PEPTOID DESIGN WITH ADAPTIVE GEOMETRIC SEARCH

1. Introduction

Peptoids are artificial proteins synthesised in the chemical labs, first developed around 20 years ago Zuckermann et al. [1994]. To endorse them with important chemical functions such as binding the ends of selected side chains with a particular metal ion, their side chains need to be of certain types to be able to stablize in positions that are close to the binding configurations. When the peptoids' low energy states resemble these binding configurations, they are more likely to bind with the metal ion since chemical structures always tend towards their low energy states. Various metal bindings are observed in natural proteins and their binding positions are recorded. Thus we can synthesise peptoids whose low energy states resemble these observed binding positions in natural proteins. However, for one potential peptoid design the choices of the side chains on the chosen backbone can be of an exponential order and hence impossible to be tested through trial-and-error lab experiments. Therefore we need an efficient methodology to predict the low energy states of the peptoid designs. Predicting the low energy state configurations of proteins is often called the "protein folding problem", which is the problem we are trying to solve in the case for peptoids.

Our approach in the peptoid design is a two-step process. In the first step, we consider the most influential energies in the peptoid and conduct an efficient geometric search 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.

2. Adaptive geometric search

In peptoids, since the energies of the bond angles and the bond lengths are larger than the energies of the torsion angles and other energies by an order of 10^2, in the first step we are going to consider only the lowest energy bond lengths and bond angles with the torsion angles as variables in the energy function. This makes our approach very different from that of Rosetta's. The rotamer space is based on physical chemistry considerations, e.g. trans is more stable than cis. Ours is purely geometrical.

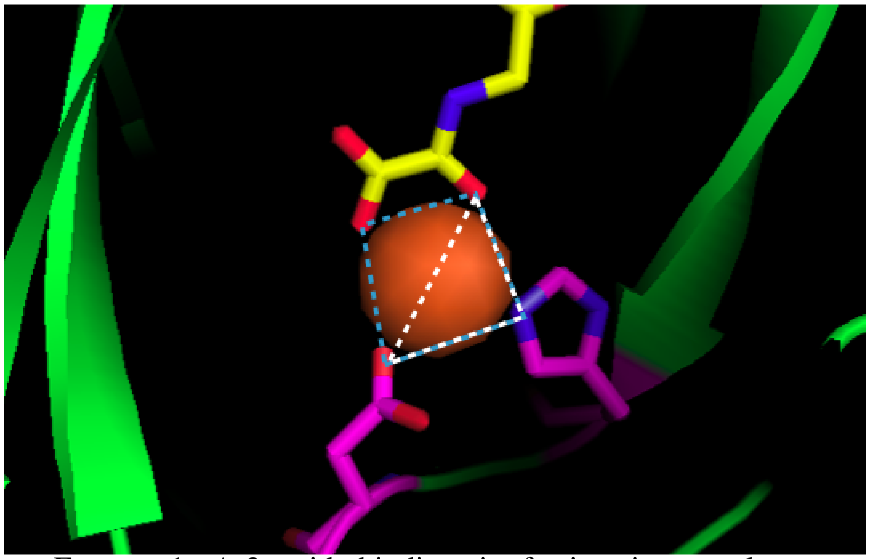

FIGURE 1. A 3-reside binding site for iron in natural protein. The dotted blue polygon is called the "binding configuration" and each vertex of the polygon is called the "binding node." Taking the right one of the two binding nodes in the top side chain as the designated node, the dotted white triangle is the "target polygon."

As seen in Figure 1, we can consider the binding configuration as a polygon whose vertices are the binding nodes. For each side chain, all the possible positions of one binding node (often the end node of the side chain) form a manifold in 3D. If there are multiple binding nodes from the same side chain (as seen in Figure 1), we designate one binding node and positions of all other binding nodes from the same side chain depend on the position of the designated node. Thus the task is to find one potential binding node from each manifold such that they form a "desirable configuration" (and later possibly check that all dependent nodes have good positions). Then let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be the target polygon. In this paper all polygons are denoted by putting curly brackets around their ordered vertices. We define the error ϵ of a configuration or a polygon $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ by its distance to the target configuration $\mathcal{P}$ defined as

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

Following the standard notation, we use capital letters to denote points in space and we write AB to denote the length of the line segment joining the points A and B. Let ϵ_T be the maximum error we allow to account for the smaller energies we are ignoring and errors due to the discretisation of the manifolds. If $\epsilon < \epsilon_T$, we call the configuration or polygon $\mathcal{S}$ a "desirable configuration".

