PEPTOID DESIGN WITH ADAPTIVE GEOMETRIC SEARCH

1. Introduction

Peptoids are artificial proteins synthesised in the chemical labs, first developed around 20 years ago(ref). To endorse them with important chemical functions such as binding the ends of side chains with metal ions, their side chains need to be of certain types to be able to stablize in positions close to the binding configurations. When the peptoids' low energy states resemble these binding configurations, they are more likely to bind with metal ions since chemical structures tend towards low energy states. Various metal bindings are observed in natural proteins so we can measure these binding positions. Then we can synthesize peptoids whose low energy states resemble these observed binding positions in natural proteins that bind. However, for one peptoid design the choices of side chains on one particular backbone are exponential and impossible to be carried out in labs with the brute force of synthesising and testing them all. Therefore we need an efficient methodology to predict the low energy states of these peptoids to be synthesised. Predicting the low energy states of proteins is often called the hard problem of protein folding, which is the problem we are trying to solve in the case for peptoids.

Our approach in peptoid design is a two step process. In the first step, we consider the most influential energies in protein folding. Then an efficient geometric search is conducted to eliminate all the impossible designs. In the second step, designs that passed the quick initial screening will be further examined in the comprehensive protein folding software called Rosetta. This two step process efficiently saves all the time that the majority impossible designs would take to be evaluated by Rosetta and to be synthesised.

2. Adaptive geometric search

In peptoids, since energies of bond angles and bond lengths are larger than energies of torsion angles etc. by a magnitude of 100 or more(be more precise), in the first step we are going to consider just the torsion angles as variables in the energy function. Furthermore we are going to consider only the lowest energy bond lengths and bond angles because with all other minor energies ignored, this condition gives the lowest energy configuration of the peptoid.

Without loss of generality, consider the binding configuration as a polygon whose vertices are the binding sites of the configuration. For each binding site or each vertex of the polygon, all the possible positions of the potential binding node (often the end node) on all possible side chains form a manifold, or a series of surfaces in 3D space. The task is to find one potential binding node from each manifold such that the polygon whose vertices are these potential binding nodes is a "desirable configuration." Let $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ be the target binding configuration. In this paper all polygons are denoted by putting curly brackets around their vertices. We define the error ϵ of a configuration or polygon $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ by its distance to the target configuration $\mathcal{P}$, that is,

$$\epsilon = \max_{1\leq i,j\leq k} |P_iP_j - S_iS_j|.$$

Following the standard notation, we use capital letters to denote a point in space and two points written next to each other to denote the length of the line segment connecting them. Let ϵ_T be the maximum error we allow due to the discretisation of manifolds in computing. If $\epsilon < \epsilon_T$, we call configuration or polygon $\mathcal{S}$ a "desirable configuration".

Given a target polygon $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$, a tolerance $\epsilon_T \geq 0$ and one edge (P_i, P_j). Let $\mathcal{C}_i, \mathcal{C}_j$ be two cubes with side length l and the distance between their centers d. Then we have the following results.

Theorem 1. *If $d < P_iP_j - \epsilon_T - \sqrt{3}\,l$ or $d > P_iP_j + \epsilon_T + \sqrt{3}\,l$, then there are no polygons $\{\tilde{P}_1, \tilde{P}_2, \dots, \tilde{P}_n\}$ within ϵ_T distance from polygon $\mathcal{P}$ where $\tilde{P}_i \in \mathcal{C}_i, \tilde{P}_j \in \mathcal{C}_j$.*

