to the distances used in the definition of the spheres as *discretization distances*, whereas we will refer to the other available distances as *pruning distances*. This second group of distances can in fact be exploited for selecting the paths on $\mathcal{T}$ corresponding to solutions where all the distances are satisfied. Naturally, it is not advisable to construct $\mathcal{T}$ and then to perform such a path selection: the reader is referred to Section 2.4 for an algorithm we developed for an efficient exploration of $\mathcal{T}$.

The structure of $\mathcal{T}$ also suggests alternative representations for DDGP solutions. At every layer of the tree, a particular position for the corresponding vertex is selected by choosing either the right-hand or the left-hand branch. This left/right information can be coded with a binary variable, and an entire path can be seen as a sequence of binary values [129]. As a consequence, a DDGP solution can also be represented by an integer number ranging from 0 to $2^{|V|-K}$.

Naturally, these different representations may become more complex when the DDGP instance also contains edges of $E \setminus E'$. In this case, when the distance $\delta(u_i, v)$, related to the reference vertex u_i, belongs to $E \setminus E'$, the intersection of the spheres with the spherical shell provides two arcs. Therefore, the two corresponding nodes on $\mathcal{T}$ are related to two continuous sets of positions. When verifying the pruning distances, it may be discovered that subsets of positions in these arcs are actually not feasible. Therefore, differently from the case where all distances are exact, paths on $\mathcal{T}$ are feasible if there exist subsets of positions, for each of its arcs, that are feasible.

One strategy for dealing with interval distances is to *discretize* the two arcs obtained with the intersections by selecting a predefined number D of sample positions

Figure 2.3: *An example of approximated tree. The nodes in red are originally associated to arcs; in the approximation, they are replaced by $D = 2$ nodes containing sample positions extracted from the arcs.*

[87]. In this case, a node of the tree $\mathcal{T}$, representing an arc, can be replaced by D nodes representing D selected sample positions (see Fig. 2.3). We say that D is the *discretization factor*. Replacing $\mathcal{T}$ with its approximation $\mathcal{T}'$ allows for performing a search over discrete sets of positions. However, some (if not all) solutions related to $\mathcal{T}$ may not be present in $\mathcal{T}'$. A discussion on the consequences of this fact is given in Section 2.11.

2.2 Graph Rigidity and Discretization

The concept of discretizability for a DGP instance is close to the concept of *rigidity* for a graph [68, 75, 82]. We give the following definition.

Definition 2.2.1 *Given a connected graph $S = (V_S, E_S)$, we say that the pair (S, χ) is a skeletal structure if*

$$\chi : V \longrightarrow \mathbb{R}^K$$

is a realization of the graph S such that $\chi(u) \neq \chi(v)$ for each $u, v \in V_S$ for which $u \neq v$.

Two skeletal structures (S, χ_1) and (S, χ_2) are said *isometric* if

$$\forall \{u, v\} \in E_S, \quad ||\chi_1(u) - \chi_1(v)|| = ||\chi_2(u) - \chi_2(v)||.$$

Moreover, we say that two skeletal structures (S, χ_1) and (S, χ_2) are *congruent* if

$$\forall \{u, v\} \in V_S \times V_S, \quad ||\chi_1(u) - \chi_1(v)|| = ||\chi_2(u) - \chi_2(v)||.$$

The two definitions above show that congruency implies isometry, because the set of constraints to be satisfied in order to verify the isometry case are also included in the set of constraints related to congruency (see equations above). These preliminary definitions allow us to define "graph rigidity" [75].

Definition 2.2.2 *Given a connected graph $S = (V_S, E_S)$, if every pair of isometric skeletal structures (S, χ_1) and (S, χ_2) are also congruent, then we say that the graph S is rigid.*

Poorly speaking, a graph G is rigid when, given a realization of the graph, it is not possible to apply continuous deformations to such a realization that are able to preserve the distances in E_S. It is easy to verify that graphs G representing some class of DDGP instances in a given dimension K are rigid [85]: the structure of the graph allows in fact for discretizing the set of valid realizations, the ones that are compatible with the distances associated to the edges. There exist, however, rigid graphs that do not correspond to DDGP instances. An example is Desargues graph [81], which is rigid but does not allow for the discretization for $K > 1$. Some authors studied the relationships between graph rigidity and the number of solutions of a DGP instance (see for example [68]).

2.3 The Consecutivity Assumption

The consecutivity assumption is an additional assumption for the discretization of DGP instances that can imply the definition of search trees $\mathcal{T}$ having some additional properties, that can be of particular interest.

Definition 2.3.1 *Given a simple weighted undirected graph G and a discretization order on V, we say that the ordering satisfies the "consecutivity assumption" if:*

$$\forall v \in \{K+1, \ldots, |V|\}, \quad u_1 = v - K, u_2 = v - K + 1, \ldots, u_K = v - 1.$$

Let

$$S(v, G) = \{\{u, v\} \in E \,:\, u \in V\}.$$

The following proposition characterizes the DDGP instances for which the consecutivity assumption is satisfied.

Proposition 2.3.2 *Let G be a simple weighted undirected graph representing a DDGP instance for which the discretization order satisfies the consecutivity assumption. Assumption* **(b)** *in Def. 2.1.1 is equivalent to:*

(b') $\forall v \in \{K+1, \ldots, |V|\}$, *we have*

- **(b'.2)** $G_v \equiv G[\{v-K, v-K+1, \ldots, v-2, v-1, v\}]$ *is a clique and* $|S(v, G_v) \cap E'| \geq K - 1$;
- **(b'.3)** $\mathcal{V}_S(v-K, v-K+1, \ldots, v-2, v-1) > 0$, *if* $K > 1$.

proof. In this proof, we consider the following set of vertices:

$$V_v = \{v-K, v-K+1, \ldots, v-2, v-1, v\}.$$

If the vertex is $v-1$, we have:

$$V_{v-1} = \{v-K-1, v-K, v-K+1, \ldots, v-3, v-2, v-1\}.$$

Let us begin by proving that assumption **(b)** implies assumption **(b')**. When the consecutivity assumption is satisfied, we can rewrite assumption **(b)** as follows:

$$\forall v \in \{K+1, \ldots, |V|\}, \quad \begin{cases} \{\{v-K, v\}, \{v-K+1, v\}, \ldots, \{v-2, v\}\} \subset E', \\ \{v-1, v\} \in E, \\ \mathcal{V}_S(v-K, v-K+1, \ldots, v-2, v-1) > 0. \end{cases} \tag{2.1}$$

The proof proceeds by induction: let us suppose that $G[V_{v-1}]$ is a clique, which immediately implies that edges are incident to all pairs of vertices of $V_v \setminus \{v\}$. Moreover, assumption **(b)** ensures that all edges connecting the vertices in $V_v \setminus \{v\}$ and the vertex v are available (see equ. (2.1)). It follows therefore that $G[V_v]$ is a clique. Since only $\{v-1, v\}$ can belong to $E \setminus E'$ by assumption **(b)**, the minimal cardinality for $S(v, G[V_v]) \cap E'$ is $K-1$. This proves by induction that assumption **(b)** implies assumption **(b')**. Notice that assumption **(b.3)** and assumption **(b'.3)** are equivalent.

Let us consider now the implication **(b')** $\Rightarrow$ **(b)**. By assumption **(b'.2)**, $G[V_v]$ is a clique, so that $S(v, G[V_v])$ has cardinality K. Assumption **(b'.2)** also implies that $S(v, G[V_v]) \cap E'$ has minimal cardinality $K-1$, so that the K^{th} edge of $S(v, G[V_v])$ can actually belong to $E \setminus E'$. This implies that assumption **(b)** is satisfied. □

This theoretical result was initially published in [124] but was limited to instances containing only exact distances, and for dimension $K = 3$. The proposed characterization of DDGP instances where the consecutivity assumption is satisfied proposes a different representation of the associated discretization orders. Not only the orders begin with an initial clique (see assumption **(a)**), but they can in fact be entirely represented as sequences of overlapping cliques. This gives an immediate advantage: all those cliques can be a priori verified, before attempting the construction of the tree $\mathcal{T}$. Naturally, even if not ordered in sequence in a discretization order satisfying the consecutivity assumption, those cliques exist in G and they can all be verified for feasibility. However, when the discretization order is a sequence of cliques, all discretization distances *belong* to such cliques, and their a priori verification ensures that the tree $\mathcal{T}$ can be fully constructed.

It was already proved, moreover, that DDGP instances with the consecutivity assumption define search trees $\mathcal{T}$ that are symmetric. Section 2.8 will present a discussion on how tree symmetries can be exploited for improving the exploration of $\mathcal{T}$. Then, Chapter 3 will present a wide discussion on the different kinds of discretization orders that can be defined for the DDGP, with and without consecutivity assumption, and will present some methods for their identification. The additional properties on $\mathcal{T}$ induced by the consecutivity assumption come at the cost of the NP-hardness for the problem of finding discretization orders with consecutivity assumption (this problem has polynomial complexity otherwise, see [83] and [24]).

Algorithm 1 The BP algorithm's main framework

```
1: BP(v, G, D)
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 the two candidate arcs and add them to the list L
9:    else
10:      compute the two candidate positions and add them to the list L
11:   end if
12:   for i = 1, ..., |L| do
13:      if (L(i) is an arc) then
14:         take D samples from the arc; set N = D;
15:      else
16:         set N = 1;
17:      end if
18:      // verifying the feasibility of the computed positions
19:      for j = 1, ..., N do
20:         if (x_v^{i,j} is feasible) then
21:            BP(v + 1, G, D);
22:         end if
23:      end for
24:   end for
25: end if
```

2.4 The Branch-and-Prune Algorithm

The basic idea behind the Branch-and-Prune (BP) algorithm finds its origins in a previous approach to DG (mentioned, among others, in Section 1.2): the Build-Up algorithm. When working only with exact distances, the necessary assumptions for running the Build-Up algorithm are in fact satisfied when adding one supplementary reference distance in the assumption **(b)** of Def. 2.1.1 (as well as in assumption **(b')** of Prop. 2.3.2). In these hypotheses, the possible positions for the current vertex v can be obtained by intersecting K spheres, whose intersection gives one singleton with probability 1 [139]. Therefore, the search domain of the Build-Up algorithm corresponds with the unique solution to the problem. The main intuition in [22], where a very preliminary version of the BP algorithm was proposed, is that a finite number of vertex positions can still be identified under weaker assumptions. Naturally, these weaker assumptions imply the definition of a search domain that grows exponentially with the size of the instances (see Section 2.1), but it allows to deal with a much larger set of DGP instances.

The BP algorithm was formally introduced in [95]. It performs a systematic exploration of the search tree $\mathcal{T}$, defined in Section 2.1. As already remarked, when instances containing interval distances are considered, the original tree $\mathcal{T}$ is replaced with an approximated tree where arc nodes are substituted with a predefined number of nodes representing sample positions (extracted from the arcs).

The tree of possible solutions 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 are placed in their unique positions (see Section 2.5), and then the search actually starts with the vertex having rank $K + 1$ in the associated discretization order. At each step, the candidate positions for the current vertex v are computed, and the search is branched. Depending on the available distance information, the set of candidate positions may contain either two singletons, or two disjoint arcs. In the latter case, every arc can be approximated with a subset of D sample positions.

Pruning devices can be employed for discovering infeasible vertex positions. The main pruning device exploits the so-called pruning distances of DDGP instances. As soon as a vertex position is found to be infeasible, then the corresponding branch can be pruned and the search can be backtracked. Thanks to the pruning phase, the size of the tree can be kept relatively small, so that an exhaustive search on the remaining branches becomes feasible for some subsets of instances.

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, G is a simple weighted undirected graph representing an instance of the DDGP, and D is the discretization factor. Once the initial clique has been realized, the BP algorithm can be invoked recursively, starting from the vertex v having rank $K + 1$.

There are two main phases that can be identified in the BP algorithm. In the *branching* phase, new candidate positions for the current vertex v are generated. Different methods can be employed in practice for the computation of such candidate positions (see Section 2.6). Moreover, once computed, the feasibility of candidate positions needs to be verified, and this can be done during the *pruning* phase of the BP algorithm. To this purpose, various pruning devices can be conceived and integrated with the BP algorithm (see Section 2.7).

The correctness of the BP algorithm is consequence of the use of the pruning devices. The algorithm terminates when it reaches a leaf node, or before (when no candidate positions are feasible at a given recursion level). It can also continue the exploration after the identification of the first solution, in order to construct the entire solution set. At the end of the execution, if all branches that did not reach a leaf node are removed from $\mathcal{T}$, then the entire solution set of the DDGP instance is obtained.

2.5 Realizing the Initial Clique

Since the initial clique in assumption **(a)** is a complete graph where all edges correspond to exact distances, there exists only one valid realization of this clique in $\mathbb{R}^K$, modulo total translations, rotations and reflections.

The construction of this realization can be performed vertex by vertex, and by selecting position coordinates in a way to avoid redefining solutions that can in fact be obtained by applying total translations, rotations, and reflections to other solutions. In this section, we will use the symbols v_1, v_2, ..., v_K to refer to the vertices of the initial clique, and the symbols x_1, x_2, ..., x_K for the corresponding positions. Even if irrelevant to the discretization order, we can suppose that a total order relationship exists for the vertices of this clique, given by the superscripts of every v_k.

In order to prevent translations, we can fix the coordinates of the first clique vertex v_1 at $x_1 = (0, 0, \ldots, 0) \in \mathbb{R}^K$. The possible positions for v_2 belong to the sphere centered in $(0, 0, \ldots, 0)$ and having radius $\delta(v_1, v_2)$. However, in order to prevent rotations around v_1, we can fix an axis passing through v_1, and obtain in this way two possible positions x_2 for v_2. In order to prevent solutions to reflect over the hyper-plane that is orthogonal to the axis passing through v_1 and v_2, we can fix one of these two possible positions for v_2.

The procedure continues similarly for the other vertices v_k, with $3 \leq k \leq K$. For every new v_k to be added, the problem is restricted to $\mathbb{R}^{k-1}$, where $k-1$ vertices have already been placed in a unique position x_v, and all distances (belonging to E') between all those vertices and v_k are known. Therefore, the possible positions for v_k can be obtained by intersecting $k-1$ spheres in $\mathbb{R}^{k-1}$: when the discretization assumptions are satisfied, this gives as a result two positions [85, 124]. However, in order to prevent reflections over the hyper-plane that is orthogonal to the one passing through v_1, v_2, ..., v_{k-1}, we can arbitrarily select one of these two possible positions.

At the end, this procedure provides every vertex in the initial clique with a unique position. All solutions that are subsequently constructed from this realization for the initial clique are independent from each other (i.e. none of them can be obtained from another solution by applying the above mentioned transformations). In fact, while any translation is prevented by the way x_1 is defined, the way the other $K-1$ vertices of the clique are computed ensures that no rotations and no reflections around the hyper-planes formed by subsets of vertices can be performed.

It is interesting to remark that, in all DDGP instances, the first $K+1$ vertices form as well a clique. The same procedure detailed above can be therefore applied. However, for the vertex v_{K+1}, it is not possible to select only one position. The total reflection around the hyper-plane given by the first K vertices defines the main symmetry of the search trees $\mathcal{T}$ (see Section 2.8).

2.6 Computing Vertex Positions

The method employed to compute the candidate positions at each recursive call of Alg. 1 has a fundamental importance. While looking for candidate vertex positions for v, it is supposed that K reference vertices for v are already positioned on the current branch of $\mathcal{T}$. In the following, in order to avoid including too complex notations, the discussion will focused on the three-dimensional case, i.e. for $K = 3$.

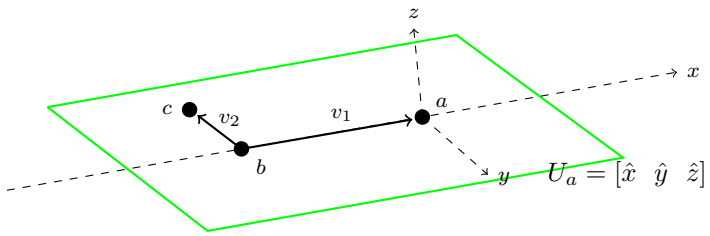


Figure 2.4: *The reference vertices a, b and c induce a local system of coordinates.*

However, both methods discussed below can be extended for any K greater or equal to 1 (the reader is referred to [53, 103]).

When $K = 3$, the discretization assumptions ensure that there exist 3 reference vertices $\{a, b, c\}$ for the current vertex v. These reference vertices define a local coordinate system centered at the vertex a [54, 152], illustrated in Fig. 2.4.

When the three reference distances belong to E', three spheres are defined, whose intersection gives 2 points, with probability 1 [85]. The two points x_v^+ and x_v^- for vertex v are symmetric with respect to the plane defined by the reference vertices. When one of the three distances belongs instead to $E \setminus E'$, the intersection involves two spheres and one spherical shell, which results in two arcs (see Fig. 2.4). These two arcs correspond to two intervals, $[\underline{\omega}_v^+, \bar{\omega}_v^+]$ and $[\underline{\omega}_v^-, \bar{\omega}_v^-]$, for the angle ω_v. In order to discretize these intervals, D points can be selected from the two arcs. This selection can be performed in different ways: (i) D distances can be extracted from the intervals (with equal step from one to the next one); (ii) D angles can be extracted from the angle intervals (still with a constant step); (iii) D equidistant points can be selected from the obtained arcs. All these techniques are simple to implement, and they are equivalent in terms of complexity. In all situations, after performing this selection, the problem is reduced to the one of computing the intersection among three spheres. Therefore, we will suppose in the following that the available discretization distances are exact.

From the equations of the spheres in the three-dimensional space, we can deduce that the points belonging to the intersection of the three spheres can be obtained by solving the following system of quadratic equations:

$$\begin{cases} ||x_v - x_a||^2 &= \delta_{v,a}^2 \\ ||x_v - x_b||^2 &= \delta_{v,b}^2 \\ ||x_v - x_c||^2 &= \delta_{v,c}^2. \end{cases}$$

This particular quadratic system can be solved by calculating the solutions of two linear systems [27]. However, solution methods for both quadratic and linear systems can lead to numerical instabilities [124]. We will present, in Sections 2.6.1 and 2.6.2, two numerically stable methods for the computation of vertex positions at each recursive call of the BP algorithm.

2.6.1 Matrix Accumulation Method

In order to compute the possible positions for the vertex v, we exploit the information about the three reference vertices a, b and c, and the corresponding distances $\delta_{a,v}$, $\delta_{b,v}$ and $\delta_{c,v}$. This information is available because the discretization assumptions are satisfied. We will use the symbol θ_v for referring to the angle formed by the two segments (v,a) and (a,b), and we will use the symbol ω_v for referring to the angle formed by the two planes defined by the triplets (a,b,c) and (b,a,v). The cosine of the angles θ_v and ω_v can be computed by exploiting the positions of the reference vertices a, b and c, as well as the discretization distances $\delta_{a,v}$, $\delta_{b,v}$ and $\delta_{c,v}$:

$$\begin{cases} \cos\omega_v &= \dfrac{\cos\theta_{c,b,v} - \cos\theta_{a,b,v}\cos\theta_{a,b,c}}{\sin\theta_{a,b,v}\sin\theta_{a,b,c}}, \\ \\ \cos\theta_v &= \cos\theta_{b,a,v} = \dfrac{\delta_{a,b}^2 + \delta_{a,v}^2 - \delta_{b,v}^2}{2\,\delta_{a,b}\,\delta_{a,v}}. \end{cases} \tag{2.2}$$

By following [85], we can compute the two possible positions for the vertex v by:

$$\begin{bmatrix} x_v^{(1)} \\ x_v^{(2)} \\ x_v^{(3)} \\ 1 \end{bmatrix} = B_1 B_2 \cdots B_v \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

where the matrices are defined inductively:

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} -1 & 0 & 0 & -\delta_{1,2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_3 = \begin{bmatrix} -\cos\theta_3 & -\sin\theta_3 & 0 & -\delta_{2,3}\cos\theta_3 \\ \sin\theta_3 & -\cos\theta_3 & 0 & \delta_{2,3}\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and finally

$$B_v = \begin{bmatrix} -\cos\theta_v & -\sin\theta_v & 0 & -\delta_{v-1,v}\cos\theta_v \\ \sin\theta_v\cos\omega_v & -\cos\theta_v\cos\omega_v & -\sin\omega_v & \delta_{v-1,v}\sin\theta_v\cos\omega_v \\ \sin\theta_v\sin\omega_v & -\cos\theta_v\sin\omega_v & \cos\omega_v & \delta_{v-1,v}\sin\theta_v\sin\omega_v \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

This procedure is very efficient and works well in the practice [85, 87, 130]. It requires, however, that the reference vertices a, b and c are the ones that immediately precede v, i.e. the consecutivity assumption needs to be satisfied (see Section 2.3). As remarked above, finding an order satisfying this assumption is NP-hard, while orders that do not satisfy the consecutivity assumption can be obtained automatically in polynomial time (see Chapter 3).

