Molecular channel detection in macromolecules

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

by

Rakesh Malviya

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

JULY 2013

Acknowledgements

I wish to convey my most sincere gratitude to Dr. Vijay Natarajan for his unparalleled support in helping me complete this project. I thank Talha Bin Masood for his guidance and help in this work.

I am indebted with gratitude to my parents, my sister and brother for being a constant source of inspiration during the entire tenure of my studies. I also wish to extend my thanks to my friends for providing me moral as well as intellectual support.

Thank you all,

Rakesh Malviya
July, 2013.

Abstract

Study of molecular interaction and investigation of binding sites can solve many open questions in biochemistry and molecular biology. Molecular channels are paths leading to the binding sites or paths traversed by ions and solvent molecules through the macromolecule. Behavior of protein molecule depends on these channels. We have proposed a new algorithm which uses regular triangulation of atoms to find molecular channels.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Molecular interactions play a very important role in physiology of organisms. Molecular interactions are also studied pharmacologists to develop safe and effective medication. Channels, cavities, voids, and pockets are some of the important features of macromolecules that are studied by researchers to understand relationships between the structures and activities of macromolecules. For example, finding whether a molecule (say A) can reach another molecule's (say P) site of interaction through its (P's) molecular channels, key features of membrane channels, the geometry of ribosomal polypeptide exit channels, the architecture of biomolecular complexes, etc.

Several different methods for finding and visualizing molecular channels have been proposed previously. Some of them use geometrical models like Voronoi diagrams[16][14] and Delaunay triangulations while others have used grid based approach[17]. Several new techniques of visualization of molecular channels has been proposed[14] which include path illumination, surface clipping and more.

In this work, we have applied the notion of regular triangulation to find and represent molecular channels (described in Chapter 5). Molecular channels are described and defined in detail in chapter 2. We have also investigated different ways of visualizing molecular channels, which are implemented in collaboration with other students.

Chapter 2

PROBLEM DEFINITION

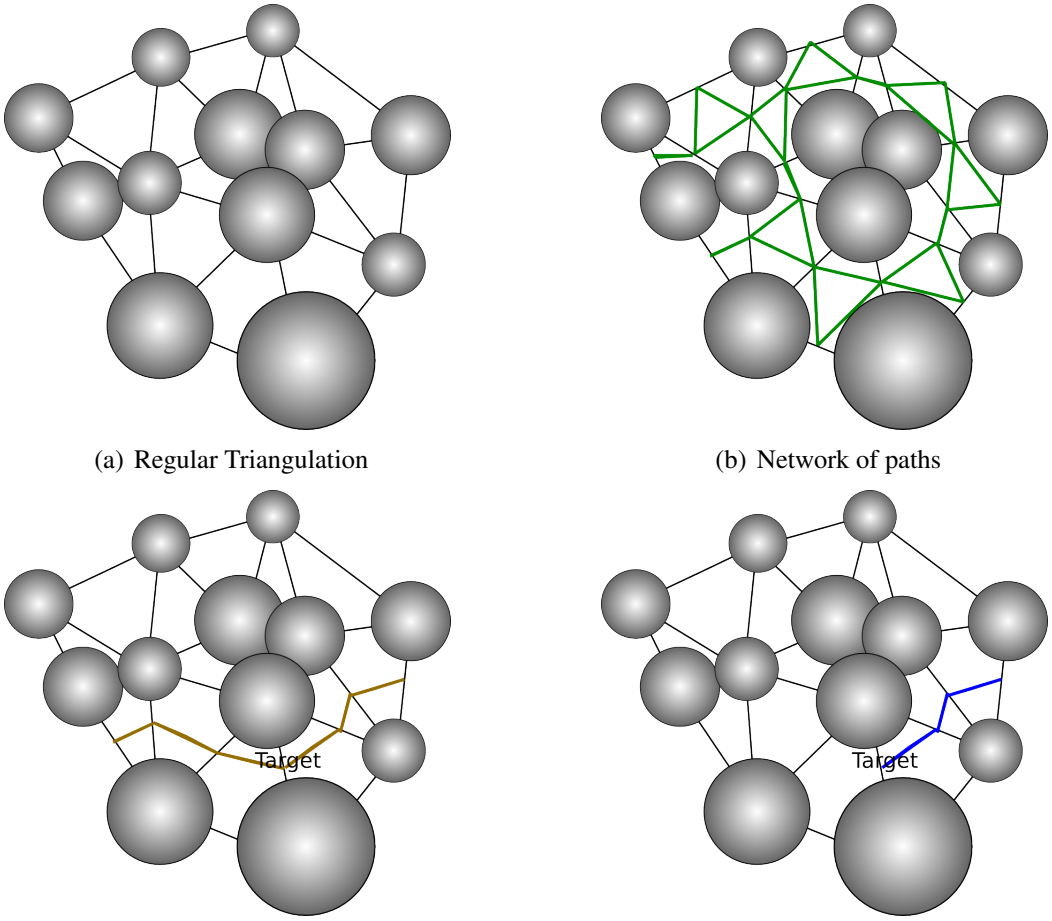
A given macromolecule may have a large number of possible paths which can be accessed by a given solvent molecule. We define all these possible paths as **Network of paths** through the molecule. Figure 2.1(a) shows regular triangulation of a macromolecule and corresponding network of paths in Figure 2.1(b).

