# Bound Smoothing with a Biased Random-Key Genetic Algorithm

**Thiago Alves de Queiroz**[1], **Ivan da Silva Sendin**[2], **and Marcos Aurélio Batista**[3]
[1]Institute of Mathematics and Technology, Federal University of Goiás - Campus Catalão, Catalão, Goiás, Brazil.
E-mail: taq@ufg.br.
[2]Faculty of Computing, Federal University of Uberlândia, Uberlândia, Minas Gerais, Brazil.
E-mail: sendin@ufu.br.
[3]Department of Computer Science, Federal University of Goiás - Campus Catalão, Catalão, Goiás, Brazil.
E-mail: marcos.batista@catalao.ufg.br.

**Abstract**— *In this work we solve a subproblem of the distance geometry problem in molecular conformation. The latter aims to determine the three-dimensional structure of a molecule from a set of imprecisely distances. We are interested only in the bound smoothing subproblem, which aims to tighten bounds from a set of lower and upper bounds on distance for pair of atoms. We apply a Biased Random-Key Genetic Algorithm to solve this subproblem, where each entry of chromosomes indicates how to decrease a bound, with the fitness function measuring the violation of the triangle inequalities. Experimental results show the effectiveness of this algorithm to solve randomly generated instances for which real distances between atoms are known in advance.*

**Keywords:** Molecule Problem; Distance Geometry Problem; Triangle Inequality; Biased Random-Key Genetic Algorithm

## 1. Introduction

A important problem in computational biology is to determine the structure of a molecule, as a protein [1], [2]. Some experimental techniques have been applied to measure interatomic distances for molecules, as the Nuclear Magnetic Resonance (NMR) [3], [4] and X-ray crystallography [5]. After measuring the distance between atoms, the next step is to construct a three-dimensional structure, that is, solve a variant of the distance geometry problem, which is NP-hard accordingly to [6].

The distances measured with NMR technique are not precise, so many of these distances need to be given in lower and upper bounds. Then, it is necessary to tighten the bounds in order to construct a well structured molecule. In this case, a distance geometry problem has to be solved [7]. In accordance with [8], this problem can be organized in three subproblems (steps): (i) bound smoothing, (ii) embedding, and (iii) optimization.

Step (i) aims to improve the bounds given on the distance of each pair of atoms as well as to compute bounds for those distances that experimental methods could not estimate, so aiming to obtain tight bounds for each pair of atoms. In step (ii), once the bounds are tightened, it is computed approximate tree-dimensional coordinates for the atoms, and they are further improved in step (iii). Steps (ii) and (iii) are repeated until to satisfy all lower and upper bounds. More details of the distance geometry problem are given in [9].

Experimental approaches that have been used to calculate a distance for a pair of atoms are not precise. On the NMR, the structure must have a small quantity of residues, so the distance can be estimated for pairs of atoms that are at a short distance (smaller than 6 Å) On the other hand, the X-ray crystallography imposes a crystallization of the structure, which is very time consuming [10].

As only a set of distances can be estimated from experimental approaches, it is necessary to compute the remaining ones to obtain a complete structure. The distance geometry problem is closely related to those in the areas of sensor network localization [11], image recognition and euclidean distance matrix completion [12], an extensive review about distance geometry is avaliable at [13].

The major contribution to the distance geometry problem with applications in the field of molecular conformation was given by [9]. They proposed an algorithm, which determines coordinates for atoms from experimental data. Their algorithm solves the three subproblems previously mentioned. Another approaches used to solve the distance geometry problem were proposed by [14], based on graph reduction, in which an input graph is decomposed in subgraphs and, next, an embedding problem is solved; by [15], based on global optimization to solve a weighted function, which does not need to consider all bounds, so this function is transformed on easier functions that are quickly solved; by [16], that considered the geometric buildup algorithm, which can solve the problem in linear time if all distances are known; by [17], based on stochastic search; and, by [10] that developed a new version of the geometric buildup algorithm for a generalized version of the distance geometry problem. The latter aims to find the equilibrium positions and the maximal bounds for each pair of atoms.

We are only interested on step (i), that is, to solve the bound-smoothing subproblem. The objective of the Bound-

smoothing is to check consistency and give information about the distance between pairs of atoms. So, this implies to tighten the initial lower and upper bounds that are know as well as to compute bounds for pair of atoms whose distance is unknown.

The bound-smoothing problem has been solved so far essentially by checking triangle (and tetrangle) inequalities for atoms-triplet (atoms-quadruple), since tighter bounds can be obtained if satisfying these inequalities. The computation of the triangle inequality is straightforward, while the tetrangle one requires to calculate the Cayley-Menger determinant, which is constructed on each quadruple of atoms (atoms-quadruple) and it has worst-case complexity time of $O(n^4)$.