2.6.2 Change of Basis Method

This method for the computation of the vertex positions is a generalization of the one in Section 2.6.1. Therefore, we begin the discussion with analyzing the matrices B_i (see equ. (2.3)). The first three elements of the last column of the matrix correspond to the spherical coordinates of a vertex position in a certain system of coordinates, where a is its center, the x-axis is defined in such a way that b is on its negative side, and the y-axis (orthogonal to the x-axis) is defined such that the vertex c is on the xy-plane and has negative y coordinate (see Fig. 2.4). We remark that this setting allows us to have a clockwise orientation for the angles ω_v, in a way that the minimum distance between c and v is achieved when $\omega_v = 0$ (equivalently, we have the maximal achieved distance when $\omega_v = \pi$). Naturally, the z-axis is normal to the xy-plane. In the following, we will refer to this coordinate system as the *system defined in a*.

Similarly, we define a new matrix U_a which is able to convert position coordinates from the system defined in a to the system defined by the canonical system (the one defined by the initial clique, see Section 2.5). Let v_1 be the vector from b to a and v_2 be the vector from b to c (see Fig. 2.4). The x-axis for the system in a can be defined by v_1, and the unit vector in this direction is $\hat{x} = v_1/\|v_1\|$. Moreover, the vectorial product $v_1 \times v_2$ gives the vector that defines the z-axis, whose corresponding unit vector is $\hat{z}$. Finally, the vectorial product $\hat{x} \times \hat{z}$ provides the vector that defines the y-axis (let the unit vector be $\hat{y}$).

These three unit vectors are the columns of the matrix $U_a = [\hat{x} \;\; \hat{y} \;\; \hat{z}]$, whose role is to directly convert vertex positions from the coordinate system defined in a to the canonical system. Once the matrix U_a has been computed, the canonical Cartesian coordinates for a candidate position for the vertex v can be obtained by:

$$x_v(\omega_v) = x_a + U_a \begin{bmatrix} -\delta_{a,v} \cos\theta_v \\ \delta_{a,v} \sin\theta_v \cos\omega_v \\ \delta_{a,v} \sin\theta_v \sin\omega_v \end{bmatrix}. \tag{2.4}$$

The two angles θ_v and ω_v, computed as in equ. (2.2), correspond to the spherical coordinates of vertex i. The two possible positions for the vertex v, x_v^+ and x_v^-, correspond to the two possible opposite values, ω_v^+ and ω_v^-, for the angle ω_v.

2.6.3 Arc reduction technique

As discussed in the previous section, D sample positions can be extracted from the two arcs that are obtained when intersecting two spheres with one spherical shell.

This allows to approximate the original tree $\mathcal{T}$, containing either positions or arcs on its nodes, with another tree $\mathcal{T}'$ consisting of only vertex positions. In this section, we describe a procedure that can be executed *before* selecting the D sample positions per arc, so that all these selected positions are at least feasible for the DDF pruning device (see Section 2.7.1). This procedure allows therefore to avoid generating sample positions that can immediately be discarded at the same layer when applying the DDF pruning device.

Our adaptive scheme is based on the idea to identify, before the branching phase of the algorithm, the subset of positions on the two computed arcs that is feasible with respect to all pruning distances that can be verified at the current layer [57]. Let us suppose that, at the current layer v, the distance $\delta_{c,v}$ is represented by the interval $[\underline{\delta}_{c,v}, \bar{\delta}_{c,v}]$. By using equ. (2.4), two intervals for the angle ω_v can be identified: $[\underline{\omega}_v^+, \bar{\omega}_v^+] \subset [0, \pi]$ and $[\underline{\omega}_v^-, \bar{\omega}_v^-] \subset [\pi, 2\pi]$, such that the distance constraints

$$\begin{array}{rcccl} & & \|x_a - x_v(\omega_v)\| & = & \delta_{a,v}, \\ & & \|x_b - x_v(\omega_v)\| & = & \delta_{b,v}, \\ \underline{\delta}_{c,v} & \leq & \|x_c - x_v(\omega_v)\| & \leq & \bar{\delta}_{c,v}, \end{array} \tag{2.5}$$

are satisfied.

Let us suppose there is a vertex $u \in \{w < v \mid v \notin \{a, b, c\}\}$, such that the distance $d_{u,v}$ is known and lies in the interval $[\underline{\delta}_{u,v}, \bar{\delta}_{u,v}]$. The solution set of the inequalities

$$\underline{\delta}_{u,v} \leq ||x_u - x_v(\omega_v)|| \leq \bar{\delta}_{u,v}$$

consists of intervals for ω_v that are compatible with the distance $d_{u,v}$. A discussion about how to solve the inequalities (2.5) is given in details in [57], where all possible scenarios are taken into consideration.

The feasible positions for the vertex v can be therefore obtained by intersecting the two previously computed arcs (in bold in Fig. 2.4), and *several* spherical shells. The final subset of $\mathcal{C}$, which is compatible with all available distances, can be found by intersecting the arcs obtained for each pruning distance with the two initial disjoint arcs, given by equ. (2.5). From this final set, we can extract $2D$ sample positions, that all satisfy the DDF pruning device.

We remark that similar results can be obtained by working with Clifford algebra [9, 10]. Clifford algebra has the advantage to provide a formal language for the representation of the sphere intersections, for which a mathematical expression can be derived. However, the complexity of such expressions can increase when dealing with larger instances, and the common practice is therefore to replace the mathematical expressions with sampled points. As a consequence, even if the Clifford algebra approach essentially differs from the one given above, the same information is finally exploited in both methods.

2.7 Pruning Devices

Pruning devices are essential in the BP algorithm for focusing the search on the feasible parts of the search tree.

2.7.1 Direct Distance Feasibility

The simplest pruning device for the BP algorithm is the Direct Distance Feasibility (DDF) criterion [85], which consists in verifying the ϵ-feasibility of constraints involving distances between the current vertex v and previously placed vertices:

$$\forall\{w,v\} \in E, \text{ with } w < v \text{ and } w \notin \{a,b,c\}, \quad \underline{\delta}_{w,v} - \epsilon \leq \|x_w - x_v\| \leq \bar{\delta}_{w,v} + \epsilon,$$

where $\{a,b,c\}$ is the set of the reference vertices for v. The distances in E that are involved in the above constraints are for this reason generally referred to as "pruning distances".

DDF works very well in practice when the pruning distances are exact; it can however be less effective when interval distances with larger ranges are available. Moreover, it cannot take into consideration pruning distances related to vertices associated to further layers of the tree. An attempt of early use of these pruning distances can be found in [1].

2.7.2 Shortest-Path-based pruning device

This pruning device is based on the point-to-point Dijkstra shortest-path searches on Euclidean graphs. Consider vertices u, v, h with $u < v < h$ such that $\{u,h\} \in E$, i.e. the distance $\delta_{u,h}$ is known. Let us suppose that the BP algorithm already placed the vertex u in the position x_u, and that the feasibility for the position x_v of the vertex v needs to be verified. Let $D(v,h)$ be an upper bound to the distance $||x_v - x_h||$ for all possible solutions to the problem. It has been proved in [85] that, if the inequality

$$D(v,h) < ||x_u - x_v|| - \delta_{uh}$$

holds, then the search node for the vertex position x_v can be pruned. One way for computing the upper bound $D(v,h)$ is by locating the shortest path between the vertex v and the vertex h of the graph G [88].

2.7.3 vdW radii pruning device

This pruning device is particularly designed for DDGP instances related to molecules (see Section 1.3.1). Even if it is generally represented by one point, an atom fills a certain portion of space: its nucleus, consisting of neutrons and protons, has a predetermined volume, while electrons orbit around this nucleus, at a given distance from it. Therefore, a more accurate representation for an atom is a sphere centered in its nucleus (which corresponds to the center of the atom) and having radius equal to the distance between the nucleus and the orbiting electrons. This distance can be estimated for each kind of atom, and it is generally referred to as *van der Waals* (vdW) radius [20].

When two atoms are chemically bonded, their *clouds* of electrons tend to overlap. If they are not bonded, however, repulsion forces do not allow their clouds to be too close to each other. The vdW pruning device is therefore based on this simple idea:

when two atoms are not bonded, their relative distance should be greater than the sum of the two corresponding vdW radii. This verification can be applied to all pairs of atoms for which no pruning distance is available.

Differently from the DDF pruning device, a distance lower bound is only available. When the relative distance between two vertices u and v, with $u < v$ and the vertex v just positioned, goes below the predefined threshold, then the current candidate position x_v for v can be rejected [56]. Since nonbonded atoms can actually admit a small overlap, we can set the distance lower bound to the 80% of the sum of the two vdW radii. The vdW radii, for different pairs of atoms, can be obtained from different sources: one example is [102].

2.7.4 Torsion Angle Feasibility

Along with the list of lower and upper bounds on the distances, NMR experiments can also provide information on the torsion angles of protein backbones (see Section 1.3.1). Three different torsion angles can be defined along the backbone main chain N–C_α–C–N–...:

$$\begin{aligned} \phi &\equiv (\mathrm{C}, \mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}), \\ \psi &\equiv (\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}, \mathrm{N}), \\ \omega &\equiv (\mathrm{C}_\alpha, \mathrm{C}, \mathrm{N}, \mathrm{C}_\alpha). \end{aligned}$$

The angle ϕ is the angle defined by the two planes (C, N, C_α) and (N, C_α, C). The angle ψ is the angle defined by the two planes (N, C_α,C) and (C_α, C, N). Finally, the torsion angle ω is usually very close to π, because there is a peptide bond that does not allow this subset of atoms to take any other configuration. The other two angles ϕ and ψ, instead, can vary in larger ranges [140].

Even if the BP algorithm is not based on the torsion angle representation of the protein backbone, but rather on a representation at atomic level, the torsion angles ϕ and ψ can be easily computed every time the four atomic positions needed for their computation are available. As soon as the value for one of these angles is obtained, we can check if it satisfies the known lower and upper bounds provided by NMR experiments: the last positioned atom can therefore be pruned if the computed angle does not satisfy this constraint. This pruning device is called Torsion Angle Feasibility (TAF) [130].

2.7.5 Secondary Structure Feasibility

Subsets of atoms of a protein can fold in local structures which are very typical in proteins. Such local structures are referred to as *secondary structures*, and they are mainly represented by α-helices and β-sheets (see Section 1.3.1). In both cases, these secondary structures are stabilized through hydrogen bonds between pairs of amino acids. More precisely, given a pair (aa_i, aa_j) of amino acids belonging to the same secondary structure, there is a hydrogen bond between the hydrogen H (the one bound to the N) of amino acid aa_i and the oxygen O (bound to the C of amino acid

aa_j). This hydrogen bond forces the involved atoms, and in particular the hydrogen H of aa_i and the oxygen O of aa_j, to be near each other.

As a consequence, the torsion angles ϕ and ψ are constrained to vary in predefined ranges when the corresponding amino acids fold in α-helix or β-sheet. The bounds on the torsion angles can therefore be refined by using this information, so that more accurate values can be used when invoking the pruning device TAF. Moreover, in the case of α-helices, it is known that the amino acid aa_j is always aa_{i+4}. Therefore, there are two atoms, one in aa_i and another in aa_{i+1}, that form a hydrogen bond. As a consequence, a new edge, concerning the distance between the hydrogen H of aa_i and the oxygen O of aa_{i+4}, can be included to the distance list encoded by G. The possibility to add this new distance for each amino acid in α-helices reflects the strong regularity of this secondary structure; β-sheets are instead less regular: for each aa_i, it is not straightforward to predict the corresponding aa_j.

In order to prune conformations which do not satisfy the restrictions given by the protein secondary structures, we use the *chemical shift index* [13] provided by NMR experiments to identify the subset of amino acids that are supposed to fold in α-helix or in β-sheet. The technique described in [146] is able to find good estimates of the torsion angles related to amino acids having a given chemical shift index. However, when it is not necessary to have tight bounds on the torsion angles, we can simply consider intervals that are centered in $-60°$ for both ϕ and ψ (typical values for α-helices), or centered in $135°$ and $-120°$ (typical values for ϕ and ψ, respectively, in β-sheets) [13].

The Secondary Structure Feasibility (SSF) pruning device is therefore based on the idea of refining bounds for the torsion angles and/or of adding new distances to the considered instance [130]. We remark that the SSF pruning device can be employed even if the oxygen atoms belonging to the protein backbones are not explicitly included in the instances. This is actually possible when the atoms C, N and H (bonded to the N) are included in the instance. In this case, the coordinates of the oxygen O can be computed by intersecting three spheres which are centered in these three atoms (C, N and H), and having as radii the corresponding distances from O. This intersection of spheres can be computed by solving two linear systems (see Section 2.6), and this procedure can provide in general two possible sets of coordinates for the oxygen O. However, since the four atoms O, C, N and H are supposed to lie on a common plane, only one position for O should be given as a solution.

2.7.6 Lennard Jones pruning device

This pruning device is based on the overall internal energy of a molecule. As it is well-known, an accurate description of all interactions among the atoms in a molecule can be very complex, so that the overall energy is generally approximated by taking into consideration the most important interactions.

The vdW energy is the one existing between single pairs of non-bonded atoms [65] (in this case, we consider both repulsion and attraction forces). However, for modeling the overall energy, it is possible to consider the sum of pairwise LJ potentials

12–6:

$$E_{LJ} = \sum_{u,v} 4\varepsilon_{u,v} \left[\left(\frac{\rho_{u,v}}{\delta_{u,v}} \right)^{12} - \left(\frac{\rho_{u,v}}{\delta_{u,v}} \right)^{6} \right],$$

where $\varepsilon_{u,v}$ and $\rho_{u,v}$ are two parameters that can be defined by the relationships between the pairs of atoms u and v [91]. The parameter $\rho_{u,v}$ is the distance where the pair potential is zero, whereas $\varepsilon_{u,v}$ is the well depth. The minimum value for the LJ pair potential is $-\varepsilon_{u,v}$ achieved in $2^{1/6}\,\rho_{u,v}$ (which corresponds to the vdW radius).

During the execution of the BP algorithm, every time a leaf node is reached (at layer n), a complete conformation is found, and its energy E_n can be computed. Let us suppose that $\hat{E}_n$ is the lowest energy found so far. The basic idea behind the LJ pruning device is to verify in advance whether new branches of the tree (at different layers) can actually contain conformations with an energy that can be potentially smaller than $\hat{E}_n$. This can be done by computing a lower bound on the energy concerning all the conformations belonging to a common branch.

Depending on the range in which the inter-atomic distances can vary, however, we can compute a more (or less) accurate lower bound for the actual value of the energy. In case the BP algorithm is currently positioned on the layer v, then we can compute a partial energy value $E_{n(\leq v)}$ (computed by using the available coordinates) and a lower bound $L_{(>v)}$ on the energy $E_{n(>v)}$ (approximated by summing the minimum values given by the Lennard Jones terms for which the distance in the realization is not available yet). Therefore, if $E_{n(\leq v)} + L_{(>v)} > \hat{E}_n$, there is no hope to identify a conformation with an energy smaller than $\hat{E}_n$ while exploring the current branch of the tree. This branch can therefore be pruned [56].

We point out that the LJ pruning device considers implicitly the vdW pruning device (see Section 2.7.3). When the vdW and LJ pruning devices work together, it is appropriate to apply LJ only after vdW.

2.8 Symmetries of the Search Domain

Several symmetries can be identified in the search domains obtained with the discretization. Such symmetric solutions are actually already contained in the original (continuous) domain, but the discretization process allows to isolate them in the new discrete set. This is of particular interest in some specific applications (such as the one in Section 1.3.1, where different levels of symmetries were already studied [104]). In the following, we will present some general results that are independent from the applications.

Let G be a graph representing an instance of the DDGP for which the corresponding discretization order satisfies the consecutivity assumption. Recall that the edge set E can be partitioned in the subsets E' and $E \setminus E'$ that separate the available distances in exact (priority 1) and inexact distances, respectively; moreover, the subsets E_d and E_p can be defined so that they contain the discretization distances

and the pruning distances, respectively. The following discussion is restricted to the case $E = E'$ and $K = 3$. However, all results can be extended for general K.

The discretization assumptions ensure that 3 edges $\{u, v\}$ related to the current vertex $v \in V$ are contained in E_d, and that every vertex u has a rank smaller than v in the associated discretization order. The existence of a pruning edge $\{w, v\} \in E_p$, with $w < v$, allows us to define the set of possible positions for v as an intersection of four spheres (instead of only three spheres, as it is done in the general step of the BP algorithm). When all the distances are exact, this intersection gives one singleton. We will suppose in the following that, when the fourth distance is available at a given layer, it is always independent from the discretization distances.

As discussed in Section 2.4, solutions of a DDGP instance can be represented in different ways. One possibility is to represent each solution as a path from the tree root to one of the feasible leaf nodes (see Fig. 2.5). Another efficient way to represent solutions is through a vector of binary variables. Fig. 2.5 shows the search tree $\mathcal{T}$ corresponding to a small DDGP instance with 9 vertices. The colors mark the presence of 8 paths on the tree representing the 8 solutions.

During the execution of the BP algorithm (see Section 2.4), we can have the following two extreme situations [127]. If E_p is empty, then the feasibility of no computed vertex position can be verified, and no branches of the search domain can be pruned. As a consequence, the corresponding instance has 2^{n-3} solutions, which correspond to the 2^{n-3} leaf nodes of the search tree (recall that $n = |V|$). In this situation, the execution of the BP algorithm is exponential, and provides, as a solution set, the entire search tree.

If E_p is not empty, the presence of a pruning distance allows us to select some of the candidate positions, and hence to select a subset of branches on which we can focus our search (see Section 2.7.1). If there is a distance $\{u, v\} \in E_p$, then there is no branching on the tree at layer v. As a consequence, the number of tree branches at layer $v - 1$ is the same as the number of branches at layer v.

The second extreme situation is therefore the following. Let us suppose that, for each layer $v > K$ of the search tree, there is a pruning distance $\{u, v\} \in E_p$, and that all involved distances are exact. In such a case, branching is only allowed at layer K, and there are only 2 leaf nodes, corresponding to two solutions. These two solutions are symmetric w.r.t. the plane defined by the initial clique (see Section 2.5). We refer to the symmetry at layer $K = 3$ of the search tree as the *first symmetry*. This symmetry is present in all DDGP instances

The most interesting situation is therefore the one in which E_p is not empty, but where pruning distances are not present at each layer. For each node at layer $v - 1$, there are two new branches at layer v, but one of the two (or even both) can be pruned later on at a further layer if there is a pruning distance [98]. In fact, a pruning distance does not have to directly concern v for making a branch rooted at v infeasible: a distance between two vertices u and w such that $u + 3 < v < w$ is indeed sufficient. In other words, there is a feasible branching at layer v (i.e. a branching defining two branches of length $|V|$) in the *only* case in which there is no pruning distance $\{u, w\}$ *passing over* the layer v. More formally, the layers of the

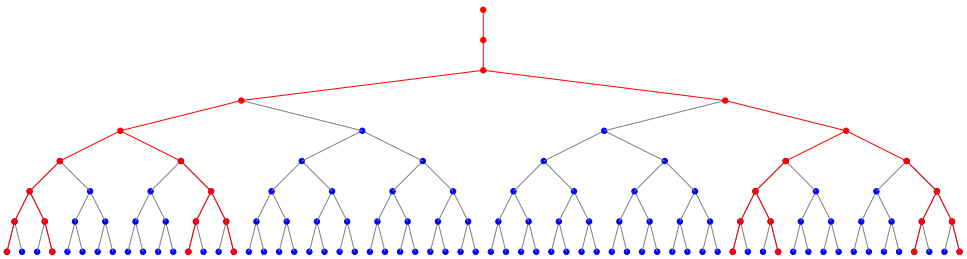


Figure 2.5: *A small binary tree related to an instance containing 9 vertices where the symmetry set S is* $\{4, 6, 8\}$.

search tree where there is feasible branching are the ones contained in the set:

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

We refer to S as the *symmetry set* of the graph G.

Since $v = 4$ is always contained in S, pruning can never occur at this layer, and this reflects the presence of the first symmetry. In general, for each $v \in S$, there is a duplication of feasible branches of the tree.

Let us consider a tree layer $v \in S$. Consider that the root of our search tree is actually $v - 3$, and that the next tree layers are sorted in accordance with the discretization order. This modified search tree corresponds to a sub-instance of the original problem that keeps both properties of discretizability and consecutivity assumption. For this reason, the modified tree has a symmetry at its layer 4: this layer corresponds to the layer v of the original tree. This intuitively shows that there is a symmetry at layer v of the original tree. The formal proof of this result can be found in [98, 99]. The same idea cannot be applied to vertices $v \notin S$, because the presence of a pruning distance crossing over v implies the possibility to prune at least one of the two branches rooted at v. We say therefore that a layer v is *symmetric* if $v \in S$.

There are three symmetries in the tree depicted in Fig. 2.5. The first one is the first symmetry, at layer 4. Other two symmetries are also present in this tree. One is at layer 6: four symmetric branches are rooted at vertex positions for vertex 6. Similarly, eight symmetric branches start from vertex positions at layer 8, where another symmetry is present. In this example, therefore, $S = \{4, 6, 8\}$. The total number of solutions for this instance is in fact $2^{|S|} = 8$.