A **Molecular Channel(s)** is defined as the set of particular paths from the network of paths which are always accessed by a given solvent molecule in a given macromolecule. Therefore a different solvent molecules may have different molecular channels in a given macromolecule, because of following:

1. Size of solvent molecule.
2. Hydrophobic properties of solvent molecule or atoms along a molecular channel.
3. Electrostatic interaction of solvent molecule and atoms along a molecular channel.

Our aim is to develop a tool that uses regular triangulation to find molecular channels in a macromolecule. Our algorithm finds shortest and widest path(s) from the network of paths, that can be taken by a given solvent. We call such paths, **Significant paths**. We find significant path from network of path by giving cost to each path based on the following formula,

$$cost = length/width^2$$



(a) Regular Triangulation

(b) Network of paths

(c) Significant paths

(d) Molecular channel

Figure 2.1: (a) Regular triangulation of Atoms. (b) Possible network of paths(in green) through Atoms. (c) Significant paths(in brown) to Target location from network of paths in Figure (b). (d) Molecular channel(in blue) identified by the user from significant paths in Figure (c).

Hence significant path is path with minimum cost. Significant paths may contain molecular channels. The intuition is that the ions or solvents choose shortest and widest paths.

The tool we developed helps the user to visualize individual paths from significant paths and their properties such as electrostatic potential. This helps the user to identify Molecular channels from significant paths.

Chapter 3

RELATED WORK

Many tools have been developed which try to find cavities and pockets in molecules. CASTp[5] uses geometrical models like alpha complexes and regular triangulation[13][6] to find cavities on protein surface. POCKET[12] and SURFNET[11] try to find cavities by filling them with spheres of given probe radius, and then creating their surfaces and volumes. In POCKET spheres are placed on a fixed regular lattice and adjusted in size until they touch the nearest atom. SURFNET considers relevant pairs of atoms and places a sphere midway between them and reduces its size if it conflicts with any neighboring atom.

The problem of molecular channel detection is studied by many research groups by employing different approaches. Of these approaches, geometry based approaches are significant from computational science perspective. Here we shall describe some of the most relevant ones.