In addition to the most basic exhaustive search method, the canonical way to search for two points that are distance l apart from two sets of N points each is to apply well-separated pair decompositions resulting in a computational complexity $\mathcal{O}(N \log N)$ in terms of N Callahan and Kosaraju [1995]. With an approximation margin $\eta\epsilon_T$, our algorithm improves the complexity to $\mathcal{O}(N)$ in terms of N.

We are going to use octrees in this algorithm, which is one of the popular data structures for computation on 3D objects. Octrees are tree structures whose nodes correspond to 3D cubes. Each node has eight children by subdividing each side of the cube by the middle in the x, y and z dimensions. All the 3D objects are stored in the leaf nodes in the octrees. Octrees have various stopping criteria to stop the tree from splitting including thresholding the maximum number of objects in a node, i.e. the octree splits only the nodes containing more than a certain number of objects. For our problem the 3D objects are points in 3D space and the stopping criterion is the minimum cube length ℓ_s, that is, the octree splits a node only if its corresponding cube has sides of length at least $2\ell_s$. Moreover, all empty nodes, i.e. nodes whose corresponding cubes contain no points, in the octrees are discarded.

To search for the desirable configurations, the algorithm first samples points from each manifold and then builds an octree for each manifold based on these sample points with the stopping criterion of the minimum cube length ℓ_s. Notice that with this stopping criterion all leaf cubes are on the lowest level in the trees. Then the algorithm compare two octrees at a time by searching adaptively into the cubic regions that pass the necessary condition 1 (see below). We call a pair of cubes that pass the necessary condition 1 a "possible pair". The algorithm finds all the possible cube pairs on each level until it ends up with the set of all possible pairs of leaf cubes. Then the sufficient condition 2 (see below) is tested on all these pairs of leaf cubes to determine whether to accept or reject all the pairs of points inside them. At the end all the pairwise desirable cubes are combined through a matrix product.

Given a target polygon $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, a tolerance $\epsilon_T \geq 0$ and one edge (P_i, P_j), let $\mathcal{C}_i, \mathcal{C}_j$ be two nonempty cubes with size ℓ and the distance between their centers d, where $i, j \in [1, 2, \ldots, n], i \neq j$. Then we have the following theorems.

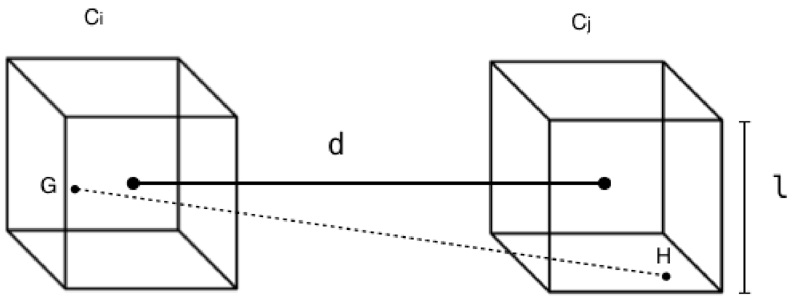

FIGURE 2.