Let us consider the solution in Fig. 2.5 corresponding to the second leaf node (from left to right). The binary vector corresponding to this solution is

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

where we suppose that 0 represents the choice *left*, and 1 represents *right* (the first three zeros are associated to three vertices of the initial clique). Since there is a symmetry at layer 6, another solution to the problem can be easily computed by

repeating all branch choices from the root node until the layer 5, and by inverting all other choices. On the binary vector, repeating means copying, and inverting means flipping. So, another solution to the problem is

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

This solution corresponds to the third leaf node in Fig. 2.5.

This procedure can be exploited for speeding up the BP algorithm, because the entire solution set can actually be computed by exploiting one known solution and the symmetry set S [123, 125]. The set S is able to provide a priori information on the quantity and on the *location* of the symmetries in search trees. If the current layer is related to a vertex $v \in S$, then, for each x_{v-1} at the previous layer, both newly generated positions for x_v are feasible. If $v \notin S$, instead, only one of the two positions can be part of a branch leading to a solution. The other position is either infeasible or it defines a branch that will be pruned later on at a further layer v, in correspondence with a pruning distance whose graph edge $\{u, w\}$ is such that $u + 3 < v \leq w$. Therefore, we can exploit such an information for performing the selection of the branches that actually define a solution to the problem. When $v \notin S$ (only one position is feasible), it is not known which of the two branches is the correct one. This is the reason why at least one solution must be obtained before having the possibility of exploiting the symmetries for computing all the others.

Let us suppose that the solution $\mathbf{s}_1$ has already been computed (by applying the BP algorithm, for example), and that it is the leftmost feasible branch in Fig. 2.5:

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

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

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

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

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

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

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

During this phase, pruning distances in E_p do not need to be verified, because all branches constructed by symmetry are feasible. This simple procedure is at the basis of the symmetry-driven BP (symBP) algorithm, detailed in the next section. We remark that ongoing research is aimed at formally proving that DDGP instances without consecutivity assumption can also generate search trees $\mathcal{T}$ having interesting properties that are close to symmetry.

2.8.1 The symmetry-driven BP

This variant of the BP algorithm is for DDGP instances satisfying the consecutivity assumption and consisting of exact distances only. After the positioning of the initial clique at the root of the search tree (see Section 2.5), the BP algorithm recursively calls itself for a complete exploration of the search tree (see Section 2.4). At each call, the two possible positions for the vertex v, x_v^0 and x_v^1, are computed (by one of the two methods in Sections 2.6.1 and 2.6.2), and their feasibility is verified by the DDF pruning device (see Section 2.7.1).

The basic idea in the symmetry-driven BP (symBP) algorithm is as follows. In order to fully exploit the information given by the symmetry set S, it is necessary to obtain at least one solution. Therefore, as far as no solutions are found, symBP basically behaves like the standard BP: it tries to reach the first leaf node of the search tree. During this phase, the set S is only used for avoiding invoking the DDF pruning device when working on symmetric vertices (by definition, no pruning distances are available). After the identification of the first solution, all other solutions can then be simply computed by exploiting the information about the symmetries given by the symmetry set S.

Algorithm 2 is a sketch of symBP. In the symBP call, there are the input arguments necessary for a BP call, as well as some additional ones. Naturally, symBP needs as an argument the set S containing the BP tree symmetries. Moreover, the flag `first_sol_found` is used for monitoring whether solutions were already found, and the last obtained solution is stored in *prev*, a vector of binary variables. Notice that it is not necessary to give the BP argument D, because necessary only when some discretization distances belong to $E \setminus E'$.

At the beginning, symBP checks whether the last vertex (ranked $n = |V|$) has been placed during the previous recursive call. In such a case, the flag `first_sol_found` is updated (a solution has just been found), and all the branches rooted at $v \notin S$ and not defining a solution are removed from the tree. If at least one solution is known, indeed, the 0/1 choices are already available for each $v \notin S$. Since, at each layer, one of the choices brought to the identification of a solution, the other one can now be discarded. Only partial branches rooted at $v \in S$ are kept, i.e. we only consider pairs of symmetric branches. Every time the algorithm is invoked with a vertex $v \in S$, the two possible positions for this vertex are computed, and the algorithm is invoked twice, once for each computed position. The DDF pruning device is not executed because, by definition, there are no pruning distances at the current layer.

If instead $v \notin S$, it is important to distinguish between two situations. If no solutions have been found so far, then the information regarding the symmetries of the search tree cannot be exploited yet, and the standard BP procedure is implemented. During this phase, the binary vector *prev* is kept updated so that it will contain, as soon as it is identified, the first found solution encoded in binary format. Notice that, if a solution is found when considering the position x_v^0 of v, it is useless to consider the position x_v^1 (recall that $v \notin S$).

Finally, if $v \notin S$ and at least one solution has already been found, symBP simply

Algorithm 2 The symBP algorithm

```
1: symBP(v, G, S, prev,first_sol_found)
2: if (v > |V|) then
3:    // one solution is found
4:    let first_sol_found = yes;
5:    remove all branches such that:
5:          • their root is u ∉ S, and
5:          • their leaf nodes v_f < n.
6: end if
7: // working with a symmetric layer
8: if (v ∈ S) then
9:    compute x_v^0;
10:   let prev(v) = 0;
11:   symBP(v + 1, G, S, prev,first_sol_found);
12:   compute x_v^1;
13:   let prev(v) = 1;
14:   symBP(v + 1, G, S, prev,first_sol_found);
15: end if
16: // non-symmetric layer, no solutions found yet
17: if (v ∉ S and first_sol_found = no) then
18:   compute x_v^0;
19:   if (x_v^0 is feasible) then
20:      let prev(v) = 0;
21:      symBP(v + 1, G, S, prev,first_sol_found);
22:   end if
23:   if (first_sol_found = no) then
24:      compute x_v^1;
25:      if (x_v^1 is feasible) then
26:         let prev(v) = 1;
27:         symBP(v + 1, G, S, prev,first_sol_found);
28:      end if
29:   end if
30: end if
31: // non-symmetric layer, at least one solution was found
32: if (v ∉ B and first_sol_found = yes) then
33:   compute x_v^{¬prev(v)};
34:   let prev(v) = ¬prev(v);
35:   symBP(v + 1, G, S, prev,first_sol_found);
36: end if
```

reconstructs other solutions by exploiting the symmetries and the last computed solution encoded in binary format. There is no branching phase, because only $x_v^{\neg prev(v)}$ can be feasible if $x_v^{prev(v)}$ was feasible for the previous solution. There is no pruning

either, because all generated positions are feasible.

Ongoing research aims at extending the symBP algorithm so that it can better exploit the information in the symmetry set *during* the search of the first solution (see [46]), as well as to extend the overall methodology to DDGP instances for which the consecutivity assumption is not satisfied, and that contain interval distances.

2.9 Parallel Computing Strategies

In recent years, we have proposed two different parallelization strategies for the BP algorithm. The first one [129] (in chronological order) is designed for instances of the DDGP that also satisfy the consecutivity assumption. The main idea is to split instances in sub-instances consisting of subsets of consecutive vertices. In this way, every sub-instance satisfies the discretization assumptions (and the consecutivity assumption), and they can therefore be solved independently by the processors of a parallel machine. However, some of the pruning distances, the ones concerning pairs of vertices that belong to two different sub-instances, cannot be considered during the parallel phase. For this reason, a further step is necessary when all processors exchange their local solutions, where those *crossing* pruning distances need to be verified.

An overlap of K vertices between consecutive sub-instances allows for reconstructing all vertex positions in a common Cartesian system. In the original paper [129], it is supposed that all processors of the parallel machine have the same computational power, so that the subdivision in sub-instances can be simply performed by separating the original instance in p sub-instances having the same size, where p is the number of available processors. However, the number of omitted pruning distances during the parallel phase increases when p is larger. To overcome to this issue (causing the final step of combining local solutions to be more expensive than the parallel phase), it was more recently proposed in [46] to separate the sub-instances in accordance with the symmetry set S associated to the original instance (see Section 2.8). This strategy is able to create sub-instances with no, or very few, crossing pruning distances. It can however be applied only to instances having nontrivial symmetries (i.e. not only the first symmetry), and it is likely to produce sub-instances of different sizes, to be subsequently assigned to processors having the proper power (more powerful processors should take larger instances).

Finally, another parallelization strategy was proposed for all DDGP instances (satisfying or not the consecutivity assumption) in [60]. Weaker discretization assumptions allow us to reorder the vertices forming every sub-instance in a way to reduce the number of pruning distances that cross over the sub-instances. The problem of efficiently partitioning the vertices of G in sub-instances was however not treated yet for the general case.

2.10 MD-jeep

Different implementations of the BP algorithms have been used for performing computational experiments, sometimes with the idea to verify experimentally a new introduced feature. The basic implementation, in C programming language, that better reflects the description of the algorithm in this chapter, is given by the software MD-jeep[1] that is distributed under the GNU General Public Licence (v.2). A description of the main features of the release 0.1 of MD-jeep can be found in [131].

The BP algorithm was successfully applied for solving DDGP instances, and particularly for instances of the protein structure determination problem (see Section 1.3.1) and of dimensionality reduction (see Section 1.3.2). Many instances of these two problems have been artificially generated, with the aim to control some parameters and verify the robustness of the algorithm with respect to different values for such parameters. Some experiments related to protein conformations were presented, for example, in [58, 85, 87, 89, 121, 124]. We worked with artificially generated instances of dimensionality reduction in [61].

Moreover, MD-jeep was employed for solving *genuine* protein instances, where the set of distances was deduced from NMR experiments, in [23, 130]. In [85], MD-jeep was compared against DGSOL [112], and against an SDP-based facial reduction method [79] (see Section 1.2). MD-jeep was also compared to an implementation of the symBP algorithm in [123, 125] (see Section 2.8.1), and it was compared to an extended version of the BP algorithm, where a set of pruning distances is relaxed to avoid drastic pruning, in [58]. Finally, in [56], when working on the biological application, the BP algorithm was integrated with new energy-based pruning devices. The algorithm variant in [126] is a preliminary step in adapting the BP algorithm for a much more challenging problem in biology, known as *protein folding*, where the distance information is replaced by energy information.

2.11 An Open Problem: Dealing with Uncertainty

There are two main sources of uncertainty that can spoil the results obtained by the BP algorithm. First of all, one source of uncertainty is given by the possible presence, in a given DGP instance, of wrong distance assignments. For example, in the application related to protein conformations (see Section 1.3.1), some wrong assignments of the available distances may be performed (this problem is discussed in Section 1.4).

Moreover, the BP algorithm described in Section 2.4, even when the method in Section 2.6.3 is implemented for improving the branching phase of the algorithm, is basically a heuristic when some discretization distances belong $E \setminus E'$. In fact, even with large values for the discretization factor D, there is no guarantee to select the sample positions from the arcs that will be feasible w.r.t. the overall set of pruning distances.

[1] http://www.antoniomucherino.it/en/mdjeep.php

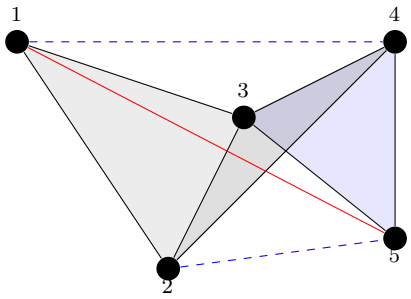


Figure 2.6: *An example of realization of five points in* $\mathbb{R}^3$.

The main issue comes from the way sample positions are extracted from the arcs obtained during the intersections [58]. In fact, these positions are selected independently at each layer of the tree and, particularly for small D values, it is *not* likely that they are compatible with each other and with other pruning distances available at further layers. Suppose v is the current vertex, and that the interval distance $[\underline{\delta}_{c,v}, \bar{\delta}_{c,v}]$ is used in the discretization. Even if we assume that there exists a distance value $\delta^*_{c,v} \in [\underline{\delta}_{c,v}, \bar{\delta}_{c,v}]$ which is compatible with all other distances in E, we cannot ensure that a finite number D of samples extracted from $[\underline{\delta}_{c,v}, \bar{\delta}_{c,v}]$ is sufficient to sample the distance $\delta^*_{c,v}$.

In order to illustrate this fact, consider the following example where five points in $\mathbb{R}^3$ are realized (Fig. 2.6). Suppose that the straight lines represent exact distances, and let the black lines be the exact discretization distances. Moreover, the dashed blue lines are the interval distances (used to compute the possible positions of vertices 4 and 5) and the red straight line represents the only pruning distance (that can be used to validate the computed positions for vertex 5). The associated distances are the following: $\delta_{1,2} = \delta_{2,3} = \delta_{2,4} = \delta_{3,4} = \delta_{3,5} = \delta_{4,5} = 1$, $\delta_{1,3} = \sqrt{2}$, $\delta_{1,4} = \sqrt{x} \in [0.5, 2]$, $\delta_{1,5} = \sqrt{3}$, $\delta_{2,5} = \sqrt{y} \in [0.5, 2]$.

According to the Cayley-Menger conditions [96, 147], for this set of distances to be realizable in $\mathbb{R}^3$, it is necessary that

$$\begin{vmatrix} 0 & 1 & 2 & x & 3 & 1 \\ 1 & 0 & 1 & 1 & y & 1 \\ 2 & 1 & 0 & 1 & 1 & 1 \\ x & 1 & 1 & 0 & 1 & 1 \\ 3 & y & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{vmatrix} = 0,$$

where the above matrix is a bordered distance matrix and $|\cdot|$ denotes its determinant. The solution set of this equation (the values for the missing distances $x = \delta^2_{1,4}$ and $y = \delta^2_{2,5}$) is represented by the blue curve in Fig. 2.7. It is easy to see that, unless the grid is sufficient refined (number of samples D is sufficient large), a valid pair of distances $(\delta^2_{1,4}, \delta^2_{2,5})$ can be sampled with probability 0.

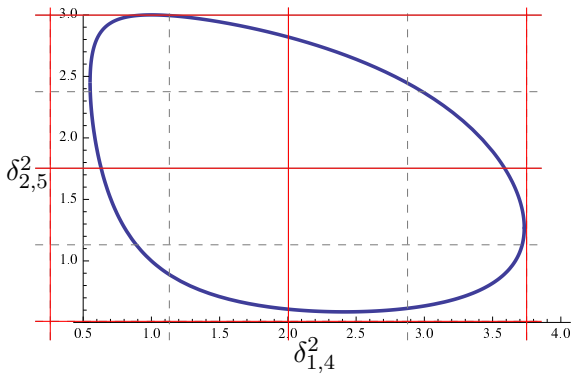


Figure 2.7: *Solution set for the five point Cayley-Menger determinant with* $\delta^2_{1,4}$ *and* $\delta^2_{2,5}$ *as missing distances.*

One possible extension for the BP algorithm for dealing with this kind of situations is given in [132]. When the feasibility check does not allow the algorithm to pass a certain tree layer, the idea is to grant the algorithm the permission to continue the exploration of the current branch, because the infeasibility may be the consequence of what discussed above. If this is the case, the algorithm should therefore be able to construct the rest of the solutions while carrying a relatively small error. If not, then other infeasibilities will be found at subsequent layers: after a predefined number of constraint violations, the algorithm finally prunes the current branch.

This extension of the BP algorithm was tested on instances such that $E_d \subset E'$, and with rather low percentage of introduced errors. In fact, when the number of allowed constraint violations is larger, the search tree $\mathcal{T}$ can experience an important growth, so that the computational cost is drastically increased.

Future works will be devoted to novel strategies for the efficient use of the vertex positions included in the arcs obtained by the sphere intersections. An adaptation of BP to the one-dimensional case was recently published in [117], which keeps its deterministic side even when working with interval distances. The extension of this result to the general case is not however straightforward: some initial results aiming at dealing with the general case were recently published in [53, 58, 103, 149].

Moreover, a new research direction consists in using continuous solvers for global optimization for *correcting* the partial solutions that, generated at a certain layer of the tree, do not satisfy all distance constraints. This idea is motivated by the fact that, at every layer of the search tree where the pruning devices may fail, the partial solution can be a quite good approximation of a valid (partial) solution. In fact, all distances are satisfied by the partial solution, except a small subset of distances: the pruning distances for which DDF fails. Continuous optimization solvers can therefore come to help for correcting these partial solutions *before* exploring subsequent layers of the tree.

Chapter 3

Discretization Orders

3.1 Making Order

Let $G = (V, E, d)$ be a simple weighted undirected graph representing an instance of the DGP (see Definitions 1.1.2 and 1.1.3), in any dimension $K > 0$. Recall that the vertex set V represents a set of objects, and that the edges in E indicate whether the distance between two vertices u and $v \in V$ is known or not. The function d associates the numerical value of the distance to a given pair of vertices; this value can be either exact, approximated, or represented by a real-valued interval. Moreover, d can also provide information about the priority level of every available distance (see Section 1.1, Def. 1.1.3). Recall that E' is the subset of E related to the exact distances in G (or equivalently, the set of distances having priority 1). Recall that E_d is the set of discretization distances in G, while $E_p = E \setminus E_d$ is the set of pruning distances.

We will begin our discussion by introducing some definitions, by following [55]. Let $\mathbb{P}(V)$ be the powerset of V.

Definition 3.1.1 *A covering sequence*

$$r : \mathbb{N}_+ \longrightarrow \mathbb{P}(V)$$

of length p on V, denoted by $(r(1), \ldots, r(p))$, is an ordered set of non-empty subsets $r_i \subseteq V$, ranked by $1, 2, \ldots, p$, such that

$$\bigcup_{i=1}^{p} r_i = V. \tag{3.1}$$

The condition in equ. (3.1) in Def. 3.1.1 ensures that every vertex $v \in V$ belongs to at least one subset $r(i)$ of the sequence, with $i \leq p$. The indices $i \leq p$ are the *ranks* of these subsets where the vertices of V belong to. In this chapter, we will employ the compact notation r_i for $r(i)$.

With a little abuse, we will refer to a sequence r as a *partial vertex order* on V. The length of a partial order r can be either finite or infinite. Even if the infinite

case is not suitable for the applications, this can be a property of orders where vertex repetitions are allowed (see for example [87, 116]). Given a partial order r, the set

$$R = \{u \in V | \exists i \neq j : u \in r_i \cap r_j\}$$

contains all vertices of V that are repeated at least once in the order.

Definition 3.1.2 *A partial vertex order on V "without repetitions" is a covering sequence r such that $R = \emptyset$. Similarly, a partial vertex order on V "with repetitions" is a covering sequence r such that $R \neq \emptyset$.*

As an immediate consequence of its definition, the set R is empty if and only if the intersection of every pair r_i and r_j in the partial order, with $i \neq j$, is empty. This implies the absence of repetitions. Only in this case, the typical properties of partial orders are actually satisfied (see Prop. 3.1.3). This is why calling a sequence r "a partial order on V" is a little abuse. However, we prefer to use the same name (a *partial order*) for both kinds of orders in the following discussion. In a strict mathematical sense, only orders not allowing for vertex repetitions satisfy the reflexivity, the antisymmetry and the transitivity properties.

Proposition 3.1.3 *A partial vertex order on V without repetitions is a "partial order", in a mathematical sense.*

proof. Let us define an order relationship "$\leq$" for the vertices of V that is based on the partial vertex order r. Given u and $v \in V$, we say that $u \leq v$ if there exist two distinct indices i and j, for which $u \in r_i$ and $v \in r_j$, so that $i \leq j$. We say that $u = v$ if $i = j$. We point that the relationship $u = v$ implies that the two vertices belong to the same subset r_i, and not that they are the same vertex.

The reflexivity property for the order relationship comes immediately from the hypothesis that the vertex order admits no repetitions, so that every vertex v can only belong to a unique r_i.

Let us prove the antisymmetry property of the relationship. If we suppose that $u \leq v$ and $v \leq u$, then we have that $i \leq j$ and $j \leq i$. By the (total) order relationship on the ranks of the order, we have therefore that $i = j$. As a consequence, $u = v$.

Finally, we prove that the transitivity property is satisfied. If $u \leq v$ and $v \leq w$, there must exist three indices i, j and k such that $i \leq j$ and $j \leq k$. Therefore, we have $i \leq k$, and so $u \leq w$. □

An order r is obviously not total in general because the same rank can be associated to more than one vertex (belonging to the same subset r_i such that $|r_i| > 1$).

Definition 3.1.4 *A partial order $r : \mathbb{N}_+ \longrightarrow \mathbb{P}(V)$ is total if $|r_i| = 1$ for each $i \in \mathbb{N}_+$.*

Notice that a total order can be either with or without repetitions.

In order to verify the number of reference vertices related to each subset r_i, we introduce the two following subsets of edges:

$$\Lambda_\alpha(r_i, v) = \{(u, v) \in E \,|\, \exists j < i, : u \in r_j\},$$

$$\Lambda_\beta(r_i, v) = \{(v, u) \in E \,|\, \exists j \geq i, : u \in r_j \text{ and } (v, u) \notin \Lambda_\alpha(r_i, v)\}.$$

In the following, we will suppose that $v \in r_i$, but both sets Λ_α and Λ_β can actually be defined even for vertices $v \notin r_i$ (see discussion below). Notice that, when $R \neq \emptyset$, the same edge (u, v) can appear more than once in the definition of Λ_α, because u may be contained in more than one r_i. However, since Λ_α is a set, no redundant distance information is considered. It follows that $\Lambda_\alpha \cap \Lambda_\beta = \emptyset$. The edges in Λ_α are the ones that are supposed to play the role of reference distances in the discretization of the DGP with partial order r.

