Adaptive Geometric Search for Protein Design

1. Introduction

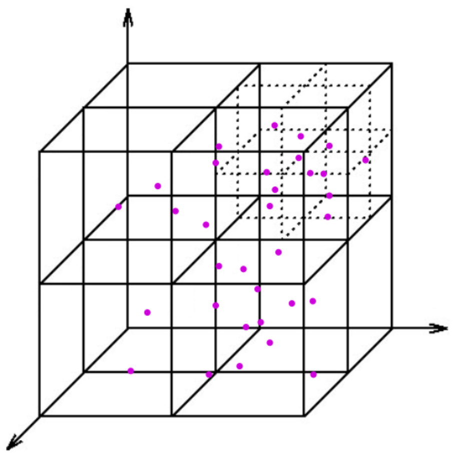
2. Method

2.1. Overview

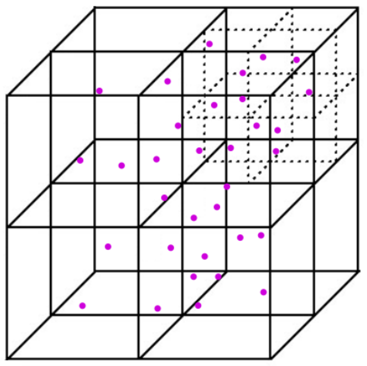
Our approach in the peptoid design is a two-step process. In the first step, we consider the most influential energies in the peptoid and conduct an efficient geometric search to eliminate all the impossible designs. In the second step, designs that passed the quick initial screening will be further 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.

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 (Figure 1) by searching adaptively into the cubic regions that pass the necessary condition 6 (see below). We call a pair of cubes that pass the necessary condition 6 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 7 (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.



(a) Side chain i



(b) Side chain j

Figure 1. Purple points are points sampled from manifolds that correspond to two side chains i and j. Part of the octree nodes (cubes) are illustrated.

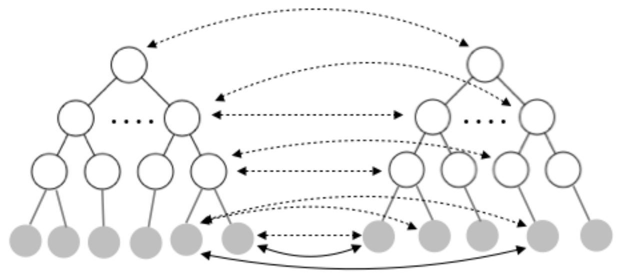


Figure 2. An illustration of the adaptive search between two octrees. Dotted lines point out the possible cube pairs on each level. Solid lines link the desirable leaf cube (gray nodes) pairs that pass the sufficient condition 7.

2.2. Theory : Necessary and Sufficient Conditions

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 nonempty cubes with size ℓ and the distance between their centers d, where $i, j \in [1, 2, \dots, n], i \neq j$. Then we have the following theorems.

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. See Appendix A. □

Theorem 6 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 6" in the future to refer to the condition defined in Theorem 6. 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 7" 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. See Appendix A. □

Notice that the condition of Theorem 7 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 $i = 1, 2, \dots, n$. Algorithm 1 gives the pseudo code of the adaptive geometric search algorithm. Figure 5 illustrates the algorithm.

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

```
1:  $trees$ = $[t_1, t_2, \dots, t_n]$
2:  $h$ = depth of the octrees $t_i$ for $i = 1, 2, \dots, n$
3:  for $i, j \in [1, 2, \dots, n], i \neq j$ 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 $k \in [1, 2, \dots, h]$ do
9:          for $(b_0, b_1)$ in $combos[k]$ do
10:             $combos[k + 1]$ += $Compare1(b_0, b_1, l^*, \epsilon_T)$
11:         end for
12:     end for
13:     for $(b_0, b_1)$ in $combos[h]$ do
14:         $pairs$ += $Compare2(b_0, b_1, l^*, \epsilon_T)$
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
```

2.3. Algorithm Complexity

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. *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 $[\ell^* - \epsilon_T, \ell^* - (1-\eta)\epsilon_T) \cup (\ell^* + (1-\eta)\epsilon_T, \ell^* + \epsilon_T]$.*