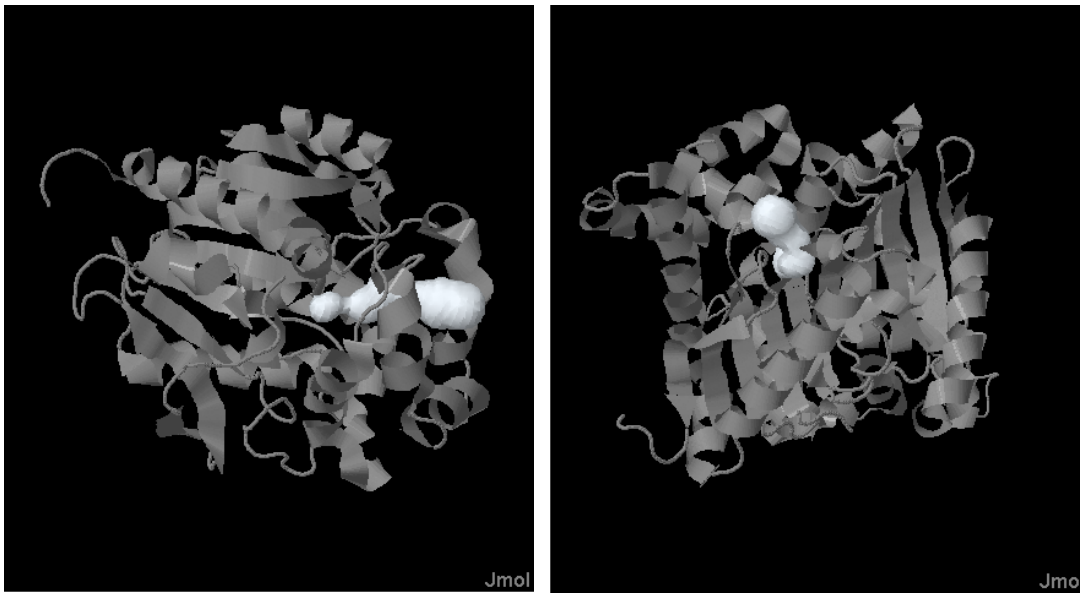
The following group of tools create surfaces and volumes of channels using probe spheres. SURFNET[11], HOLLOW[8] and 3V[20] uses probe sphere of variable radius, and VOIDOO[9] uses water molecule as probe. HOLLOW and 3V uses grid based approach to find surfaces and volumes of channels.

The following approaches try to find molecular channels in proteins. Most of them require user-defined starting position. One such tool is HOLE[19] which uses a Monte Carlo simulated annealing procedure to find the best route for a sphere with variable radius to squeeze through the channel. CAVER[17] uses grid based path search method, where each grid has weight

proportional to the distance from the molecule, and Dijkstra's shortest path algorithm is used to detect path. MOLE[16] uses Voronoi diagram which is provided as graph to Dijkstra's shortest path algorithm, where edge weights in graph are proportional to length of edge and inversely proportional to square distance to the molecule. Lindow et al.[14] used Voronoi diagram of spheres to detect molecular channels in similar fashion as MOLE where edge weight was equal to edge length. They also developed a filtering pipeline to extract significant paths.

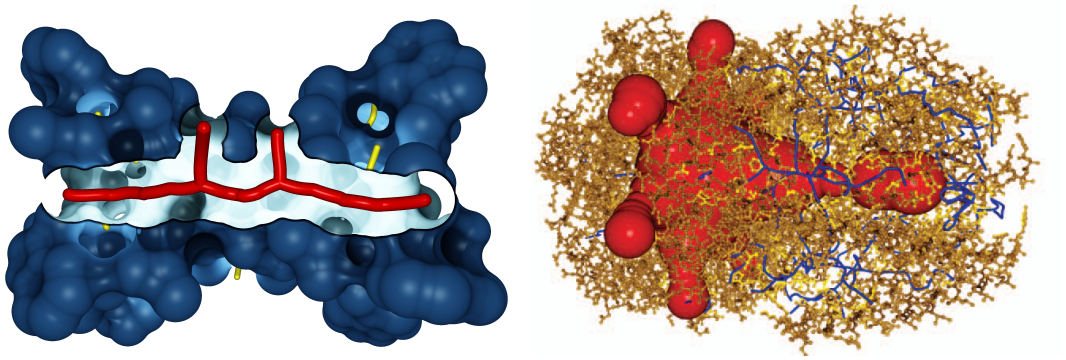
3.1 Known Molecular Channels

Figure 3.1 shows some known molecular channel computed by MOLE algorithm. Figure 3.2 shows some known molecular channel computed by algorithm described by Lindow et al.



(a) Haloalkane dehalogenase (PDB: 1CQW) (b) Carbonic Anhydrase (PDB:3EYX)

Figure 3.1: Known channels computed by MOLE algorithm. Figures from official website of MOLE[16]. (a) Haloalkane dehalogenase (PDB: 1CQW). (b) Carbonic Anhydrase (PDB: 3EYX).