The same applies to the set Λ_β, with the difference that edges in Λ_β correspond to distances that are to be used as a reference for vertices in further ranks. Edges in Λ_α cannot be contained in Λ_β, because, once used for finding candidate positions for v, they do not bring any additional information when used as a reference from the same vertex v, that appears for the second time at further layers. It is important to notice that, during the execution of an algorithm such as BP, the repetitions of the same vertex can only be placed in the same position of their previous copies. The information about distances concerning the repetitions may actually be exploited for verifying the amplitude of the error propagation during the search (see for example [87]).

Sets similar to Λ_α and Λ_β were already introduced in some publications for similar purposes (see [54] and [114]), but they were defined as subsets of V. The decision, taken more recently in [115] and [55], to work instead on subsets of edges was guided by the fact that it essentially simplifies the theory and the notations.

As remarked above, the sets Λ_α and Λ_β have a different meaning when the vertex v does not belong to r_i. If $v \in r_k$, with $k > i$, then only the reference distances of v until rank i are counted in Λ_α (while the ones having ranks between $i + 1$ and k are omitted); if instead $k < i$, then not all reference distances for v may be counted, because the focus in this case is on the ones having reference vertices with rank smaller than k. Similar observations can be made for Λ_β when $v \notin r_i$.

Based on the above sets of edges, we introduce the following counters:

$$\alpha(r_i) = \min_{v \in r_i} |\Lambda_\alpha(r_i, v)|, \quad \beta(r_i) = \max_{v \in r_i} |\Lambda_\beta(r_i, v)|,$$

$$\alpha_{ex}(r_i) = \min_{v \in r_i} |\Lambda_\alpha(r_i, v) \cap E'|, \quad \beta_{ex}(r_i) = \max_{v \in r_i} |\Lambda_\beta(r_i, v) \cap E'|.$$

By considering the counters defined above, as well as the definition of DDGP (Def. 2.1.1, Section 2.1), we give the following definition.

Definition 3.1.5 *A partial discretization order in dimension K is a partial order $r : \mathbb{N}_+ \longrightarrow \mathbb{P}(V)$ such that:*

(a) $G[r_1, \ldots, r_K] \equiv (C, E_C)$ *is a clique with* $|C| = K$ *and* $E_C \subset E'$*;*

(b) $\forall i \in \{K+1, \ldots, |r|\}$, $\alpha(r_i) \geq K$ *and* $\alpha_{ex}(r_i) \geq K-1$*,*

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

Notice that a partial discretization order can either be with or without repetitions. The *total* discretization order is a special case. In general, therefore, the number of constraints in assumption **(b)** is smaller than $|V| - K$: if the inequalities are satisfied for r_i, they are also satisfied for every $v \in r_i$, because the α and α_{ex} values are minima. When referring in general to partial or total orders, we will simply write "discretization orders".

Proposition 3.1.6 *Necessary condition for* G *to admit a partial discretization order in dimension* K *is that, for every suitable order* r *on* V*,*

$$\forall i \in \{1, 2, \ldots, |r|\}, \quad \alpha(r_i) + \beta(r_i) \geq K.$$

proof. Suppose that there exists a rank $i \in \mathbb{N}_+$, for a certain partial order r, for which $\alpha(r_i) + \beta(r_i) < K$. By definition, there exists $\hat{v} \in r_i$ such that the cardinality of $|\Lambda_\alpha(r_i, \hat{v})|$ is minimal and equal to $\alpha(r_i)$. Since $\beta(r_i)$ is instead a maximal cardinality, $|\Lambda_\beta(r_i, \hat{v})|$ is at most equal to $\beta(r_i)$. Therefore, for the vertex $\hat{v}$, $|\Lambda_\alpha(r_i, \hat{v})| + |\Lambda_\beta(r_i, \hat{v})| < K$, which implies the absence of a sufficient number of edges for this vertex for constructing a discretization order (i.e. no r_i containing $\hat{v}$ can satisfy the two discretization assumptions). □

Notice that a similar necessary condition is related to the counters $\alpha_{ex}(r_i)$ and $\beta_{ex}(r_i)$. The condition

$$\forall i \in \{1, 2, \ldots, |r|\}, \quad \alpha_{ex}(r_i) + \beta_{ex}(r_i) \geq K - 1,$$

is a necessary condition for the orders r to be partial discretization orders.

Every set r_i defines a subgraph of G: let E_i be the edge set of this induced subgraph. E_i can be either empty, or contain some edges, or define a fully connected graph. E_i can contain either distances belonging to E' or $E \setminus E'$. Notice however that, when E_i contains edges, those edges are not supposed to belong to E_d: the discretization distances necessary to these vertices in r_i are related to vertices in r_j's, with $j < i$, whereas discretization distances having the vertices in r_i as a reference are supposed to concern vertices of r_j, with $j > i$. Every internal order for the vertices of each r_i is admissible.

The consecutivity assumption on partial discretization orders allows to have additional and important properties on DGP trees $\mathcal{T}$ obtained with the discretization process (see Section 2.3). For example, the consecutivity assumption allows to verify in advance whether the discretization distances in the set $\{u_1, u_2, \ldots, u_K\}$, for every $v \in r_i$, are compatible to each other before running a solution method such as the BP algorithm. Moreover, it was proved that DGP search trees related to discretization orders with consecutivity assumption are symmetric (see Section 2.8). Orders

satisfying the consecutivity assumption should therefore be preferred in general, but the problem of automatically detecting such orders is NP-hard [24]. We provide in Section 3.2 some tools for aiding the identification of discretization orders with consecutivity assumption; some of such orders were handcrafted and are presented in Section 3.3. The automatic tools presented in Sections 3.5 and 3.6 consider instead vertex orders that do not satisfy the consecutivity assumption.

3.2 Pseudo de Bruijn Graphs

Graphs of *de Bruijn* [32] are generally employed for formalizing problems of DNA assembly [26, 44]. We propose the use of a graph B which is an extension of the classical de Bruijn graph. If G represents a DGP instance, the vertices of our *pseudo* de Bruijn graph $B = (V_B, A_B)$ are $(K+1)$-cliques of the graph G, where K is the dimension. A vertex $b \in V_B$ can be seen as a subsequence of $K+1$ vertices of G that form a clique, with a variable internal ordering.

In the standard de Bruijn graph, there is an arc from b to c if they admit an overlap. More precisely, if the ending of the subsequence b coincides with the beginning of the subsequence c, then the arc (b, c) is added to the arc set A_B. Since the vertices in V_B cannot be considered as static objects (the internal order of the vertices is not a constant), it was necessary to extend the standard definition of de Bruijn graph for the purposes of our studies. Consider for example that $c \in V_B$ is a $(K+1)$-clique composed by edges of E': in this case, the $K+1$ vertices in the clique can be reordered $(K+1)!$ times. If instead $b \in V_B$ contains an edge of $E \setminus E'$, there are $2(K-1)!$ permutations of the vertices that allow the extremes of the interval distance to be the first and the last vertex in the clique. When working with discretization orders, every overlap needs to have length equal to K.

Definition 3.2.1 *There is a K-overlap from the vertices b to the vertex c of V_B if there exists an internal order for the vertices in b and an internal order for the vertices in c for which the K-suffix of b coincides with the K-prefix of c.*

This definition applies to any kind of clique (either consisting of exact distances, or containing interval data).

Definition 3.2.2 *A K-valid path $P = \{p_1, p_2, \ldots, p_n\}$ on B is a sequence without repetitions of K-overlapping cliques p_i where the internal order of each clique is preserved when referring to the neighboring p_{i-1} and p_{i+1}, and every $u \in V$ is included in at least one clique p_i.*

A K-valid path on B naturally implies the definition of a partial discretization order. Alg. 3 is a sketch of a simple algorithm that can be executed for constructing an order r from a K-valid path P. The algorithm assigns the first K labels to the vertices in $p_1 \in P$ (the internal order of the clique must be preserved in this case). Then, for all other p_j, with $j \geq 2$, the last vertex of the clique p_j, in the internal order, is added to the vertex order.

Algorithm 3 Constructing an order r from a K-valid path on B

```
1: find_induced_order (P)
2: i = 1
3: for all u ∈ p_1 in the internal order do
4:    r_i = u; i++
5: end for
6: for (j = 2, ..., n) do
7:    u = last vertex in internal order of p_j
8:    r_i = u; i++
9: end for
10: return r;
```

Proposition 3.2.3 *Any order r constructed by Alg. 3 from a K-valid path P on B is a discretization order for which the consecutivity assumption is satisfied.*

proof. By construction. □

Notice that the same vertex can appear in more than one clique of G, so that the order induced by a K-valid path P can either be without or with repetitions.

A simple verification for the existence of a K-valid path on B is to check its connectivity. Naturally, if B is not connected, no valid paths can be constructed. But even when B is connected, a K-valid path on B may not exist. This is the case for example for the protein backbone [105]. To overcome this issue, *auxiliary cliques* can be included in B. When dealing with protein backbones, the identification of a K-valid path on B is in fact only possible when auxiliary cliques are included in the pseudo de Bruijn graph B.

Definition 3.2.4 *An auxiliary $(K+1)$-clique is a clique*

$$\{v_1, v_2, \ldots, v_K, v_1\}$$

where $\{v_1, v_2, \ldots, v_K\}$ is a K-clique of V having edges in E'.

It is important to remark that several auxiliary cliques can be generated from one K-clique, depending on the selected internal vertex order. The set of vertices $\{v_1, v_2, \ldots, v_K, v_1\}$ evidently form a clique, because the distances between the duplicated v_1 and all other vertices are known. Moreover, the distance between the first and second copy of v_1 is exact and equal to 0.

Auxiliary cliques allow for locally reordering a given subset of vertices, so that a K-overlap can become possible with other cliques. Every time an auxiliary clique is involved, a vertex is repeated in the sequence, exactly K places after its previous copy.

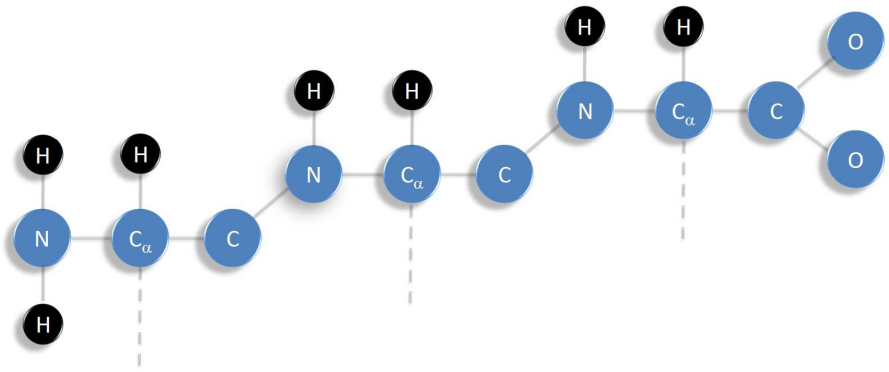


Figure 3.1: *The chemical structure of the considered 3-amino acid protein backbone. Some atoms are omitted because they can be positioned uniquely once the considered atoms have been placed. Side chains may be attached to the atoms C_α through the bonds represented by the dashed gray lines.*

3.3 Handcrafted Orders for Protein Backbones

We will consider a small polypeptide model composed by 3 amino acids, representing the first, the generic and the last amino acid of a protein sequence (see Fig. 3.1). For every chemical bond (light gray lines in the picture), there is a known exact distance that can be considered for the discretization. Moreover, the relative distance between atoms bonded to another common atom is known, and can also be considered as exact. Finally, every quadruplet of consecutive bonded atoms form a torsion angle, from which a lower and an upper bound can be obtained for the distance between the first and the last atom of the quadruplet. Since peptide bonds, which chemically connect consecutive amino acids, give a rigid configuration to a part of the backbone structure, some of the distances derived from torsion angles can also be considered as exact [105].

Table 3.1 shows the (non-auxiliary) cliques that can be found in the 3-amino acid backbone. Only information deduced from its chemical structure are considered in the table: the distances derived from NMR experiments (see Section 1.3.1) are not considered. In fact, the interest is in finding orders that are suitable for every protein backbone, so that only instance-independent distances are used for defining the 4-cliques of the pseudo de Bruijn graph B.

A discretization order for the protein backbones was proposed in [87]. This order was handcrafted and satisfies the consecutivity assumption. In terms of pseudo de Bruijn graph, the handcrafted order corresponds to the 3-valid path in dimension 3 in Fig. 3.2. Notice that the two hydrogens bonded to the nitrogen atom N^1 of the first amino acid, as well as the two oxygens bonded to the carbon C^3 of the last amino acid, are here omitted. In fact, positions for these atoms can be calculated at the end of the computation, when a position has already been assigned to all other

name	*atoms*				*edge* $\{r_{i-3}, r_i\}$
c_1	N^1	C^1_α	H^1_α	C^1	exact
c_2	H^1_α	C^1_α	C^1	N^2	interval
c_3	C^1_α	C^1	N^2	H^2	exact
c_4	C^1_α	C^1	N^2	C^2_α	exact
c_5	C^1	N^2	H^2	C^2_α	exact
c_6	H^2	N^2	C^2_α	H^2_α	interval

name	*atoms*				*edge* $\{r_{i-3}, r_i\}$
c_7	N^2	C^2_α	H^2_α	C^2	exact
c_8	H^2_α	C^2_α	C^2	N^3	interval
c_9	C^2_α	C^2	N^3	H^3	exact
c_{10}	C^2_α	C^2	N^3	C^3_α	exact
c_{11}	C^2	N^3	H^3	C^3_α	exact
c_{12}	H^3	N^3	C^3_α	H^3_α	interval
c_{13}	N^3	C^3_α	H^3_α	C^3	exact

Table 3.1: *4-cliques contained in the graph representing an instance related to a 3-amino acid backbone. Auxiliary cliques are not reported.*

$$
\begin{aligned}
&\textit{(first amino acid)} && \diamondsuit \to c_1 \to c_2 \\
&\textit{(second amino acid)} \quad \to && c_4 \to c_5 \to \diamondsuit \to \diamondsuit \to c_6 \to c_7 \to \diamondsuit \to c_8 \to \diamondsuit \\
&\textit{(third amino acid)} \quad \to && c_{10} \to c_{11} \to \diamondsuit \to \diamondsuit \to c_{12} \to c_{13} \ .
\end{aligned}
$$

Figure 3.2: *A handcrafted path for the protein backbones. The symbol* $\diamondsuit$ *indicates that an auxiliary clique is used in the order.*

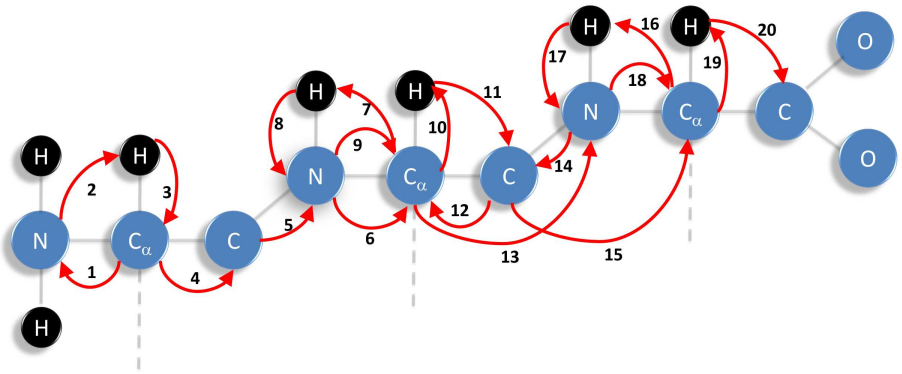


Figure 3.3: *An handcrafted discretization order with repetitions for protein backbones.*

atoms. The de Bruijn graph representation of the handcrafted order starts with the auxiliary clique (C^1_α,N^1,H^1,C^1_α). There are 7 auxiliary cliques; in general, for a protein backbone consisting of n_{aa} amino acids, $1 + 4 \cdot (n_{aa} - 2) + 2$ auxiliary cliques are necessary for constructing this path. Notice that the second amino acid can be repeated as many times as necessary in a protein backbone formed by $n_{aa} > 3$ amino acids. This order is depicted in Fig. 3.3.

In [87], we have analyzed in details the behavior of the BP algorithm when working with this handcrafted order for protein backbones. We consider a very simple instance having the main structure of the model in Fig. 3.3. The weight

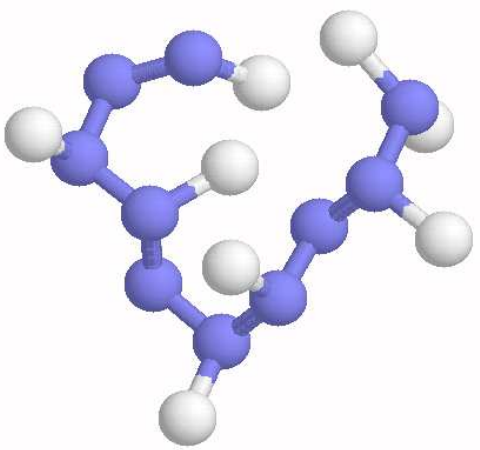


Figure 3.4: *A simple test instance.*

function d has no priorities π associated to the distances, and is constructed as follows. Given four real numbers a, b, $\underline{c}$ and $\bar{c}$ such that $2a > b$ and $\underline{c} < \bar{c}$, the distance between every pair $\{u, v\}$ of bonded atoms is $\delta_{u,v} = a$; the distance between every pair $\{u, v\}$ of atoms two covalent bonds apart is $\delta_{u,v} = b$. The distance function d maps every pair $\{u, v\}$ of atoms three covalent bonds apart to a discrete set of D values in the interval $[\underline{c}, \bar{c}]$. As is, the obtained instance is discretizable and has no edges in E_p. Thanks to the discretization, the search domain of such an instance is a tree, where paths from the root to the leaf node represent solutions. With an empty E_p, the solution set corresponds to this entire search domain. Therefore, we randomly choose a path in the search domain (by selecting one if its leaf nodes), and we derived additional interval distances from it:

$$[\delta_{u,v} - \varepsilon, \delta_{u,v} + \varepsilon],$$

where ε is a small positive constant. In order to simulate NMR data, we only considered distances smaller than 5Å, that we have included in the set E_p. Fig. 3.4 shows the three-dimensional conformation related to the generated test instance, whose distances were generated with $\varepsilon = 0.3$.