Theorem 1. *If $d < P_iP_j - \epsilon_T - \sqrt{3}\,\ell$ or $d > P_iP_j + \epsilon_T + \sqrt{3}\,\ell$, then there are no pairs of points $(G, H) \in \mathcal{C}_i \times \mathcal{C}_j$ such that $|GH - P_iP_j| \leq \epsilon_T$.*

Proof. For any two points $G \in \mathcal{C}_i, H \in \mathcal{C}_j$ as shown in Figure 2, if $d < P_iP_j - \epsilon_T - \sqrt{3}\,\ell$, by the triangle inequality we have,

$$GH \leq d + \sqrt{3}\,\ell < P_iP_j - \epsilon_T - \sqrt{3}\,\ell + \sqrt{3}\,\ell = P_iP_j - \epsilon_T.$$

If $d > P_iP_j + \epsilon_T + \sqrt{3}\,\ell$, again by the triangle inequality,

$$GH \geq d - \sqrt{3}\,\ell > P_iP_j + \epsilon_T + \sqrt{3}\,\ell + \sqrt{3}\,\ell - \sqrt{3}\,\ell = P_iP_j + \epsilon_T.$$

□

Theorem 1 suggests a "necessary condition" for any two cubic regions on the same level of the trees to contain any desirable pairs of points. We are going to call it the "necessary condition 1" in the future to refer to the condition defined in Theorem 1. There are other necessary conditions based on the positions of all the points in the cubes, but in comparison this condition is a tighter condition and is more efficient for computation. On the other hand, we have the following "sufficient condition 2" for all pairs of points from two leaf cubes to be desirable.

Theorem 2. *If $P_iP_j - \epsilon_T + \sqrt{3}\,\ell \leq d \leq P_iP_j + \epsilon_T - \sqrt{3}\,\ell$, then all pairs of points $(G, H) \in \mathcal{C}_i \times \mathcal{C}_j$ satisfy $|GH - P_iP_j| \leq \epsilon_T$.*

Proof. As shown in Figure 1, for any points $G \in \mathcal{C}_i, H \in \mathcal{C}_j$, we have $d - \sqrt{3}\,\ell \leq GH \leq d + \sqrt{3}\,\ell$. If $P_iP_j - \epsilon_T + \sqrt{3}\,\ell \leq d \leq P_iP_j + \epsilon_T - \sqrt{3}\,\ell$. Substituting the tighter bound of d on each side of the inequality we have $P_iP_j - \epsilon_T \leq GH \leq P_iP_j + \epsilon_T$. □

Notice that the condition of Theorem 2 is only possible when $P_iP_j - \epsilon_T + \sqrt{3}\,\ell \leq P_iP_j + \epsilon_T - \sqrt{3}\,\ell$, or when $\ell \leq \epsilon_T/\sqrt{3}$. Since the leaf cubes of the octrees must have length $\ell_T \leq 2l_s$, we require $\ell_s \leq \epsilon_T/(2\sqrt{3})$.

Let t_i be the octree generated from manifold $\mathcal{A}_i$ for $i = 1, 2, \ldots, n$. Algorithm 1 gives the pseudo code of the adaptive geometric search algorithm. Figure 3 illustrates the algorithm.

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:  $h =$ depth of the octrees $t_i$ for $i = 1, 2, \ldots, n$
3:  **for** $i, j \in [1, 2, \ldots, n], i \neq j$ **do**
4:      $pairs$ = []
5:      $l^* = P_iP_j$
6:      $combos$ = [[] for $x$ in range($h + 1$)]
7:      $combos[0]$ = [$(t_i, t_j)$]
8:      **for** $k \in [1, 2, \ldots, h]$ **do**
9:          **for** $(b_0, b_1)$ in $combos[k]$ **do**
10:             $combos[k + 1] \mathrel{+}= Compare1(b_0, b_1, l^*, \epsilon_T)$
11:         **end for**
12:     **end for**
13:     **for** $(b_0, b_1)$ in $combos[h]$ **do**
14:         $pairs \mathrel{+}= Compare2(b_0, b_1, l^*, \epsilon_T)$
15:     **end for**
16:     Append all $(p, q) \in pairs$ as edges to the graph $G$
17: **end for**
18: Search $G$ for the desirable polygon

19: # Check the necessary condition 1
20: **function** COMPARE1($b_0, b_1, l^*, \epsilon_T$)
21:     **return** $[(c_i, c_j)$ for $(c_i, c_j)$ in $b_0.children \times b_1.children$ if
        $|(c_i.center, c_j.center) - l^*| \leq \epsilon_T + \sqrt{3}\, c_i.length]$
22: **end function**