(a) Gramicidin (PDB: 1GRM) (b) Mechanosensitive channel (PDB: 2OAU)

Figure 3.2: Known channels computed by algorithm described by Lindow et al. Figures from paper by Lindow et al.[14]. (a) Gramicidin (PDB: 1GRM). (b) Mechanosensitive channel (PDB: 2OAU).

Chapter 4

BACKGROUND

Following section discusses definition of some important geometrical tools used by researchers to find molecular channels and cavities in macromolecules. For this report let us define S to be a set of points in $\mathbb{R}^d$.

4.1 Simplices and Simplicial complexes

In geometry, a simplex is generalization of notion of a triangle or tetrahedron to arbitrary dimensions.

As defined in Munkres[15] a k-simplex Δ^T is convex hull of T, where T is affinely independent set of points such that

$$T \subset S \text{ and } \mid T \mid = k+1 \leq d+1.$$

Hence vertex is known as 0-simplex, edge as 1-simplex, triangle as 2-simplex and tetrahedron as 3-simplex. A face of Δ^T is the convex hull of a non-empty subset of T. A face is proper if subset is proper. We denote $\tau \leq \Delta^T$ if τ is a face and $\tau < \Delta^T$ if it is proper face.

A simplicial complex is a finite collection of simplices C such that $\Delta^T \in C$ and $\tau \leq \Delta^T$ implies $\tau \in C$, and $\Delta_1^T, \Delta_2^T \in C$ implies $\Delta_1^T \bigcap \Delta_2^T$ is either empty or face of both.

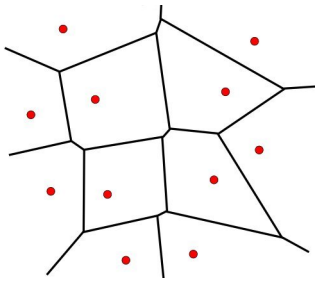


Figure 4.1: Voronoi diagram of input points(in red)

4.2 Voronoi diagram and Delaunay triangulation

Delaunay triangulations are simplicial complexes which are used in many geometrical methods described in previous work. Voronoi diagrams are dual of Delaunay triangulations, see Aurenhammer et al.[1] for detail discussion.

Voronoi cell of a point $p \in S$ is the set of points in $\mathbb{R}$ for which p is closer than any other point $q \in S$, i.e.

$$V_p = \{x \in \mathbb{R}^d | \ \|x - p\| \ \leq \|x - q\| \ , \forall q \in S\}.$$

Since boundary of V_p is formed by half planes between p and $q \in S$, V_p is convex polytope in $\mathbb{R}^d$. These polytopes meet each other at their faces forming the Voronoi diagram as shown in Figure 4.1.

Weighted Voronoi cell or power cell of point $q \in S$ is the set of points in $\mathbb{R}^d$ for which p is closest weighted point, i.e

$$V_p = \{x \in \mathbb{R}^d | \ d(p, x) \leq d(q, x), \forall q \in S\},$$

where $d(p, x) = \|x - p\|^2 - w_p$ and w_p is weight of point p.

The Delaunay triangulation, Del S, is dual to the Voronoi diagram. Specifically, whenever two Voronoi cells share a common side then the edge connecting the two corresponding points belongs to the Delaunay triangulation, and whenever three Voronoi cells share a common corner the triangle spanned by the three corresponding points belongs to the Delaunay triangulation. These interactions are also captured by the nerve of the Voronoi diagram, which is collection of subsets whose cells have non-empty common intersection, i.e.

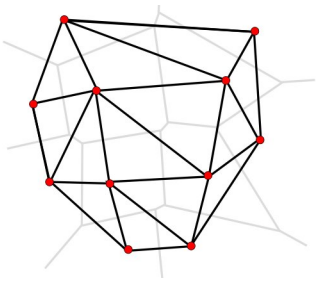


Figure 4.2: Delaunay triangulation of above Voronoi diagram

$$Nrv(VorS) = \{X \subseteq VorS \mid \bigcap X \neq \varnothing\}.$$

The Delaunay triangulation is geometric realization of nerve. The weighted case can similarly be constructed and is called a weighted Delaunay triangulation or regular triangulation.

4.3 Alpha Shapes and Alpha Complexes

In this section we will review alpha shapes and alpha complexes and their weighted counterparts. The following paragraph, quoted from Edelsbrunner et al.[7], explains alpha-hulls and alpha-shapes very intuitively.