Fig. 3.5 shows the search tree $\mathcal{T}$ related to this instance, which can be explored by the BP algorithm. The positions of the first three atoms can be obtained by applying the method in Section 2.5. Branching starts at level 4, in correspondence with the atom C^1_α. Due to the first symmetry (see Section 2.8), we can discard one of the branches at level 4, and focus on only one of them. At level 5, we have the first duplicated atom, the nitrogen N^1 which already appeared at level 1. Therefore, there is no branching, because the new copy of N^1 can only be placed in the same position of its previous copy. The first hydrogen in the vertex order on which we need to branch appears at level 6. This is the hydrogen H^1_α. Since the distance between this atom and the previous H^1 is an interval, we need to discretize the interval and

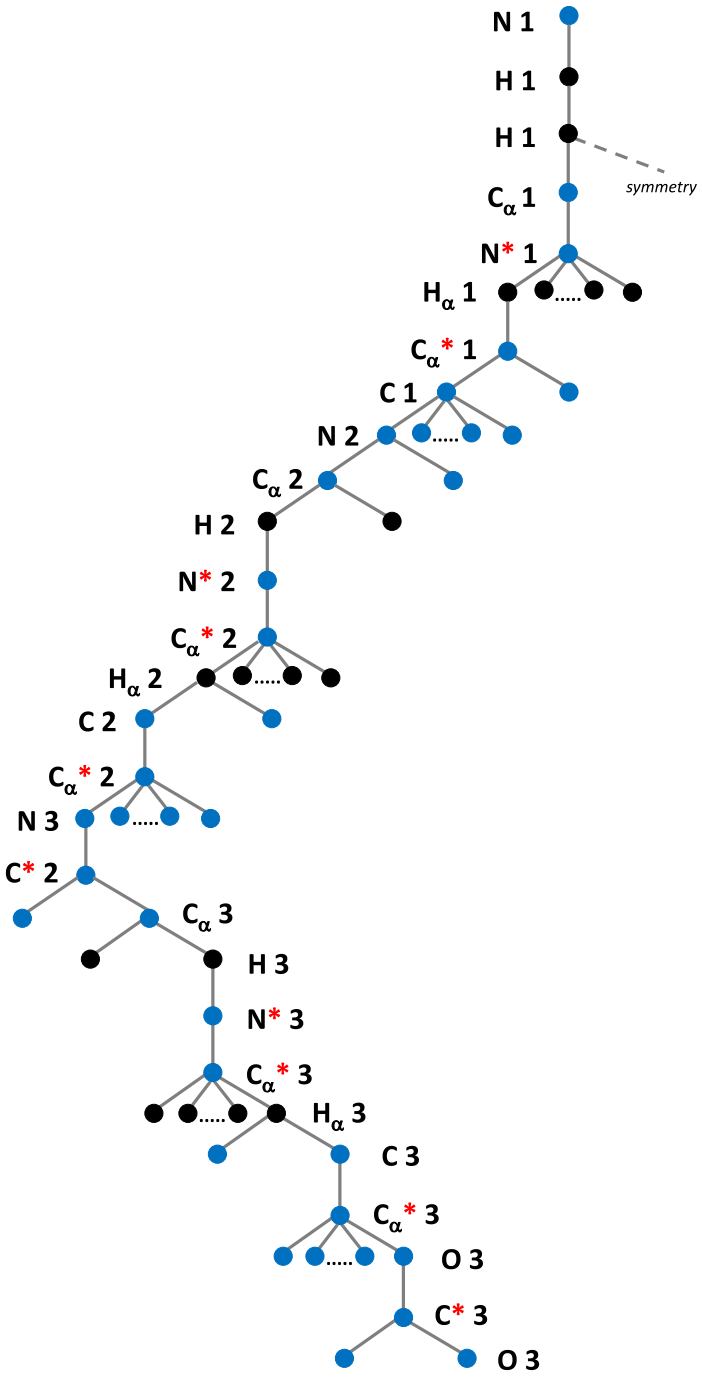


Figure 3.5: *Part of the search tree $\mathcal{T}$ of our small test instance.*

layer	*atom*	*amino acid*	*repeated?*	*branches w/out pruning*	*branches with pruning*
1	N	1	no	1	1
2	H	1	no	1	1
3	H	1	no	1	1
4	C_α	1	no	2	2
5	N	1	yes	2	2
6	H_α	1	no	24	**18**
7	C_α	1	yes	24	18
8	C	1	no	48	36
9	N	2	no	576	360
10	C_α	2	no	1152	720
11	H	2	no	2304	**10**
12	N	2	yes	2304	10
13	C_α	2	yes	2304	10
14	H_α	2	no	27648	**70**
15	C	2	no	55296	140
16	C_α	2	yes	55296	140
17	N	3	no	663552	1400
18	C	2	yes	663552	1400
19	C_α	3	no	1327104	2800
20	H	3	no	2654208	**4**
21	N	3	yes	2654208	4
22	C_α	3	yes	2654208	4
23	H_α	3	no	31850496	**9**
24	C	3	no	63700992	18
25	C_α	3	yes	63700992	18
26	O	3	no	764411904	52
27	C	3	yes	764411904	52
28	O	3	no	1528823808	**10**

Table 3.2: *The number of branches, step by step, at the different layers of the search tree $\mathcal{T}$, with and without pruning. In bold face, the number of branches potentially subject to be reduced by the use of pruning devices.*

take from it D exact distances (see Section 2.4). As a consequence, $2D$ branches are added at level 6 of the binary tree. At level 7, we find another duplicated atom, and therefore, there is no branching. After this atom, we have a sequence of 3 atoms that are neither duplicated nor hydrogens: depending on the fact that an interval needs to be discretized or not, only two or $2D$ branches are added to the tree. The first hydrogen of the second amino acid is at level 11. Since the distance between C^1 and H^2 is in E', we have only two branches. The other cases are similar to the ones already discussed.

Table 3.2 provides the number of branches at each layer of the tree that we obtained during our experiments. The last but first column of Table 3.2 shows the number of branches of the full tree, where no pruning is applied. Without pruning, the full tree has 1528823808 nodes at level 28. The last column shows the

$$
\begin{array}{rcl}
\textit{(first amino acid)} & & c_1 \to c_2 \\
\textit{(second amino acid)} & \to & c_3 \to c_5 \to \diamondsuit \to c_6 \to \diamondsuit \to c_7 \to c_8 \\
\textit{(third amino acid)} & \to & c_9 \to c_{11} \to \diamondsuit \to c_{12} \to \diamondsuit \to c_{13}\ .
\end{array}
$$

Figure 3.6: *A de Bruijn guided path for protein backbones. The symbol $\diamondsuit$ indicates that an auxiliary clique is used in the order.*

corresponding values when pruning is applied (the discretization factor D is set to 5). We point out that, in conjunction with the DDF pruning device (see Section 2.7.1), we employed the vdW pruning device (see Section 2.7.3), with a very safe threshold for all minimal vdW radii (0.5Å).

3.4 An Optimal Order with Repetitions

Fig. 3.6 shows another possible path for the 3-amino acid backbone depicted in Fig. 3.1. There are two auxiliary cliques in second amino acid, and other two auxiliary cliques in the third one. As a consequence, two atoms are duplicated in each amino acid in the corresponding induced order. In general, for n_{aa} amino acids, $2 \cdot (n_{aa} - 1)$ repetitions are necessary. The internal order of the starting clique c_1 is: N^1, H^1, C^1_α, C^1. Naturally, this is only one possible path that can be identified on the pseudo de Bruijn graph B.

The path in Fig. 3.6 requires fewer auxiliary cliques than the one in Fig. 3.2. In order to verify whether there are other possible paths for which the number of necessary auxiliary cliques is smaller (implying therefore fewer repetitions), one could attempt the construction of all possible 3-valid paths on the graph B by an exhaustive search. Naturally, even if an exhaustive search might be feasible for small instances, this is not an advisable procedure. For the considered 3-amino acid backbone, it is possible to prove that the discretization order induced by the path in Fig. 3.6 is optimal in terms of length.

Theorem 3.4.1 *Let G be a graph representing a DGP instance related to a protein backbone. For every amino acid in the protein backbone with rank greater than 2, every discretization order for its atoms requires at least 2 repetitions.*

proof. In a path starting with c_2 (see Table 3.1), the 4^{th} place in the induced order can be either for H^1_α or for N^2, because of the constraint on the internal orders for the interval clique c_2 (refer to Def. 3.2.2). However, in order to construct a path to c_6 (and not to c_1), it is necessary to choose the internal order where N^2 is in position 4. At this point, the clique c_2 admits a 3-overlap with both cliques c_3 and c_4, and whichever the chosen clique is, the clique c_5 can follow either c_3 or c_4. The position of the atom H^2 in the induced order is the 5^{th} (when c_3 is chosen) or the 6^{th} (when c_4 is chosen). In order to add c_6 immediately after c_5, the atom H^2 should be instead in position 4, which is taken by N^2. However, the position 4 was fixed by c_2 at the

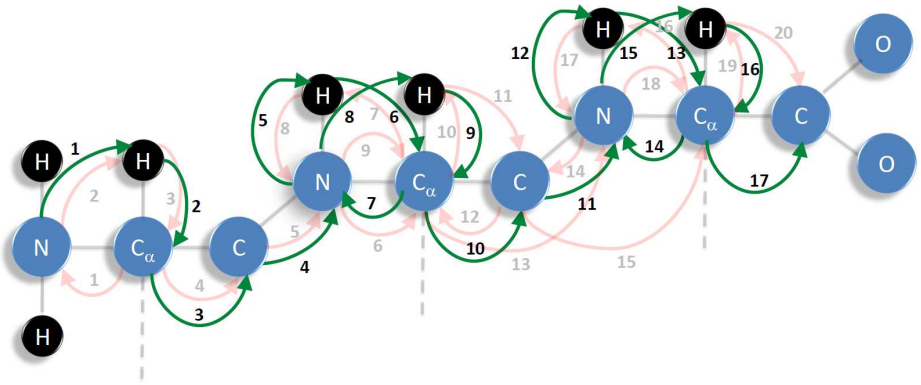


Figure 3.7: *An optimal (in terms of length) discretization order for the protein backbone (in green). In the background, in light red, a previously proposed handcrafted order.*

beginning of the path. An auxiliary clique is therefore necessary for adjusting the internal order of c_5 and for making it possible to have a 3-overlap with the clique c_6. Naturally, the use of an auxiliary clique before c_6 might be avoided if a different path is rather constructed, where auxiliary cliques need to be however involved earlier. This implies that at least one auxiliary clique is necessary for constructing a path on B from c_2 to c_6.

Similarly, it is possible to prove that at least one auxiliary clique is needed to step from the clique c_6 to the clique c_8. Because of the repetitive structure of protein backbones, the theorem is proved. □

Fig. 3.7 graphically shows the discretization order induced by the de Bruijn guided path, in green. Since this path is basically a sequence of 15 cliques, $4 + 14$ atoms (all atoms contained in the initial clique, plus one atom for all others) are included in this order (repetitions are also counted). Fig. 3.7 also shows the order presented in Section 3.3, in light red, to facilitate comparison. In the previous order, there are more repetitions: there are 18 cliques in total, and therefore there are $4 + 17$ atoms in the induced order. The order induced by the de Bruijn guided path is optimal, as Theorem 3.4.1 shows.

We point out that, in [28, 29], discretization orders were handcrafted for the 20 amino acids that are involved in the protein synthesis. However, the optimality of such orders, in the sense given above, was not verified yet.

3.5 Constructing Minimal-Rank Discretization Orders

In the next sections, we will focus on partial discretization orders for which the consecutivity assumption is not satisfied. Moreover, we will restrict our discussion

to orders without repetitions, because the use of repetitions is not strictly necessary when the consecutivity assumption is not imposed.

Proposition 3.5.1 *For every discretization order with repetitions, there exists a discretization order without repetitions.*

proof. Suppose that r is a discretization order with repetitions, where the vertex v appears twice. Let j' and j'' be the two indices for which $v \in r_{j'} \cap r_{j''}$, with $j' < j''$. For every $i > j''$, the reference distances for the vertices in r_i may therefore be related to both $r_{j'}$ and $r_{j''}$. The contribution to the counter $\alpha(r_i)$ is however 1 and not 2, because the two distances (edges) coincide. As a consequence, if v is removed from $r_{j''}$, another valid order is obtained, where there are no repetitions of the vertex v. By iterating this procedure for all repeated vertices v, one can construct a discretization order without repetitions. □

Notice that the method used in the proof of Prop. 3.5.1 can actually be used for transforming a discretization order with repetitions in another order without repetitions. However, the consecutivity assumption, which may be satisfied by the order with repetitions, is likely to be lost during this transformation.

Our starting point for an automatic detection of discretization orders (without repetitions, without consecutivity assumption) will be an extension of the algorithm that was initially proposed in [83], successively tailored to interval distances in [114], and more recently modified for dealing with partial orders in [115]. A sketch of this algorithm is in Alg. 4. The basic idea is to initially identify a K-clique from G and, for each of them, to attempt the construction of a partial discretization order having this initial clique. Alg. 4 makes use of the function `filter` (see Alg. 5), which is able to select, for every rank i, all vertices for which the discretization assumptions in Def. 3.1.5 are satisfied. Only vertices that do not appear already in the ordering (i.e. having already rank $j < i$) are here considered, for avoiding repetitions. In a first analysis, let us consider that the function `optimize` in Alg. 4 simply returns its input argument.

The orders r that are constructed by applying Alg. 4 have the property that every generic set r_i contains the vertices that could not be considered for inclusion in any r_j, with $j < i$, because otherwise the discretization assumptions would not be satisfied. The internal order for the vertices in every 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 some total orders which are not compatible with r are also discretization orders. The fact that a certain vertex v belongs to r_i suggests that it can be included in no r_j's with $j < i$, but this vertex may be placed in sets r_j with $j > i$, as far as it is not a *necessary reference* in between. Therefore, orders that differ from the original r can be constructed by moving vertices from their original subset r_i to other suitable subsets r_j, with $j > i$.

Algorithm 4 An algorithm for finding minimal-rank partial discretization orders

find minimal-rank discretization orders (G)
// initial clique
choose a K-clique (C, E_C) of G such that $E_C \subset E'$
set $r_1 = \{u_1, u_2, \ldots, u_K\}$;
set $A = V \setminus C$
set $i = K + 1$
// constructing the rest of the order
while $(A \neq \emptyset)$ **do**
  $r_i = \mathrm{A}$;
  $r_i = \texttt{filter}(r_i)$;
  if $(r_i = \emptyset)$ **then**
    break: no possible orders; **choose** another initial clique
  else
    $r_i = \texttt{optimize}(r_i)$;
    let $A = A \setminus r_i$
    let $i = i + 1$
  end if
end while
return r;

Algorithm 5 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 $u = \arg\min_{v \in r_i} |\Lambda_\alpha(r_i, v)|$
end while
while $(\alpha_{ex}(r_i) < K - 1$ and $r_i \neq \emptyset)$ **do**
  $r_i = r_i \setminus \{u\}$, where $u = \arg\min_{v \in r_i} |\Lambda_\alpha(r_i, v) \cap E'|$
end while
return r_i

Given the initial clique C, Alg. 4 firstly 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 in A, the ones that are "allowed" to be in r_i. This selection is performed by the routine `filter` which verifies, for every $v \in A$, if 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 turn, that there is no partial order for the initial clique C satisfying assumptions **(a)** and **(b)** of Def. 3.1.5. We give the following characterization of the discretization orders that can be obtained by Alg. 4.

Proposition 3.5.2 *Let G be a simple weighted undirected graph representing a DGP instance, and let C be a K-clique of G. When it exists, the partial discretization order r for G obtained by Alg. 4, having C as initial clique, has the following properties:*

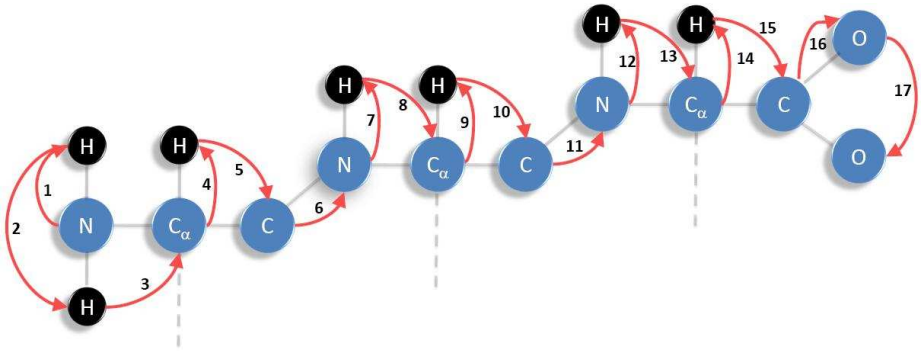


Figure 3.8: *An order for the discretization of protein backbones found by Alg. 4.*

1. $\forall i \in \{2, \ldots, |r|\}$, *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. For a given r_i, if $v \in A$ is not returned by `filter`, it 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 (with $j < i$). In other words, r_i is the first possible rank for v.

Let us suppose, by contradiction, that there exists an order r_{short} that is shorter in length than the order r found by Alg. 4. In this case, the vertices belonging to the last set r_p, with $p = |r|$, are moved in r_{short} in subsets $\hat{r}_j$ having a smaller rank. Therefore, for the property 1 in this same proposition, $\alpha(\hat{r}_j) < K$ or $\alpha_{ex}(\hat{r}_j) < K-1$. As a consequence, the discretization assumptions cannot be satisfied by the order r_{short}. Contradiction. □

Fig. 3.8 shows an order found by Alg. 4 for our small protein backbone model composed by 3 amino acids. For every vertex r_i in the order with $i > 3$, $\alpha(r_i) \geq 3$ and $\alpha_{ex}(r_i) \geq 2$. Therefore, the set of Euclidean objects to be intersected in BP is always given by spheres and, at most, one spherical shell. Notice that, depending on the considered atoms, different sets of reference distances can be available. In the second amino acid, for example, the atom N has two exact reference distances (the preceding atoms C_α and C), and one reference is an interval (the preceding atom H). In this case, the 3 atoms are the ones that immediately precede N. Moreover, supposing that N has index i in the order, the interval distance is related to the edge $\{r_{i-2}, r_i\}$. For the atom H in the second amino acid, instead, the edges that contribute in α are, from the closest to the farthest, $\{r_{i-1}, r_i\}$, $\{r_{i-2}, r_i\}$, and $\{r_{i-4}, r_i\}$, which is an interval.

The order in Fig. 3.8 was obtained from the initial clique (N,H,H). If other cliques are considered, other possible orders can be identified by the algorithm. An interesting example is given by the initial clique (O,O,C), formed by the last (in the

natural ordering) three atoms of the considered protein backbone. In such a case, the algorithm finds the reverse of the order in Fig. 3.8.

3.6 Searching for Optimal Orders

In Section 3.5, a very simple, yet efficient, algorithm for the identification of minimal-rank partial discretization orders is presented. Alg. 4, with the function `filter` in Alg. 5 and an initial clique, gives as a result one partial order, from which total orders can be constructed. As already remarked in Section 3.5, several different total orders can be obtained in practice from one given partial order. In this section, our aim is to develop methods 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 DGP search trees $\mathcal{T}$, for its exploration to be more efficient. We point out that the idea of optimizing the set of objectives during the search of discretization orders was initially proposed in [59].

Once an initial clique has been defined (see Alg. 4), for every vertex v that does not belong to this clique, the dependency of v on other vertices can be analyzed. 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. Moreover, all vertices with a greater rank need to be considered after v. Only vertices that share the same rank with v can appear either before or after v in the order.

The discretization order that Alg. 4 is able to construct is therefore subject to quite strong constraints. All sets r_i containing only one vertex v impose a relative positioning for v in all possible orders, while sets r_i with a higher cardinality contain vertices whose internal ordering can be modified. The idea is to obtain optimal internal orderings for these sets so that our considered discretization orders become optimal, in the sense described below.

Alg. 4 is able to provide us with an initial partial order. Every partial order compatible with the initial minimal-rank discretization order can be obtained by moving vertices in close r_i sets, and/or by splitting sets r_i in smaller sets. Prop. 3.5.2 shows that, in practice, these initial orders are such that every vertex v is always included in the first set r_i where it satisfies the discretization assumptions. Moving a vertex v to subsequent ranks (when possible) 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 sets, can produce ordering having instead a longer length. Both operations (moving vertices and splitting sets r_i) can be performed by *optimizing* the sets r_i.

The objectives over partial orders r that we consider are simple functions, i.e. they have the form

$$f(A) = \sum_v a_v \mathbf{1}_A(v),$$

where a_v is a real coefficient and $\mathbf{1}_A(v)$ is an indicator function over a finite set A,

defined as

$$\mathbf{1}_A : \begin{cases} 1 & \text{if } v \in A; \\ 0 & \text{if } v \notin A. \end{cases}$$

Consequence of the fact that f is simple is that there exists a unique subset of A for which it is maximized (or minimized). We give the following definition.

Definition 3.6.1 *An objective for optimal orders is a simple function* $f_\ell : \mathbb{P}(V) \setminus \{\emptyset\} \longrightarrow \mathbb{R}$.

Notice that the restriction to simple functions does not allow us to consider any kind of objective. Let us consider, for example, the following objective:

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

It is clear that f is maximized when A contains only one element, but all subsets of A with cardinality 1 achieve the maximum, and thus there is no uniqueness of the maximizer.

We also consider that more than one objective f_ℓ can be used during the optimization process. In this case, we will assign priorities to the objectives. Henceforth, the subscript $\ell \in \mathbb{N} \cup \{0\}$ 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 3.6.2 *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*

$$\begin{aligned} &\max_{r \subset r_i^{(M)}} f_M(r) \\ &\quad \text{s.t. } r_i^{(M)} = \arg \max_{r \subset r_i^{(M-1)}} f_{M-1}(r) \\ &\qquad \text{s.t} \ldots \\ &\qquad\quad \text{s.t. } r_i^{(2)} = \arg \max_{r \subset r_i^{(1)}} f_1(r), \end{aligned} \tag{3.2}$$

where $r_i^{(1)}$ *is the initial set of vertices admitting rank* i*.*

Multi-level optimization is a class of generally very hard optimization problems [11, 25]. 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 $r_i^{(1)}$, containing all the vertices that are candidates for being placed in the subset r_i. In other words, all the vertices in $r_i^{(1)}$ satisfy the discretization assumptions. In order to solve our simple multi-level optimization problem, the objectives f_ℓ can be optimized one after the other,

Algorithm 6 The function `optimize`

```
1: function optimize(r_i)
2: for (each objective f_ℓ, with ℓ = 1, 2, ..., M) do
3:     r_i = {v ∈ r_i : f_ℓ is optimized}
4: end for
```

by taking into consideration their priority levels. Alg. 6 is a sketch of a function devoted to this task, to be invoked at line 14 of Alg. 4.

To sum up, Alg. 4 can be divided into two main parts. First, a K-clique C of G is identified, that plays 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 is defined. To this purpose, the set is initially filtered by invoking the function `filter`, where all vertices satisfying the discretization assumptions 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 again by invoking the function `optimize`. This functions removes from the set 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 3.6.3 *Let $G = (V, E, d)$ be a simple weighted undirected graph representing an instance of the DGP. When they exist, Alg. 4, used in conjunction with Alg. 5 and Alg. 6, is able to construct partial discretization orders for G, which are optimal.*

proof. The correctness of Alg. 4, using the routine `filter` (Alg. 5), was proved in Prop. 3.1. Since `optimize` (Alg. 6) 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 after application of `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 eligible 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 functions as objectives. □

After the optimization of the objectives, the new set r_i can be included in the ordering, and Alg. 4 can step to the next iteration $i + 1$. As remarked above, all vertices that were removed from r_i by the function `optimize` are candidate vertices for inclusion in further sets r_j, with $j > i$. Therefore, all of them can be selected again by the function `filter`. One immediate 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).

