In Proceedings of Workshop on Computational Optimization 2020

# MD-JEEP: a New Release for Discretizable Distance Geometry Problems with Interval Data

A. Mucherino,* D.S. Gonçalves,† L. Liberti,‡ J-H. Lin,§
C. Lavor,¶ N. Maculan‖

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

†Centro de Ciências Físicas e Matemáticas, Universidade Federal de Santa Catarina, Florianópolis, Brazil.
douglas.goncalves@ufsc.br

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

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

¶Department of Applied Mathematics (IMECC-UNICAMP), University of Campinas, Campinas (SP), Brazil.
clavor@unicamp.br

‖COPPE, Federal University of Rio de Janeiro, Rio de Janeiro (RJ), Brazil.
maculan@cos.ufrj.br

***Abstract*—With the most recent releases of MD-JEEP, new relevant features have been included to our software tool. MD-JEEP solves instances of the class of Discretizable Distance Geometry Problems (DDGPs), which ask to find possible realizations, in a Euclidean space, of a simple weighted undirected graph for which distance constraints between vertices are given, and for which a discretization of the search space can be supplied. Since its version 0.3.0, MD-JEEP is able to deal with instances containing interval data. We focus in this short paper on the most recent release MD-JEEP 0.3.2: among the new implemented features, we will focus our attention on three features: $(i)$ an improved procedure for the generation and update of the boxes used in the coarse-grained representation (necessary to deal with instances containing interval data); $(ii)$ a new procedure for the selection of the so-called discretization vertices (necessary to perform the discretization of the search space); $(iii)$ the implementation of a general parser which allows the user to easily load DDGP instances in a given specified format. The source code of MD-JEEP 0.3.2 is available on GitHub, where the reader can find all additional details about the implementation of such new features, as well as verify the effectiveness of such features by comparing MD-JEEP 0.3.2 with its previous releases.**

## I. INTRODUCTION

Let $G = (V, E, d)$ be a simple weighted undirected graph, where vertices represent given objects (depending on the applications), and edges between two vertices indicate that the relative distance between the two vertices is known [6]. The weight function $d$ associates the numerical value of the distance to every edge of $E$. This numerical value can be represented either by a singleton (in this case, we say that the distance is *exact*), or rather by an interval representing several possible distance values, delimited by a lower and an upper bound. In the general case, therefore, for two vertices $u$ and $v \in V$ for which $\{u, v\} \in E$, the distance $d(u, v)$ is an interval $[\underline{d}(u, v), \bar{d}(u, v)]$.

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

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

*exists such that*

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

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

The function $x$ is called a *realization* of the graph $G$. We say that a realization $x$ that satisfies all constraints in equ. (1) is a *valid realization*. The DGP has several interesting applications, such as the one arising in structural biology for the determination of protein structures (see for example [1]), and the sensor network localization problem [4]; the reader is referred to [9] for more information about the applications.

In the last years, we have been focusing our research on a special class of DGP instances where the search space can be discretized and reduced to a tree, by transforming in this way the problem into a combinatorial problem [8]. Let $E'$ be the subset of the edge set $E$ such that the weight associated

to the edges are "degenerate" intervals, i.e. intervals such that $\underline{d}(u,v) = \bar{d}(u,v)$.

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

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

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

We will refer to assumptions **(a)** and **(b)** as the *discretization assumptions*. Such assumptions can be verified once a vertex ordering has been associated to the vertex set $V$, which we call a *discretization order* when the two assumptions are satisfied. For more details about discretization orders, the reader is referred to [3], [13].

In the following, for a given vertex $v$, we will refer to all vertices $u$ such that $u < v$ and $\{u,v\} \in E$ as *reference vertices*. When constructing the realization $x$ in the vertex ordering given by the discretization order, feasible positions for the reference vertices have already been computed when one is searching for positions for the current vertex $v$. The positions for the reference vertices, together with the corresponding distances, can therefore be exploited for defining the set of feasible positions for $v$.

Not all reference vertices are however necessary for the definition of a *preliminary set* of possible positions for $v$. As assumption **(b)** suggests, only $K$ reference vertices are actually necessary. However, not all possible subsets of $K$ reference vertices are able to satisfy assumption **(b)**, i.e. at least $K-1$ vertices need to be connected to a distance which is considered as exact (belonging to the subset $E'$), and they need to admit a realization for which the volume $\mathcal{V}_S(u_1,u_2,\ldots,u_K)$ is strictly positive (see definition above). We refer to the selected subset of reference vertices as *discretization vertices*; similarly, the distances related to discretization vertices are called *discretization distances*. Notice that the concept of discretization vertex and distance is local and related to the current vertex $v$.