Proof. If there exists a polygon $\{\tilde{P}_1, \tilde{P}_2, \dots, \tilde{P}_n\}$ within ϵ_T distance from polygon $\mathcal{P}$ where $\tilde{P}_i \in \mathcal{C}_i, \tilde{P}_j \in \mathcal{C}_j$, then by the definition of the distance between polygons, we have $\tilde{P}_i\tilde{P}_j \in [P_iP_j - \epsilon_T, P_iP_j + \epsilon_T]$. Since any points $P \in \mathcal{C}_i, Q \in \mathcal{C}_j$ must satisfy $PQ \in [d - \sqrt{3}\,l, d + \sqrt{3}\,l]$, we derive that

$$P_iP_j + \epsilon_T \geq d - \sqrt{3}\,l,$$
$$P_iP_j - \epsilon_T \leq d + \sqrt{3}\,l.$$

Written in another form, we have $P_iP_j - \epsilon_T - \sqrt{3}\,l \leq d \leq P_iP_j + \epsilon_T + \sqrt{3}\,l$. The theorem states the contraposition of this conclusion. □

Theorem 1 suggests a necessary condition to determine whether two cubic regions are plausible to contain any desirable pairs of points. There are other necessary conditions based on positions of points in the cubes, but in comparison this condition is tighter and more efficient for computation. On the other hand, sufficient conditions exist for all pairs of points from a pair of cubes being desirable. Again we use the sufficient condition based on the center positions of cubes.

Theorem 2. *If $P_iP_j - \epsilon_T + \sqrt{3}\,l \leq d \leq P_iP_j + \epsilon_T - \sqrt{3}\,l$, then all pairs of points from $\mathcal{C}_i, \mathcal{C}_j$ are within ϵ_T distance from P_iP_j.*

Proof. The distance l' between all pairs of points from $\mathcal{C}_i, \mathcal{C}_j$ satisfies $d - \sqrt{3}\,l \leq l' \leq d + \sqrt{3}\,l$. If $P_iP_j - \epsilon_T + \sqrt{3}\,l \leq d \leq P_iP_j + \epsilon_T - \sqrt{3}\,l$, then substituting the tighter bound of d on each side of the inequality we have $P_iP_j - \epsilon_T \leq l' \leq P_iP_j + \epsilon_T$. $\square$

Notice that the condition of Theorem 2 is only possible when $P_iP_j - \epsilon_T + \sqrt{3}\,l \leq P_iP_j + \epsilon_T - \sqrt{3}\,l$, or $l \leq \epsilon_T/\sqrt{3}$. Thus we design an adaptive search algorithm which searches for plausible cubic regions on each level according to Theorem 1, and checks for sufficiency when the cube length l satisfies $l \leq \epsilon_T/\sqrt{3}$. Pairs of cubes that satisfy the necessary condition but not the sufficient condition are refined into sub-cubes, until when they contain so few points such that a thorough search over every pair of points is not costly. This suggests limiting the maximum number of points in all the cubes, which in turn suggests using the data structure of an octree. However in octrees two nodes in comparison can be of different status, that is, comparison of an intermediate node with a leaf node on the same level. To efficiently refine an area in relation to a point, we need the following theorem.

Theorem 3. *Given a point G, a cube $\mathcal{C}$ with length l and the target distance P_iP_j. Let d be the distance from G to the center of $\mathcal{C}$. If $d > P_iP_j + \epsilon_T + \frac{\sqrt{3}}{2}\,l$ or $d < P_iP_j - \epsilon_T - \frac{\sqrt{3}}{2}\,l$, then there are no polygons $\{\tilde{P}_1, \tilde{P}_2, \ldots, \tilde{P}_n\}$ within ϵ_T distance from polygon $\mathcal{P}$ where $\tilde{P}_i \in \mathcal{C}$, $\tilde{P}_j = G$. If $P_iP_j - \epsilon_T + \frac{\sqrt{3}}{2}\,l \leq d \leq P_iP_j + \epsilon_T - \frac{\sqrt{3}}{2}\,l$, then GH for all points $H \in \mathcal{C}$ are within ϵ_T distance from P_iP_j.*

Proof. The proof is similar to the proofs of Theorem 1 and 2, except now G is a fixed point and thus $GH \in \left[d - \frac{\sqrt{3}}{2}\,l, d + \frac{\sqrt{3}}{2}\,l\right]$ for any point $H \in \mathcal{C}$. $\square$

Again we check both the necessary and sufficient conditions when comparing a point to sub-cubes of a cubic region. If necessary but not sufficient conditions are satisfied by some cube, we refine the search there until on the last level all cubes either violate the necessary condition, or satisfy the sufficient condition, or are leaf nodes in the octree and all points are exhaustively searched.

