Proceedings of the Federated Conference on Computer Science and Information Systems pp. 591–599 DOI: 10.15439/2016F213 ACSIS, Vol. 8. ISSN 2300-5963

A New Approach to the Discretization of Multidimensional Scaling

W. Gramacho*†, A. Mucherino†, J-H. Lin‡, C. Lavor§

*Federal University of Tocantins, Palmas-TO, Brazil.
wgramacho@uft.edu.br

†IRISA, University of Rennes 1, Rennes, France.
antonio.mucherino@irisa.fr, warley.gramacho-da-silva@irisa.fr

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

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

***Abstract*—Given a set of points in a Euclidean space having dimension $K > 0$, we are interested in the problem of finding a realization of the same set in a Euclidean space having a lower dimension. In most situations, it is not possible to preserve all available interpoint distances in the new space, so that the best possible realization, which gives the minimal error on the distances, needs to be searched. This problem is known in the scientific literature as the Multidimensional Scaling (MDS). We propose a new methodology to discretize the search space of MDS instances, with the aim of performing an efficient enumeration of their solution sets. Some preliminary computational experiments on a set of artificially generated instances are presented. We conclude our paper with some future research directions.**

I. Introduction

GIVEN a set of points X in a Euclidean space $\mathbb{R}^K$, with $K > 0$, Multidimensional Scaling (MDS) consists in finding a realization of X in $\mathbb{R}^k$, with $0 < k < K$, such that the interpoint distances in $\mathbb{R}^K$ are preserved as much as possible [3]. The *initial* dimension K is generally very large, while the new dimension k is generally a priori unknown. However, for a fixed *destination* dimension k, the MDS can be seen as a particular class of the Distance Geometry Problem (DGP) [16]. In fact, in the DGP, suitable embeddings of a given simple weighted undirected graph $G = (V, E, d)$ are searched, in a way that the distances between embedded vertices u and $v \in V$ are as close as possible to the weights on the edges $(u, v) \in E$, when available. In the MDS, the graph G can be simply deduced from the original set of points in $\mathbb{R}^K$. A *valid embedding* is an embedding of G satisfying all distance constraints, with a given tolerance.

In recent works, it was shown that the DGP can be discretized when some particular assumptions are satisfied [17]. The discretization makes it possible to work with a finite search space, which is otherwise continuous. This search space has the structure of a tree, which is binary when all available distances can be considered as exact. The DGP is NP-hard [27], and even if its discretization does not reduce its complexity [14], it allows for employing a Branch & Prune (BP) framework for an ad-hoc exploration of the discretized search space [14], [24]. The present work is a preliminary step for the *discretization* of the MDS. In this work, in order to mainly focus on the problem discretization, we will suppose that the topology of the embeddings in both dimensions K and k can be represented well by considering all real inter-point distances.

We warn the reader that a previous work, devoted to the discretization of the MDS, was already published in [1]. However, differently from that work, we will consider, in our first analysis, the basic MDS without any additional constraints. This decision was taken with the aim of developing, first of all, an efficient procedure for the discretization of the MDS, to be extended thereafter for tackling more complex problems. In fact, differently from [1], the algorithms for discretized MDS that we propose in this paper are particularly tailored to this special class of problems.

The rest of the paper is organized as follows. In Section II, we will focus on previous works for the discretization of the DGP. Then, in Section III, we will make a parallel between the DGP and the MDS, while trying to extend and adapt the methodologies, already developed for the DGP, for the discretization of the MDS. Our computational experiments will be presented in Section IV. Finally, Section V will conclude the paper.

II. Distance Geometry Problem

The Distance Geometry Problem (DGP) consists in embedding a simple weighted undirected graph $G = (V, E, d)$ in a k-dimensional space so that all weights d_{uv} on the edges of G are realized as distances between the positions assigned to its vertices [16]. More formally, the DGP asks whether it is possible to find an embedding $x : V \to \mathbb{R}^k$ satisfying constraints based on the available edge weights:

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

Notice that, in these distance constraints, we can obtain the equation of a hyper-sphere in $\mathbb{R}^k$ by fixing one of the two vertices in a certain position. The discretization of the DGP is based on the idea to intersect as many hyper-spheres as necessary to obtain a discrete set of possible positions for a given vertex $v \in V$ (see Section II-A).

A classical approach to the DGP is to reformulate it as an unconstrained global optimization problem [23]. The satisfaction of the constraints based on the distances can be measured by computing the difference between the actual value $||x_u - x_v||$ of the distance, and its expected value d_{uv}. In order to verify the overall satisfaction of the available constraints, a penalty function can be introduced, whose general term is related to the generic constraint. Various penalty functions can be defined for the DGP, and one of the most used penalty functions is the so-called Medium Distance Error (MDE):

$$MDE(x) = \frac{1}{|E|} \sum_{(u,v)\in E} \frac{|\,||x_u - x_v|| - d_{uv}\,|}{d_{uv}}. \qquad (1)$$

Finding the global minimum of this penalty function allows to obtain solutions to the DGP. If all distances are compatible to each other and they are not affected by numerical errors, the MDE value for a valid embedding x needs to be equal to zero.