Historically, the word "discretization" has been employed because the corresponding set of vertices and distances (when they are exact) allows us to reduce the set of feasible positions for $v$ to a discrete set; when not enough exact distances are present, however, so that the $K^{th}$ distance is actually represented by a nondegenerate interval (for which $\underline{d}(u,v) < \bar{d}(u,v)$), then the preliminary position set for the current vertex $v$ is instead continuous. In the latter case, this set has the property of being the set with minimal dimensionality that can be obtained by exploiting the smallest subset of reference distances: the use of any other available distance (not marked as a "discretization" distance) would in fact not reduce the dimensionality of the set, which is bounded to remain equal to 1 (only the use of an exact distance can make the dimensionality drop to 0, which goes against our hypothesis).

In the general case [8], the preliminary position set which can be obtained by using the discretization vertices and distances for a given vertex $v$ consists of two singletons (when all discretization distances are exact), or two arcs (otherwise). The Branch-and-Prune (BP) algorithm (see Section II) is based on the idea of constructing and exploring a search tree containing, at every layer $v$, the subsets of vertex positions extracted from the preliminary position sets. In case of a discrete set, every position can be assigned to a different tree node; otherwise, every node can rather contain the disjoint components of the set (corresponding to the two arcs). The additional distances (not used for the construction of the preliminary set) can be exploited to verify the feasibility of the points assigned to the tree nodes: when none of the node points are feasible w.r.t. these additional distances, then the corresponding branch of the tree can be pruned. For this reason, these additional distances are named *pruning distances*. Notice that, even if locally the nodes can contain continuous geometrical objects having dimension up to 1, the tree is a discrete structure and the general problem is therefore combinatorial, even if locally continuous. This particular structure of the search tree has in fact inspired the BP algorithm.

The first releases of MD-JEEP were able to deal with instances consisting of exact distances only [10]. Since its version 0.3.0, MD-JEEP is on GitHub[1] and implements a coarse-grained representation for the nodes of the search tree, which makes it possible to solve instances containing interval data (more details will be given in Section II). The basic idea behind the coarse-grained representation is to assign, to every node of the search tree, a pair consisting of one selected vertex position and of a $K$-dimensional box containing additional positions that are feasible w.r.t. all reference distances. In fact, as the search proceeds by realizing more and more vertices by following the discretization order, more and more pruning distances need to be verified and satisfied, so that the initial selections of vertex positions in the $K$-dimensional boxes may not always be feasible. In case the minimum and maximum distance between pairs of boxes indicates that the distances may be satisfied by other points in the boxes, then a *refinement step* can be performed, which basically consists in selecting new positions for previous vertices inside their own boxes.

In this short paper, we will present some new features introduced in the last release of MD-JEEP (version 0.3.2), which mainly aim at improving the performances of the implemented BP algorithm. In Section II, we will give a brief description of the BP algorithm and of the coarse-grained representation. In Section III, we will present three of the most relevant features introduced in last MD-JEEP release: a new

[1] https://github.com/mucherino/mdjeep

procedure to create and expand the $K$-dimensional boxes used in the coarse-grained representation, two new small procedures for the selection of the discretization vertices (from the set of reference vertices), and a general parser for loading instances having various formats (allowing in this way a larger target of applications without the need of format conversions). The source code of MD-JEEP 0.3.2, as well as the code of its previous releases, is available on GitHub, together with some sets of DDGP instances: the effectiveness for these newly introduced features can as well be verified by the reader. Section IV will conclude the paper with some future works.

## II. A COARSE-GRAINED REPRESENTATION FOR THE BP ALGORITHM

The basic idea behind the Branch-and-Prune (BP) algorithm is to construct the search tree by performing a depth-first tree search where new potential candidate positions (to be assigned to tree nodes) for the current vertex $v$ are computed by using its discretization vertices and distances, and by verifying the feasibility of such new positions by exploiting the pruning distances [5]. When a new position is feasible, then the branch having this position as a root is subsequently explored, until a leaf node of the tree is reached; otherwise, when the position does not respect the pruning distances, then this new branch is *virtually* pruned: it is removed from the tree which one may construct by using only discretization distances, but in the implementations it is actually not generated at all.

