DGA 2013, pp. 119 – 124.

# Quaternions as a tool for merging multiple realization trees*

Felipe Fidalgo[1] and Jaime Rodriguez[2]

[1]*Department of Applied Mathematics, IMECC - UNICAMP, Campinas, Brazil,* felipefidalgo@ime.unicamp.br

[2]*Department of Mathematics, UNESP, Ilha Solteira, Brazil,* jaime@mat.feis.unesp.br

**Abstract** This work uses Quaternion Algebra as a tool to improve the resolution of the Multiple Realization Trees method which solves the Discretizable Molecular Distance Geometry Problem (DMDGP) by dividing the instances into smaller pieces producing more than one binary tree of realizations. Quaternion Rotations are used here to merge such trees, saving positions of memory and causing a decreasing in the number of operations.

**Keywords:** Molecular Geometry, Quaternion Algebra, Distance Geometry, Branch-and-Prune Algorithm

## 1. Introduction

From known distance values for pairs of atoms in a molecule, it is possible to formulate an inverse problem called *Molecular Distance Geometry Problem (MDGP)* [1, 5] which consists of finding a 3-D conformation for the molecule such that it satisfies the distance constraints. Such data usually come from chemical knowledge (like atomic bond lengths and bond angles) combined with a physical experimental method called *Nuclear Magnetic Resonance (NMR)* [3]. With additional assumptions, Lavor et. al [5] proposed a discrete formulation for a subclass of the MDGP, which is called *Discretizable Molecular Distance Geometry Problem (DMDGP)*. In addition, an efficient method was also proposed for solving this problem, the Branch-and-Prune (BP) algorithm, which generates a binary tree with all possible solutions [5]. To make this method faster, Nucci et. al [6] proposed another one which uses the BP algorithm more than once, generating more than one tree, as shown in Section 2. Finally, Section 3 shows how to use quaternions in order to make the method, proposed by Nucci et. al, even more efficient, comparing it with the rotation matrix approach.

## 2. Multiple Realization Trees

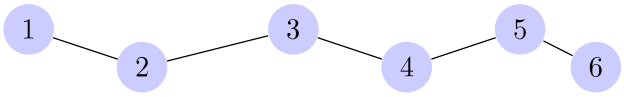
Given a molecule $M$ in a backbone-chain shape, with an order $<$ on its set of atoms $\{1, \ldots, n\}$, we can split it into an union of intervals in an increasing order like

$$M = M_1 \cup M_2 \cup \ldots \cup M_k, \tag{1}$$

where $M_j = [a_j, b_j]$, $a_1 = 1$, $b_k = n$, $1 \leq a_j \leq a_{j+1} \leq n$, $b_j - a_{j+1} \geq 2$ and $j = 2, \ldots, k-1$. The whole molecule can be represented as the interval $M = [1, n]$. For example: let $M = [1, 6]$

*The authors would like to thank to the brazilian research agencies CNPq and FAPESP for the financial support.

be a molecule. So, it can be splitted into the union $M = M_1 \cup M_2$, where $M_1 = [1, 4]$ and $M_2 = [2, 6]$.




This division motivates what Nucci et. al [6] called as the method of *Multiple Realization Trees (MRT)*. Before describing it, we give some important definitions. A *Valid Realization* of a DMDGP instance $M$ corresponds to a bijective embedding of it in $\mathbb{R}^3$ which satisfies the distance constraints, i.e., a three-dimensional conformation for $M$. A *Realization Tree*, in our case, is a binary-tree graph which represents, in a depth-first fashion, *all* the valid realizations which solves the DMDGP. A *Feasible Branch* is the name we give for each of the branches of a Realization Tree, i. e., each feasible branch represents one embedding of $M$ in $\mathbb{R}^3$.

The MRT method applies the BP algorithm in each interval $M_j$, producing $k$ realization trees $T_j$. We denote by $T$ the realization tree which represents all the feasible realizations for throughout the DMDGP instance $M$. Back to our example, the BP method provides the trees $T_1$ and $T_2$, as in the figures below.

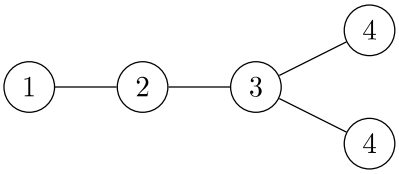



Figure 1: Tree $T_1$: 4 levels.