Let $t_1, t_2, \ldots, t_n$ be the octrees generated by each manifold. Recall that a manifold is a set of potential binding sites corresponding to a side chain.

We offer two adaptive search algorithms below, a basic version Algorithm 1 and a full version Algorithm 2 that takes into consideration of sufficient conditions and point-cube adaptive search. Notice that the algorithm can be further optimized by checking and deleting cubes on each level over all n octrees.

Algorithm 1 Adaptive Geometric Search ($\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}, \mathcal{P}, \epsilon_T$)

```
1: $trees$ = $[t_1, t_2, \ldots, t_n]$
2: for $i$ in range($n$) do
3:     $pairs$ = []
4:     $l = P_i P_{(i+1)\, mod\, n}$
5:     $t_1, t_2$ = trees[i], trees[(i+1) mod n]
6:     $depth = min(t_1.depth, t_2.depth)$
7:     $combos$ = [[] for x in range(depth+1)]
8:     $combos[0]$ = [$(t_1, t_2)$]
9:     for $i$ in range($depth + 1$) do
10:        for $(b_0, b_1)$ in $combos[i]$ do
11:            if $b_0, b_1$ are not leaves then
12:                $combos[i+1] \mathrel{+}= Compare(b_0, b_1, l, \epsilon_T)$
13:            else
14:                $pairs \mathrel{+}= SearchE(b_0, b_1, l, \epsilon_T)$
15:            end if
16:        end for
17:    end for
18:    $M_i = \{M_{i,j} = 1$ for $(p_i, p_j) \in pairs; M_{i,j} = 0$ otherwise$\}_{n\times n}$
19: end for
20: Trace($\prod_{i=1}^{n} M_i$)

21: function COMPARE($b_0, b_1, l, \epsilon_T$)
22:    return $[(c_i, c_j)$ for $(c_i, c_j)$ in $b_0.children \times b_1.children$ if
       $|(c_i.center, c_j.center) - l| \leq \epsilon_T + \sqrt{3}\, c_i.length]$
23: end function

24: function SEARCHE($b_0, b_1, l, \epsilon_T$)
25:    return $[(p_i, p_j)$ for $(p_i, p_j)$ in $b_0.containedPoints \times$
       $b_1.containedPoints$ if $|p_i p_j - l| \leq \epsilon_T]$
26: end function
```