There are two main applications of the DGP that can be commonly found in recent publications. One application arises in biology, and it is concerned with the identification of protein conformations by exploiting some known interatomic distances that can be obtained by experimental techniques [19]. Another common application is given by the so-called Sensor Network Localization Problem (SNLP) [2], [5], where the positions of the sensors forming a network need to be identified.

As already mentioned above, when some particular assumptions are satisfied, the DGP can be discretized, so that its search space can be reduced from a continuous (and infinite) domain to a discrete (and finite) domain. We will give some details about the discretization of the DGP in Section II-A, and we will focus on discretization orders (vertex orders for G which make the discretization assumptions satisfied) in Section II-B.

A. The Discretization

Let $G = (V, E, d)$ be a simple weighted undirected graph representing a DGP instance. Let $G[\cdot]$ be the subgraph of G induced by a subset of vertices in V; let $\mathcal{V}_S(\cdot)$ be the volume of the simplex defined by the vertices given as arguments. The discretization in dimension $k > 0$ of a DGP instance can be performed when there exists an order defining sequence $r : i \in \mathbb{N} \longrightarrow v \in V \cup \{\Diamond\}$ of length $|r|$ (for which $r_i = \Diamond$ if $i > |r|$) such that the following two assumptions are satisfied:

Algorithm 1 The BP algorithm.

```
1:  BP($i, k, G, r, \varepsilon$)
2:  let $v = r_i$;
3:  compute $x'_v$ in dimension $k$;
4:  if ($x'_v$ is feasible with tolerance $\varepsilon$) then
5:     if ($i = |r|$) then
6:        print new solution;
7:     else
8:        BP($i+1, k, G, r, \varepsilon$);
9:     end if
10: end if
11: compute $x''_v$ in dimension $k$;
12: if ($x''_v$ is feasible with tolerance $\varepsilon$) then
13:    if ($i = |r|$) then
14:       print new solution;
15:    else
16:       BP($i+1, k, G, r, \varepsilon$);
17:    end if
18: end if
```

(a) $G[\{r_1, r_2, \dots, r_k\}]$ is a clique;
(b) $\forall i \in \{k+1, \dots, |r|\}$, there exist k "reference" vertices, i.e. there exist $j_1, j_2, \dots, j_k$ such that
   1) $j_1 < i$, $j_2 < i$, …, $j_k < i$;
   2) $\{(r_{j_1}, r_i), (r_{j_2}, r_i), \dots, (r_{j_k}, r_i)\} \subset E$;

   for which
$$\mathcal{V}_S(r_{j_1}, r_{j_2}, \dots, r_{j_k}) > 0.$$

Vertex orders satisfying assumptions (a) and (b) are named *discretization orders*: more details about these special orders are given in Section II-B. The class of DGP instances for which at least one discretization order exists for the corresponding graph is named Discretizable DGP (DDGP) [24].

The search space of DDGP instances is finite. It has the structure of a tree, where nodes contain candidate vertex positions, organized layer by layer. In fact, assumption (a) allows us to place the first k vertices given by the vertex order r in k fixed positions. In this way, we can avoid to consider congruent solutions that can be obtained by rotating and/or translating other solutions. Assumption (b) ensures the existence of at least k *reference vertices* for every vertex having a rank $i > k$: a vertex u can play the role of "reference" for a given vertex v if u precedes v in the vertex ordering r, and $(u,v) \in E$. We say that the corresponding distances d_{uv} are the *reference distances* for the vertex v. We exploit the k reference vertices for defining k spheres centered in the reference vertices and having as radius the corresponding reference distances: in dimension k, the intersection of these k spheres provides us with a subset of vertex positions having cardinality 2 [8], [14]. The condition on the volume of the simplex in assumption (b) ensures that this sphere intersection does not provide a full circle (which is obviously not discrete).

We employ a Branch & Prune (BP) algorithm [25] for the solution of DDGP instances (see Alg. 1 for a sketch). The algorithm recursively calls itself for the exploration of the search tree. In the algorithm call, i is the current rank in the given vertex order r: it is supposed that the coordinates of all vertex positions on the current branch are stored in the global memory. The value of k corresponds to the dimension in which we wish to embed the graph G. ε is our tolerance for the errors that can affect the generated vertex coordinates: as soon as new vertex positions are computed by performing the sphere intersection, in fact, their feasibility is verified by exploiting additional distances that were not used in the discretization process. In order to do so, we consider the corresponding terms of equation (1), and we declare as “infeasible” a new generated position if at least one of such terms is greater than the predefined tolerance ε. Once a new solution is found by the BP algorithm, the quality of such a solution can be verified by applying equation (1): since all its terms are smaller than our tolerance ε, we expect to obtain good-quality solutions.