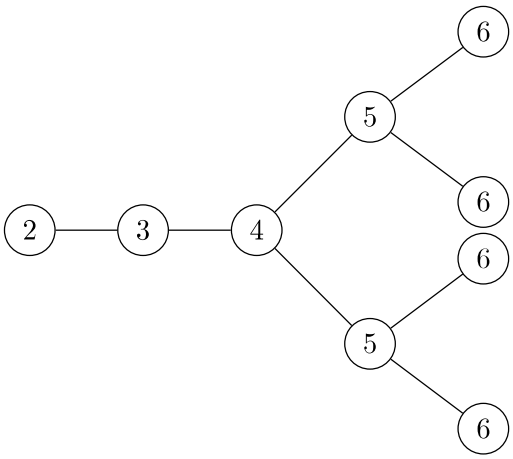



Figure 2: Tree $T_2$: 5 levels.

It is necessary to merge all the trees following the same order of split (1), aiming to get realizations of the whole molecule. The procedure is merging each branch from one tree to all branches from the other, one at a time. We denote the $t^{th}$ branch of a tree $T_p$ as $T_{p,t}$.

In order to produce mergeable trees, one has to assume that two consecutive intervals $M_p$ and $M_{p+1}$ have, at least, three atoms in the intersection [6]. Consider, then, the two consecutive trees $T_p$ and $T_{p+1}$ relative to the previously mentioned intervals. As they have three levels of intersection, we consider the tree $T_p$ to be fixed, calling it *Base Tree*, and we move the other tree $T_{p+1}$, which we name *Sliding Tree*, towards $T_p$ using Euclidean transformations in order to preserve lenghts and angles. Assume that the last three atoms of the base interval are $i, j$ and $k$, respectively ordered, and let $T_{x,y}(z)$ be the generic notation for the position of the atom $z \in \{1, 2, \ldots, |T_{x,y}|\}$ in the branch $y$ of the binary tree $x$. Also, if the number of feasible branches in a tree $T$ is denoted by $|T|$, then the final number of feasible realizations of $M$, provided by the MRT method, is $r$, which is defined by the multiplication

$$r = |T_1||T_2| \ldots |T_k| \leq |T|.$$

Three Euclidean transformations are necessary to merge the arbitrary branches $T_{p+1,t}$ and $T_{p,q}$. The first one is a *translation* that makes $T_{p+1,t}(i) \to T_{p,q}(i)$, shown in Figure 2.

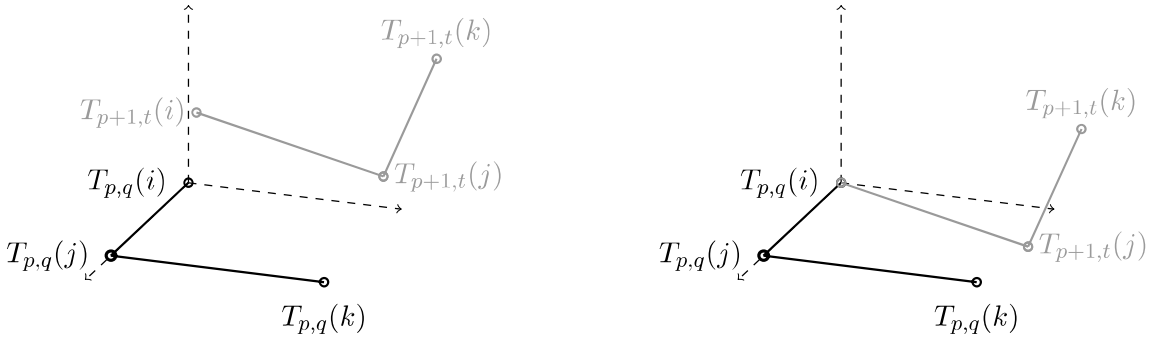



Figure 3: First Euclidean transformation: a translation of both realized branches.

We also want to make $T_{p+1,t}(j) \to T_{p,q}(j)$, without losing what we have built with the translation. Let us denote $E_p = T_{p,q}(j) - T_{p,q}(i)$ and $E_{p+1} = T_{p+1,t}(j) - T_{p+1,t}(i)$ and let $\theta$ be the angle between $E_p$ and $E_{p+1}$. We apply a *plane rotation* of $\theta$ in the branch $T_{p+1,t}$, as one can see in Figure 4.