23: # Check the sufficient condition 2
24: **function** COMPARE2($b_0, b_1, l^*, \epsilon_T$)
25:     **if** $|(b_0.center, b_1.center) - l^*| \leq \epsilon_T - \sqrt{3}\, b_0.length$ **then**
26:         **return** $[(b_0, b_1)]$
27:     **else**
28:         **return** []
29:     **end if**
30: **end function**
```

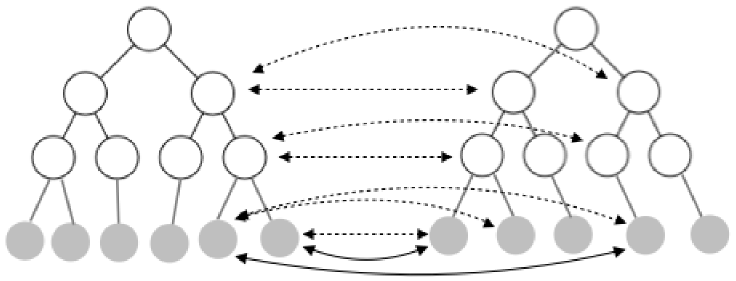

FIGURE 3. An illustration of the adaptive search between two octrees. Dotted lines point out the possible cube pairs on each level. Solid lines link the desirable leaf cube (gray nodes) pairs that pass the sufficient condition 2.

3. ANALYSIS OF THE ALGORITHM

The adaptive geometric search algorithm has three parts, building the octrees, adaptively searching every two octrees and the graph search. Let N be the number of sample points from each manifold. For convenience we build all octrees with the same initial cube length ℓ_0. The time complexity of building an octree with initial cube length ℓ_0 and minimum cube length ℓ_s is $\mathcal{O}(\log_2(\ell_0/\ell_s)N)$.

Next we compute the time complexity of the adaptive search between any two octrees (without losing generality) called t_1, t_2. Let the corresponding polygon edge length be l^*. Then we have the following results.

Theorem 3. *In terms of the initial cube length ℓ_0 and the minimum cube length ℓ_s, the time complexity of the adaptive search algorithm 1 is $\mathcal{O}\left((\ell_0/\ell_s)^6\right)$.*

Proof. Let d be the depth of the octrees t_1, t_2. Let $\psi_k(t)$ be the number of nodes on the k-th level in the octree t. Recall that ℓ_0 denotes the length of the root cubes of the octrees t_1, t_2. Since all the cubes have the minimum length ℓ_s, we have $\ell_0/2^d \geq \ell_s$, or $d \leq \log_2(\ell_0/\ell_s)$. Then the total number of computations $\mathcal{N}$ satisfies

$$\begin{aligned}
\mathcal{N} &= \mathcal{O}(\sum_{k=1}^{d} \psi_k(t_1)\psi_k(t_2)) \\
&= \mathcal{O}(\sum_{k=1}^{d} 8^k \cdot 8^k) \\
&= \mathcal{O}(64^d) \\
&= \mathcal{O}(64^{\log_2(\ell_0/\ell_s)}) \\
&= \mathcal{O}((\ell_0/\ell_s)^6).
\end{aligned}$$

□

Theorem 4. *If we set $\ell_s = \eta \frac{\epsilon_T}{4\sqrt{3}}$ for any $0 < \eta < 1$, then the adaptive geometric search algorithm 1 returns all the pairs of points whose distances are within the set $[\ell^* - (1-\eta)\epsilon_T, \ell^* + (1-\eta)\epsilon_T]$, and some but possibly not all the pairs of points whose distances are within the set $[\ell^* - \epsilon_T, \ell^* - (1-\eta)\epsilon_T) \cup (\ell^* + (1-\eta)\epsilon_T, \ell^* + \epsilon_T]$ with time complexity $\mathcal{O}\left((\frac{\ell_0}{\eta\, \epsilon_T})^6\right)$.*

Proof. Let ℓ_T be the length of the leaf cubes. By the definition of l_s, we have $\ell_T < 2l_s = \eta\frac{\epsilon_T}{2\sqrt{3}}$. Thus $\ell_T < \epsilon_T/\sqrt{3}$ and the sufficient condition 2 can be tested. If the sufficient condition 2 is rejected on a pair of cubes $\mathcal{C}_1, \mathcal{C}_2$, then the distance d between them satisfies $d > \ell^* + \epsilon_T - \sqrt{3}\,\ell_T$ or $d < \ell^* - \epsilon_T + \sqrt{3}\,\ell_T$. Let G, H be any two points such that $G \in \mathcal{C}_1, H \in \mathcal{C}_2$. By the triangle inequality, we have