Therefore, the literature has been focused on algorithms to solve instances with a large number of atoms by means of the tetrangle inequality. A parallel algorithm working in a shared memory architecture for bound smoothing was proposed in [8]. Their algorithm spends $O(n^3 logn)$ time with $O(\frac{n}{longn})$ processors, and it is based on classify the atoms in independent sets and, then, to color vertices of the resultant graph. They solved satisfactorily instances with up $430$ atoms, for which their algorithm required only $5\%$ of the time spent by a sequential algorithm.

Another parallel algorithm was developed in [18] for the Beowulf-type cluster of PCs, which requires $O(\frac{n^4}{p})$ time for $p < \frac{n}{6}$ processors. This algorithm organizes atoms in subsets, so atoms-quadruple can be classified at one out of five types according to the subset that each atom is in. They solved one instance with $630$ atoms with efficiency arriving at $60\%$.

In order to solve the bound-smoothing problem satisfactorily, we considered the biased random-key genetic algorithm (BRKGA) [19]. The BRKGA has been applied with successes to solve classical optimization problems, such as in the area of covering [20], packing [21], scheduling [22], and clustering [23]. A C++ framework of this algorithm was made available in Internet by [24].

This paper is organized as follows. In Section 2, a formal description of the problem is given, while in Section 3, the Biased Random-Key Genetic algorithm is introduced and it is explained how this algorithm is used to solve the bound-smoothing problem. Numerical experiments are presented in Section 4 and they give details about the algorithm and its efficiency to deal with this problem. Some randomly generated instances are solved and the results are compared with the original structure. Finally, some conclusions and directions for further works are given in Section 5.

## 2. Bound-smoothing Problem

In the Bound-smoothing Problem (BsP), let $\mathcal{S}$ be the set of pairwise atoms with coordinates in $\mathbb{R}^3$, for $m$ atoms. In the set $S_1 \subset \mathcal{S}$, for each pair of atoms $\{a, b\} \in S_1$, it is known the exact Euclidean distance $d_{ab}$. On the other hand, given $S_2 \subset \mathcal{S}$, for each pair $\{i, j\} \in S_2$, we have $l_{ij} \leq d_{ij} \leq u_{ij}$, where $l_{ij}$ and $u_{ij}$ are the lower and upper bounds on the distance $d_{ij}$. And, for the last set, namely $S_3 \subset \mathcal{S}$, there is no information on the distance $d_{kl}$ for each pair $\{k, l\} \in S_3$. Moreover, $S_1 \cup S_2 \cup S_3 = \mathcal{S}$ and $S_1 \cap S_2 \cap S_3 = \emptyset$. The objective of the BsP is to compute bounds for the pair of atoms in $S_3$, and then, to tighten the bounds in $S_2$ and $S_3$ while satisfying distance constraints, as those imposed by triangle and tetrangle inequalities.

In the case of the triangle inequalities, any three points (atoms-triplet) have to satisfy the triangle inequality, so we can improve the lower and upper bounds computing this inequality for all atoms-triplet. In other words, for $(i, j, k)$, with $l_{ij} \leq d_{ij} \leq u_{ij}$, $l_{ik} \leq d_{ik} \leq u_{ik}$ and $l_{jk} \leq d_{jk} \leq u_{jk}$, the triangle inequality $|d_{ik} - d_{jk}| \leq d_{ij} \leq d_{ik} + d_{jk}$ must hold. Then, the bounds can be improved by:

$$\begin{aligned} \overline{u}_{ij} &= \min\{u_{ij},\ u_{ik} + u_{jk}\}, \\ \overline{l}_{ij} &= \max\{l_{ij},\ l_{ik} - u_{jk},\ l_{jk} - u_{ik}\}, \end{aligned} \tag{1}$$

where $\overline{l}_{ij}$ e $\overline{u}_{ij}$ are the new values for the lower and upper bounds, respectively.

Observe that $\overline{u}_{ij}$ can be calculated independently of $\overline{l}_{ij}$. Moreover, if $u_{ik} \geq \overline{u}_{ik}$, $u_{jk} \geq \overline{u}_{jk}$ and $\overline{l}_{ij} = \max\{l_{ij},\ l_{ik} - u_{jk},\ l_{jk} - u_{ik}\}$, then we can again improve $\overline{l}_{ij}$ performing $\overline{l}_{ij} = \max\{l_{ij},\ l_{ik} - \overline{u}_{jk},\ l_{jk} - \overline{u}_{ik}\}$. Whenever $\overline{l}_{ij} > \overline{u}_{ij}$, it follows that the original values are restored.