Algorithm 2 Adaptive Geometric Search ($\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}, \mathcal{P}, \epsilon_T$)

```
    $trees$ = $[t_1, t_2, \ldots, t_n]$
2:  for $i$ in range($n$) do
        $pairs$ = []
4:      $l = P_i P_{(i+1)\, mod\, n}$
        $t_1, t_2$ = trees[i], trees[(i+1) mod n]
6:      $depth = min(t_1.depth, t_2.depth)$
        $combos$ = [[] for x in range(depth+1)]
8:      $combos[0]$ = [$(t_1, t_2)$]
        for $i$ in range($depth + 1$) do
10:         for $(b_0, b_1)$ in $combos[i]$ do
                $l_b = \sqrt{3}\, b_0.length$
12:             $d =$ distance between $b_0, b_1$
                if $l - \epsilon_T + l_b \leq d \leq l + \epsilon_T - l_b$ then
14:                 $pairs$ += [$(p_i, p_j)$ for $(p_i, p_j)$ in $b_0.containedPoints \times b_1.containedPoints$]
                else if neither $b_0, b_1$ are leaves then
16:                 $combos[i+1]$ += $Compare(b_0, b_1, l, \epsilon_T)$
                else if $b_0, b_1$ are both leaves then
18:                 $pairs$ += $SearchE(b_0, b_1, l, \epsilon_T)$
                else
20:                 for $p$ in leaf $b_i$, $i = 0$ or $1$ do
                        $pairs$ += $AdaptSearch(p, b_{1-i}, l, \epsilon_T)$
22:                 end for
                end if
24:         end for
        end for
26:     $M_i = \{M_{i,j} = 1$ for $(p_i, p_j) \in pairs; M_{i,j} = 0$ otherwise$\}_{n \times n}$
    end for
28: Trace($\prod_{i=1}^{n} M_i$)

    function COMPARE($b_0, b_1, l, \epsilon_T$)
30:     return [$(c_i, c_j)$ for $(c_i, c_j)$ in $b_0.children \times b_1.children$ if $|(c_i.center, c_j.center) - l| \leq \epsilon_T + \sqrt{3}\, c_i.length$]
    end function

32: function SEARCHE($b_0, b_1, l, \epsilon_T$)
        return [$(p_i, p_j)$ for $(p_i, p_j)$ in $b_0.containedPoints \times b_1.containedPoints$ if $|p_i p_j - l| \leq \epsilon_T$]
34: end function
```

```
function ADAPTSEARCH($p, b, l, \epsilon_T$)
    $pairs$ = []
    $combos$ = [[] for x in range($b.depth + 1$)]
    $combos[0]$ = [$b$]
    for $i$ in range($b.depth + 1$) do
        for $c$ in $combos[i]$ do
            $l_c = \sqrt{3}/2 \times c.length$
            $d =$ distance between $p$ and $c.center$
            if $l - \epsilon_T + l_c \leq d \leq l + \epsilon_T - l_c$ then
                $pairs \mathrel{+}= [(p, q)$ for $q \in c.containedPoints]$
            else if $c$ is not a leaf then
                $combos[i+1] \mathrel{+}= CompareP(p, c, l, \epsilon_T)$
            else
                $pairs \mathrel{+}= SearchEP(p, c, l, \epsilon_T)$
            end if
        end for
    end for
    return $pairs$
end function

function COMPAREP($p, c, l, \epsilon_T$)
    return [$b_i$ for $b_i$ in $c.children$ if $|(p, b_i.center) - l| \leq \epsilon_T + \sqrt{3}/2\, b_i.length$]
end function

function SEARCHEP($p, c, l, \epsilon_T$)
    return [$(p, q)$ for $q$ in $c.containedPoints$ if $|pq - l| \leq \epsilon_T$]
end function
```

3. ANALYSIS OF ALGORITHM

The algorithm has two parts, the adaptive search on two octrees and the matrix multiplication (including the computation of the trace) part. Let S be the upper bound for the number of ones in all matrices M_i for $i = 1, 2, \ldots, n$. Let N be the number of sample points in each manifold. Since M_i's are sparse matrices, computing their product has time complexity at most $\mathcal{O}(SN)$. The complexity of computing the trace is also linear in N. Thus the second part has time complexity $\mathcal{O}(N)$.

Now let's consider the adaptive search part of the algorithm. For simplicity, we assume the sample points are evenly distributed in octrees t_1, t_2. Let n_T be the stopping threshold for both octrees, i.e. the maximum points per cube. In the best case scenario, the algorithm finds only one pair of cubes plausible on each level of the octrees. In this case the time complexity is $\mathcal{O}(d)$ where d is the smaller of the depths of two octrees. In the worst case scenario, the algorithm would be searching through all pairs of cubes at each level of the octrees until the last pair of the lowest level cubes are reached, which are also the leaves of the octrees. The algorithm's time complexity would be $\mathcal{O}(N_1N_2)$ where N_i is the number of cubes or nodes in octree t_i for $i = 1, 2$. If the sample points are evenly distributed, the number of cubes on each level of the octree is linear in N and thus this time complexity is equivalent to that of the exhaustive search $\mathcal{O}(N^2)$. However the next theorem shows that the worst scenario can be easily avoided. Let l_0 be the length of the initial cubes, or the root of each octree. We call the root of the octree level zero. Then the eight children of the root are the first level cubes with length $l_0/2$. And the eight children of the first level cubes are the second level cubes with length $l_0/4$, etc.