The complexity of one single BP call corresponds to the complexity of computing twice a new vertex position, and of verifying whether the exploration of the tree should continue in those two directions or not. If, for example, x'_v is feasible (see Alg. 1), then this position could be part of a solution, and therefore the branch of the binary tree rooted at x'_v needs to be explored. In this case, the algorithm invokes itself for computing the possible positions for the next rank. Instead, if for example x''_v is not feasible with the given tolerance, then the current branch does not contain any solution. It is therefore pruned: the algorithm does not invoke itself in this case. This algorithm phase is fundamental and named *pruning phase*.

There are two main approaches to deal with the uncertainty on the values of the available distances. In [15], for example, intervals are used for representing such an uncertainty, and the terms of equation (1) are adapted for providing a positive measure only when the actual distance is not contained in the given interval (for both feasibility verification during the execution of BP, and computation of the MDE function once a new solution is found). However, in some applications (such as the MDS), the uncertainty on the distance values is not known a priori, and therefore our approach will consist in considering instances with only exact distances, which we will need to tackle by admitting larger values for the tolerance ε. In other words, since the range of the intervals to be assigned to uncertain distances cannot be predefined, we will try to define them during the execution of the BP algorithm: the tolerance ε represents in this case the maximal allowed interval range.

B. Vertex Orders

A *discretization order* is a vertex order associated to the graph G such that the discretization assumptions (a) and (b) are satisfied [9], [13]. Given a graph G representing a DGP instance, one question that can immediately arise is whether this instance belongs to the DDGP class or not. When instances are stored in text files, an order may be implicitly associated to the vertices of G, but such an order may not satisfy the discretization assumptions. In order to find suitable discretization orders, a greedy algorithm was proposed in [13] and extended subsequently for dealing with interval distances [20]. A heuristic, that has as worst-case complexity the one of the greedy algorithm, was also proposed in [10], for dealing more efficiently with very large instances.

In some particular applications (the reader can refer for example to [15]), additional assumptions on the discretization orders are required, and the problem of finding a discretization order may become NP-hard [4]. Naturally, the above-mentioned greedy algorithm cannot guarantee the generation of a vertex order that, apart from the discretization assumptions, could as well satisfy the additional ones.

For example, we say that a discretization order satisfies the *consecutivity assumption* if the reference vertices for a given vertex v, and v itself, are consecutive in the order. Orders satisfying the consecutivity assumption can be seen as sequences of overlapping cliques, which can be searched on the pseudo de Bruijn graphs introduced in [21]. Even if it implies the execution of an exponential search, the use of such pseudo de Bruijn graphs aided the identification of discretization orders satisfying the consecutivity assumption for the backbones of protein instances.

The complexity of this ordering problem can also increase when *optimal* discretization orders are required. We say that an order is optimal when the rank assigned to every vertex maximizes some predefined objective functions [22]. For example, when low-rank vertices have the maximal number of reference vertices, a direct impact on the width of the corresponding search tree can be observed: it becomes smaller. Several objectives, together with their priority orders, can be defined and used for finding optimal discretization orders. If the objectives are given by a set of simple functions, then the problem of finding an optimal discretization order (without any other additional assumption) still has a polynomial complexity, and the greedy algorithm can be extended for dealing with this class of vertex orders [9].

III. Multidimensional Scaling

Since one of the first pioneer papers on this topic [29] in 1952, the MDS received an increasing interest. Surveys on the MDS can be found in the scientific literature: a recent example is [11], published in 2013. In the cited survey, the MDS 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 (generally 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, are identified. The MDS has a wide range of applications, and the interested reader can refer to the citations in [11] for additional information about these applications.

One of the best-known methods for dimensionality reduction, especially in the field of mathematical statistics, is Principal Component Analysis (PCA) [7], [12]. PCA was proposed by Karl Pearson in 1901 [26], and it consists of an orthogonal transformation that can project an ensemble of the high-dimensional data 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, this 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 by Tenenbaum et al [28]. Unlike PCA, Isomap method has a better ability for successfully capturing the intrinsic global geometric features of the given set of points. It was noted recently, however, that the application of the Isomap method is able to provide good results only when transferring the data between two similar geometries [6].

Let X be a set consisting of points of the Euclidean space $\mathbb{R}^K$. Since the coordinates for each point in X are known, it is possible to compute their relative distances. With this information, we can define a simple weighted undirected graph $G = (V, E, d)$. In the graph, every vertex $v \in V$ represents one single point in X, and an edge $(u, v) \in E$, together with the weight that is associated to it, represents the relative distance between the two corresponding points in X. All graphs G constructed in this way have the following two properties: they are complete, and all the weights associated to the edges (the distances) are exact. These graphs represent instances of the DGP.