> "Think of $\mathbb{R}^3$ filled with Styrofoam and the points of S made of more solid material, such as rock. Now imagine a spherical eraser with radius alpha. It is omnipresent in the sense that it carves out Styrofoam at all positions where it does not enclose any of the sprinkled rocks, that is, points of S. The resulting object will be called the alpha-hull. To make things more feasible we straighten the surface of the object by substituting straight edges for the circular ones and triangles for the spherical caps. The obtained object is the alpha-shape of S."

Edelsbrunner et al.[**?**] discusses alpha shapes as follows. The eraser intuition discussed above has a equivalent view that arises when we consider union of disks centered at a given point. Let p_1, p_n be the point in S, $B_i(\alpha)$ be closed disk with center p_i and radius $\alpha > 0$, and let $U(\alpha) = \bigcup_{i=1}^{n} B_i(\alpha)$ be the union of disks. If $x \in U(\alpha)$ belongs to Voronoi cell V_i, it also belongs to $B_i(\alpha)$. This implies $V_i \bigcap U(\alpha)$ is equal to $V_i \bigcap B_i(\alpha)$. Let $R_i(\alpha) = V_i \bigcap B_i(\alpha)$,

than realization of nerve of $R_i(\alpha)$ gives alpha complex C_α. Alpha-shape is given by union of all simplices in alpha complex.

Now we generalize alpha complexes to weighted case. Given $\alpha \in \mathbb{R}$, let $B_i(\alpha)$ be closed disk with center p_i and radius $(\alpha^2 + w_i)^{1/2}$. If $\alpha^2 + w_i < 0$ than root is imaginary and $B_i(\alpha)$ is empty. Similar to unweighted case consider intersection of union of disks and corresponding power cells for given α, and obtain the weighted alpha complex using notion of nerve.

There are only finitely many alpha complexes C_i, which when ordered by inclusion of simplices give rise to the notion of filtration of Delaunay triangulation, i.e.

$$\varnothing = C_0 \subset C_1 \subset C_2 ..... \subset C_n = Del\ S$$

Chapter 5

MOLECULAR CHANNELS IN MACROMOLECULES

5.1 Motivation

Previous approaches have following shortcomings:

1. CAVER[17] uses grid based path search method so user has to choose between accuracy and performance.

2. MOLE[16] uses Voronoi diagram of atoms(taken as points) so they don't take radius of atoms in consideration because of which they cannot accurately find significant paths for a given solvent size.

3. Lindow et al.[14] uses Voronoi diagram of spheres to find significant paths. Voronoi diagram of sphere does not have a dual triangulation.

Regular triangulation and its dual weighted Voronoi diagram has been used in some popular visual analysis tools like PyMol[4] for computing pockets, surface area and volume in proteins. This motivated us to design an algorithm to find molecular channels using regular triangulation.

5.2 Intuition

The intuition is that a given solvent molecule can only pass through any triangle in the regular triangulation in Figure 2.1(a) by entering via an edge and exiting through another edge of that triangle. So, the possible network of paths(path graph) through this regular triangulation is as shown in Figure 2.1(b).

Similarly in 3D a given solvent molecule can only enter a tetrahedron in a regular triangulation through a triangle and exit through a second triangle of that tetrahedron.

We use this idea to develop our algorithm and compute network of paths through a given macromolecule, as described in Section 5.3 below. To find significant paths in a given macromolecule from the network of paths we use the same idea as that was used in MOLE[16], as described in Section 5.4 below.

5.3 Algorithm to find network of paths through a given macromolecule

Input: Regular triangulation of atoms in a given macromolecule.
Output: Path Graph, $G^{path} = (V^{path}, E^{path})$ as a network of paths through the macromolecule.

Our algorithm has following steps:

1. Create Path Graph, $G^{path} = (V^{path}, E^{path})$.
2. Process Path Graph. Here the goal is to find maximum size of solvent allowed by each edge, $e^{path} \in E^{path}$ and length of that edge.

In the following sections we will describe the individual steps of our algorithm in more detail.