Proof. See Appendix A. □

Lemma 4. *Set $\ell_s = \eta\frac{\epsilon_T}{4\sqrt{3}}$ for some $0 < \eta < 1$. Then for any cube $\mathcal{C}_1$ in an octree t_1, there are at most $\frac{4\pi}{3}(3\sqrt{3} + 2\frac{4\sqrt{3}}{\eta})\left(3(\frac{\ell^*}{\ell_s})^2 + (\frac{3\sqrt{3}}{2} + \frac{4\sqrt{3}}{\eta})^2\right)$ cubes $\mathcal{C}_2$ on the same level from another octree t_2 such that $(\mathcal{C}_1, \mathcal{C}_2)$ are possible pairs, that is, they satisfy the necessary condition 6.*

Proof. See Appendix A. □

Theorem 5. *Recall that ℓ_0 denotes the initial cube length and the minimum cube length $\ell_s = \eta\frac{\epsilon_T}{4\sqrt{3}}$. Let n_m be defined as in Lemma 9. Then the time complexity of the adaptive search part of Algorithm 1 is $\mathcal{O}\left(\frac{1}{\eta^6\epsilon_T^5}\right)$.*

Proof. See Appendix A. □

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 $i, j \in [1, 2, \ldots, n], i < j$. We view the leaf cubes as vertices and possible pairs of them as undirected edges in the graph. If we want to produce all the desirable n-tuple cubes, then by induction it's easy to see that the upper bound on the time complexity is $\mathcal{O}(\prod_{1\le i<j\le n} s_{ij})$.

In practice we can do much better. Consider building a directed graph by giving directions to the edges to form a n-cycle of groups of cubes from $t_1, t_2, \ldots, t_n$. Finding strongly connected components in this directed graph first would in most cases greatly reduce the search space with only a linear cost $\mathcal{O}(\sum_{1\le i<j\le n} s_{ij})$.

In summary we state the total time complexity of the algorithm.

$$\begin{aligned} & \mathcal{O}\left(n\log_2(\frac{\ell_0}{\ell_s})N + \frac{n^2}{\eta^6\epsilon_T^5} + \prod_{1\le i<j\le n} s_{ij}\right) \\ = \ & \mathcal{O}\left(n\log_2(\frac{4\sqrt{3}\,\ell_0}{\eta\,\epsilon_T})N + \frac{n^2}{\eta^6\epsilon_T^5} + \prod_{1\le i<j\le n} s_{ij}\right) \\ = \ & \mathcal{O}\left(n\log_2\left(\frac{\ell_0}{\eta\epsilon_T}\right)N + \frac{n^2}{\eta^6\epsilon_T^5} + \prod_{1\le i<j\le n} s_{ij}\right). \end{aligned}$$

(1)

(2)

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 (4.1) 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 $\mathcal{C}(S)$ in the last term of the complexity formula (4.1) becomes the dominating term. However the number of results s_{ij}'s can be significantly reduced when we take optimal dihedral angles instead of uniform sampling from $[0, 2\pi]$.

3. Experiments

3.1. Scaffold Matcher

In immunology, an antigen is any structural substance that serves as a target for the receptors of an adaptive immune response and the epitope is the specific part of the antigen that an antibody binds to. Because this is important to drug development, we want to design protein structures that bind to the epitopes of antigens.

In designing epitope-binding proteins, the algorithm can be used to select the best scaffolds or the substitute backbone structures given the hot-spot residues, i.e. the residues that readily bind to the target epitope. In designing a binding with the protein "mdm2", we want to match an OOP type of backbone to the hot-spot residues, leu, phe, trp stubs, that are binding to "mdm2" in Figure 3.

3.1.1. Methods

There are two parts of the algorithm. In step 1, we search through all possible backbones for a matching triangle to the target triangle. In step 2, for every match result from step 1 the connecting atom's bond angles are checked against the optimal bond angle. If a match passes step 2, it's returned as a final result. Otherwise we continue the iteration in step 1.