Theorem 4. *Given the target polygon edge length l, octrees t_1, t_2 with root cubes of length l_0 and distance d between the centers of them. If $\epsilon_T < \frac{14-5\sqrt{3}}{32}\, l_0$, then all pairs of cubes on level 3 or below can not be plausible.*

Proof. For any two cubes $\mathcal{C}_1 \in t_1, \mathcal{C}_2 \in t_2$ from some level k, let c_1, c_2 be the centers and l_c be the common side length. By Theorem 1, in order for $\mathcal{C}_1, \mathcal{C}_2$ to be a plausible pair, we have

$$-\sqrt{3}\, l_c - \epsilon_T + l \leq c_1c_2 \leq \sqrt{3}\, l_c + \epsilon_T + l.$$

Let $\mathcal{L}$ be the set of distances between all pairs of cubes from level k. Depending on the relative configuration of the initial cubes t_1, t_2, the set $\mathcal{L}$ has different ranges. Notice that c_1 and c_2 lie at least $l_c/2$ distance away from all the faces of t_1 and t_2. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the cubes that are l_c smaller than t_1 and t_2 on each side length, but centered and positioned as t_1, t_2. Thus $c_1 \in \mathcal{B}_1$ and $c_2 \in \mathcal{B}_2$. Consider the sphere $\mathcal{S}_1$ inside $\mathcal{B}_1$ where the center of $\mathcal{S}_1$ is the center of the cube $\mathcal{B}_1$, and $\mathcal{S}_1$ is tangent to all six faces of $\mathcal{B}_1$. Similarly define $\mathcal{S}_2$ by $\mathcal{B}_2$. Notice that $\mathcal{S}_1, \mathcal{S}_2$ have radius $\frac{1}{2}(l_0 - l_c)$. Since both $\mathcal{S}_1$ and $\mathcal{S}_2$ are entirely contained by $\mathcal{B}_1$, $\mathcal{B}_2$ and $c_1 \in \mathcal{B}_1, c_2 \in \mathcal{B}_2$, if $\mathcal{B}_1, \mathcal{B}_2$ are disjoint, we have the relationship $\min(\mathcal{L}) \leq \min d(\mathcal{S}_1, \mathcal{S}_2) \leq \max d(\mathcal{S}_1, \mathcal{S}_2) \leq \max(\mathcal{L})$. Since $\min d(\mathcal{S}_1, \mathcal{S}_2) = d - l_0 + l_c$ and $\max d(\mathcal{S}_1, \mathcal{S}_2) = d + l_0 - l_c$, in order for all pairs of cubes on level k to be plausible we have

$$\min d(\mathcal{S}_1, \mathcal{S}_2) = d - l_0 + l_c \geq l - \sqrt{3}\, l_c - \epsilon_T$$
$$\max d(\mathcal{S}_1, \mathcal{S}_2) = d + l_0 - l_c \leq l + \sqrt{3}\, l_c + \epsilon_T.$$

Thus,

$$\begin{aligned} l-&\sqrt{3}\, l_c - \epsilon_T + l_0 - l_c \leq d \leq l + \sqrt{3}\, l_c + \epsilon_T - l_0 + l_c \\ &\Longrightarrow \qquad \epsilon_T \geq l_0 - (1+\sqrt{3})\, l_c. \end{aligned}$$

On the second level $l_c = l_0/4$. Then we can always set ϵ_T to be smaller than $\frac{3-\sqrt{3}}{4}\, l_0$ and violate the necessary condition for all pairs of cubes on this level to be plausible.