A very simple but nice example of MDS is given in [11] 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 precise, 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 translations and rotations). In general, however, solutions where the overall distance information is precisely verified may not exist, so that approximated mappings need to be searched.

Let G be the simple weighted undirected graph defined as described above from a known set in dimension K. In the example of the 4 US cities, the distances are measured by approximating a surface on Earth with a plane containing the 4 cities: a small error is introduced in the distances because of the plane approximation of the Earth region. In general, every graph is embeddable in dimensions $k \geq |V|$ without introducing any error in the distances. However, when the number of vertices in V is large, the dimensionality is huge. The main interest therefore 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. In the example of the 4 US cities, the embedding of the corresponding graph G needs to be searched in dimension $k = 2$. Recall that K is the initial dimension of our MDS instances, while k is the destination dimension.

In this work, we will extend the concept of discretizability to MDS instances (see Section III-A), and we will provide two variants of the BP algorithm that are particularly tailored to the MDS (in both Sections III-A and III-B).

A. The Discretization

The methodology that we propose for discretizing the MDS is strictly related to the previous works on the discretization of the DGP (see Section II-A). The entire theory of the discretization can be in fact inherited almost unchanged: the novelties in this paper are mostly *operational*: we propose some ad-hoc strategies to be integrated in our new algorithms.

Since all relative distances are generally available in MDS instances, the main task that is required for performing the discretization is the selection, for every vertex $v \in V$ that does not belong to the initial clique, of a k-plet of reference vertices. We point out that this task has a fundamental importance. In fact, once a k-plet of reference vertices has been selected for the vertex v, the corresponding reference distances remain fixed in the search tree, i.e. we cannot allow the introduction of any error in the corresponding distances.

The main idea for our first variant on the BP algorithm for the MDS is to try to consider all possible k-plets of reference vertices, and to choose, at each recursive call, the k-plet for which the minimal error is observed during the pruning phase. If this minimal error is larger than a predetermined tolerance $\varepsilon > 0$, then the current branch of the tree is pruned and the search is backtracked.

In order to measure the quality of partial embeddings (up to the current rank of the discretization order), we need to consider the following function, that we name *partial* MDE (pMDE, see equation (1)):

$$pMDE(x, r, i) = \frac{1}{|E(r, i)|} \sum_{(u,v) \in E(r,i)} \frac{|\, ||x_u - x_v|| - d_{uv} \,|}{d_{uv}}, \tag{2}$$

where

$$E(r, i) = \{(u, v) \in E \mid \exists j < i \, : \, u = r_j, \, v = r_i\}.$$

Notice that, while the vertex ordering associated to G is irrelevant for the computation of the MDE, it becomes important for the pMDE. The value of $pMDE(x, r, i)$ is in fact the average error introduced in the partial embedding x of G up to the rank i, in the vertex ordering r.

The sketch of our BP variant for the MDS is given in Alg. 2. The algorithm keeps the general structure of Alg. 1, but, every time it invokes itself recursively, it verifies all the possible k-plets of reference vertices for the current vertex v, and chooses the best k-plet p_{best} in terms of introduced error pMDE. We are aware that the choice of the k-plet is greedy, and that it might have, in theory, a negative impact on the computations. However, our computational experiments (see Section IV) show that our methodology works quite well in conjunction with the algorithm pruning phase, which does not allow the overall introduced error to grow more than desired.

Algorithm 2 A variant of the BP algorithm for the MDS.

```
1: BP($i, k, G, r, \varepsilon$)
2: let $v = r_i$;
3: for (every $k$-plet $p$ of reference vertices for $v$) do
4:     compute $x'_v$ and $p$MDE($x'_v$) in dimension $k$;
5:     compute $x''_v$ and $p$MDE($x''_v$) in dimension $k$;
6: end for
7: let $p_{best}$ be the $k$-plet leading to the lowest $p$MDE;
8: let $\hat{x}'_v$ and $\hat{x}''_v$ be the two positions for $v$ given by $p_{best}$;
9: if ($pMDE(\hat{x}'_v, r, i) < \varepsilon$) then
10:     if ($i = |r|$) then
11:         print new solution;
12:     else
13:         BP($i+1, k, G, r, \varepsilon$);
14:     end if
15: end if
16: if ($pMDE(\hat{x}''_v, r, i) < \varepsilon$) then
17:     if ($i = |r|$) then
18:         print new solution;
19:     else
20:         BP($i+1, k, G, r, \varepsilon$);
21:     end if
22: end if
```