5.3.1 Create Path Graph

Given a two-dimensional regular triangulation as shown in Figure 2.1(a). We can create a path graph, $G^{path} = (V^{path}, E^{path})$ as shown in Figure 2.1(b). Each vertex in path graph, $v^{path} \in V^{path}$ corresponds to each edge in the regular triangulation. Create an edge, $e^{path} \in E^{path}$ between any two vertices in path graph if there corresponding edges in the regular triangulation belongs to same triangle in the regular triangulation.

Similarly given a three-dimensional regular triangulation of atoms in a macromolecule $Del_W(S)$, we can create path graph, $G^{path} = (V^{path}, E^{path})$ as follows. Each vertex in path graph, $v_i^{path} \in V^{path}$ corresponds to each triangle, t_i^{del} in the regular triangulation. If a triangle , t_i^{del} in the regular triangulation is on convex hull of macromolecule than mark corresponding path vertex as boundary point. Create an edge, $e_{ij}^{path} = (v_i^{path}, v_j^{path}) \in E^{path}$ between any two vertices(v_i^{path} and v_j^{path}) in path graph if there corresponding triangles(t_i^{del} and t_j^{del}) in the regular triangulation that belongs to same tetrahedron in the regular triangulation.

Therefore, if there are x tetrahedra in a 3-dimensional regular triangulation there exist $4x$ vertices and $6x$ edges in corresponding path graph. Also Shewchuk et al.[18] proved that in the worst case the number of tetrahedra of regular triangulation $Del_W(S)$ is $O(n^2)$, where $n = |S|$. If the vertices in regular triangulation are uniformly or nearly uniformly distributed , the expected number of tetrahedra is nearly $O(n)$. In case of macromolecules like proteins the atoms are nearly uniformly distributed, hence number of vertices and edges in the path graph are also $O(n)$.

5.3.2 Process Path Graph

The goal of this step of our algorithm is to find the maximum size of solvent allowed by each edge, $e^{path} \in E^{path}$ in the path graph, G^{path} and the length of that edge.

To understand this step first we will consider the path graph of regular triangulation in two-dimensions. Note that in a two-dimensional path graph there exist path edge between any two edges of a triangle in regular triangulation, hence these two edges have a common vertex in regular triangulation. Therefore we will iterate through each vertex in regular triangulation.

It consists of following substeps:

1. Select an unvisited vertex v from regular triangulation. As shown in Figure 5.1 where the red vertex is selected. Set selected vertex v as visited.

2. Find 2-ring neighbour vertices of the selected vertex in regular triangulation using a breadth first search method. In Figure 5.1, light green vertices are 1-ring neighbours of v whereas green and light green vertices are 2-ring neighbours of v.

3. Identify all triangles t_i in regular triangulation that contains the selected vertex v. For each triangle t_i, we can identify edges $e_1^{t_i}$ and $e_2^{t_i}$ in triangle t_i. Let e_i^{path} be the path edge that pass between edges $e_1^{t_i}$ and $e_2^{t_i}$. Now to find the maximum size of solvent allowed by e_i^{path}(defined as $size_i^{max}$) and to find length of e_i^{path}(defined as len_i) do the following steps:

   (a) Identify all the vertices in 2-ring neighbourhood which are in the sector formed by selected vertex v as center and edges $e_1^{t_i}$ and $e_2^{t_i}$. The sector is shown in yellow in Figure 5.1. Selected vertex and identified vertices are shown in Figure 5.2.

   (b) For each vertex v_j in the sector, find the radius R^{v_j} of the smallest circle touching atoms at v and v_j. Such circles are shown in blue in Figure 5.2.

   (c) Store minimum of R^{v_j} over all v_j as R_i^{min}. Such a circle with radius R_i^{min} is shown in blue in Figure 5.3.

   (d) **RollCheck:** Now, roll the circle with minimum radius, R_i^{min} in the sector as shown in Figure 5.3. While rolling check if the circle intersects any other atom in the sector. The implementation of this step finds the rolled position of the circle with minimum radius on each line segment between v and each vertex v_j in the sector as shown in Figure 5.3. At each rolled position, we check whether or not circle intersects with any of the vertices in the sector. If it does than shrink its radius, R_i^{min} until there are no intersections. Therefore this step has running time complexity of $O(n_s^2)$, where n_s is the number of vertices in the sector.