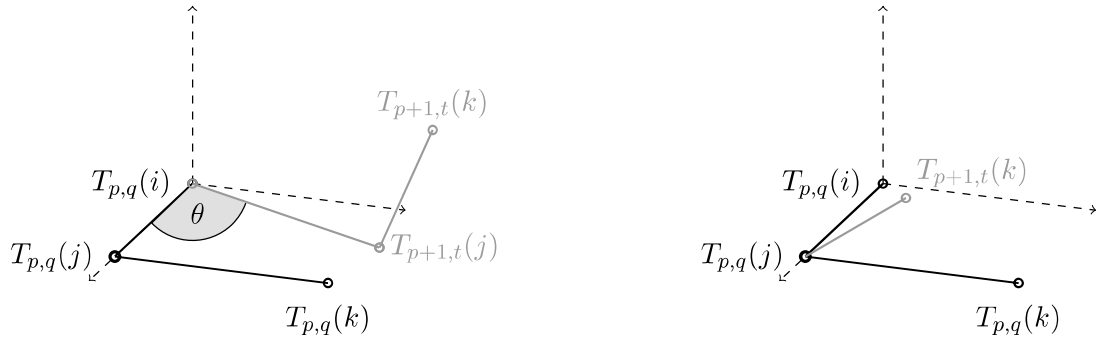



Figure 4: Second Euclidean transformation: a plane rotation in terms of $\theta$.

Finally, after one translation and one rotation, consider $F_p = T_{p,q}(k) - T_{p,q}(j)$ and $F_{p+1} = T_{p+1,t}(k) - T_{p+1,t}(j)$. We want to move the sliding branch $T_{p+1,t}$ such that it satisfies $T_{p+1,t}(k) \to T_{p,q}(k)$, without moving anything else which has been already transformed previously. We define the rotation axis, whose attitude $L$ is spanned by $T_{p,q}(j) - T_{p,q}(i)$, and consider the plane $\mathbb{P}$, orthogonal to this axis. Let $\mathbf{P} = I_3 - LL^T$ be the matrix that gives the orthogonal projection to $\mathbb{P}$.

Then, the projections of $F_p$ and $F_{p+1}$ in $\mathbb{P}$ are, respectively,

$$P_p = \mathbf{P}F_p \qquad \text{and} \qquad P_{p+1} = \mathbf{P}F_{p+1}.$$

Now, let $\varphi$ be the angle between $P_p$ and $P_{p+1}$. Thus, we rotate $F_{p+1}$ towards $F_p$ in $\varphi$ about the axis spanned by $L$, as it is shown in Figure 5. So we do with the remaining structure. Therefore, both realizations are connected and supposed to respect all original distance and angle constraints.

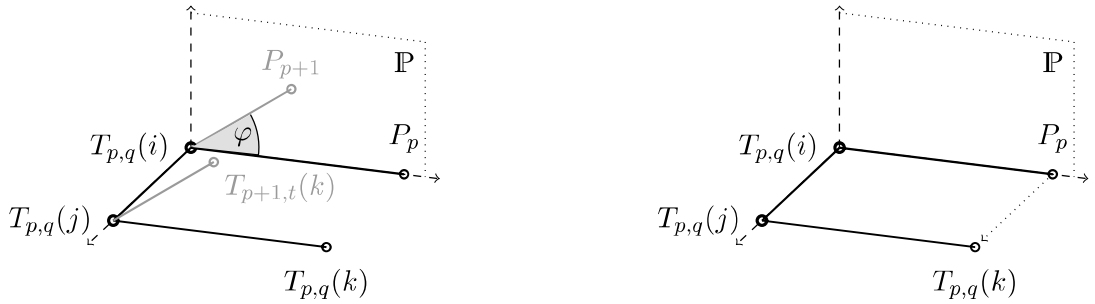



Figure 5: Third transformation: a spatial rotation in terms of the projected angle $\varphi$.

Following this outline, all the trees are connected and their branches consist of feasible points that solve the DMDGP. In addition, we remark that all rotations are computed using matrices.

## 3. Merging Trees with Quaternion Rotations

All general rotations in real 3-D space can be represented by an axis, spanned by a unitary vector $\mathbf{n}$, and an angle $\theta$. Using this information, one can build the matrix $R_{\mathbf{n},\theta}$ which carries out a general rotation and can be determined by using the matrix form of *Rodrigues' Rotation Formula* [7]

$$R_{\mathbf{n},\theta} = I + \sin(\theta)J(\mathbf{n}) + (1 - \cos(\theta))J(\mathbf{n})^2, \tag{2}$$

where $J(\mathbf{n})$ is a skew-symmetric $3 \times 3$ - matrix generated by the elements of $\mathbf{n}$ as

$$J(\mathbf{n}) = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

Such rotation matrices need 37 arithmetic operations to be determined, according to Equation (2), and 9 positions of memory to be stored. In addition, it is necessary to use more 15 operations to multiply it for a vector $\mathbf{v}$ we want to rotate, totalizing 52 arithmetic operations.