In comparison with Alg. 1, Alg. 2 has a higher complexity. The worst-case complexity is achieved (as in Alg. 1) when the pruning phase is never able to prune away infeasible tree branches. In this case, the algorithm needs to invoke itself

$$\sum_{i=k+1}^{|r|} 2^{i-k}$$

times. At each call, moreover, it is necessary to select the best k-plet of reference vertices in a set of $i-1$ vertices, where i is the rank in r of the current vertex. The number of combinations of k objects among $i-1$ objects, without repetitions and without assigning a specific order to them, are

$$\frac{(i-1)!}{k!(i-k-1)!}.$$

Therefore, the complexity of finding all these possible combinations depends upon the current layer of the tree i. As a consequence, the worst-case complexity of Alg. 2 is

$$\sum_{i=k+1}^{|r|} \frac{2^{i-k}(i-1)!}{k!(i-k-1)!}.$$

B. Vertex Orders

The reader might have remarked that, while vertex orders have been presented as a fundamental concept for the discretization of DGPs, there is only a quick mention to vertex orders in the previous section, devoted instead to the discretization of the MDS. In fact, the completeness of graphs G representing MDS instances ensures that all vertex orders allow for the discretization (the discretization assumptions in Section II-A are always satisfied). However, as already done for the DGP (see Section II-B), we may want to select optimal vertex orders for a given instance of the MDS, which may help us in improving the performances of the BP algorithm.

In terms of partial orders, a vertex order that allows for the discretization of the MDS can be any order with an initial k-clique at the first rank, and all other vertices at rank two. In other words, once the initial clique has been embedded, as well as some other vertices until the vertex v, any of the remaining vertices can be embedded as next. This observation leaded us to develop another variant of the BP algorithm for the MDS.

Consider we have already embedded m vertices of our graph G, with $m > k$ and $m < |V|$. Instead of considering a predefined vertex ordering, we can assume here that every non-embedded vertex can be the candidate to be embedded as next. Moreover, for each of such candidate vertices, different k-plets can be selected to play the role of reference vertices in the discretization (see Section II-A). For every candidate vertex, and for every k-plet, an error on the pruning distances can be computed, and the best vertex and k-plet can be selected together. If the minimal obtained error is greater than our tolerance ε, then the current tree branch needs to be pruned.

The sketch of this second variant of the BP algorithm for the MDS is given in Alg. 3. In the algorithm call, the discretization order r is initially empty, and it is constructed

Algorithm 3 Another variant of BP for the MDS.

```
1: BP($i, k, G, r, \varepsilon, \bar{V}$)
2: for (every new candidate vertex $v \in \bar{V}$) do
3:     for (every $k$-plet $p$ of reference vertices for $v$) do
4:         compute $x'_v$ and $p$MDE($x'_v$) in dimension $k$;
5:         compute $x''_v$ and $p$MDE($x''_v$) in dimension $k$;
6:     end for
7: end for
8: let $\bar{v} \in \bar{V}$ be the vertex leading to the lowest $p$MDE;
9: let $r_i = \bar{v}$;
10: let $p_{best}$ be the $k$-plet leading to the lowest $p$MDE;
11: let $\hat{x}'_{\bar{v}}$ and $\hat{x}''_{\bar{v}}$ be the two positions for $\bar{v}$ given by $p_{best}$;
12: if ($pMDE(\hat{x}'_{\bar{v}}, r, i) < \varepsilon$) then
13:     let $\bar{V} = \bar{V} \setminus \{\bar{v}\}$;
14:     if ($\bar{V} = \emptyset$) then
15:         print new solution;
16:     else
17:         BP($i+1, k, G, r, \varepsilon, \bar{V}$);
18:     end if
19: end if
20: if ($pMDE(\hat{x}''_{\bar{v}}, r, i) < \varepsilon$) then
21:     let $\bar{V} = \bar{V} \setminus \{\bar{v}\}$;
22:     if ($\bar{V} = \emptyset$) then
23:         print new solution;
24:     else
25:         BP($i+1, k, G, r, \varepsilon, \bar{V}$);
26:     end if
27: end if
```

step by step during the several recursive calls. In this situation, backtracking also means changing vertex ordering, because every tree branch may admit a different optimal order. In practice, at each recursive call, the best vertex $\bar{v}$ and the best k-plet, which lead to the minimal pMDE value, are selected at the same time. This double choice is performed, as in Alg. 2, in a greedy manner. In the algorithm call, we also added the set $\bar{V}$, which is supposed to contain the vertices of the graph that have not been yet embedded.

The complexity of this algorithm is evidently higher, in comparison with Algs. 1 and 2. With respect to Alg. 2, an additional task needs to be performed at every recursive call of the algorithm: it is necessary to select the next vertex to be considered. This task requires a search over the remaining vertices to be embedded (with complexity $n - i$, where $n = |V|$) of the optimal one, i.e of the one for which it is possible to identify the best k-plet of reference vertices. Therefore, the overall worst-case complexity of Alg. 3 is

$$\sum_{i=k+1}^{|r|} \frac{2^{i-k}(|V|-1)(i-1)!}{k!(i-k-1)!},$$

which is much higher of the original complexity of Alg. 1.