A straightforward algorithm to update the bounds considers all atoms-triplet and first it computes upper bounds. Next, all triplets are checked for lower bounds. Algorithm 1 summarizes these steps.

**Algorithm 1:** Triangle inequality for the BsP.

**Input** : Lower bound $L$; Upper bound $U$; Sets $S_1$, $S_2$ and $S_3$.
**Output**: Tightened lower and upper bounds $\overline{L}$ and $\overline{U}$, respectively.

1.1 Let $T$ be the set of all atoms-triplet.
1.2 **foreach** *triplet* $(i, j, k) \in T$ **do**
1.3  **if** $\{i, j\} \notin S_1$ **then**
1.4  $\overline{u}_{ij} \leftarrow \min\{u_{ij},\ u_{ik} + u_{jk}\}$.
1.5 $\overline{U} \leftarrow$ all upper bounds $\overline{u}$.
1.6 $U \leftarrow \overline{U}$.
1.7 **foreach** *triplet* $(i, j, k) \in T$ **do**
1.8  **if** $\{i, j\} \notin S_1$ **then**
1.9  $aux = \max\{l_{ij},\ l_{ik} - u_{jk},\ l_{jk} - u_{ik}\}$.
1.10  **if** $aux \leq \overline{u}_{ij}$ **then**
1.11  $\overline{l}_{ij} \leftarrow aux$.
1.12  **else**
1.13  $\overline{l}_{ij} \leftarrow l_{ij}$.
1.14 $\overline{L} \leftarrow$ all lower bounds $\overline{l}$.
1.15 **return** *sets* $\overline{L}$ *and* $\overline{U}$.

We consider that $u$ is from $U$ and $l$ is from $L$, while $\overline{u}$ is from $\overline{U}$ and $\overline{l}$ is from $\overline{L}$. After to tighten the bounds by applying the triangle inequality algorithm, a bound may still have some slack, that is, $u_{ik} + u_{jk} - u_{ij} > 0$ and $l_{ij} - l_{ik} - u_{jk} > 0$. In this case, it can be eliminated by computing all-pairs shortest paths [25].

A better way to improve the bounds is to check the tetrangle inequalities for all atoms-quadruple. The tetrangle inequality is associated with the calculation of Cayley-Menger determinants. For a quadruple $(i, j, k, r)$, its determinant $\det_{CM}(d_{ij}, d_{ik}, d_{ir}, d_{jk}, d_{jr}, d_{kr})$ is:

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & (d_{ij})^2 & (d_{ik})^2 & (d_{ir})^2 \\ 1 & (d_{ji})^2 & 0 & (d_{jk})^2 & (d_{jr})^2 \\ 1 & (d_{ki})^2 & (d_{kj})^2 & 0 & (d_{kr})^2 \\ 1 & (d_{ri})^2 & (d_{rj})^2 & (d_{rk})^2 & 0 \end{vmatrix} > 0 \qquad (2)$$

Therefore, for each pair of atoms $\{k, r\}$, we obtain 32 inequalities from eq. (2) for the lower bound $l_{kr}$ as well as 32 inequalities for the upper bound $u_{kr}$. However, only seven of these inequalities are non-redundant, with three for the upper bound $u_{kr}$:

$$\begin{aligned} &\det_{CM}(l_{ij}, u_{ik}, u_{ir}, u_{jk}, u_{jr}, u_{kr}) > 0, \\ &\det_{CM}(u_{ij}, l_{ik}, l_{ir}, u_{jk}, u_{jr}, u_{kr}) > 0, \\ &\det_{CM}(u_{ij}, l_{ik}, l_{ir}, u_{jk}, u_{jr}, u_{kr}) > 0. \end{aligned} \qquad (3)$$

And, another four ones for the lower bound $l_{kr}$:

$$\begin{aligned} &\det_{CM}(u_{ij}, u_{ik}, l_{ir}, l_{jk}, u_{jr}, l_{kr}) > 0, \\ &\det_{CM}(u_{ij}, l_{ik}, u_{ir}, u_{jk}, l_{jr}, l_{kr}) > 0, \\ &\det_{CM}(l_{ij}, l_{ik}, u_{ir}, l_{jk}, u_{jr}, l_{kr}) > 0, \\ &\det_{CM}(l_{ij}, u_{ik}, l_{ir}, u_{jk}, l_{jr}, l_{kr}) > 0. \end{aligned} \qquad (4)$$

Among the values of $u_{kr}$ computed in eq. (3), the smallest one is compared to the initial $u_{kr}$, and the minimum between them are set to $\overline{u}_{kr}$. Similarly, the largest value of $l_{kr}$ among those computed in (4) is compared to the given $l_{kr}$, and the maximum between them are set to $\overline{l}_{kr}$.