Theorem 3.6.4 *Alg. 4, used in conjunction with Alg. 5 and Alg. 6, has polynomial complexity.*

proof. Line 3 of Alg. 4 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 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 are simple functions: a simple loop over the vertices in r_i is sufficient for identifying the subset of optimal vertices. The complexity of the function `optimize` is therefore $O(nM)$, where M is the total number of objectives. As a consequence, the total complexity of Alg. 4 is polynomial. □

3.7 Objectives to Be Optimized

Some objectives to be optimized during the search for discretization orders are presented in this section. 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 present three objectives that are suitable for all classes of DGPs. They are mostly based on properties of reference distances for the vertices belonging to the vertex subsets obtained at line 10 of Alg. 4. Recall that the function `filter` (see Alg. 5) allows to identify all vertices, that do not appear yet in the order, for which the discretization assumptions in Def. 3.1.5 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 $\{u, v\}$, the difference in ranks for the two vertices u and v.

The fourth objective that we present is instead particularly conceived for the class of DGP related to molecular conformations (see Section 1.3).

3.7.1 Early use of exact distances

The counter $\alpha_{ex}(r_i)$ is able to provide the minimum number in r_i of reference distances that belong to E'. By applying Alg. 5, 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 $K-1$. However, larger values for $\alpha_{ex}(r_i)$ would have a positive impact on the search tree.

Our objective $f_A(r_i)$ is $\alpha_{ex}(r_i)$. In fact, the earlier are the exact distances used for generating the search tree, the smaller is the tree width. 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.

3.7.2 Early pruning

The counter $\alpha(r_i)$ is similar to $\alpha_{ex}(r_i)$, but it considers all distances in E. When the value of $\alpha(r_i)$ is maximized, not only distances necessary for the discretization are available (as ensured by Alg. 5), 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 early layers of the tree, its width can be reduced. This objective was initially considered in [83], and subsequently in [59].

3.7.3 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 the distance $\delta_{v,w}$ "crosses" several other vertices. In this situation, the search over the tree can be rather expensive, because candidate positions for v can be discovered to be infeasible only when reaching the tree layer corresponding to w (in the situation where no position for w can be found such that $\delta_{v,w}$ is satisfied). In order to reduce the impact of this kind of pruning distances on the search, we consider the following objective:

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

which provides the maximal difference between the rank i and the ranks belonging to 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.

3.7.4 Early use of distances between pairs of hydrogens

When dealing with molecules, the distance information can come from two different sources: the chemical composition of the molecule and the NMR experiments (see Section 1.3.1). The use of NMR distances at early layers of the search tree can help reducing the tree width, because these distances are likely to be represented by tighter intervals (because not only based on geometrical constraints). Recall that distances obtained by NMR experiments are generally between pairs of hydrogen atoms. 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):

$$\alpha_{\mathbb{H}}(r_i) = \min_{v \in r_i} |\Lambda_\alpha(r_i, v) \cap \mathbb{H}|.$$

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

3.8 An Open Problem: Orders and Consecutivity

When reading this chapter, the reader may have guessed by him/herself an important open problem that concerns discretization orders. We presented handcrafted orders, orders guided by a pseudo de Bruijn graph approach, both satisfying the consecutivity assumption, but we were not able to present an automatic tool for the detection of this kind of orders. As shown in [24], the problem to identify a discretization order that satisfies the consecutivity assumption is NP-hard.

The methodology presented in Section 3.6 cannot be simply extended for an automatic construction of total orders satisfying the consecutivity assumption. In terms of optimal orders, a discretization order with consecutivity assumption optimizes the following objective:

$$g(r_i) = \begin{cases} 1 & \text{if } G[\{r_{i-K}, \ldots, r_{i-1}\}] \text{ is a clique,} \\ 0 & \text{otherwise.} \end{cases}$$

However, the objective g is not a simple function (see Def. 3.6.1). As a consequence, different subsets of r_i can give the same value to $g(r_i)$. Since more than one optimal value can be selected, the search needs to branch over all possible optimal sets. The use of non-simple function, such as g, would therefore make the entire methodology presented in Section 3.6 of exponential complexity. This observation agrees with the NP-hardness of the problem of finding discretization orders with consecutivity assumption.

Future works will be aimed at the development of more sophisticated techniques for the automatic detection of discretization orders with the consecutivity assumption. One possible research line is to verify whether an order satisfying the consecutivity assumption can be deduced from a pre-computed discretization order obtained, for example, by Alg. 4.

Chapter 4

Opening to New Applications

4.1 Introducing the Dynamics

Very recently, we have begun investigating some new research directions in the context of the DGP, where some new applications are taken into consideration. Motivated by the fact that such applications are dynamical in nature, we recently worked for introducing a new class of the DGP, named *dynamical* DGP (dynDGP) [119, 133].

In order to consider the temporal component in DGPs, we suppose that the vertex set of the graph G, representing an instance of the dynDGP, is actually the set product of two sets: the set V, which contains a predefined number of objects (which correspond to the *static* vertices of the classical DGP), and the set $T \subset \mathbb{N}_+$, consisting of the first $n = |T|$ integer strictly positive numbers representing the time as a sequence of discrete steps.

When V is replaced by $V \times T$ in G, the vertex (v, t) of the graph is an ordered pair that represents a given object v at a certain instant t. In the following, we will use the compact notation v_t for the vertex $(v, t) \in V \times T$. The edge set E contains pairs $\{u_q, v_t\}$, whose weights provide the information about the distance between two vertices u and v at times q and t, respectively. As in Def. 1.1.3, we suppose that two values can be associated to every edge in E by the weight function d: a distance value δ (most likely, in the applications considered in this chapter, an approximation of this distance), and a priority level π.

Given a graph $G = (V \times T, E, d)$, finding a solution of a dynDGP consists in identifying a realization

$$x : V \times T \longrightarrow \mathbb{R}^K$$

such that the violations on the distance constraints are as small as possible, where higher importance is given to distances with higher priority level π (see the equivalent definition of the *static* DGP in Def. 1.1.3). The penalty function σ (see equ. (1.3)) can be adapted for the dynDGP as follows:

$$\sigma(X) = \sum_{\{u_q, v_t\} \in E} \pi(u_q, v_t) \left(\|x_u^q - x_v^t\| - \delta(u_q, v_t) \right)^2,$$

where x_v^t indicates the position in $\mathbb{R}^K$ of the object v at time t. In dynDGP applications, the realization x represents an *animation* of the objects in V over the time steps in T.

The simplest approach to the dynDGP would be to tackle it as a classical (i.e. static) DGP, where the fact that distances may concern the same object at two different times, or two objects at the same time, is neglected. However, this information should instead be exploited by solution methods for improving their performances. We recently began studying this particular class of DGP instances with the aim of exploiting their dynamical component in solution methods. By reviewing some recent literature, we found out that different kinds of applications can give rise to problems that can in fact be formulated as a dynDGP.

Since the vertex set of G is a set product, it is evident that subgraphs of G concerning either the object set V or the time set T can have interesting meanings in the context of the dynDGP. Moreover, such subgraphs may have some important properties, that allow for verifying the relative dependence of the objects in V during the animation.

The subgraph $G_t = G[V \times \{t\}]$, induced by the set product between V and only one temporal value t, corresponds to one *frame* of the animation at a fixed time. Let E_t be its edge set. In some dynDGP applications, this subgraph may contain a graph S, to which a skeletal structure may be associated (see Section 2.1, Def. 2.2.2).

Definition 4.1.1 *Given a graph G representing a dynDGP instance, we say that G admits skeletal structure (S, χ) if, for every $t \in T$, S is subgraph of G_t and*

$$\forall \{u, v\} \in E_S, \quad ||\chi(u) - \chi(v)|| = \delta(u_t, v_t).$$

Depending on the application at hand, dynDGP instances may admit or not a skeletal structure.

Similarly, the subgraph $G_v = G[\{v\} \times T]$ represents a sub-instance of the dynDGP instance where only one object of V is concerned. Let E_v be its edge set. We also consider the subgraphs $G_v^{(\underline{t},\bar{t})}$ corresponding to $G[\{v\} \times \{\underline{t}, \dots, \bar{t}\}]$, where $\underline{t}$, $\bar{t} \in T$ with $\underline{t} < \bar{t}$. Let $E_v^{(\underline{t},\bar{t})}$ be the edge set of $G_v^{(\underline{t},\bar{t})}$. In terms of dynDGP, a realization of the subgraphs G_v represents a possible *trajectory* of a fixed object $v \in V$ over time.

The dynDGP instances for the applications that we consider can be constructed from known initial animations, representing simulations or rough approximations of the solutions. In general, such initial animations are given by the trajectories x_v^t of the objects $v \in V$ for all times $t \in T$. Our approach consists in representing such initial animations by the relative distances between pairs of vertices u_q and v_t, and to *manipulate* them by introducing new distance constraints, so that new animations, having some particular desired properties, can be obtained. The set of original distances, with some new added distances, defines an instance of the dynDGP.

In this chapter, we will review some very recent research lines concerning a selected set of applications for the dynDGP. Crowd simulations, air traffic control and multi-robot systems can share a similar dynDGP modeling where no particular skeletal structure can be identified in the corresponding graph G. We will discuss

this particular case in Section 4.2. In Section 4.3, we will instead focus on human motion retargeting, where G does admit a skeletal structure, which can be exploited for the solution of the instance.

4.2 Animating Objects with no Skeletal Structure

We consider in this section a set of applications where the graph G, representing an instance of the dynDGP, admits no skeletal structure. This is the case when working with animations of independent (or almost independent) objects, such as pedestrians in a crowd [67, 108, 136], "aircraft" [137, 141], and multi-robot systems [47, 109, 144]. In the first two applications, given the initial configuration of a set of moving objects, with known directions and speeds, our aim is to predict their future configurations while avoiding, for example, any kind of collisions. We will not enter in the details of every mentioned application, but we will rather consider a simplified problem where animations of moving objects in the plane are manipulated by formulating a dynDGP. However, we will discuss in more details the multi-robot system application in Section 4.2.2.

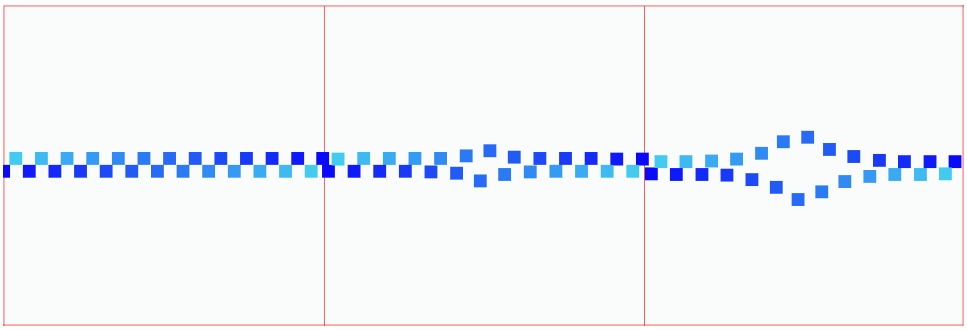
Given a known animation of a set of objects, we can obtain a representation of the animation that is based on inter-object distances. This allows us to construct an initial graph G, which is complete, because all distances can be computed from a given animation. However, these distances are not all strictly necessary for the representation of the original animation.

The most important distances for describing the objects' dynamics are those allowing to represent their motion, which are the inter-frame distances between objects at different times t. Moreover, since there is no skeletal structure associated to the dynDGP instance, it is not strictly necessary to consider distances between the objects in a common frame. This kind of distances can be considered in a second step, when including the new distance constraints in the graph G. For example, in order to avoid collisions occurring in initial rough simulations, we can impose that all distances between objects at the same frame are greater than a predefined positive real threshold Δ.

More formally, instances of the dynDGP related to applications with no skeletal structure can be modeled by a graph having an edge set satisfying the following property:

$$\bigcup_{t \geq \iota+1} \bigcup_{v \in V} E_v^{(t-\iota,t)} \subset E, \tag{4.1}$$

where $E_v^{(t-\iota,t)}$ is the edge set of the subgraph $G[\{v\} \times \{t-\iota, \ldots, t\}]$, and the depth parameter $\iota > 0$ is the number of previous immediate positions for an object that are used to represent its movement. The corresponding distances δ can be either considered as is, or modified in order to impose some desired effects. Different priority levels π can be associated to such distances, depending on which ones are more important in the animation (for example, the distances between two vertices concerning the same object v at two consecutive times t_1 and t_2 represent the movement



(a) **(b)** **(c)**

Figure 4.1: *The trajectories of two objects moving one towards the other. Different degrees of blue are used for representing different time steps.* **(a)** *the original animation;* **(b)** *the dynDGP solution found by imposing* $\Delta = 0.1$*;* **(c)** *the dynDGP solution found by imposing* $\Delta = 0.2$*.*

speed, which is likely to be preserved if a high priority is associated to this distance). Notice the use of the inclusion operator "$\subset$" in equ. (4.1): additional edges, with the corresponding distances, may be included to E, instead of only modifying the existing ones. In general, all added or modified distances need to have a high priority with respect to the original distances.

4.2.1 Preliminary computational experiments

This section presents some very preliminary computational experiments for the dynDGP model presented in the previous section. Fig. 4.1 shows a simple animation, that is subsequently manipulated. In the original animation, two objects are initially positioned at the opposite sides of a 2D box of size 1.0×1.0, and they subsequently move towards each other. At the central frame, both objects are positioned near the center of the box, where they come very close to each other.

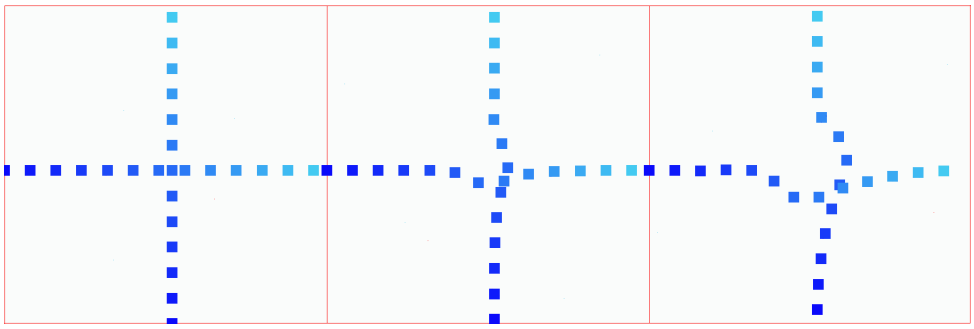
By representing this animation by distances, we can construct the graph G with the edge set E as in equ. (4.1), where our depth parameter ι was set to 3. Moreover, we can add the following set of edges:

$$E^+ = \big\{\{u_t, v_t\} \,:\, u, v \in V,\, u \neq v \text{ and } ||x_u^t - x_v^t|| \leq \Delta\big\} \subset E,$$

where Δ is a strictly positive real number, and with the following associated distance constraints:

$$\forall \{u_t, v_t\} \in E^+, \quad \delta(u_t, v_t) \in [\Delta, +\infty]. \tag{4.2}$$

Since the initial animation can serve as a good starting point for the desired solution, we employ a Spectral Projected Gradient (SPG) algorithm for the solution



(a) **(b)** **(c)**

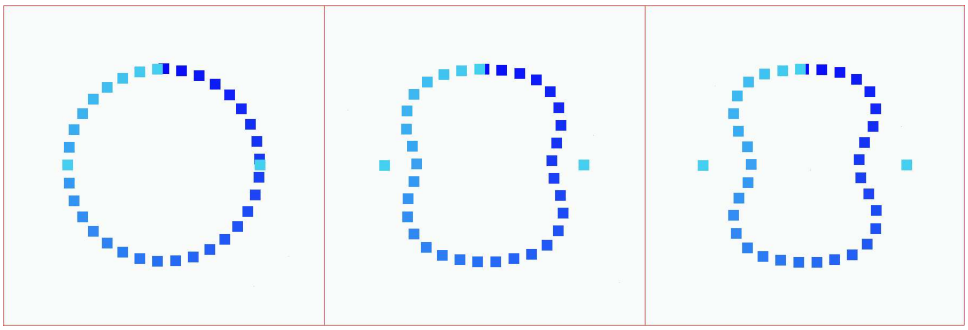
Figure 4.2: *An example similar to the one in Fig. 4.1, where the two objects have orthogonal trajectories.* **(a)** *the original animation;* **(b)** *dynDGP solution with* $\Delta = 0.1$*;* **(c)** *dynDGP solution with* $\Delta = 0.2$*.*

of this dynDGP instance. Gradient descent methods are typically used for local optimization, where the direction given by the vector opposite to the function gradient is explored for minimizing the function values. A crucial point is the identification of the *step* along the opposite gradient direction.

In [52], the gradient descent was coupled with a closed-form formula capable of providing the step α that improves the convergence properties of the gradient method, which depend upon the Hessian matrix of σ. For this reason, this class of methods is known as *spectral* gradient methods, for which proof of convergence was given for strictly convex functions (recall that our function σ does not belong to this category). Moreover, it was noticed that the function values do not decrease monotonically during the iterations of the spectral gradient. Therefore, as in [119], we use the closed-form formula for α proposed in [52] only to define an initial value that we subsequently refine with a non-monotone line search [18, 162]. We implemented a spectral gradient algorithm with projection (SPG [17]), which allows to bound every distance in the corresponding intervals (the interval is degenerate in case of exact distances, and $+\infty$ can be replaced by a sufficiently large real value in equ. 4.2).

Fig. 4.1(b) shows a realization of G where Δ was set to 0.1; Fig. 4.1(c) shows the equivalent result for $\Delta = 0.2$. SPG was used for solving the obtained dynDGP instance (one unique instance for the entire animation), where the original animation was given as a starting point. The experiment in Fig. 4.2 is very similar to the one in Fig. 4.1, the only difference being that the original paths of the two objects are orthogonal in the second one.

The experiment in Fig. 4.3 shows an initial animation where one object performs a clock-like motion. On its way, however, it finds two obstacles. The dynDGP instance is therefore conceived in order to preserve this motion as much as possible, but at the same time so that the object can avoid the collisions with the obstacles. The



(a) (b) (c)

Figure 4.3: *A clock-like trajectory that initially finds two obstacles on its way.* **(a)** *the original animation;* **(b)** *dynDGP solution with* $\Delta = 0.1$*;* **(c)** *dynDGP solution with* $\Delta = 0.2$.

edge set E of the graph representing this dynDGP instance is defined as explained above for the experiment in Fig. 4.1. Higher priorities are given to the distances introduced for avoiding the collisions.

These experiments are very preliminary but they show the possible use of DGP tools for the solving this class of problems. While the dynDGP can be studied from a general point of view and independently from the different applications, the tools for its solution can be tailored to some special cases. Just to give an example, in crowd simulations, the *personal space* that every object constructs around him/herself can be play a fundamental role in order to model the interactions with the others [50], so that more natural animations can be obtained.

4.2.2 The particular case of multi-robot systems

The application on multi-robot systems has some particular additional properties [47, 109, 144]. We focus our attention on local on-board sensors that are capable to measure the distance between mobile robots. The main problem is to implement decentralized group localization and formation control algorithms, with the aim of deploying a highly autonomous robot teams in "non-trivial" environments (e.g. inside buildings, underwater, underground, or even in deep space).

In recent works, the formation control is performed by ensuring that the robots are able to define, at every time $t \in T$, a skeletal structure (S, χ) that is rigid [161]. The existence or absence of an edge in S is generally associated to the distance range between two robots (if too large, the robots cannot communicate), and to the presence of obstacles (that does not allow the robots to obtain the necessary relative measurements).

Given a pre-recorded animation of a multi-robot system, we can represent it by relative distances for creating a graph G associated with an animation (see Sec-

tion 4.2). As for the other applications, there is no skeletal structure in G. However, since it is imposed to the robots to form a rigid structure at every frame, the pair (G_t, x^t) is in fact a rigid skeletal structure for every $t \in T$.

As in Section 4.2, new distance constraints can be included in the dynDGP instance, and a new manipulated animation, describing the new trajectories for the robots, can be obtained by solving such an instance. However, in the particular application concerning multi-robots formations, we have an additional constraint imposing that (G_t, x^t) keeps defining a rigid skeletal structure, in order to ensure a good level of communication among the robots. Even if this constraint is satisfied by the pre-recorded animation, the introduction of new distances in G can define new animations where this constraint may not be satisfied (when two robots are constrained to get further apart, for example, they may not be able to communicate anymore). The rigidity of each (G_t, x^t), for every $t \in T$, is therefore an additional constraint that we will consider in future works in the context of the dynDGP.

4.3 Human Motion Retargeting

Character animation is nowadays largely used in movie and video game industries. Typically, a 3D model and its associated skeletal structure (see Section 4.1.1) are designed by an artist, then animated either manually or using recorded motion capture data. However, it is often the case that motions created for a specific character, or captured from a given actor, need to be reused on characters with a different morphology: this problem is known in the specialized literature as *motion retargeting*.

Examples of motion retargeting include adapting motions to preserve important relationships between body parts (e.g. a character's hand touching its chin when nodding) or between body parts and the environment (e.g. ensuring that feet remain planted on the ground during locomotion support phases). Motion retargeting is especially important when using motion capture data, where the differences between morphologies of the human actor and of the character to animate raise adaptation issues.