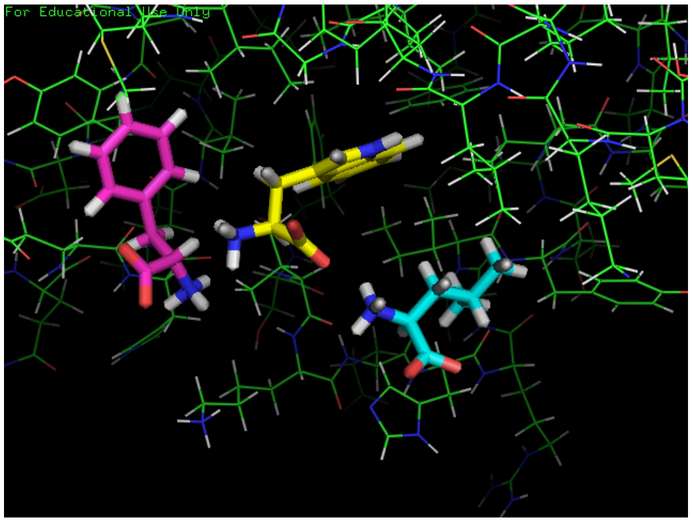
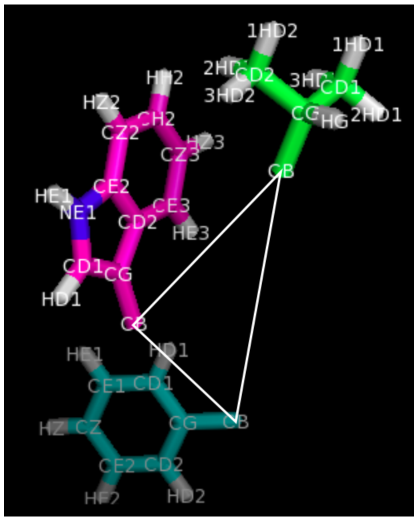
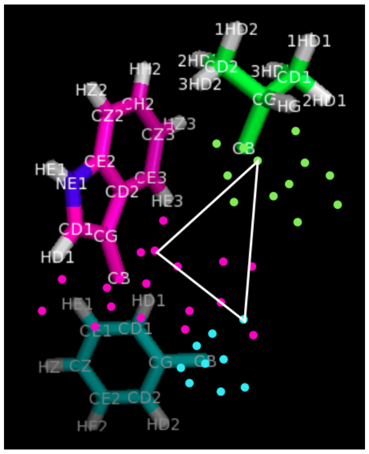


Figure 3. Three hot-spot residues are leu(cyan), phe(magenta), trp(yellow) stubs. The background protein is "mdm2", the protein the three hot-spot residues are binding with.

The target triangle is made up of $C\beta$'s of the hot spot residues, illustrated in Figure 7(a). The algorithm simply searches through the possible take-off position combinations, four triangles in this example (Figure 7(c)), from every backbone for a match in shape within the error bound. Notice that in this case all $C\beta$'s are fixed due to the short lengths of hot spot residues. With longer hot spot residues, there will be a manifold of all the possible $C\beta$'s for each hot spot residue (see Figure 7(b)). For every possible take-off position combination as the target shape, adaptive geometric search can be used to find all the matches.

Once we have the matching shapes, we calculate the corresponding matrices R's of rotation and translation such that after applying these tranformations R's backbones are connected onto the hotspot residues at atoms $C\beta$'s. Finally we just check if the bond angles at the connecting atoms are within some error bound to the optimal bond angles (see Figure 5).


(a) fixed target triangle


(b) possible $C\beta$ sites

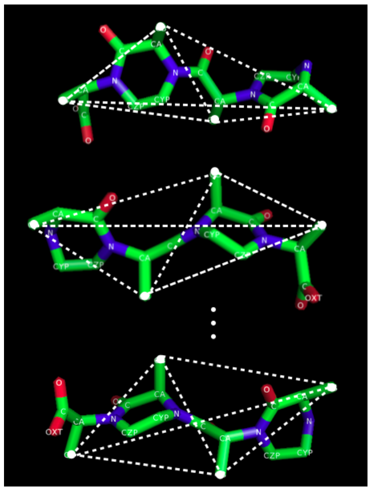

(c) 17 possible backbone matches

Figure 4. (a) the simple case: the $C\beta$'s of the hotspot residues are fixed. (b) the general case: there are multiple sites for the $C\beta$'s of each hotspot residue. (c) the list of 17 candidate backbones: white dots denote the $C\beta$'s. Each backbone has four $C\beta$'s and thus four possibly matching triangles.