   (e) Set $size_i^{max} = R_i^{min}$ and $len_i =$ length of arc of roll.

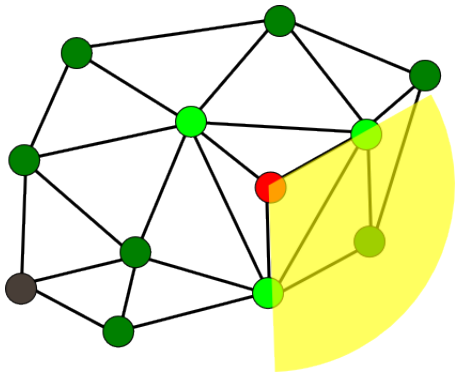


Figure 5.1: 2-ring neighbour vertices(in light green and green) of selected vertex(in red)

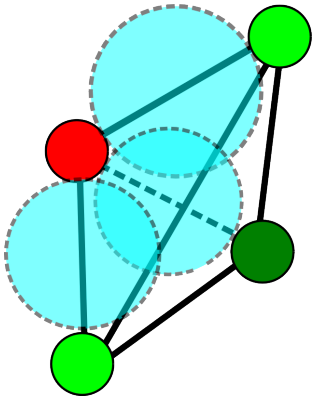


Figure 5.2: Smallest circle touching atoms at selected vertex in red and each of the vertices in sector. As described in Step 3b of Section 5.3.2.

4. If there are no unvisited vertex in regular triangulation exit, else goto Step 1.

Now we will consider the three-dimensional case. Note that in the three-dimensional path graph there exist a path edge between any two triangles of a tetrahedron in a regular triangulation. Hence these two triangles have a common edge in regular triangulation. Therefore we will iterate through each edge in the regular triangulation.

1. Select an unvisited edge e^{del} from regular triangulation.

2. Find 2-ring neighbour vertices of the selected edge in regular triangulation using a breadth first search method.

3. Identify all tetrahedra tet_i^{del} in regular triangulation that contains the selected edge e^{del}. For each tetrahedron tet_i^{del}, identify triangles $t_1^{tet_i}$ and $t_2^{tet_i}$. Let e_i^{path} be the path edge that

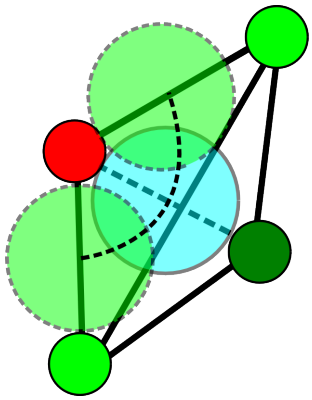


Figure 5.3: Rolling of circle with minimum radius, R_i^{min} in the sector as described in Step 3d of Section 5.3.2.

pass between triangles $t_1^{tet_i}$ and $t_2^{tet_i}$. Now to find the maximum size of solvent allowed by e_i^{path}(defined as $size_i^{max}$) and to find length of e_i^{path}(defined as len_i), do the following steps:

(a) Identify all the vertices in 2-ring neighbourhood which are in the sector formed by selected edge e^{del} as center and triangles $t_1^{tet_i}$ and $t_2^{tet_i}$.

(b) For each vertex v_j in the sector and find the radius R^{v_j} of the smallest sphere touching atoms at e^{del} and v_j also called tangent sphere. We find tangent sphere using method described in Langlet[10]. Tangent sphere to three atoms is shown in Figure 5.4.

(c) Store minimum of R^{v_j} over all v_j as R_i^{min}.

(d) **RollCheck:** Now, roll the tangent sphere with minimum radius, R_i^{min} in the sector. While rolling check if the tangent sphere intersects with any other atom in the sector. The implementation of this step finds the rolled position of the tangent sphere with minimum radius on each triangle formed by e^{del} and each vertex v_j in the sector. At each rolled position, we check whether or not the tangent sphere intersects with any of the vertices in the sector. If it does than shrink its radius R_i^{min} until there are no intersections. Therefore this step has running time complexity of $O(n_s^2)$, where n_s is the number of vertices in the sector.