A classical human motion representation consists in a sequence of local rotations for each edge of the corresponding skeletal structure (S, χ). This skeletal structure can either consist in a simplification of the human skeleton (see Fig 4.4), or it can represent more accurately its morphology, by employing a more complex structure that is generally referred to as *mesh structure* (see next section). Without specific retargeting techniques, the local rotations over the edges of S are simply transferred from the original to the target skeletal structure. However, this process often leads to undesired results [51, 66, 134, 135]. In the context of motion retargeting, the vertices of the graph S are generally called *joints*, and, in case of structures such as the one in Fig 4.4, the corresponding edges are called *bones*.

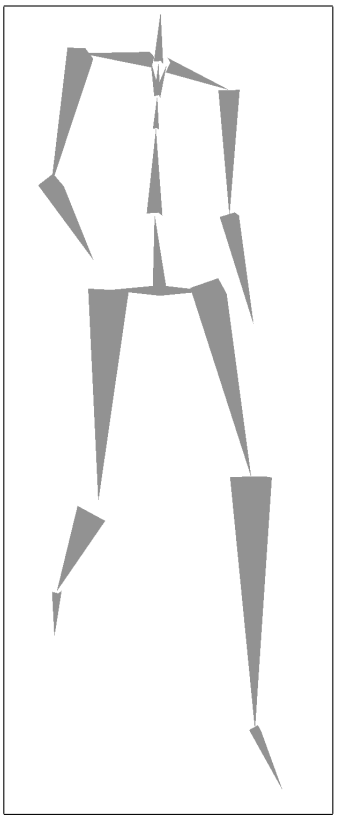


Figure 4.4: *A possible representation of the human skeleton.*

A general definition of motion retargeting can therefore be as follows. It is the process of adapting the motion of a source character to a target character having a different morphology, where characters are represented by suitable skeletal structures. Early motion retargeting approaches relied on space-time constraints in order to preserve desirable qualities of the original motion [51]. Others proposed to use Inverse Kinematics (IK), combined for example with prioritized constraints [90], end-effector importances [160] or intermediate/normalized skeletons [80, 110]. However, all these approaches strongly depend upon determining or manually specifying what are the important constraints for retargeting motions.

Our initial works on motion retargeting find inspiration from an approach based on the so-called *interaction mesh* [70]. The interaction mesh provides an efficient and implicit representation of spatial relationships between body parts of two interacting characters, as well as between the character and its environment [4, 72]. Our approach to motion retargeting relies on a more general graph representation of character dynamics, which is a complete supergraph of the skeletal structure (S, χ) used for the representation of the character.

4.3.1 Instances with non-rigid skeletal structure

As in Section 4.2, our dynDGP approach consists in representing the original motion of a human character by relative distances (between pairs of joints, in this case) and, with this distance information, to construct the graph G associated to the dynDGP. In this application, however, since a character animation can be seen as sequence of character postures, we only consider, in our preliminary works, distances between vertices of the skeletal structure, at different times t, representing either bone lengths or relative movements. As a consequence, the edge set of our graph G is

$$E = \bigcup_{t \in T} E_t,$$

and it contains all edges that relate joints at same times t. When considering the source skeletal structure (S, χ_1), for which the original animation is known, all dis-

tances associated to the edges in E can be computed. However, we are interested in reproducing the animation for a new skeletal structure (S, χ_2), where the graph S is common to the two structures, but the associated realizations differ. Replacing χ_1 with χ_2 implies that some distances $||\chi_1(u) - \chi_1(v)||$ may be different from $||\chi_2(u) - \chi_2(v)||$, for some $\{u, v\} \in E_S$.

When defining the dynDGP instance, therefore, the original distances computed from the animation of (S, χ_1) need to be modified so that they are compatible with the target structure (S, χ_2) [120]. In doing so, we also modify accordingly the distances related to edges in G that do not belong to the skeletal structure. This is a very delicate step, because the inclusion of large changes in the modified distances may cause large incompatibilities with other distances, and spoil the computations. Consider for example an arm represented by shoulder v_1, elbow v_2 and wrist v_3, with distance $\delta(v_1, v_3)$ so that the angle in v_2 is 90°. Now suppose to modify the bone lengths $\delta(v_1, v_2)$ and $\delta(v_2, v_3)$ in a way that $\delta(v_1, v_2) + \delta(v_2, v_3) = \delta(v_1, v_3)$. When a new realization is searched by considering these distances, the obtained arm would be completely extended (the angle in v_2 is now 180°). Evidently, this solution does not correspond to the original posture of the arm.

We propose therefore to extract the information about the posture of (S, χ_1), represented by the position matrix X_1, before representing the original animation by distances and constructing the new dynDGP instance. Our procedure is based on the idea to compute all shortest paths $P_{u,v} = \{p_1, \ldots, p_k\}$ between pairs of distinct joints, where $p_1 = u$, $p_k = v$ and, for every $i = 1, \ldots, k-1$, we have $\{p_i, p_{i+1}\} \in E_S$. The term "shortest" is meant to make reference to the number of edges on which the path walks from the joint u to the joint v; it is not related neither to δ, not to π. We refer to the sum of the distance values over a path $P_{u,v}$ as the *weight* $\tau_{u,v}$ of the shortest path $P_{u,v}$, computed as:

$$\tau_{u,v} = \sum_{i=1}^{(|P_{u,v}|-1)} ||\chi(p_i) - \chi(p_{i+1})||.$$

Once all shortest paths $P^{(1)}_{u,v}$ over the graph S are computed, we normalize the computed distances with the weights $\tau^{(1)}_{u,v}$ of the corresponding shortest path:

$$\forall \{u, v\} \in E_t, \quad \delta^N_{u,v} = \frac{||x_u - x_v||}{\tau^{(1)}_{u,v}},$$

where x_v is the position of the generic vertex v in the posture X_1. Notice that all distances related to bone lengths are equal to 1 after the normalization, and that distances close to 0 indicate an inter-joint contact, while non-bone distances close to 1 are related to completely extended postures. We point out that the idea to normalize relative distances is not completely new, and that it was partially exploited for example in [80] in a morphology-independent representation of the motions, which is however not solely based on distance information.

In order to impose the posture in X_1 to the skeletal structure (S, χ_2), we apply first of all the formula for computing a new distance matrix:

$$\forall \{u, v\} \in E_t, \quad \delta_{u,v} = \tau^{(2)}_{u,v} \cdot \delta^N_{u,v}, \tag{4.3}$$

where $\tau^{(2)}_{u,v}$ is the weight of the shortest path $P^{(2)}_{u,v}$ over the graph S. This formula makes it possible to reconstruct correctly the bone lengths of (S, χ_2) while modifying accordingly the original distances in X_1, for them to be adapted to the new bone lengths of (S, χ_2). It is easy to verify that the issue discussed in the example above (concerning the modified posture of an arm) is overcome with this procedure, but it is important to remark as well that it is unlikely that the final set of distances $\delta_{u,v}$ obtained by equ. (4.3) defines a DGP where all such distances can be satisfied without introducing any violation (in the sense of Def. 1.1.2).

Intuitively, distances between joints that are close in the skeletal structure (i.e. corresponding to shortest paths $P_{u,v}$ over fewer bones, consider again the example of the arm) can be approximated better than others (for example the distance between a hand and a foot is more difficult to approximate). For this reason, every computed distance $\delta_{u,v}$ is coupled with the priority level $\pi_{u,v}$, that is based on the cardinality $|P_{u,v}|$ of the corresponding shortest path:

$$\pi_{u,v} = \frac{|P_{\max}| - |P_{uv}| + 2}{|P_{\max}|}, \tag{4.4}$$

where $P_{\max}$ is the longest shortest path that can be defined over the two skeletal structures (S, χ_1) and (S, χ_2). Notice that, in correspondence with the bone lengths, the priority $\pi_{u,v}$ is maximal and equal to 1; the smallest possible priority value is given by $2/|P_{\max}|$. This distance information, obtained by applying equ. (4.3) for all frames of an original animation of (S, χ_1), together with the associated priority levels (see equ. (4.4)), defines a graph G representing a dynDGP instance.

We solved such dynDGP instances by the non-monotone SPG (see Section 4.2.1), which was applied frame by frame with the realization found at frame $t - 1$ as a starting point (only exception is for the first frame, that takes the first frame of the original motion as a starting point). Our preliminary results can be viewed in the video clip associated to the conference paper [15], freely available from the conference website[1], where these results were recently presented.

4.4 Future Works: the Discretizable dynDGPs

In the preliminary results presented in this Chapter, we considered dynDGP instances without skeletal structure (see Section 4.2.1) and with non-rigid skeletal structure (for motion retargeting, see Section 4.3.1). However, in the latter case, different kinds of character models can actually lead to the definition of dynDGPs having a skeletal structure that is rigid. An example is the mesh structure, used

[1] `https://dl.acm.org/citation.cfm?id=3136466`

frequently in computer graphics, that was already mentioned at the beginning of Section 4.3.

As already observed in Chapter 2, the discretization of a DGP instance and the rigidity of the graph G representing such an instance are two very close concepts. Given a dimension K, every DDGP instance has an underlying graph G that is rigid in dimension K. The inverse implication is not true in general. However, in the dynDGP, a skeletal structure can be present at every frame, and the graph G is a supergraph containing all such skeletal structures, which may be rigid. Therefore, even if this is not true in general, we can say that there are "good chances" that the graph G, related to a dynDGP with rigid skeletal structure, admits the discretization.

An example is given by the kind of mesh structure that is basically formed by a closed triangular grid. Such structures generally work in spaces having dimension $K = 3$. In terms of graph, every triangle of the mesh is a 3-clique, while two overlapping triangles induce a subgraph which misses only one edge to form a 4-clique. As in the work presented for the protein backbones in Chapter 3, this missing distance information can be estimated, so that the discretization is actually possible.

The possibility to discretize opens the door to a completely new research line in the field of motion retargeting. Given a discretizable dynDGP, there are several questions that may raise. For example, is it possible to find optimal discretization orders for this new class of instances (see Section 3.6)? Or it is rather more convenient to pre-define the vertex orders for an entire class of problems, as it was done for protein backbones in Section 3.3? And, in the BP algorithm, can we conceive new pruning devices that can come to help for the search to focus on the feasible branches of the tree $\mathcal{T}$ (see Section 2.7)? What do the symmetries of the search tree represent in the context of the dynDGP (see Section 2.8)? Can we exploit such symmetries to speed up the search, or to obtain better results? Finally, are discretizable dynDGP instance more parallelizable than general DDGP instances (see Section 2.9)? This is material for several more years of research on the DGP.

Acronyms and Main Notations

Main Notations

n	problem size
K	realization dimension
V	object/vertex set
T	time set (represented by the first n positive integer numbers)
$\mathcal{D}$	distance list (without predefined structure)
E	edge set
κ	the assignment function
δ	the inverse of the assignment function
π	function assigning priority levels to pairs of vertices in E
σ	penalty function measuring the constraint violations
x	realization of graph G in a Euclidean space
X	matrix representation of the realization x
d	weight function of graph G, corresponding to δ, and π values (optional)
$G = (V, E, d)$	simple weighted undirected graph representing an instance of the DGP
E'	subset of E consisting of exact/priority 1 distances
E_d	subset of E consisting of discretization distances
E_p	subset of E consisting of pruning distances
$G[\cdot]$	subgraph induced by a subset of vertices
B	pseudo de Bruijn graph
$\mathcal{T}$	search tree obtained with the discretization
D	number of sample positions extracted from arcs
S	symmetry set of G
$\mathbb{P}(V)$	powerset of V
r	(partial, total, optimal, discretization) vertex order
$\alpha(r_i)$	counter of adjacent predecessors of vertices ranked i in r
$\alpha_{ex}(r_i)$	as $\alpha(r_i)$, but restricted to distances in E'
$\beta(r_i)$	counter of adjacent successors of vertices ranked i in r
$\beta_{ex}(r_i)$	as $\beta(r_i)$, but restricted to distances in E'
$V \times T$	vertex set of dynDGP instances
$G_v = G[\{v\} \times T]$	sub-instance of dynDGP instance where $v \in V$ is fixed
$G_t = G[V \times \{t\}]$	sub-instance of dynDGP instance where $t \in T$ is fixed
$G_v^{(\underline{t},\bar{t})}$	sub-instance of dynDGP instance given by $G[\{v\} \times \{\underline{t}, \ldots, \bar{t}\}]$
(S, χ)	skeletal structure having graph S and realization χ
τ	weight of shortest path between two vertices of a connected graph

Acronyms

BP	Branch-and-Prune
DDF	Direct Distance Feasibility (pruning device)
DG	Distance Geometry
DGP	DG Problem
DDGP	Discretizable DGP
dynDGP	dynamical DGP
LJ	Lennard Jones (pruning device)
MDS	MultiDimensional Scaling
NMR	Nuclear Magnetic Resonance
PCA	Principal Component Analysis
PDB	Protein Data Bank
symBP	symmetry-driven BP
SNLP	Sensor Network Localization Problem
SP	Shortest-Path (pruning device)
SPG	Spectral Projected Gradient
SSF	Secondary Structure Feasibility (pruning device)
TAF	Torsion Angle Feasibility (pruning device)
uDGP	unassigned DGP
vdW	van der Waals (pruning device)

Bibliography

[1] A. Agra, R. Figueiredo, C. Lavor, N. Maculan, A. Pereira, C. Requejo, *Feasibility Check for the Distance Geometry Problem: an Application to Molecular Conformations*, International Transactions in Operational Research **24**, 1023–1040, 2017.

[2] D.K. Agrafiotis, *Stochastic Proximity Embedding*, Journal of Computational Chemistry **24**(10), 1215–21, 2003.

[3] D.K. Agrafiotis, D. Bandyopadhyay, E. Yang, *Stochastic Proximity Embedding (SPE): a Simple, Fast and Scalable Algorithm for Solving the Distance Geometry Problem*, In: [128], 291–311. 2013.

[4] R.A. Al-Asqhar, T. Komura, M.G. Choi, *Relationship Descriptors for Interactive Motion Adaptation*, Proceedings of the 12^{th} ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA13), 45–53, 2013.

[5] J. Alencar, C. Lavor, T.O. Bonates, *A Combinatorial Approach to Multidimensional Scaling*, IEEE conference proceedings, 2014 IEEE International Congress on Big Data, 562–569, 2014.

[6] A.Y. Alfakih, A. Khandani, H. Wolkowicz, *Solving Euclidean Distance Matrix Completion Problems via Semidefinite Programming*, Computational Optimization and Applications **12**, 13–30, 1999.

[7] 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.

[8] 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: [128], 377–412, 2013.

[9] R. Alves, A. Cassioli, A. Mucherino, C. Lavor, L. Liberti, *Adaptive Branching in iBP with Clifford Algebra*, Proceedings of Distance Geometry and Applications (DGA13), Manaus, Amazonas, Brazil, 65–69, 2013.

[10] 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.

[11] G. Anandalingam, T.L. Friesz, *Hierarchical Optimization: An Introduction*, Annals of Operations Research **34**(1), 1–11, 1992.

[12] F. Beck, M. Burch, S. Diehl, D. Weiskopf, *A Taxonomy and Survey of Dynamic Graph Visualization*, Computer Graphics Forum, **36**(1), 133–159, 2017.

[13] J.M. Berg, J.L. Tymoczko, L. Stryer. *Biochemistry*, W.H. Freeman publications (6^{th} edition), 2006.

[14] 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.

[15] A. Bernardin, L. Hoyet, A. Mucherino, D.S. Gonçalves, F. Multon, *Normalized Euclidean Distance Matrices for Human Motion Retargeting*, ACM Conference Proceedings, Motion in Games 2017 (MIG17), Barcelona, Spain, November 2017.

[16] S.J.L. Billinge, Ph.M. Duxbury, D.S. Gonçalves, C. Lavor, A. Mucherino, *Assigned and Unassigned Distance Geometry: Applications to Biological Molecules and Nanostructures*, Quarterly Journal of Operations Research **14**(4), 337–376, 2016.

[17] E.G. Birgin, J.M. Martínez, *Large-Scale Active-Set Box-Constrained Optimization Method with Spectral Projected Gradients*, Computational Optimization and Applications **23**, 101–125, 2002.

[18] E.G. Birgin, J.M. Martínez, M. Raydan, *Nonmonotone Spectral Projected Gradient Methods on Convex Sets*, SIAM Journal on Optimization **10**, 1196–1211, 2000.

[19] 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.

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

[21] A.A. Canutescu, A.A. Shelenkov, R.L. Dunbrack jr., *A Graph-Theory Algorithm for Rapid Protein Side-Chain Prediction*, Protein Science **12**, 2001–2014, 2003.

[22] R.S. Carvalho, C. Lavor, F. Protti, *Extending the Geometric Build-Up Algorithm for the Molecular Distance Geometry Problem*, Information Processing Letters **108**, 234–237, 2008.

[23] 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.

[24] A. Cassioli, O. Günlük, C. Lavor, L. Liberti, *Discretization Vertex Orders in Distance Geometry*, Discrete Applied Mathematics **197**, 27–41, 2015.

[25] B. Colson, P. Marcotte, G. Savard, *An Overview of Bilevel Optimization*, Annals of Operations Research **153**, 235–256, 2007.

[26] P.E.C. Compeau, P.A. Pevzner, G. Tesler, *How to Apply de Bruijn Graphs to Genome Assembly*, Nature Biotechnology **29**, 987–991, 2011.

[27] I.D. Coope, *Reliable Computation of the Points of Intersection of n Spheres in n-space*, ANZIAM Journal **42**, 461–477, 2000.

[28] V. Costa, A. Mucherino, C. Lavor, L.M. Carvalho, N. Maculan, *On Suitable Orders for Discretizing Molecular Distance Geometry Problems related to Protein Side Chains*, IEEE Conference Proceedings, Federated Conference on Computer Science and Information Systems (FedCSIS12), Workshop on Computational Optimization (WCO12), Wroclaw, Poland, 397–402, 2012.

[29] 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.

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

[31] J. Dattorro, *Convex Optimization and Euclidean Distance Geometry*, $\mathcal{M}\epsilon\beta oo$, Palo Alto, 2005.

[32] N.G. de Bruijn, *A Combinatorial Problem*, Koninklijke Nederlandse Akademie v. Wetenschappen **49**, 758–764, 1946.

[33] R. de Freitas, B. Dias, N. Maculan, J. Szwarcfiter, *Distance Geometry Approach for Special Graph Coloring Problems*, Technical report 1606-04978v1, arXiv, 2016.

[34] J. de Leeuw, *Differetiability of Kruskal's Stress at a Local Minimum*, Psychometrika **49**, 111–113, 1984.

[35] J. de Leeuw, *Convergence of the Majorization Method for Multidimensional Scaling*, Journal of Classification **5**, 163–180, 1988.

[36] F. Delille, *Principes de Cosmographie*, Librairie Classique de Jules Delalain et Fils, 6^{eme} édition, 1865.

[37] H.B. Demuth, M.H. Beale, O. De Jess, M.T. Hagan, *Neural Network Design*, 2^{nd} edition, Martin Hagan, USA, 2014.

[38] M.M. Deza, E. Deza, *Encyclopedia of Distances*, 756 pages, Springer, 2016.

[39] I. Dokmanic, R. Parhizkar, J. Ranieri, M. Vetterli, *Euclidean Distance Matrices: Essential Theory, Algorithms, and Applications*, IEEE Signal Processing Magazine **32**(6), 12–30, 2015.

[40] Q. Dong, Z. Wu, *A Linear-Time Algorithm for Solving the Molecular Distance Geometry Problem with Exact Interatomic Distances*, Journal of Global Optimization **22**, 365–375, 2002.

[41] Q. Dong, Z. WU, *A Geometric Build-Up Algorithm for Solving the Molecular Distance Geometry Problem with Sparse Distance Data*, Journal of Global Optimization **26**, 321–333, 2003.

[42] M.J. Duan, M.H. Li, L. Han, S.H. Huo, *Euclidean Sections of Protein Conformation Space and their Implications in Dimensionality Reduction*, Proteins **82**, 2585–2596, 2014.

[43] Ph.M. Duxbury, L. Granlund, S.R. Gujarathi, P. Juhas, S.J.L. Billinge, *The Unassigned Distance Geometry Problem*, Discrete Applied Mathematics **204**, 117–132, 2016.

[44] T. Ellis, T. Adie, G.S. Baldwin, *DNA Assembly for Synthetic Biology: from Parts to Pathways and Beyond*, Integrative Biology **3**, 109–118, 2011.

[45] X. Fang, K.-C. Toh, *Using a Distributed SDP Approach to Solve Simulated Protein Molecular Conformation Problems*, In: [128], Springer, 351–376, 2013.

[46] F. Fidalgo, D.S. Gonçalves, C. Lavor, L. Liberti, A. Mucherino, *A Symmetry-based Splitting Strategy for Discretizable Distance Geometry Problems*, to appear in Journal of Global Optimization, 2018.

[47] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, B. Stewart, *Distributed Multirobot Exploration and Mapping*, Proceedings of the IEEE **94**(7), 1325–1339, 2006.

[48] 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.