The coarse-grained representation initially proposed in [12] is based on the idea to assign, to every node of the search tree, a $K$-dimensional box $B_v$ containing feasible positions for the current vertex $v$, together with a selected position $x_v \in B_v$. The function

$$z : v \in V \longrightarrow (x_v, B_v) \in \mathbb{R}^K \times \mathbb{R}^{2K}$$

is constructed as the search proceeds over the tree branches. When the vertex $u$ is used as a reference for current vertex $v$, the selected position $x_u$ is the one used as a reference to compute the preliminary position set for the vertex $v$. However, this position $x_u$ can be subsequently changed to another one in $B_u$ to allow satisfying the discretization, as well as the pruning distances, available at the tree layer $v$. The change of the selected position in the set $B_u$ is performed in the part of the algorithm that we name the *refinement step*. Since the version 0.3.0 of MD-JEEP, the refinement step is performed by invoking a Spectral Projected Gradient (SPG) with non-monotone line-search [2].

Once a tree leaf node is reached, the solution to the DDGP instance can be simply obtained by extracting the positions $x_v$ from the function $z$. The use of the boxes $B_v$ can then be useful to verify how different this latest found solution is from other possible solutions that the algorithm will encounter in the further exploration of the tree (for more details, see [11] and the *resolution parameter* recently introduced in BP).

## III. NEW IMPLEMENTED FEATURES

This section describes the three main features recently implemented in MD-JEEP; we will make reference to the release MD-JEEP 0.3.2. For lack of space, we will not present computational experiments in this short paper: the reader can easily verify, by using for example the instances available on MD-JEEP's GitHub repository, the usefulness of these new features by comparing the performances of MD-JEEP 0.3.2 with its previous versions. As for example, MD-JEEP 0.3.1 was not able to solve (the given answer was: 0 found solutions) some of the instances in `proteinSet2`. This set of instances (available on the repository) consists in the same instances of `proteinSet1`, but where the distances are given at low resolution (only 3 decimal digits).

### A. Bound expanding procedure

Since the version 0.3.0 of MD-JEEP, it was empirically noticed that the size of the boxes $B_v$ (over the various dimensions) was too small to have a successful refinement step in the BP algorithm (see Section II). This issue was initially solely attributed to the convergence properties of the SPG method, where it is known that too tight bounds (the size of the boxes $B_v$) may harm its capability to converge to a local optimum. For this reason, since the version 0.3.0 of MD-JEEP, even if initially in a very primitive form, a *bound expanding* procedure was included in the code.

A recent deeper analysis shows however that the necessity to expand the boxes does not come solely from the need of less stringent bounds in SPG. Fig. 1 (left-hand side) shows an illustration in 2D of the procedure to generate the position $x_v$ (in green) and the box (in gray) from a computed arc (in magenta) for a three-dimensional instance (see Section II). At the center of the dashed circle, it is indicated that its center depends on both the reference vertices $u_1$ and $u_2$; the reference vertex $u_3$ is used for delimiting the arcs on the circle. Except for the very first vertices having a small rank in the discretization ordering (for which the preliminary position sets are discrete), the boxes related to $u_1$ and $u_2$ contain an infinite number of possible positions for the two reference vertices, from which two positions, say $x_u^1$ and $x_u^2$, are selected. Moreover, these two positions $x_u^1$ and $x_u^2$ are subject to change after every refinement step. When the box in Fig. 1 (left-hand side) is constructed, however, only the currently selected positions $x_u^1$ and $x_u^2$ are taken into consideration.

In the hypothesis where the distance between $u_1$ and $u_2$ is fixed to an exact value, the radius of all circles, obtained with different positions $x_u^1$ and $x_u^2$, is constant. Its placement, however, *floats* in the space, and so does its projection in 2D (see Fig. 1, right-hand side). There are therefore several arcs to take into consideration, which depend on the positions of the reference vertices: the actual region of space containing the feasible positions for the current vertex $v$ is actually a larger box (in blue in the picture).