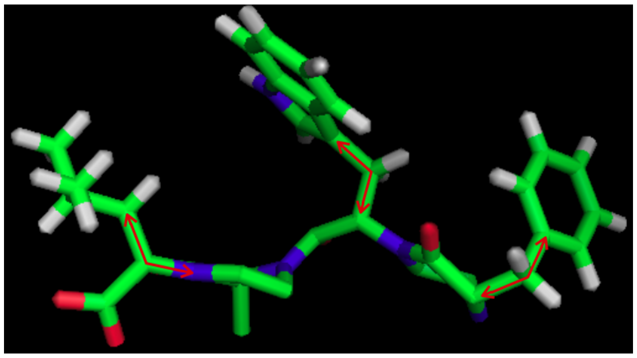


Figure 5. The bond angles at the connecting atoms between hotspot residues and the backbone are highlighted in red.

```
**Algorithm 2** Scaffold Match $(\{A_1, A_2, \ldots, A_n\}, \{P_1, P_2, \ldots, P_m\}, \delta, \delta_A)$
      $results = []$
  2:  **for** $i = 1, \ldots, m$ **do**
          $\{\tilde{P}_1, \tilde{P}_2, \ldots, \tilde{P}_k\} = AdaptiveGeometricSearch(\{A_1, A_2, \ldots, A_n\}, P_i, \delta)$
  4:      **for** $j = 1, \ldots, k$ **do**
              $R_j = CalculateTranformation(\tilde{P}_j, P_i)$ # calculates the tranformation matrix from $P_i$ to $\tilde{P}_j$
  6:          **if** $CheckAngle(R_j B_{P_i}, S_1, S_2, S_3)$ **then**
                  $results = results + [R_j B_{P_i}]$
  8:          **end if**
          **end for**
 10:  **end for**
```

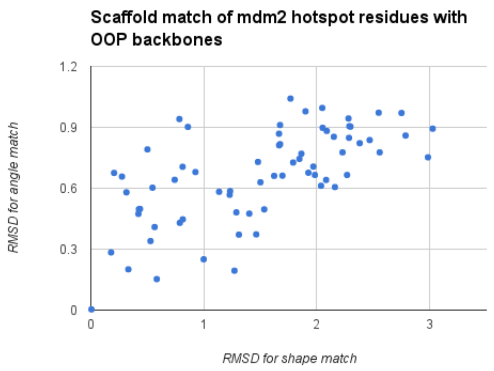
3.1.2. Algorithm Analysis

Let A_i be the manifold of possible positions of the connecting atom on the i-th hotspot residue. For example, in Figure 7(b) points in colors are sampled from manifolds A_1, A_2 and A_3 respectively. Let P_j be the j-th polygon of the backbone take-off position combination and for example, there are 4×17 of them in Figure 7(c). Let B_P denote the atoms positions matrix corresponding to the backbone where the target polygon P comes from. Let S_i denote the atoms positions matrix for the i-th residue. Let δ be the distance error bound and δ_A be the angle error bound. Then we describe in peudocode Algorithm 2.

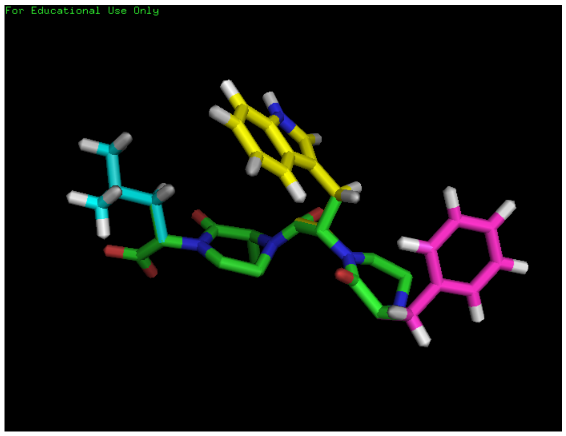
Let $\mathcal{C}$ denote the time complexity for adaptive geometric search. Recall in Algorithm 2 that m is the number of target polygons from backbone take-off site combinations. Then the time complexity of the scaffold matching algorithm is $\mathcal{O}(\mathcal{C}m)$.

3.1.3. Result

In the search process we scored all the possible matches by the RMSD (root mean square deviation) values for both shape match and angle match in Figure 6(a). Our algorithm picked the candidate at the origin which has the lowest RMSDs. In Figure 6(b) we show this best design for the OOP backbone of the hot-spot residues[1]. A sanity test in Rosetta shows its energy score is a low 4.67 with a potential energy score 4.59 after further minimization, which means this protein is likely very stable in practice. The algorithm run time is $0.02 \sim 0.12$ seconds whereas running the same design and producing the same results in Rosetta takes ~ 18 minutes using the scripts of Dr. Kevin Drew.