For an inequality $\det_{CM}(d_{ij}, d_{ik}, d_{ir}, d_{jk}, d_{jr}, d_{kr}) > 0$, in which we want to compute the value of $d_{kr}$ knowing the remaining ones, it is necessary to solve a quadratic equation to find its roots. Algorithm 2 presents a straightforward way to tighten lower and upper bounds by computing the tetrangle inequalities.

In order to get better results, it is recommend first apply the triangle algorithm 1. Next, the tetrangle algorithm 2 can be repeated until the largest change in any of the bounds be smaller than a given tolerance value. Unhappily, apply the tetrangle inequality a lot of times may bring on a slow ratio of convergence, once the total number of iterations grows quickly. Observe that Algorithm 2 has worst-case complexity time of $O(n^4)$ as pointed out in [8].

**Algorithm 2:** Tetrangle inequality for the BsP.

**Input** : Lower bound $L$; Upper bound $U$; Sets $S_1$, $S_2$ and $S_3$.
**Output**: Tightened lower and upper bounds $\overline{L}$ and $\overline{U}$, respectively.

2.1 Let $T$ be the set of all atoms-quadruple.
2.2 **foreach** *quadruple* $(i, j, s, t) \in T$ **do**
2.3  **foreach** *pair* $\{k, r\}$ *out of the six ones in* $(i, j, s, t)$ **do**
2.4  **if** $\{k, r\} \notin S_1$ **then**
2.5  $u'_{kr} \leftarrow$ minimum $u_{kr}$ from eq. (3).
2.6  $\overline{u}_{kr} \leftarrow \min\{u_{kr},\ u'_{kr}\}$.
2.7 $\overline{U} \leftarrow$ all upper bounds $\overline{u}$.
2.8 $U \leftarrow \overline{U}$.
2.9 **foreach** *quadruple* $(i, j, s, t) \in T$ **do**
2.10  **foreach** *pair* $\{k, r\}$ *out of the six ones in* $(i, j, s, t)$ **do**
2.11  **if** $\{k, r\} \notin S_1$ **then**
2.12  $l'_{kr} \leftarrow$ maximum $l_{kr}$ from eq. (4).
2.13  $\overline{l}_{kr} \leftarrow \max\{l_{kr},\ l'_{kr}\}$.
2.14 $\overline{L} \leftarrow$ all lower bounds $\overline{l}$.
2.15 **return** *sets* $\overline{L}$ *and* $\overline{U}$.

## 3. Biased Random-Key Genetic Algorithm

The BRKGA was introduced in [26] and [27]. Algorithm 3 summarizes a typical BRKGA framework, as that provided in [24].

The BRKGA has two key features that distinguish it from others traditional genetic algorithm: a standardized chromosome encoding that uses a vector with random keys (*alleles*) uniformly over the interval $[0, 1)$; and, a well-defined evolutionary process which uses parameterized uniform crossover [28].

Therefore, the BRKGA performs the crossover without caring about the feasibility of the solutions generated for the new individuals. This is possible because of the standard chromosome encoding that is responsible to guarantee the feasibility of the decoding function.

The decoding function is the most important of the BRKGA, since it maps the real vector for a valid solution of the problem under consideration, and it is also more time consuming than the other parts of the algorithm. In some problems where feasibility is hard to achieve, the decoder may generate invalid solutions, so a penalty factor is necessary into the fitness for the algorithm converges.

According to algorithm 3, the parameters that must be specified in a BRKGA framework are the size of the chromosomes $t$, the size of the population $pop$, the size of

**Algorithm 3:** BRKGA algorithm.

Generate an initial population $Pop$.
**while** *not attend a stopping criteria* **do**
  **Decode** each chromosome of $Pop$ and extract a solution and its fitness.
  Sort chromosomes of $Pop$ in non-increasing order of fitness. Consider the top $pop_e$ chromosomes to be in an elite set $Eli$.
  Next generation $Qn$ receives $Eli$.
  Next generation $Qn$ receives $pop_\mu$ randomly-generated new chromosomes.
  Generate $off \leftarrow pop - pop_e - pop_\mu$ chromosomes offspring using the parameterized crossover, for a random parent from $Eli$ and another from $Pop \backslash Eli$.
  Next generation $Qn$ receives $off$.
  $Pop \leftarrow Qn$.
**return** *chromosome with the best fitness.*

the elite set $pop_e$, the number of mutants $pop_\mu$ introduced at each generation, and the inheritance probability $\rho_e$. In the framework of [24], it is also allowed to define the number of independent populations $k$ and the number of threads for parallel decoding $MAXT$.