$$GH \geq d - \sqrt{3}\,\ell_T > \ell^* + \epsilon_T - 2\sqrt{3}\,\ell_T > \ell^* + (1-\eta)\epsilon_T,$$

or

$$GH \leq d + \sqrt{3}\,\ell_T < \ell^* - \epsilon_T + 2\sqrt{3}\,\ell_T < \ell^* - (1-\eta)\epsilon_T.$$

Therefore, in rejecting all pairs of points in $\mathcal{C}_1 \times \mathcal{C}_2$ we may have rejected some pairs of points whose distances are within the set $[\ell^* - \epsilon_T, \ell^* - (1-\eta)\epsilon_T) \cup (\ell^* + (1-\eta)\epsilon_T, \ell^* + \epsilon_T]$. From Theorem 3 the time complexity of the algorithm is $\mathcal{O}((\ell_0/\ell_s)^6)$. Substituting in $\ell_s = \eta\frac{\epsilon_T}{4\sqrt{3}}$, we have the time complexity equal to $\mathcal{O}\left((\frac{\ell_0}{\eta\,\epsilon_T})^6\right)$. □

Now we consider the last part of the algorithm, the graph search. Let s_{ij} be the number of possible leaf cube pairs between octrees t_i, t_j for $i, j \in [1, 2, \ldots, n], i \neq j$. Then by induction the complexity of computing desirable n-tuple cubes is $\mathcal{O}(\prod_{i,j\in[1,2,\ldots,n], i\neq j} s_{ij})$. If we pick at most M points from each cube and pass them through a basic no-clashing, no-crossing quality check, then to produce desirable polygons we need to perform a further computation of complexity $\mathcal{O}(M^n)$.

Therefore if we view l_0 as a constant, the total time complexity of the algorithm is

$$\begin{aligned}
& \mathcal{O}\left(n\log_2(\frac{\ell_0}{\ell_s})N + n^2\left(\frac{\ell_0}{\eta\,\epsilon_T}\right)^6 + \left(\prod_{\substack{i,j\in[1,2,\ldots,n]\\ i\neq j}} s_{ij} + M^n \right)\right) \\
= \; & \mathcal{O}\left(n\log_2(\frac{4\sqrt{3}\,\ell_0}{\eta\,\epsilon_T})N + \frac{n^2}{(\eta\,\epsilon_T)^6} + \left(\prod_{\substack{i,j\in[1,2,\ldots,n]\\ i\neq j}} s_{ij} + M^n \right)\right) \\
= \; & \mathcal{O}\left(n\,(1-\log_2(\eta\,\epsilon_T))\,N + \frac{n^2}{(\eta\,\epsilon_T)^6} + \left(\prod_{\substack{i,j\in[1,2,\ldots,n]\\ i\neq j}} s_{ij} + M^n \right)\right).
\end{aligned}$$

In practice we usually search for a triangle or a 4-sided polygon as the target polygon, i.e. $n = 3$ or 4. When $n = 3$, depending on the parameters η, ϵ and N the computation time varies but all three terms in the complexity formula are typically of the same order. When there are large numbers of possible pairs s_i's and/or often when $n = 4$, the term $\mathcal{O}(\prod_{i,j\in[1,2,\ldots,n],i\neq j} s_{ij})$ in the last term of the complexity formula becomes the dominating term.

REFERENCES

P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42:67–90, 1995.

R. N. Zuckermann, E. J. Martin, et al. Discovery of nanomolar ligands for 7-transmembrane g-protein-coupled receptors from a diverse n-(substituted)glycine peptoid library. *J. Med. Chem.*, 37(17):2678–2685, 1994.