IV. Computational experiments

A. Generation of MDS instances

We consider artificially generated instances, with $K > 3$ and $k = 3$. Let n be the number of points in the original set $X \subset \mathbb{R}^K$; let $e \in [0,1]$. The procedure we employ for the instance generation consists in the following steps:

- we generate n random points in the cube of $\mathbb{R}^3$ with sides equal to 1;
- we add $K - 3$ coordinates to each point, having random values extracted from the interval $[0, e]$;
- we compute all relative distances between point pairs in $\mathbb{R}^K$;
- we define a simple weighted undirected graph G containing this distance information.

We point out that the error e introduced in these instances is distributed over the set of distances, as well as over the added coordinates, in a quite uniform manner. This property is unlikely to be satisfied by real instances in general. We also remark that the overall introduced error per instance naturally depends upon the value of e, but also on the distance size n.

B. Preliminary experiments

All the experiments presented in this section have been performed on a laptop computer equipped with an Intel Core i5 with 2.4 GHz and 8GB/1600 MHz RAM, running Mac OS X 10.11.3.

Table I shows some experiments where a set of instances that are embeddable in dimension 3 are considered (our introduced error e, in fact, is here set to 0). We fix the destination dimension to $k = 3$, while various initial dimensions K are taken into consideration. The ε value is set to 0.001 in all the experiments, even if $e = 0$, in order to deal with some errors introduced by imprecise computer arithmetics. In all generated instances, the total number of solutions is always 2 (when G is embeddable in dimension k, then the total number of solutions is 2, because all distances are available [18]). We point out however that, in general when an approximated embedding is searched, the introduced errors in the distances may lead to the identification of multiple solutions. The experiments lasting more than 2 hours were aborted. When executing Algs. 1 and 2, a vertex order was predefined. In Alg. 1, moreover, the selected reference vertices are always the k immediate preceding ones in the predefined ordering r. The computational time is measured in seconds.

The table shows that Alg. 2 is able to find better quality solutions w.r.t. the ones obtained by the standard Alg. 1, in terms of MDE. Such an improvement does not appear so evident instead when comparing Alg. 2 and Alg. 3: the MDE values remain mostly unchanged. It seems therefore that it is not important in which place of the ordering a vertex is located, but mostly which its reference vertices are. The computational times that we report reflect the theoretical worse-case complexities provided in Sections III-A and III-B. For instances with $n = 200$, Alg. 3 would have taken more than 2 hours to run to termination.

Table II shows some computational experiments where valid embeddings cannot be identified in the destination dimension k without introducing some errors in the distances. All considered instances have size $n = 20$, and part of the experiments already reported in Table I are here provided again for completeness. As the error e increases, the MDE values in the found solutions show that the quality of the solutions decreases as well. In some cases, Alg. 1 is not able to find any solution, because of the propagation of such errors on our search tree. Alg. 2 is instead always able to get the solutions, and the MDE reflects the initial introduced error e. Alg. 3 is always able to find solutions as well (the increase in the complexity is here not so important because all instances are small), and it finds better quality results in some cases.

Table III shows some experiments with our artificially generated instances having larger size ($n =$ 50, 100 and 200), for larger errors (we consider $e \in \{0.01, 0.02\}$) and where we compare Alg. 2 to Alg. 3 only. The table shows that the quality of the solutions found by the two algorithms is comparable (in this set of experiments, only one time Alg. 3 was able to do better than Alg. 2), but the computational time grows too much when executing Alg. 3. In fact, the instances with $n = 200$ would have taken more than 2 hours, and these experiments were therefore aborted.

Finally, only Alg. 2 is considered in our last table of experiments. These experiments are aimed at stressing our algorithm with high initial dimensions; the destination dimension is still fixed to 3. As already pointed out, the errors introduced in our instances are quite uniformly distributed over the set of distances, and over the coordinates. This is the reason why we consider small errors e, but repeated uniformly over the n vertices forming the instance, and its $K - 3$ additional dimensions. In fact, as Table IV shows, the MDE values

TABLE I
COMPUTATIONAL EXPERIMENTS ON A SET OF INSTANCES THAT ARE EMBEDDABLE IN DIMENSION 3.