## 3.1 Decoder Phase

First, we apply the triangle algorithm 1 to tighten the bounds in $S_2$ and, next, to compute initial bounds for the pair of atoms in $S_3$. We do not consider the tetrangle algorithm 2 due its cycling and slow convergence. Let $L$ and $U$ be the new lower and upper bounds after apply the triangle algorithm.

The aim of the decoder is to extract from chromosomes a good solution for the BsP, in which $L$ and $U$ are tightened in accordance with a fitness function.

For the BRKGA, each chromosome $ch$ has size equal to $|S_2| + |S_3|$, where each allele is sequentially associated to a pair of atoms from these sets. For a pair $a = \{i, j\}$, its allele with value $v_a$ ranging in $[0, 1)$ is used to compute new bounds as specified in Algorithm 4.

We apply Algorithm 4 at each allele of the chromosome $ch$, so new lower $\overline{L}$ and upper $\overline{U}$ bounds are obtained according to the values in $ch$ and the input bounds $L$ and $U$.

Observe that $\overline{L}$ and $\overline{U}$ do not have bounds such that $\overline{l}_{ij} < l_{ij}$ and $u_{ij} > \overline{u}_{ij}$ for each pair of atoms $\{i, j\}$. On the other hand, some new bounds cannot satisfy triangle inequalities and, then, we compute the violation of these inequalities by Algorithm 5.

**Algorithm 4:** Bounds for a given allele.

**Input** : Lower bound $L$; Upper bound $U$; pair $a = \{i, j\}$ of atoms; value $v_a$ of the allele.
**Output**: New bounds for $a = \{i, j\}$.

4.1 $avg \leftarrow u_{ij} - l_{ij}$.
4.2 $rg \leftarrow \frac{avg \times v_a}{2}$.
4.3 $\overline{l}_{ij} \leftarrow l_{ij} + rg$.
4.4 $\overline{u}_{ij} \leftarrow l_{ij} - rg$.
4.5 **return** $\overline{l}_{ij}$ *and* $\overline{u}_{ij}$.

**Algorithm 5:** Violation of triangle inequalities.

**Input** : Lower bounds $\overline{L}$ and $L$; Upper bounds $\overline{U}$ and $U$; Sets $S_1$, $S_2$ and $S_3$.
**Output**: Total violation of triangle inequalities.

5.1 Let $T$ be the set of all atoms-triplet.
5.2 $viol \leftarrow 0$.
5.3 **foreach** *triplet* $(i, j, k) \in T$ **do**
5.4  **if** $\{i, j\} \notin S_1$ **then**
5.5  $minU \leftarrow \min_{\{\forall\ k \neq i \neq j\}} \{\overline{u}_{ik} + \overline{u}_{jk}\}$.
5.6  **if** $\{i, j\} \in S_2$ **then**
5.7  **if** $minU < u_{ij}$ *and* $minU > \overline{u}_{ij}$ **then**
5.8  $viol \leftarrow viol + |\overline{u}_{ij} - minU|$.
5.9  **else**
5.10  **if** $minU > \overline{u}_{ij}$ **then**
5.11  $viol \leftarrow viol + |\overline{u}_{ij} - minU|$.
5.12 **foreach** *triplet* $(i, j, k) \in T$ **do**
5.13  **if** $\{i, j\} \notin S_1$ **then**
5.14  $maxL \leftarrow \max_{\{\forall\ k \neq i \neq j\}} \{\overline{l}_{ik} - \overline{u}_{jk},\ \overline{l}_{jk} - \overline{u}_{ik}\}$.
5.15  **if** $\{i, j\} \in S_2$ **then**
5.16  **if** $maxL > l_{ij}$ *and* $maxL < \overline{l}_{ij}$ **then**
5.17  $viol \leftarrow viol + |maxL - \overline{l}_{ij}|$.
5.18  **else**
5.19  **if** $maxL < \overline{l}_{ij}$ **then**
5.20  $viol \leftarrow viol + |maxL - \overline{l}_{ij}|$.
5.21 **return** $viol$.

The Root Mean Square Gap (RMSG) has been used to measure the distance between lower and upper bounds [8]. It corresponds to the root mean square of the difference between upper and lower bounds for all the atoms distance, as described in eq. (5).

$$RMSG(\overline{L}, \overline{U}) = \sqrt{\frac{\sum_{\{i,j\} \in \mathcal{S}} (\overline{u}_{ij} - \overline{l}_{ij})^2}{|\mathcal{S}|}} \tag{5}$$

Naturally, a small value for the RMSG may indicate that lower and upper bounds are close to each other. On the other hand, if the gap between each bound is significantly small,

a valid structure cannot be constructed in the embedding and optimization steps of the distance geometry problem, even if all triangle inequalities are satisfied. It is important to mention that to construct a valid structure is necessary to satisfy other requirements, as the angle bond imposed to some pair of atoms.