This work aims to propose a theoretical modification on the tools which are used to make rotations on three-dimensional structures in order to decrease the storage space and, consequently, the number of operations to accelerate this process. Our approach uses the Quaternion Algebra $\mathbb{H}$ [4] to do that.

Consider the unit quaternion $q = q_0 + \mathbf{q_v}$, where $q_0 \in \mathbb{R}$ and $\mathbf{q_v} \in \mathbb{R}^3$. It is possible to prove that there is an unique angle $0 \leq \theta \leq \pi$ such that $q_0 = \cos(\theta)$ and $\|\mathbf{q_v}\| = \sin(\theta)$. Then, we can rewrite $q = \cos(\theta) + \mathbf{u}\sin(\theta)$, where $\mathbf{u} = \frac{\mathbf{q_v}}{\|\mathbf{q_v}\|}$ [4]. The conjugate of $q$ can be written as $q^* = \cos(\theta) - \mathbf{u}\sin(\theta)$ [4]. Now, the following outcome characterizes a quaternion rotation by means of a linear operator[4].

**Theorem 1** (Quaternion Rotation Operator)**.** *For a unit quaternion $q = \cos(\theta) + \mathbf{u}\sin(\theta)$, the operator $R_q : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$, whose action on the vector $\mathbf{v} \in \mathbb{R}^3$ is given by $R_q(\mathbf{v}) = q\mathbf{v}q^*$, is a*

*rotation operator which rotate vectors about the axis spanned by the unit vector* $\mathbf{u}$ *through an angle* $2\theta$ *in clockwise sense.*

Explicitly, the action of $R_q$ in a vector $v \in \mathbb{R}^3$ can be derived as the *Rodrigues' Rotation Formula*

$$R_q(\mathbf{v}) = \cos(2\theta)\mathbf{v} + (1 - \cos(2\theta))\mathbf{p_u}(\mathbf{v}) + \sin(2\theta)(\mathbf{u} \times \mathbf{v}), \tag{3}$$

where $\mathbf{p_u}(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})\mathbf{u}$ is the orthogonal projection of $\mathbf{v}$ in the direction of $\mathbf{u}$.

First of all, one improvement we can realize on the use of quaternions is that it is easier to encode a rotation in a unit quaternion than in an orthogonal matrix. Moreover, Equation (3) shows that each quaternion rotation creates a local frame $(\mathbf{v}, \mathbf{p_u}(\mathbf{v}), \mathbf{u} \times \mathbf{v})$ to represent the rotated vector and, also, gives us a route on how to localize it in the space more easily.

On computational complexity, quaternion rotations require the storage of only four positions instead of nine, necessary in the use of three-dimensional matrices. In addition, 25 arithmetic operations are used to compute such rotations, fashioned such as in Equation (3). As we can see, this number of operations is reasonably less than the one used in the matrix approach. When considering more than one rotation, it seems to be even more efficient by saving space and floating-point-arithmetic operations.

Therefore, we apply these ideas as efficient tools in the merging of BP-trees. According to the MRT procedure described previously, we have to carry out two rigid rotations in the sliding structure. For each one of them, it is necessary first to determine the cosine of the rotation angle by using the usual dot product in $\mathbb{R}^3$, restricting the domain of the cosine function to the range $[0, \pi]$ in order not to allow it to reach the position determined by the angle $2\pi - \theta$ since both angles have the same cosine value. Moreover, the unitary vector which spans the oriented rotation axis can be chosen by applying the usual cross product in 3-D Euclidian Space, whose signal induces the orientation of the rotation, following the so-called *Right-Hand Rule*. Thus, the order in the cross product really matters: indeed, the axis for a rotation of a vector $\mathbf{x}$ towards another vector $\mathbf{y}$ is spanned by the vector $\mathbf{x} \times \mathbf{y}$, while the axis for the rotation of $\mathbf{y}$ towards $\mathbf{x}$ is spanned by the vector $\mathbf{y} \times \mathbf{x}$. As they satisfy the relation of anticommutativity $\mathbf{y} \times \mathbf{x} = -(\mathbf{x} \times \mathbf{y})$, the direction of the rotation is, therefore, encoded in the sign of the cross product.