(a) match results



(b) best match

Figure 6. (a) RMSD (root mean square deviation) graph for all possible OOP backbone matches with the hot-spot residues. The candidate at the origin is a clear winner having close-to-zero RMSDs for both shape and angle matches. (b) Illustration of the best OOP backbone that matches with the hot-spot residues.

[1]All figures of protein structures in this thesis are generated with PyMOL Molecular Graphics System, Version 1.7.4 Schrdinger, LLC.

A. Appendix

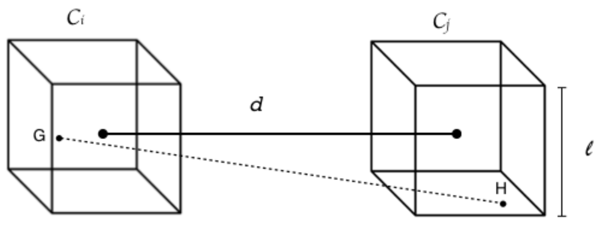


Figure 7. Cubes $\mathcal{C}_i, \mathcal{C}_j$ of size ℓ that are d distance apart.

Theorem 6. *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 7, 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 7. *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 7, 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 $P_iP_j - \epsilon_T \leq GH \leq P_iP_j + \epsilon_T$. □

Theorem 8. *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 $[\ell^* - \epsilon_T, \ell^* - (1-\eta)\epsilon_T) \cup (\ell^* + (1-\eta)\epsilon_T, \ell^* + \epsilon_T]$.*

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 7 can be tested. If the sufficient condition 7 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 $[\ell^* - \epsilon_T, \ell^* - (1-\eta)\epsilon_T) \cup (\ell^* + (1-\eta)\epsilon_T, \ell^* + \epsilon_T]$. □

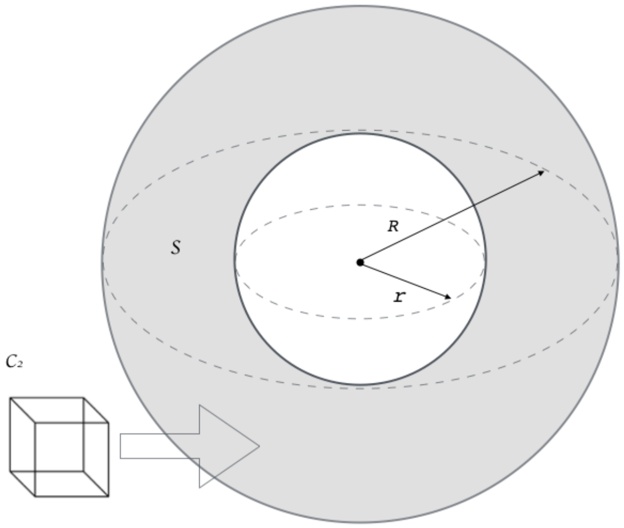


Figure 8. An illustration for Lemma 9. The outer radius $R = \ell^* + \sqrt{3}\,\ell + \epsilon_T + \frac{\sqrt{3}}{2}\,\ell$, and the inner radius $r = \ell^* - \sqrt{3}\,\ell - \epsilon_T - \frac{\sqrt{3}}{2}\,\ell$. Let $\mathcal{S}$ denote the spherical shell (in shade). How many cubes $\mathcal{C}_2$ can fit into $\mathcal{S}$?

Lemma 9. *Set $\ell_s = \eta \frac{\epsilon_T}{4\sqrt{3}}$ for some $0 < \eta < 1$. Then for any cube $\mathcal{C}_1$ in an octree t_1, there are at most $\frac{4\pi}{3}(3\sqrt{3} + 2\frac{4\sqrt{3}}{\eta})\left(3(\frac{\ell^*}{\ell_s})^2 + (\frac{3\sqrt{3}}{2} + \frac{4\sqrt{3}}{\eta})^2\right)$ cubes $\mathcal{C}_2$ on the same level from another octree t_2 such that $(\mathcal{C}_1, \mathcal{C}_2)$ are possible pairs, that is, they satisfy the necessary condition 6.*