instance		Alg. 1		Alg. 2		Alg. 3	
n	K	MDE	time	MDE	time	MDE	time
20	4	10^{-15}	0.00	10^{-17}	0.02	10^{-17}	0.09
20	5	10^{-15}	0.00	10^{-16}	0.03	10^{-16}	0.14
20	8	10^{-14}	0.00	10^{-17}	0.03	10^{-17}	0.14
20	10	10^{-14}	0.00	10^{-17}	0.03	10^{-17}	0.14
50	4	10^{-14}	0.00	10^{-17}	2.94	10^{-17}	28.01
50	5	10^{-15}	0.00	10^{-16}	2.53	10^{-16}	25.77
50	8	10^{-15}	0.00	10^{-17}	2.84	10^{-17}	28.02
50	10	10^{-14}	0.00	10^{-17}	2.81	10^{-16}	28.34
100	4	10^{-13}	0.00	10^{-16}	71.42	10^{-17}	1444.88
100	5	10^{-14}	0.00	10^{-17}	75.23	10^{-17}	1388.16
100	8	10^{-15}	0.00	10^{-17}	68.90	10^{-17}	1035.28
100	10	10^{-13}	0.00	10^{-17}	74.68	10^{-17}	1041.27
200	4	10^{-13}	0.00	10^{-17}	4289.86	-	-
200	5	10^{-14}	0.00	10^{-17}	4272.32	-	-
200	8	10^{-13}	0.00	10^{-17}	2525.45	-	-
200	10	10^{-14}	0.01	10^{-17}	2880.03	-	-

TABLE II
COMPUTATIONAL EXPERIMENTS ON A SET OF SMALL INSTANCES FOR WHICH A VALID EMBEDDING CANNOT BE FOUND IN DIMENSION 3 WITHOUT INTRODUCING AN ERROR ON THE DISTANCES.

instance				Alg. 1		Alg. 2		Alg. 3	
n	K	e	ε	MDE	time	MDE	time	MDE	time
20	4	0	0.001	10^{-15}	0.00	10^{-17}	0.02	10^{-17}	0.09
20	5	0	0.001	10^{-15}	0.00	10^{-16}	0.03	10^{-16}	0.14
20	8	0	0.001	10^{-14}	0.00	10^{-17}	0.03	10^{-17}	0.14
20	10	0	0.001	10^{-14}	0.00	10^{-17}	0.03	10^{-17}	0.14
20	4	0.01	0.01	10^{-3}	0.00	10^{-5}	0.03	10^{-5}	0.14
20	5	0.01	0.01	10^{-3}	0.00	10^{-5}	0.03	10^{-5}	0.14
20	8	0.01	0.01	-	-	10^{-5}	0.03	10^{-5}	0.14
20	10	0.01	0.01	-	-	10^{-3}	0.03	10^{-3}	0.15
20	4	0.02	0.02	10^{-3}	0.00	10^{-3}	0.03	10^{-5}	0.16
20	5	0.02	0.02	10^{-3}	0.00	10^{-3}	0.05	10^{-3}	0.22
20	8	0.02	0.02	10^{-2}	0.00	10^{-3}	0.03	10^{-3}	0.14
20	10	0.02	0.02	-	-	10^{-2}	0.03	10^{-3}	0.15
20	4	0.03	0.03	10^{-2}	0.00	10^{-3}	0.05	10^{-3}	0.21
20	5	0.03	0.03	10^{-3}	0.00	10^{-3}	0.05	10^{-3}	1.09
20	8	0.03	0.03	10^{-2}	0.00	10^{-2}	0.05	10^{-2}	0.22
20	10	0.03	0.03	-	-	10^{-2}	0.03	10^{-2}	0.17

TABLE III
COMPUTATIONAL EXPERIMENTS ON A SET OF LARGER INSTANCES, WHICH ARE AFFECTED BY A LARGER OVERALL ERROR.

instance				Alg. 2		Alg. 3		instance				Alg. 2		Alg. 3	
n	K	e	ε	MDE	time	MDE	time	n	K	e	ε	MDE	time	MDE	time
50	4	0.01	0.01	10^{-5}	2.88	10^{-5}	27.75	50	4	0.02	0.02	10^{-3}	4.80	10^{-3}	44.49
50	5	0.01	0.01	10^{-3}	2.86	10^{-3}	28.10	50	5	0.02	0.02	10^{-3}	4.86	10^{-3}	44.89
50	8	0.01	0.01	10^{-3}	3.09	10^{-3}	29.67	50	8	0.02	0.02	10^{-2}	4.61	10^{-3}	44.25
50	10	0.01	0.01	10^{-3}	3.93	10^{-3}	36.48	50	10	0.02	0.02	10^{-3}	3.31	10^{-3}	32.31
100	4	0.01	0.01	10^{-5}	87.12	10^{-5}	1622.05	100	4	0.02	0.02	10^{-3}	98.19	10^{-3}	1786.72
100	5	0.01	0.01	10^{-3}	120.72	10^{-3}	2188.86	100	5	0.02	0.02	10^{-3}	97.27	10^{-3}	1797.52
100	8	0.01	0.01	10^{-3}	118.10	10^{-3}	2136.80	100	8	0.02	0.02	10^{-3}	130.74	10^{-3}	2360.76
100	10	0.01	0.01	10^{-3}	90.67	10^{-3}	1649.14	100	10	0.02	0.02	10^{-2}	97.31	10^{-2}	1870.78
200	4	0.01	0.01	10^{-5}	5521.94	-	-	200	4	0.02	0.02	10^{-3}	1988.79	-	-
200	5	0.01	0.01	10^{-3}	4407.33	-	-	200	5	0.02	0.02	10^{-3}	4415.64	-	-
200	8	0.01	0.01	10^{-3}	2589.42	-	-	200	8	0.02	0.02	10^{-3}	4049.64	-	-
200	10	0.01	0.01	10^{-3}	3017.70	-	-	200	10	0.02	0.02	10^{-2}	4846.58	-	-