The first rotation is supposed to take $E_{p+1}$ into $E_p$, since they have the same origin. Thus, the cosine of the angle and the unitary vector which spans the correspondent axis are, respectively,

$$\cos(\theta) = \frac{\langle E_{p+1}, E_p \rangle}{\|E_{p+1}\| \, \|E_p\|} \quad \text{and} \quad \mathbf{n} = \frac{E_{p+1} \times E_p}{\|E_{p+1} \times E_p\|}. \tag{4}$$

Using the parameters developed above, we associate the following quaternion element to such rotation

$$q_{\theta,\mathbf{n}} = \cos(\tfrac{\theta}{2}) + \mathbf{n}\sin(\tfrac{\theta}{2}).$$

Then, employing the result displayed in Theorem 1, we apply the rotation in the sliding structure $T_{p+1,t}$

$$T_{p+1,t}(u) \leftarrow R_{q_{\theta,\mathbf{n}}}(T_{p+1,t}(u)), \qquad \text{for } u = 2, \ldots, |T_{p+1,t}|, \tag{5}$$

where $|T_{p+1,t}|$ is the number of vertices in this feasible branch of the realization tree $T_{p+1}$.

Further, assume $L = E_p / \|E_p\|$, $F_p = T_{p,q}(k) - T_{p,q}(j)$ and $F_{p+1} = T_{p+1,t}(k) - T_{p+1,t}(j)$. The orthogonal projection matrix, associated to the plane $\mathbb{P}$, is given by $M = I_3 - LL^T$, as we have seen. Then, the projections are given by the vectors $P_p = MF_p$ and $P_{p+1} = MF_{p+1}$. Analogously to (4), the second rotation is generated by the parameters

$$\cos(\varphi) = \frac{\langle P_{p+1}, P_p \rangle}{\|P_{p+1}\| \, \|P_p\|} \quad \text{and} \quad \mathbf{m} = \frac{P_{p+1} \times P_p}{\|P_{p+1} \times P_p\|}. \tag{6}$$

After calculating them, we define the associated quaternion to the respective rotation by

$$q_{\varphi,\mathbf{m}} = \cos(\tfrac{\varphi}{2}) + \mathbf{m}\sin(\tfrac{\varphi}{2}).$$

Therefore, we rotate the sliding structure, again as in Theorem 1, by making

$$T_{p+1,t}(u) \leftarrow R_{q_{\omega,\mathbf{m}}}(T_{p+1,t}(u)), \qquad \text{for } u = 3, \ldots, |T_{p+1,t}|, \tag{7}$$

concluding the merging of the two structures $T_{p,q}$ and $T_{p+1,t}$.

There are two rotations in this approach. The first one fixes the first vertex of the sliding structure and the second one fixes both the first and the second vertices. Then, compounding both the rotations in only one and applying it in the sliding structure leads us to reach the same resulting structure. It is reasonably easier and computationally cheaper to compose two quaternion rotations than multiplying two rotation matrices [4, 2]. Using this, we can save half of the storage space and carry out less than the half of arithmetic operations for the transformation of each point.

As a conclusion, using quaternion rotations, instead of matrices, can bring improvements either about computational time or about simplifying the method. We are in the process of implementing these ideas in order to illustrate computationally the theoretical improvements.

# References

[1] G. Crippen and T. Havel. *Distance Geometry and Molecular Conformation.* Research Studies Press, U.K., 1988.

[2] D. Eberly. Rotation Representation and Performance Issues. *A summary of representations of rotations and performance available in http://www.geometrictools.com/Documentation/RotationIssues.pdf*, 2002.

[3] T. Havel. Distance Geometry. In D. Grant and R. Harris, editors, *Encyclopedia of Nuclear Magnetic Resonance, pp. 1701–1710.* Wiley, New York. 1995.

[4] J. Kuipers. *Quaternions and Rotation Sequences.* Princeton University Press, Princeton, 1998.

[5] C. Lavor, L. Liberti, N. Maculan, and A. Mucherino. The discretizable molecular distance geometry problem. *Computational Optimization and Applications*, 52:115–146, 2012.

[6] P. Nucci, L. Nogueira, and C. Lavor. Solving the Discretizable Molecular Distance Geometry Problem by multiple realization trees. In Mucherino, A., Lavor, C., Liberti. L., and Maculan, N., editors, *Distance Geometry: Theory, Methods and Applications.* Springer, New York, 2013.

[7] C. Taylor and D. Kriegman. Minimization on the lie group SO(3) and related manifolds. *Technical Report, Yale University*, 1994.