[49] A.E. Garcia, *Large-Amplitude Nonlinear Motions in Proteins*, Physical Review Letters **68**, 2696–2699, 1992.

[50] M. Gérin-Lajoie, C.L. Richards, B.J. McFadyen, *The Negotiation of Stationary and Moving Obstructions During Walking: Anticipatory Locomotor Adaptations and Preservation of Personal Space*, Motor Control **9**, 242–269, 2005.

[51] M. Gleicher, *Retargetting Motion to New Characters*. ACM Proceedings of the 25^{th} annual conference on Computer Graphics and Interactive Techniques, 33–42, 1998.

[52] W. Glunt, T.L. Hayden, M. Raydan, *Molecular Conformations from Distance Matrices*, Journal of Computational Chemistry **14**(1), 114–120, 1993.

[53] D.S. Gonçalves, *A Least-Squares Approach for Discretizable Distance Geometry Problems with Inexact Distances*, to appear in Optimization Letters, 2018.

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

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

[56] D.S. Gonçalves, A. Mucherino, C. Lavor, *Energy-based Pruning Devices for the BP Algorithm for Distance Geometry*, IEEE Conference Proceedings, Federated Conference on Computer Science and Information Systems (FedCSIS13), Workshop on Computational Optimization (WCO13), Krakov, Poland, 335–340, 2013.

[57] 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.

[58] 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.

[59] D.S. Gonçalves, J. Nicolas, A. Mucherino, C. Lavor, *Finding Optimal Discretization Orders for Molecular Distance Geometry by Answer Set Programming*. In: "Recent Advances in Computational Optimization", S. Fidanova (Ed.), Studies in Computational Intelligence **610**, 1–15, 2015.

[60] W. Gramacho, A. Mucherino, C. Lavor, N. Maculan, *A Parallel BP Algorithm for the Discretizable Distance Geometry Problem*, IEEE Conference Proceedings, Workshop on Parallel Computing and Optimization (PCO12), 26^{th} IEEE International Parallel & Distributed Processing Symposium (IPDPS12), Shanghai, China, 1756–1762, 2012.

[61] W. Gramacho, A. Mucherino, J-H. Lin, C. Lavor, *A New Approach to the Discretization of Multidimensional Scaling*, IEEE Conference Proceedings, Federated Conference on Computer Science and Information Systems (FedCSIS16), Workshop on Computational Optimization (WCO16), Gdansk, Poland, 601–609, 2016.

[62] P. Guerry, V.D. Duong, T. Herrmann, *CASD-NMR 2: Robust and Accurate Unsupervised Analysis of Raw NOESY Spectra and Protein Structure Determination with UNIO*, Journal of Biomolecular NMR **62**, 473–480, 2015.

[63] S.R. Gujarathi, C.L. Farrow, C. Glosser, L. Granlund, P.M. Duxbury, *Ab-Initio Reconstruction of Complex Euclidean Networks in Two Dimensions*, Physical Review E**89**, 053311, 2014.

[64] P. Güntert, *Automated NMR Structure Calculation with CYANA*, Methods in Molecular Biology **278**, 353–378, 2004.

[65] H.C. Hamaker, *The London–van der Waals Attraction between Spherical Particles*, Physica **4**(10), 1058–1072, 1937.

[66] Ch. Hecker, B. Raabe, R.W. Enslow, J. DeWeese, J. Maynard, K. van Prooijen, *Real-Time Motion Retargeting to Highly Varied User-Created Morphologies*, Proceedings of ACM SIGGRAPH 2008, ACM Transactions on Graphics **27**(3), 2008.

[67] D. Helbing, I. Farkas, T. Vicsek, *Simulating Dynamical Features of Escape Panic*, Nature **407**, 487–490, 2000.

[68] B. Hendrickson, *Conditions for Unique Graph Realizations*, SIAM Journal on Computing **21**(1), 65–84, 1992.

[69] I. Herman, G. Malançon, S. Marshall, *Graph Visualization and Navigation in Information Visualization: a Survey*, IEEE Transactions on Visualization and Computer Graphics **6**(1), 24–43, 2000.

[70] E.S.L Ho, T. Komura, C-L. Tai, *Spatial Relationship Preserving Character Motion Adaptation*, Proceedings of the 37^{th} International Conference and Exhibition on Computer Graphics and Interactive Techniques, 8 pages, 2010.

[71] G.L. Holliday, A. Bairoch, P.G. Bagos, A. Chatonnet, D.J. Craik, R.D. Finn, B. Henrissat, D. Landsman, G. Manning, N. Nagano, C. O'Donovan, K.D. Puitt, N.D. Rawlings, M. Saier, R. Sowdhamini, M. Spedding, N. Srinivasan, G. Vriend, P.C. Babbitt, A. Bateman, *Key Challenges for the Creation and Maintenance of Specialist Protein Resources*, Proteins: Structure, Function, Bioinformatics **83**(6), 1005–1013, 2015.

[72] E.S.L. Ho, H.P.H. Shum, *Motion Adaptation for Humanoid Robots in Constrained Environments*, IEEE International Conference on Robotics and Automation (ICRA13), 3813–3818, 2013.

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

[74] A. Huang, C.M. Stultz, *Finding Order within Disorder: Elucidating the Structure of Proteins Associated with Neurodegenerative Disease*, Future Medical Chemistry **1**, 467–482, 2009.

[75] B. Jackson, T. Jordán, *Connected Rigidity Matroids and Unique Realizations of Graphs*, Journal of Combinatorial Theory, Series B **94**, 1–29, 2005.

[76] I.T. Jolliffe, *Discarding Variables in a Principal Component Analysis. I: Artificial Data*, Journal of the Royal Statistical Society, Series C **21**(2), 160–173, 1972.

[77] S. Kirkpatrick, C.D. Gelatt, M.P Vecchi, *Optimization by Simulated Annealing*, Science **220**, 671–680, 1983.

[78] A. Kitao, F. Hirata, N. Go, *The Effects of Solvent on the Conformation and the Collective Motions of Protein − Normal Mode Analysis and Molecular-Dynamics Simulations of Melittin in Water and in Vacuum*, Journal of Chemical Physics **158**, 447–472, 1991.

[79] N. Krislock, H. Wolkowicz, *Explicit Sensor Network Localization using Semidefinite Representations and Facial Reductions*, SIAM Journal on Optimization **20**, 2679–2708, 2010.

[80] R. Kulpa, F. Multon, B. Arnaldi, *Morphology-Independent Representation of Motions for Interactive Human-like Animations*, Proceedings of Eurographics 2005, M. Alexa, J. Marks (Eds.), Computer Graphics Forum **24**(3), 343–351, 2005.

[81] K. Kutnar, D. Marušič, *A Complete Classification of Cubic Symmetric Graphs of Girth 6*, Journal of Combinatorial Theory Series B **99**(1), 162–184, 2009.

[82] G. Laman, *On Graphs and Rigidity of Plane Skeletal Structures*, Journal of Engineering Mathematics **4**(4), 331–340, 1970.

[83] 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.

[84] C. Lavor, L. Liberti, W. Lodwick, T. Mendonça da Costa, *An Introduction to Distance Geometry applied to Molecular Geometry*, SpringerBriefs in Computer Science, New York, 54 pages, 2017.

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

[86] C. Lavor, L. Liberti, N. Maculan, A. Mucherino, *Recent Advances on the Discretizable Molecular Distance Geometry Problem*, European Journal of Operational Research **219**, 698–706, 2012.

[87] 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.

[88] 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.

[89] C. Lavor, A. Mucherino, L. Liberti, N. Maculan, *Discrete Approaches for Solving Molecular Distance Geometry Problems using NMR Data*, International Journal of Computational Biosciences **1**(1), 88–94, 2010.

[90] B. Le Callennec, R. Boulic, *Interactive Motion Deformation with Prioritized Constraints*, Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA04), 163–171, 2004.

[91] J.E. Lennard Jones, *Cohesion*, Proceedings of the Physical Society **43**, 461–482, 1931.

[92] I.A. Lewis, S.C. Schommer, B. Hodis, K.A. Robb, M. Tonelli, W.M. Westler, M.R. Sussman, J.L. Markley, *Fast and Accurate Method for Determining Molar Concentrations of Metabolites in Complex Solutions from Two-Dimensional 1H-13C NMR Spectra*, Analitycal Chemistry **79**(24), 9385–9390, 2007.

[93] L. Liberti, C. D'Ambrosio, *The Isomap Algorithm in Distance Geometry*, Proceedings of the 16^{th} International Symposium on Experimental Algorithms (SEA17), Leibniz International Proceedings in Informatics **5** (LIPIcs), London, UK, 1–13, 2017.

[94] L. Liberti, C. Lavor, *Euclidean Distance Geometry*, Springer, Berlin, 133 pages, 2017.

[95] 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.

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

[97] 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.

[98] L. Liberti, B. Masson, J. Lee, C. Lavor, A. Mucherino, *On the Number of Solutions of the Discretizable Molecular Distance Geometry Problem*, Lecture Notes in Computer Science **6831**, W. Wang, X. Zhu, D-Z. Du (Eds.), Proceedings of the 5^{th} Annual International Conference on Combinatorial Optimization and Applications (COCOA11), Zhangjiajie, China, 322–342, 2011.

[99] 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.

[100] R.S. Lima, J.M. Martínez, *Solving Molecular Distance Geometry Problems Using a Continuous Optimization Approach*, In: [128], Springer, 213–224, 2013.

[101] J.P. Linge, M. Habeck, W. Rieping, M. Nilges, *ARIA: Automated NOE Assignment and NMR Structure Calculation*, Bioinformatics **19**, 315–316, 2003.

[102] J.P. Linge, M. Nilges, *Influence of Non-Bonded Parameters on the Quality of NMR Structures: A New Force Field for NMR Structure Calculation*, Journal of Biomolecular NMR **13**(1), 51–59, 1999.

[103] D. Maioli, C. Lavor, D. Gonçalves, *A Note on Computing the Intersection of Spheres in* $\mathbb{R}^n$, ANZIAM Journal **59**, 271–279, 2017.

[104] A.G. Maldonado, J.P. Doucet, M. Petitjean, B.T. Fan, *Molecular Similarity and Diversity in Chemoinformatics: from Theory to Applications*, Molecular diversity **10**(1), 39–79, 2006.

[105] T.E. Malliavin, A. Mucherino, M. Nilges, *Distance Geometry in Structural Biology: New Perspectives*. In: [128], Springer, 329–350, 2013.

[106] A. Man-Cho So, Y. Ye, *Theory of Semidefinite Programming for Sensor Network Localization*, Mathematical Programming B **109**, 367–384, 2007.

[107] Colonel F.A. Mathieu, *Précis de Topographie – Deuxième Partie* (in French), L. Fournier & Cie, 119 pages, Paris, 1945.

[108] F.D. McKenzie, M.D. Petty, P.A. Kruszewski, R.C. Gaskins, Q-A.H. Nguyen, J. Seevinck, E.W. Weisel, *Integrating Crowd-Behavior Modeling into Military Simulation using Game Technology*, Simulation Gaming **39**(1), 10–38, 2008.

[109] E. Montijano, E. Cristofalo, D. Zhou, M. Schwager, C. Sagues, *Vision-based Distributed Formation Control without an External Positioning System*, IEEE Transactions on Robotics **32**(2), 339–351, 2016.

[110] J.-S. Monzani, P. Baerlocher, R. Boulic, D. Thalmann, *Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting*, Computer Graphics Forum **19**(3), 11–19, 2000.

[111] J. Moré, Z. Wu, *Global Continuation for Distance Geometry Problems*, SIAM Journal on Optimization **7**(3), 814–836, 1997.

[112] J. Moré, Z. Wu, *Distance Geometry Optimization for Protein Structures*, Journal on Global Optimization **15**, 219–234, 1999.

[113] H.N.B. Moseley, G.T. Montelione, *Automated Analysis of NMR Assignments and Structures for Proteins*, Current Opinion in Structural Biology **9**, 635–642, 1999.

[114] 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.

[115] 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, 234–242, 2015.

[116] 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.

[117] A. Mucherino, *On the Exact Solution of the Distance Geometry with Interval Distances in Dimension 1*. In: "Recent Advances in Computational Optimization", S. Fidanova (Ed.), Studies in Computational Intelligence **717**, 123–134, 2018.

[118] A. Mucherino, R. de Freitas, C. Lavor, *Distance Geometry and Applications*, special issue of Discrete Applied Mathematics **197**, 1–144, 2015.

[119] 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.

[120] A. Mucherino, D.S. Gonçalves, A. Bernardin, L. Hoyet, F. Multon, *A Distance-Based Approach for Human Posture Simulations*, IEEE Conference Proceedings, Federated Conference on Computer Science and Information Systems (FedCSIS17), Workshop on Computational Optimization (WCO17), Prague, Czech Republic, 441–444, 2017.

[121] A. Mucherino, C. Lavor, *The Branch and Prune Algorithm for the Molecular Distance Geometry Problem with Inexact Distances*, Proceedings of World Academy of Science, Engineering and Technology **58**, International Conference on Bioinformatics and Biomedicine (ICBB09), Venice, Italy, 349–353, 2009.

[122] A. Mucherino, C. Lavor, *Applications of Distance Geometry*, special issue of Optimization Letters, to appear, 2018.

[123] A. Mucherino, C. Lavor, L. Liberti, *A Symmetry-Driven BP Algorithm for the Discretizable Molecular Distance Geometry Problem*, IEEE Conference Proceedings, Computational Structural Bioinformatics Workshop (CSBW11), International Conference on Bioinformatics & Biomedicine (BIBM11), Atlanta, GA, USA, 390–395, 2011.

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

[125] A. Mucherino, C. Lavor, L. Liberti, *Exploiting Symmetry Properties of the Discretizable Molecular Distance Geometry Problem*, Journal of Bioinformatics and Computational Biology **10**(3), 1242009(1–15), 2012.

[126] A. Mucherino, C. Lavor, L. Liberti, N. Maculan, *Strategies for Solving Distance Geometry Problems with Inexact Distances by Discrete Approaches*, Proceedings of Toulouse Global Optimization 2010 (TOGO10), Toulouse, France, 93–96, 2010.

[127] A. Mucherino, C. Lavor, L. Liberti, N. Maculan, *On the Discretization of Distance Geometry Problems*, ITHEA Conference Proceedings, Mathematics of Distances and Applications 2012 (MDA12), Varna, Bulgaria, 160–168, 2012.

[128] A. Mucherino, C. Lavor, L. Liberti, N. Maculan (Eds.), *Distance Geometry: Theory, Methods and Applications*, 410 pages, Springer, 2013.

[129] A. Mucherino, C. Lavor, L. Liberti, E-G. Talbi, *A Parallel Version of the Branch & Prune Algorithm for the Molecular Distance Geometry Problem*, IEEE Conference Proceedings, ACS/IEEE International Conference on Computer Systems and Applications (AICCSA10), Hammamet, Tunisia, 1–6, 2010.

[130] 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.

[131] A. Mucherino, L. Liberti, C. Lavor, *MD-jeep: an Implementation of a Branch & Prune Algorithm for Distance Geometry Problems*, Lectures Notes in Computer Science **6327**, K. Fukuda et al. (Eds.), Proceedings of the 3^{rd} International Congress on Mathematical Software (ICMS10), Kobe, Japan, 186—197, 2010.

[132] A. Mucherino, L. Liberti, C. Lavor, N. Maculan, *Comparisons between an Exact and a Meta-Heuristic Algorithm for the Molecular Distance Geometry Problem*, ACM Conference Proceedings, Genetic and Evolutionary Computation Conference (GECCO09), Montréal, Canada, 333–340, 2009.

[133] A. Mucherino, J. Omer, L. Hoyet, P. Robuffo Giordano, F. Multon, *An Application-based Characterization of Dynamical Distance Geometry Problems*, under review, 2018.

[134] F. Multon, L. France, M.P. Cani-Gascuel, G. Debunne, *Computer Animation of Human Walking: a Survey*, The Journal of Visualization and Computer Animation **10**(1), 39–54, 1999.

[135] F. Multon, R. Kulpa, B. Bideau, *MKM: a Global Framework for Animating Humans in Virtual Reality Applications*. Presence: Teleoperators and Virtual Environments **17**(1), 17–28, 2008.

[136] A.H. Olivier, A. Marin, A. Crétual, J. Pettré, *Minimal Predicted Distance: a Common Metric for Collision Avoidance During Pairwise Interactions between Walkers*, Gait Posture **36**(3), 399–404, 2012.

[137] J. Omer, *A Space-Discretized Mixed-Integer Linear Model for Air-Conflict Resolution with Speed and Heading Maneuvers*, Computers and Operations Research **58**, 75–86, 2015.

[138] K. Pearson, *On Lines and Planes of Closest Fit to Systems of Points in Space*, Philosophical Magazine **2**, 559–572, 1901.

[139] M. Petitjean, *Spheres Unions and Intersections and Some of their Applications in Molecular Modeling*, In: [128], 61–83, 2013.

[140] G.N.T. Ramachandran, V. Sasisekharan, *Conformation of Polypeptides and Proteins*, Advances in Protein Chemistry **23**, 283–437, 1968.

[141] A. Richards, J.P. How, *Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming*, IEEE Conference Proceedings, American Control Conference 2002, Anchorage, AK, USA, 2002.

[142] 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.

[143] D. Scaramuzza, M.C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M.W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Hee Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J.C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, L. Meier, *Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments*, Robotics & Automation Magazin **21**(3), 15 pages, 2014.

[144] F. Schiano, A. Franchi, D. Zelazo, P. Robuffo Giordano, *A Rigidity-Based Decentralized Bearing Formation Controller for Groups of Quadrotor UAVs*, Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS16), 5099–5106, 2016.

[145] I. Schoenberg, *Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espaces distanciés vectoriellement applicable sur l'espace de Hilbert"*, Annals of Mathematics **36**, 724–732, 1935.

[146] Y. Shen, F. Delaglio, G. Cornilescu, A. Bax, *TALOS+: a Hybrid Method for Predicting Protein Backbone Torsion Angles from NMR Chemical Shifts*, Journal of Biomolecular NMR **44**(4), 213–236, 2009.

[147] M.J. Sippl, H.A. Scheraga, *Cayley-Menger Coordinates*, Proceedings of the National Academy of Sciences of the United States (PNAS) **83**, 2283–2287, 1986.

[148] A. Sit, Z. Wu, *Solving a Generalized Distance Geometry Problem for Protein Structure Determination*, Bulletin of Mathematical Biology **73**, 2809–2836, 2011.

[149] M. Souza, C. Lavor, A. Muritiba, N. Maculan, *Solving the Molecular Distance Geometry Problem with Inaccurate Distance Data*, BMC Bioinformatics **14**, S71–S76, 2013.

[150] M. Spedding, *Resolution of Controversies in Drug/Receptor Interactions by Protein Structure. Limitations and Pharmacological Solutions*, Neuropharmacology **60**, 3–6, 2011.

[151] J.B. Tenenbaum, V. de Silva, J.C. Langford, *A Global Geometric Framework for Nonlinear Dimensionality Reduction*, Science **290**, 290(5500), 2319-23, 2000.

[152] H. Thompson, *Calculation of Cartesian Coordinates and Their Derivatives from Internal Molecular Coordinates*, Journal of Chemical Physics **47**, 3407–3410, 1967.

[153] W.S. Torgerson, *Multidimensional Scaling: I. Theory and Method*, Psychometrika **17**(4), 401–419, 1952.

[154] P. Verissimo, M. Raynal, *Time in Distributed System Models and Algorithms*. In: "Advances in Distributed Systems, Advanced Distributed Computing: From Algorithms to Systems", S.K. Shrivastava, S. Krakowiak (Eds.), Springer, 1–32, 1999.

[155] 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.

[156] D. Wu, Z. Wu, *An Updated Geometric Build-Up Algorithm for Solving the Molecular Distance Geometry Problems with Sparse Data*, Journal of Global Optimization **37**, 661–672, 2007.

[157] K.P. Wu, J.M. Chang, J.B. Chen, C.F. Chang, W.J. Wu, T.H. Huang, T.Y. Sung, W.L. Hsu, *RIBRA–an Error-Tolerant Algorithm for the NMR Backbone Assignment Problem*, Journal of Computational Biology **13**(2), 229–273, 2006.

[158] Y-C. Wu, Q. Chaudhari, E. Serpedin, *Clock Synchronization of Wireless Sensor Networks*, IEEE Signal Processing Magazine **28**(1), 124–138, 2011.

[159] D. Wu, Z. Wu, Y. Yuan, *Rigid versus Unique Determination of Protein Structures with Geometric BuildUp*, Optimization Letters **2**, 319–331, 2008.

[160] M.E. Yumer, N.J. Mitra, *Spectral Style Transfer for Human Motion Between Independent Actions*, ACM Transactions on Graphics **35**4, 8 pages, 2016.

[161] D. Zelazo, A. Franchi, H.-H. Bülthoff, P. Robuffo Giordano, *Decentralized Rigidity Maintenance Control with Range Measurements for Multi-Robot Systems*, The International Journal of Robotics Research **34**(1), 105–128, 2015.

[162] H. Zhang, W.W. Hager, *A Nonmonotone Line Search Technique and its Applications to Unconstrained Optimization*, SIAM Journal of Optimization **14**(4), 1043–1056, 2004.