With this in mind, the chromosome's fitness is the violation value returned by Algorithm 5. The BRKGA objective is to minimize such violation value.

It is worth to mention that a set of numerical tests aiming to minimize the RMSG, subject to satisfy triangle inequalities for all atoms-triplet, was also conducted on instances for which the exact distance between all pair of atoms are known. However, these results shown that the gap for each bound is very small, such that the exact distances are completely out of the initial bounds $L$ and $U$.

# 4. Computational Experiments

All algorithms were implemented in the C++ programming language and the experiments occurred in a computer with 4.0 GHz Intel Core i7-4790K processor, 32 GB of memory RAM and GNU/Linux operating system. The BRKGA framework of [24] was used, where the decoder phase was implemented according to Section 3.1.

The experiments were conducted in a set of randomly generated instances as described in [29]. These instances have structure similar to that of original proteins and the exact distance between each pair of atom is known. We consider instances with the number of atoms ranging from 10 to 100, totaling 15 instances, each one with name $brN$, where $N$ is the number of atoms.

Observe that we can perfectly check whether the exact distances are in the tighter bounds of the BRKGA solution, contrary if we consider real proteins as those available in the Protein Data Bank, for which the exact distances are unknown for almost all pair of atoms. We are interested to verify whether the BRKGA can be used to solve the distance geometry problem satisfactorily, while solving the BsP and returning tighter valid bounds as well.

For each instance, first it is generated coordinates of the atoms by observing the angle between atoms-triplet or -quadruple [30], so the exact Euclidean distance $d$ is computed. Next, in order to generate lower and upper bounds according to a NMR simulation: the distance between atoms $\{i, i+i\}$ and $\{i, i+2\}$ can be measured precisely due to geometric considerations, so they are considered as the exact Euclidean distance ($l_{i,i+i} = u_{i,i+i}$ and $l_{i,i+2} = u_{i,i+2}$) and they are in the set $S_1$. The remaining bounds for each pair of atoms $\{j, k\}$ are given as:

- If $d_{jk} < 6$Å, then this distance can be estimated using the NMR, so we consider as bounds $l_{jk} = \lfloor d_{jk} \rfloor$ and $u_{jk} = \lceil d_{jk} \rceil$. This pair is in the set $S_2$;
- Otherwise, the NMR method cannot estimate the distance between $j$ and $k$, so the lower and upper bounds are unknown. In this case, $\{j, k\}$ is in the set $S_3$.

The parameters used by the BRKGA are described next. We consider one independent population executing on one thread. So, we have $pop = 10t$, where $t = |S_2| + |S_3|$ is the size of each chromosome, $pop_e = 25\%$ of the population, $pop_\mu = 15\%$ of the population, and, $\rho_e = 65\%$. The BRKGA stops when a maximum number of iterations is reached, that is, if it reaches 2000 iterations.

## 4.1 Results

We present the results for all the instances in Table 1. Each row of this table has: name of the instance; total time spent (in seconds); fitness value; RMSG for $L$ and $U$ (after to apply the triangle algorithm); RMSG for the best solution computed with the BRKGA; difference (in percentage) between the RMSGs; percentage of reduction of the bounds for all pair of atoms (this is an average value); number of bounds that were tightened (in percentage); number of bounds tightened (in percentage) for which the respective exact distance is not in.

Table 1 presents the results for the set of 15 instances. Only 2 out of these instances had fitness equal to zero (see br10 and br20), that means that all triangle inequalities are satisfied. As we can observe, the time, fitness value and RMSGs increased accordingly the number of atoms increased too.

Although the RMSG was not considered in the fitness function, its value decreased of $55.69\%$ on average (see column "RMSG BRKGA") especially due to the lower and upper bounds that were reduced too. Observe that $100\%$ of the bounds were tightened, for all the instances, where each bound had an average reduction of $44.48\%$, on average. This average reduction ranged from $42\%$ to $54\%$ considering all the instances.

For the first 5 instances (br10 to br50) the runtime was less than 1 hour. The worst runtime was for the br100, where the BRKGA required more than 40 hours. On average, the BRKGA required approximately 10 hours, so it may not be a satisfactory result, since real proteins have hundreds or thousands of atoms. On the other hand, the parameters of the BRKGA, as the maximum number of iterations and the size of the population, can be decreased as an attempt to improve the runtime.