From a technical point of view, it is not advisable to compute *all* such arcs to construct the boxes $B_v$. Rather, in MD-JEEP 0.3.2, we have implemented a simple heuristic for the bound expanding procedure where, from the initial box for the current vertex $v$ obtained as in the previous MD-JEEP

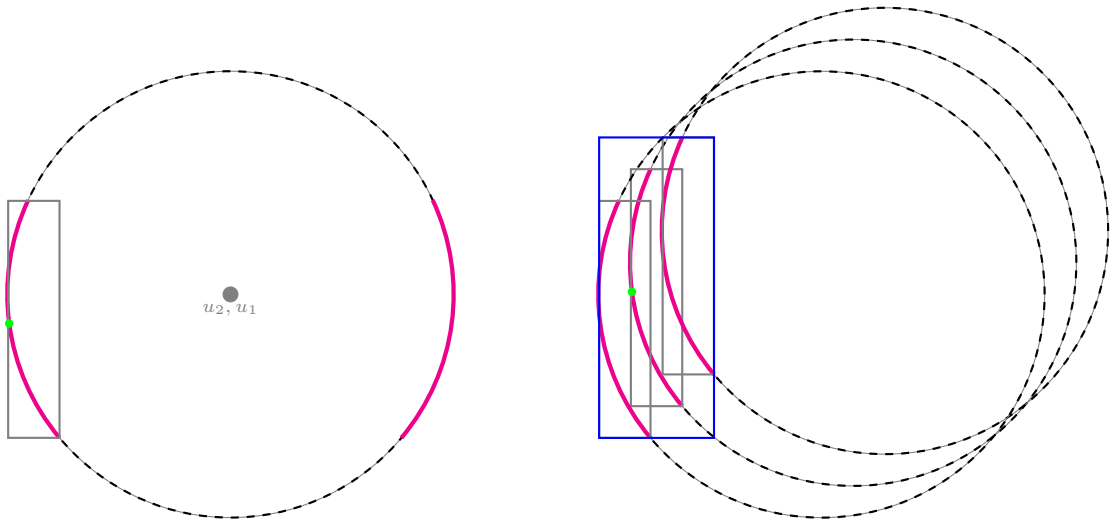


Fig. 1. On the left-hand side, the projection in 2D of two arcs representing the positions that are feasible w.r.t. the discretization distances for a given vertex $v$, from which the position $x_v$ (in green) and the box $B_v$ (in gray) can be computed. On the right-hand side, we have a similar illustration with several different positions for $u_1$ and $u_2$, which shows that the actual box (in blue) of feasible positions is larger. For simplicity, it is supposed that the distance between $u_1$ and $u_2$ is exact, and that the arc length does not depend on the positions of the reference vertices.

versions (see Fig. 1, left-hand side), we keep expanding this box as far as the new added box layer contains points that satisfy all available distances (both discretization and pruning distances). The selected position (in green in Fig. 1) is still extracted from the initial arc.

Naturally, what is obtained for $B_v$ is only an approximation of the actual box depicted in Fig. 1 (right-hand side), but our initial experiments where BP is integrated with this heuristic are already providing promising results. In fact, MD-JEEP 0.3.2, equipped with this heuristic for the definition of the boxes, is able to perform better than its previous versions. The technical details about the implementation of the heuristic can be found on the GitHub repository.

### B. Selecting the discretization vertices

The DDGP general case is the one where the set of discretization distances consists of $K-1$ exact distances, and 1 interval distance. There can however be some vertices that, in the given discretization order, refer only to exact discretization distances. In previous MD-JEEP versions, the selection of such discretization distances was performed by simply choosing the distances to the vertices that are closer in rank to the current vertex $v$. In MD-JEEP 0.3.2, we have implemented two new procedures for the selection of the discretization distances.

In both procedures, the main idea is to attempt the selection of the discretization distances that are likely to lead to the least error propagation. In case our set of reference distances contains $K$ exact distances, then we have no possible choice: this set will be our set of discretization distances. But if more than $K$ exact distances are actually available, it is possible to verify all combinations of selected distances for which the corresponding reference vertices form a clique: we therefore choose the set for which the angles formed by pairs of reference vertices is as far as possible from a multiple of $\pi/2$. Formulae for the computation of candidate positions for the vertices make in fact use of trigonometric functions such as *sine* and *cosine* [7], and it is well-known that an error in an angle near $\pi/2$ (or one of its multiples) can be consistently amplified in the sine and cosine values.

Finally, in case only $K-1$ reference distances are exact and the $K^{th}$ distance needs to be selected among the available distances represented by an interval, then we simply make the choice of selecting the interval distance with the smallest range.

### C. Adding a general parser

One of the most important technical features included in the current release of MD-JEEP is its new parser. Naturally, this is only a technical feature and does not provide any help in the solution of DDGP instances, but we decided to devote one short section to this feature because it opens the possibility for an easier exploration of the use of MD-JEEP in new applications, where the research community may have been using file formats different from the ones MD-JEEP was able to read until its previous release.