Proof. For any cube $\mathcal{C}_1$ in t_1, let ℓ be the length of $\mathcal{C}_1$. For any possible cube $\mathcal{C}_2$ on the same level from t_2, by the necessary condition 6 the distance between them d must satisfy that $\ell^* - \sqrt{3}\,\ell - \epsilon_T \leq d \leq \ell^* + \sqrt{3}\,\ell + \epsilon_T$. Thus all possible cubes $\mathcal{C}_2$ must be contained in the spherical shell $\mathcal{S}$ of inner radius $\ell^* - \sqrt{3}\,\ell - \epsilon_T - \frac{\sqrt{3}}{2}\,\ell$ and outer radius $\ell^* + \sqrt{3}\,\ell + \epsilon_T + \frac{\sqrt{3}}{2}\,\ell$ (see Figure 8). Since there are no overlapping cubes on the same level in t_2, the maximum number of the possible cubes n_m satisfies

$$n_m \leq \frac{Vol(\mathcal{S})}{Vol(\mathcal{C}_2)}$$
$$\leq \frac{4\pi}{3\ell^3}(\ell^* + \sqrt{3}\,\ell + \epsilon_T + \frac{\sqrt{3}}{2}\,\ell)^3$$

$$
\begin{aligned}
&- \frac{4\pi}{3\ell^3}(\ell^* - \sqrt{3}\,\ell - \epsilon_T - \frac{\sqrt{3}}{2}\,\ell)^3 \\
&\le \frac{4\pi}{3\ell^3}(3\sqrt{3}\,\ell + 2\epsilon_T)\left(3(\ell^*)^2 + (\frac{3\sqrt{3}}{2}\,\ell + \epsilon_T)^2\right) \\
&\le \frac{4\pi}{3}(3\sqrt{3} + 2\frac{\epsilon_T}{\ell})\left(3(\frac{\ell^*}{\ell})^2 + (\frac{3\sqrt{3}}{2} + \frac{\epsilon_T}{\ell})^2\right) \\
&\le \frac{4\pi}{3}(3\sqrt{3} + 2\frac{\epsilon_T}{\ell_s})\left(3(\frac{\ell^*}{\ell_s})^2 + (\frac{3\sqrt{3}}{2} + \frac{\epsilon_T}{\ell_s})^2\right) \\
&= \frac{4\pi}{3}(3\sqrt{3} + 2\frac{4\sqrt{3}}{\eta})\left(3(\frac{\ell^*}{\ell_s})^2 + (\frac{3\sqrt{3}}{2} + \frac{4\sqrt{3}}{\eta})^2\right).
\end{aligned}
$$

□

Theorem 10. *Recall that ℓ_0 denotes the initial cube length and the minimum cube length $\ell_s = \eta\frac{\epsilon_T}{4\sqrt{3}}$. Let n_m be defined as in Lemma 9. Then the time complexity of the adaptive search part of Algorithm 1 is $\mathcal{O}\left(\frac{1}{\eta^6\epsilon_T^5}\right)$.*

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_1. 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 $\ell_0/2^d \ge \ell_s$, or $d \le \log_2(\ell_0/\ell_s)$. Using Lemma 9 the total number of computations $\mathcal{N}$ satisfies

$$
\begin{aligned}
\mathcal{N} &= \mathcal{O}(\psi_1(t_1)\psi_1(t_2) + n_m \times 64\sum_{k=1}^{d-1}\psi_k(t_1)) \\
&= \mathcal{O}(n_m\sum_{k=1}^{d-1}\psi_k(t_1)) = \mathcal{O}(n_m\sum_{k=1}^{d-1}8^k) \\
&= \mathcal{O}(n_m 8^d) = \mathcal{O}(n_m 8^{\log_2(l_0/l_s)}) = \mathcal{O}(n_m(l_0/l_s)^3) \\
&= \mathcal{O}\left[\frac{4\pi}{3}(3\sqrt{3} + 2\frac{4\sqrt{3}}{\eta})\left(3(\frac{\ell^*}{\ell_s})^2 + (\frac{3\sqrt{3}}{2} + \frac{4\sqrt{3}}{\eta})^2\right)(\frac{l_0}{l_s})^3\right] \\
&= \mathcal{O}\left(\frac{1}{\eta\ell_s^5} + \frac{1}{(\eta\ell_s)^3}\right) \\
&= \mathcal{O}\left(\frac{1}{\eta^6\epsilon_T^5} + \frac{1}{\eta^6\epsilon_T^3}\right) \\
&= \mathcal{O}\left(\frac{1}{\eta^6\epsilon_T^5}\right).
\end{aligned}
$$

□