The last column of Table 1 shows the percentage of the exact distances that do not comply with the bounds found with the BRKGA. Note that we know the exact distance $d$ between each pair of atoms, due to the way that the instances were generated. The objective of this column is to show whether the bounds computed with the BRKGA still contain a satisfactory quantity of exact distances. Observe that, on average, more than $50\%$ of these bounds still keep the exact distance value, although the BRKGA do not know

Table 1: Results for the set of randomly generated instances.

| Name | Time (s) | fitness | RMSG triangle | RMSG BRKGA | Difference RMSG (%) | Reduction on bounds (%) | Bounds tightened (%) | Distances out (%) |
|---|---|---|---|---|---|---|---|---|
| br10 | 0.590 | 0.000 | 0.709 | 0.383 | 45.978 | 53.882 | 100.000 | 46.429 |
| br20 | 31.410 | 0.000 | 1.978 | 1.727 | 12.693 | 47.735 | 100.000 | 43.137 |
| br30 | 282.090 | 3.600 | 9.751 | 4.325 | 55.651 | 43.636 | 100.000 | 42.593 |
| br40 | 1,162.050 | 5.765 | 17.975 | 8.347 | 53.564 | 42.941 | 100.000 | 44.239 |
| br50 | 3,438.620 | 15.794 | 56.005 | 31.669 | 43.452 | 44.275 | 100.000 | 53.014 |
| br55 | 7,719.980 | 21.910 | 42.836 | 24.112 | 43.711 | 44.220 | 100.000 | 52.322 |
| br60 | 11,899.100 | 22.949 | 58.614 | 32.759 | 44.111 | 44.881 | 100.000 | 55.354 |
| br65 | 17,131.750 | 30.719 | 37.350 | 22.102 | 40.824 | 43.113 | 100.000 | 50.026 |
| br70 | 24,398.860 | 39.458 | 55.489 | 31.650 | 42.962 | 44.324 | 100.000 | 53.951 |
| br75 | 35,539.910 | 35.248 | 33.504 | 19.269 | 42.488 | 43.846 | 100.000 | 50.304 |
| br80 | 49,359.350 | 55.042 | 22.823 | 13.520 | 40.761 | 42.744 | 100.000 | 46.121 |
| br85 | 77,411.920 | 48.220 | 38.302 | 22.115 | 42.261 | 43.316 | 100.000 | 50.220 |
| br90 | 95,908.230 | 61.434 | 51.918 | 30.286 | 41.665 | 42.342 | 100.000 | 50.653 |
| br95 | 112,270.320 | 81.590 | 53.572 | 31.318 | 41.540 | 42.667 | 100.000 | 52.571 |
| br100 | 147,035.210 | 78.890 | 646.378 | 354.204 | 45.202 | 43.259 | 100.000 | 55.207 |
| **Average** | **38,905.96** | **–** | **75.15** | **41.85** | **42.46** | **44.49** | **100.00** | **49.74** |

neither consider any information about the exact distances when solving the problem, since its execution is only guided on check the triangle inequalities. This is an interesting result and it demonstrates that the BRKGA can effectively solve the distance geometry problem.

## 5. Conclusions

The bound smoothing problem (BsP) that appears as one of the subproblems of the distance geometry problem is addressed in this paper. In the BsP, we have to tighten bounds for a set of pairwise atoms while they must obey distance constraints, as the triangle or tetrangle inequalities.

We solved the BsP with the Biased Random-Key Genetic Algorithm, in which the decoder phase generates a solution from a chromosome by increasing or decreasing lower and upper bounds of the pair of atoms. The chromosome fitness is calculated according to the number of triangle inequalities that are violated.

Good solutions were obtained for the instances under consideration, since the RMSG decreased of $42.46\%$ on average, and the bounds had a reduction of $44.48\%$ on average, as well as all bounds were tightened for all the instances. On the other hand, the number of exact distances that remained in the new tighter bounds was of $50.03\%$ on average, even if all triangle inequalities were satisfied. This may lead to invalid molecules when solving the other subproblems of the distance geometry problem.

After all, we note that there is room for improvements by considering a new way to decoder chromosomes and extract solutions. It is also important to consider other fitness functions, since the computational runtime increased significantly as the number of atoms increased too, if using the triangle algorithm in the fitness function.

### Acknowledgements.

The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq), Research Support Foundation of Goiás State (FAPEG) and Federal University of Uberlândia (grant 04/1014/074-PROPP).

## References

[1] C. L. Brooks-III, M. Karplus, and B. M. Pettitt, *A Theoretical Perspective of Dynamics, Structure, and Thermodynamics*. New York: Wiley, 1988.

[2] T. E. Creighton, *Proteins: Structures and Molecular Properties. 2nd Edition*. Freeman and Company, 1993.

[3] A. T. Brünger and M. Nilges, "Computational challenges for macromolecular structure determination by x-ray crystallography and solution nmr-spectroscopy," *Quarterly Reviews of Biophysics*, vol. 26, pp. 49–125, 1993.