If $\mathcal{B}_1, \mathcal{B}_2$ have overlap, we still have $\max d(\mathcal{S}_1, \mathcal{S}_2) \leq \max(\mathcal{L})$ but $0 \leq \min(\mathcal{L}) \leq \frac{\sqrt{3}}{2} l_c$. The left side equality holds when there are c_1, c_2's that coincide. The right side equality holds when all c_2's lie on the corners of the cubes centered by c_1's and all c_1's lie on the corners of the cubes centered by c_2's. Thus in this case,

$$\begin{gathered} \frac{\sqrt{3}}{2}\, l_c \geq l - \sqrt{3}\, l_c - \epsilon_T \\ d + l_0 - l_c \leq l + \sqrt{3}\, l_c + \epsilon_T. \end{gathered}$$

Thus,

$$\begin{aligned} d + l_0 - l_c - &\sqrt{3}\, l_c - \epsilon_T \leq l \leq \frac{3\sqrt{3}}{2}\, l_c + \epsilon_T \\ \Longrightarrow \qquad & \epsilon_T \geq \frac{1}{2}\left(d + l_0 - (1 + \frac{5\sqrt{3}}{2})\, l_c \right) \\ & \quad \geq \frac{1}{2}\left(l_0 - (1 + \frac{5\sqrt{3}}{2})\, l_c \right). \end{aligned}$$

On the third level, $l_c = l_0/8$. Then we can set ϵ_T to be smaller than $\frac{14-5\sqrt{3}}{32}\, l_0$ and violate the necessary condition for all pairs of cubes on this level to be plausible. □

Since in practice our octrees are always more than three levels deep and ϵ_T is set about 1% of l_0, the worst case scenario can never happen. Furthermore, since from the third level we have at least one cube that's not plausible for refinement, for $\forall k \geq 3$, let α be the maximum portion of the cubes that's plausible. Then $\forall k$,

$$\alpha = \frac{8^k - 8^{k-3}}{8^k} = 1 - 8^{-3}.$$

Let the depth of the octrees be d. Then $N/n_T = 8^d$. Let η_k be the ratio of plausible cubes on level k and $\eta_k \leq \alpha$. Now the total number of computations for the adaptive search between t_1, t_2 is

$$\begin{aligned}&\mathcal{O}(8^2 + 8^2\eta_1 \cdot 8^2 + (8^2)^2\eta_1\eta_2 \cdot 8^2 + (8^2)^3\eta_1\eta_2\eta_3 \cdot 8^2 + \\ &\quad \cdots + (8^2)^{d-1}\eta_1\eta_2\cdots\eta_{d-1} \cdot 8^2 + (8^2)^d\eta_1\eta_2\cdots\eta_d \cdot C) \\ &\leq \mathcal{O}(8^{2d}\alpha^d) = \mathcal{O}(N^{2+\log_8\alpha}) = \mathcal{O}(N^{2-[3-\log_8(8^3-1)]}).\end{aligned}$$

So far we rigorously proved that the adaptive geometric search algorithm has time complexity at most $\mathcal{O}(N^{2-[3-\log_8(8^3-1)]} + SN)$ where S is the maximum number of desirable pairs of points between any two manifolds. Although the theoretical bound for the exponent of N is close to 2, in practice it can be much lower. Here is the reason. The time complexity of the adaptive search between two octrees is dominated by the highest order term which happens at the lowest level, depth d of the octrees. Assume again that the points are evenly distributed with ϵ distance apart for both octrees. Then all cubes on level d are leaf cubes. Let the target polygon side length be l and let the tolerance be ϵ_T. Let the initial cube length be l_0 and let the total number of points inside be N. Note that $l_0/\epsilon = N^{1/3}$. For each leaf cube of t_1 with center c, all the leaf cubes of t_2 which lie in the sphere $\mathcal{B}_c(l + \sqrt{3}\,\frac{l_0}{2^d} + \epsilon_T)$ and outside of the sphere $\mathcal{B}_c(l - \sqrt{3}\,\frac{l_0}{2^d} - \epsilon_T)$ are plausible. Let $\Delta\mathcal{B} = \mathcal{B}_c(l + \sqrt{3}\,\frac{l_0}{2^d} + \epsilon_T) \setminus \mathcal{B}_c(l - \sqrt{3}\,\frac{l_0}{2^d} - \epsilon_T)$. If there are any desirable pairs of points from t_1, t_2, the sphere shell $\Delta\mathcal{B}$ must cut across the initial cube of t_2, resulting in the following estimation.