MD-JEEP 0.3.2 introduces the "MD-JEEP files" (MDfiles), with extension `mdf` (an example is given in Fig. 2). This is basically a text file containing the main specifications for the DDGP instances to be solved. In the file, every *field* is supposed to begin with its string identifier, followed by a colon and then by its name. For example, a colon needs to be positioned between the field `instance` and the instance name. Once specified in the file, every field can be followed by a certain number of lines starting with the key-word `with`, which allows the user to define the value of the attributes related to the current field. The syntax is similar: the key-word `with` needs to be followed by the attribute name, and then by its value preceded by a colon. For example, for the

```
# an example of MDfile #

# instance field
instance: protein
with file: instances/0.3/proteinSet2/1hj0.nmr
with format: Id1 Id2 groupId1 groupId2 lb ub Name1 Name2 groupName1 groupName2
with separator: ' '

# method field
method: bp
with resolution: 5.0
with tolerance: 0.001
with maxtime: 6000

# refinement field
# (all attribute values below are the default values)
refinement: spg
with eta: 0.99
with gamma: 1.e-4
with epsobj: 1.e-7
with epsg: 1.e-8
with epsalpha: 1.e-12
with mumin: 1.e-12
with mumax: 1.e+12
```

Fig. 2. An example of MDfile.

field `instance`, the attribute `file` needs to be followed by the a string with the name of the text file containing the distance list defining the DDGP instance.

The `format` attribute of the `instance` field is what gives the new parser a general-purpose type of flexibility in accepting different file formats. This attribute is a string of characters specifying the meaning of every element in the generic line of the distance list (contained in the text file specified after the key-word `file`). With this new format specification, MD-JEEP is able to point to the necessary information such as the identifiers of the two vertices (`Id1` and `Id2`) related to this distance specified at the current line, the value of the lower (`lb`) and upper (`ub`) bound for this distance, and other additional information. When some information contained in the file is not strictly necessary for MD-JEEP to solve the instance, then the key-word `ignore` may be employed as for a format element. Additional details about this new general parser can be found in the documentation available on the GitHub repository (see `README` file).

## IV. Conclusions

We have presented three of the most important features introduced in the latest release of MD-JEEP. They represent another step ahead in the development of a general tool for solving DDGP instances. As for example, the implemented heuristic for expanding the boxes $B_v$ used in the coarse-grained representation in the BP algorithm may be replaced, in the near future, with a more efficient deterministic procedure. We expect then to provide more formal descriptions, as well as pseudo-codes, for these MD-JEEP features. Moreover, a longer term project consists in extending MD-JEEP to instances consisting of *only* interval distances, which represents another important step for the generalization of the software tool.

As for the general parser that was introduced in the current MD-JEEP release, we intend to extend it in the next releases so that it will be able to load some more complex data formats, such as the ones that are generally employed by the structural biology community. One example is given by a certain number of data formats that are used for storing information obtained by Nuclear Magnetic Resonance (NMR) [1].

## Acknowledgments

AM, LL and JHL are supported by the international project MultiBioStruct funded by ANR in France (ANR-19-CE45-0019) and by the Ministry of Science and Technology of Taiwan (MoST 109-2923-M-001 -004 -MY3). DG and CL are supported by CNPq. DG is also supported by CAPES/Print Process 88881.310538/2018-01, and CL is also thankful to FAPESP. NM thanks the COPPETEC Foundation.

## References

[1] 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: [9], 377–412, 2013.

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

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

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

[5] 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.

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

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

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

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

[10] 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.

[11] A. Mucherino, J-H. Lin, *An Efficient Exhaustive Search for the Discretizable Distance Geometry Problem with Interval Data*, IEEE Conference Proceedings, Federated Conference on Computer Science and Information Systems (FedCSIS19), Workshop on Computational Optimization (WCO19), Leipzig, Germany, 135–141, 2019.

[12] A. Mucherino, J-H. Lin, D.S. Gonçalves, *A Coarse-Grained Representation for Discretizable Distance Geometry with Interval Data*, Lecture Notes in Computer Science **11465**, Lecture Notes in Bioinformatics series, I. Rojas et al (Eds.), Proceedings of the $7^{th}$ International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO19), Part I, Granada, Spain, 3–13, 2019.

[13] J. Omer, A. Mucherino, *Referenced Vertex Ordering Problem: Theory, Applications and Solution Methods*, HAL open archives (hal-02509522, version 1), March 2020.