[4] J. Cavanagh, W. J. Fairbrother, A. G. Palmer, and N. J. Skelton, *Protein NMR Spec- troscopy: Principals and Practice*. Academic Press, 2006.

[5] J. Drenth, *Principals of Protein X-ray Crystallography*. Springer, 2006.

[6] J. B. Saxe, "Embeddability of weighted graphs in k-space is strongly np-hard," in *Proceedings of the 17th Allerton Conference in Communications, Control and Computing*, 1979, pp. 480–489.

[7] L. M. Blumenthal, *Theory and Applications of Distance Geometry*. London: Oxford University Press, 1953.

[8] K. Rajan and N. Deo, "Computational experience with a parallel algorithm for tetrangle inequality bound smoothing," *Bulletin of Mathematical Biology*, vol. 61, pp. 987–1008, 1999.

[9] G. M. Crippen and T. F. Havel, *Distance Geometry and Molecular Conformation*. England: Research Studies Press Ltd, 1988.

[10] B. Hendrickson, "The molecular problem: Determining conformation from pairwise distances," Ph.D. dissertation, Iowa State University, 2010.

[11] P. Biswas, T. Liang, T. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Journal on Transactions on Sensor Networks*, vol. 2, pp. 188–220, 2006.

[12] W. Rivera-Gallego, "A genetic algorithm for solving the euclidean distance matrices completion problem," in *Proceedings of the 1999 ACM Symposium on Applied Computing*, ser. SAC '99. New York, NY, USA: ACM, 1999, pp. 286–290. [Online]. Available: http://doi.acm.org/10.1145/298151.298353

[13] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino, "Euclidean distance geometry and applications," *SIAM Review*, vol. 56, pp. 3–69, 2014.

[14] A. Sit, "Solving distance geometry problems for protein structure determination," Ph.D. dissertation, Cornell University, 1991.

[15] J. Moré and Z. Wu, "Global continuation for distance geometry problems," *SIAM Journal on Optimization*, vol. 7, pp. 814–836, 1997.

[16] Q. Dong and Z. Wu, "A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances," *Journal of Global Optimization*, vol. 22, pp. 365–375, 2002.

[17] A. Grosso, M. Locatelli, and F. Schoen, "Solving molecular distance geometry problems by global optimization algorithms," *Computational Optimization and Applications*, vol. 43, pp. 23–37, 2009.

[18] N. Deo and P. Micikevicius, "Coarse-grained parallelization of distance-bound smoothing for the molecular conformation problem," in *Distributed Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2571, pp. 55–66.

[19] J. F. Gonçalves and M. G. C. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 17, pp. 487–525, 2011.

[20] M. G. C. Resende, R. F. Toso, J. F. Gonçalves, and R. M. A. Silva, "A biased random-key genetic algorithm for the Steiner triple covering problem," *Optimization Letters*, pp. 1–15, 2011.

[21] J. F. Gonçalves and M. G. C. Resende, "A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem," *Journal of Combinatorial Optimization*, vol. 22, pp. 180–201, 2011.

[22] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende, "A random key based genetic algorithm for the resource constrained project scheduling problems," *Computers and Operations Research*, vol. 36, pp. 92–109, 2009.

[23] C. E. Andrade, M. G. C. Resende, H. J. Karloff, and F. K. Miyazawa, "Evolutionary algorithms for overlapping correlation clustering," in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 405–412.

[24] R. F. Toso and M. G. C. Resende, "A c++ application programming interface for biased random-key genetic algorithms," AT&T Labs Research, Florham Park, New Jersey," Technical Report, 2012.

[25] A. W. M. Dress and T. F. Havel, "Shortest-path problems and molecular conformation," *Discrete Applied Mathematics*, vol. 19, pp. 129–144, 1988.

[26] J. F. Gonçalves and J. Almeida, "A hybrid genetic algorithm for assembly line balancing," *J. of Heuristics*, vol. 8, pp. 629–642, 2002.

[27] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *J. of Combinatorial Optimization*, vol. 6, pp. 299–333, 2002.

[28] W. M. Spears and K. A. DeJong, "On the virtues of parameterized uniform crossover," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 230–236.

[29] C. Lavor, L. Liberti, A. Mucherino, and N. Maculan, "On a discretizable subclass of instances of the molecular distance geometry problem," in *Proceedings of the 2009 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2009, pp. 804–805.

[30] C. Lavor, "On generating instances for the molecular distance geometry problem," in *Global Optimization*, ser. Nonconvex Optimization and Its Applications, L. Liberti and N. Maculan, Eds. Springer US, 2006, vol. 84, pp. 405–414.