$$\|\{c' | c' \in \Delta\mathcal{B}, c' \text{ is a center of leaf cube in } t_2\}\| \simeq \frac{2(\sqrt{3}\,\frac{l_0}{2^d} + \epsilon_T)}{l_0} \times 8^d.$$

Therefore the number of total computations heuristically is

$$\mathcal{O}(8^d \cdot \frac{2(\sqrt{3}\,\frac{l_0}{2^d} + \epsilon_T)}{l_0} \cdot 8^d) = \mathcal{O}(2^{5d} \cdot 2\sqrt{3} + 2^{6d}N^{-1/3} \cdot 2\epsilon_T/\epsilon) = \mathcal{O}(N^{5/3}).$$

4. Experiment

5. Applications in Computational Geometry

We basically solved the following problem. Given n sets $S_i, i = 1, 2, \cdots, n$ of points. The problem is to find all n-tuples of points, $(p_1, p_2, \ldots, p_n), p_i \in S_i$ for any i, that make a certain polygon shape within an error tolerance. One distinction from other more efficient algorithms for approximate spherical range search (e.g. Sunil Arya [2000]) is that our algorithm searches out all and only the n-tuples that are within the error tolerance, whereas most of the other faster methods produce only some results within the tolerance. So as a result of using these approximate methods, we can miss some working designs of peptoids. The adaptive geometric search method increased both the time and space complexity from the exhaustive search.

6. Applications in Computational Algebra

We can use adaptive geometric search to solve particular kinds of systems of equations. In many of these cases, computing Groebner bases is too slow to check if the system is consistent and looking for a solution.

If a system of equations includes at least one equation that can be written into the following form

$$\sum_{i=1}^{k}(f_i(x) - g_i(y))^2 = C,$$

then it can be viewed as a distance equation between points $(f_1(x), f_2(x), \dots, f_k(x))$ and $(g_1(y), g_2(y), \dots, g_k(y))$. Thus we separate the system of equations into the distance equations and the rest. We can always generate a set of manifolds based on the rest of the equations and use the adaptive geometric search algorithm to find all tuples of points, which are the solutions of the system, that satisfy the distance equations and lie on the manifolds. More specifically, after possibly needed changes of variables, let S_1 be the set of variables that appear in any of distance equations. Let S_2 be the set of variables that appear in all the other equations in the system. If $S_2 \subset S_1$, then we are ready to apply the adaptive geometric search algorithm. If however $S_2 \not\subset S_1$, then all variables in $S_2 \setminus S_1$ need to be scanned in a grid search before applying the adaptive geometric search algorithm, which may be inconvenient or not.

To generalize, we can view a system of equations as consisting of two parts. One part describes the manifolds, and the other part describes the geometric shape on the manifolds that we search for. If one can formulate a set of necessary conditions on geometric regions to contain solutions, then we can devise the adaptive geometric search algorithm according to these necessary conditions. In this way we can perhaps solve a broader category of systems of equations using adaptive geometric search algorithm.

References

D. M. M. Sunil Arya. Approximate range searching. *Computational Geometry: Theory and Applications*, 17:135–163, 2000.