TABLE IV
COMPUTATIONAL EXPERIMENTS ON A SET OF INSTANCES HAVING A HIGH INITIAL DIMENSION.

instance				Alg. 2	
n	K	e	ε	MDE	time
100	10	0.001	0.002	10^{-6}	145.77
100	20	0.001	0.002	10^{-6}	178.47
100	50	0.001	0.002	10^{-5}	142.68
100	100	0.001	0.002	10^{-5}	157.11
100	200	0.001	0.002	10^{-5}	157.61
100	300	0.001	0.002	10^{-3}	154.71
100	400	0.001	0.002	10^{-3}	132.22
100	500	0.001	0.002	10^{-3}	124.23
100	600	0.001	0.002	10^{-3}	120.62

in the found solutions strongly depend upon the difference between destination and initial dimensions. However, with a rather constant computational time, all the experiments were able to provide a good approximation of a valid embedding of G.

V. CONCLUSIONS

When the destination dimension k is fixed, the Multidimensional Scaling (MDS) can be seen as a DGP where an embedding of the graph G is available in dimension K, and it is necessary to find a valid embedding of the same graph in the given destination dimension. Actually, the graph G, which is the input of the DGP with the dimension k, can be generally obtained from the known embedding in dimension K. However, not all distances between vertices in dimension K can be realized in dimension k, and therefore approximated valid embeddings need to be searched.

This work represents the first step for the discretization of the MDS. The main ideas and the main methodology are inherited from previous works on the discretization of the DGP. We proposed two algorithms, that are variants on the BP algorithm, which was previously proposed for solving discretizable DGP instances. Our Alg. 2 seems to achieve the best trade-off between solution quality and increase in complexity w.r.t. the original BP algorithm.

The computational experiments that we have reported in this paper take into consideration a set of artificially generated instances, where the errors on the distances are introduced in a quite uniform manner. In order to deal with more realistic instances, there are several points to be improved in our approach. For example, the simplex inequalities (i.e. the triangular inequalities in dimension 3) need to be verified before executing our algorithms, and, when necessary, some distances may need to be corrected. Because of the nature of the problem, every corrected distance may imply to modify other distances accordingly, in order to avoid invalidating the geometry of the obtained embeddings.

From an algorithmic point of view, moreover, the branching phase of the algorithm can be improved by verifying, every time a given branch is pruned, whether other k-plets of reference vertices can be chosen for the vertices over the same branch. On the one hand, in fact, pruning allows us to focus the searches on the feasible parts of the tree, but with the risk, in presence of errors, to prune too much and obtain no solutions. On the other hand, however, branching over all the possible k-plets of reference vertices can be very expensive. It is necessary therefore to identify the best trade-off.

Finally, as a future work, we may also consider the possibility to let one of the k reference vertices to be related to an interval distance, as it was already done in works related to the discretization of the DGP. The decision not to consider yet interval distances during the discretization process is motivated by the fact that work is still in progress for an efficient management of interval distances during the generation of DGP discrete search spaces.

ACKNOWLEDGMENTS

The present work was entirely performed during WG's 1-year stay at IRISA, Rennes (France), which was funded by the Brazilian program "Ciencias sem Fronteiras". CL also wishes to thank FAPESP and CNPq for financial support.

REFERENCES

[1] 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.

[2] 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.

[3] I. Borg, P.J.F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer Series in Statistics, Second Edition, 355 pages, 2005.

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

[5] Y. Ding, N. Krislock, J. Qian, H. Wolkowicz, *Sensor Network Localization, Euclidean Distance Matrix Completions, and Graph Realization*, Optimization and Engineering **11**(1), 45–66, 2010.

[6] 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.

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

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

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

[10] W. Gramacho, D.S. Gonçalves, A. Mucherino, N. Maculan, *A new Algorithm to Finding Discretizable Orderings for Distance Geometry*, Proceedings of Distance Geometry and Applications (DGA13), Manaus, Amazonas, Brazil, 149–152, 2013.

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

[18] 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.

[19] T.E. Malliavin, A. Mucherino, M. Nilges, *Distance Geometry in Structural Biology: New Perspectives*. In: "Distance Geometry: Theory, Methods and Applications", A. Mucherino, C. Lavor, L. Liberti, N. Maculan (Eds.), Springer, 329–350, 2013.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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 Third International Congress on Mathematical Software (ICMS10), Kobe, Japan, 186–197, 2010.

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

[27] 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.

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

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