(e) Set $size_i^{max} = R_i^{min}$ and $len_i =$ length of arc of roll.

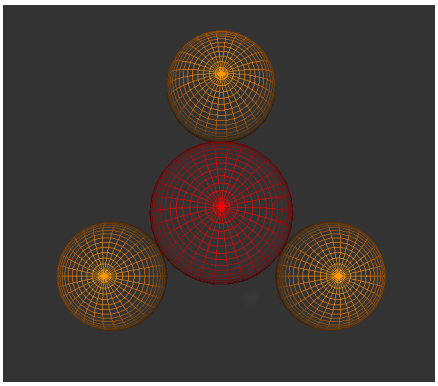


Figure 5.4: Tangent sphere (in red) to three spheres (in gold). The four spheres have coplanar centers.

4. If there are no unvisited vertex in regular triangulation exit, else goto Step 1.

5.4 Algorithm to find significant paths in a given macromolecule

Input: Path Graph, $G^{path} = (V^{path}, E^{path})$.

Input: Path vertex nearest to binding site.

Input(optional): Size of solvent molecule.

Output: Significant path(s) from path graph.

Our algorithm has following steps:

1. If solvent molecule size is given, remove all those path edges from path graph whose $size_i^{max}$ is less than size of solvent else do nothing.

2. Assign weight, $W(e)$ to each path edge in path graph as follows:

$$W(e) = len(e)/(size(e)^2 + \varepsilon)$$

where $len(e)$ is length of path edge and $size(e)$ is radius of solvent of maximum size allowed by the path edge and ε is a small number to avoid division by zero.

3. Set path vertex nearest to binding site as starting point and apply Dijkstra's shortest path algorithm to search for the shortest path from starting point to path vertices marked as boundary points.

4. To find next significant path, save the shortest path found in above step and give very high weights to its edges. Now run the Step 3 again.

5. To find a set of n significant paths, iterate between Step 3 and 4 n times.

Chapter 6

ANALYSIS

In this chapter we will mention the analysis of our algorithm to optimize it and find accuracy of results generated. The analysis is done on Intel Xeon CPU @2.00GHz.

6.1 Analysis of algorithm to find network of paths through a given macromolecule

The algorithm mentioned in Section 5.3.2 in three-dimensional case to find the maximum size of solvent allowed by each edge in path graph, G^{path} checks for all possible intersections between a solvent and atom in the molecule while passing from one triangle to another triangle of each tetrahedra in a regular triangulation. These checks guarantees that each edge in path graph allows a solvent molecule of size less than or equal to $size_i^{max}$ stored in Step 3e of the algorithm.

The following analysis is done to find avoidable checks and steps done in the above algorithm. We will refer the above mentioned algorithm as PPG1. We modify PPG1 for analysis as shown below.

6.1.1 Algorithm PPG2

Algorithm PPG2 is the same as PPG1 but it has lesser number of intersection checks in Step 3d. In PPG2 at each rolled position we check whether or not the tangent sphere intersects with the atom corresponding to vertex v_j mentioned in Step 3d, whereas in PPG1 we check intersection with all vertices in the sector. Therefore in PPG2, Step 3d has complexity of $O(n_s)$, where n_s is the number of vertices in the sector.

6.1.2 Algorithm PPG3

Algorithm PPG3 is same as PPG1, but here we do not compute the 2-ring neighbours of each selected vertex as mentioned in Step 2 of PPG1. In Step 3a of PPG3 we have only those two vertices of selected tetrahedron tet_i^{del} which are not in selected edge e^{del}.

This helps us to check whether finding 2-ring neighbour vertices of selected edge in Step 2 in PPG1 is necessary or not.

6.1.3 Algorithm PPG4

Algorithm PPG4 is same as PPG3 i.e. it doesn't compute the 2-ring neighbour vertices and further it doesn't check for any intersections.

This makes PPG4 least conservative version of PPG1. We have mentioned this version of PPG1 to compute the comparison with PPG1,PPG2 and PPG3.

6.1.4 Experiments

We perform the following tests on algorithms PPG1, PPG2, PPG3 and PPG4.

1. **Test 1:** This test has following steps.
   (a) Given a macromolecule and solvent molecule size, run all versions of PPG1, and compute processed path graph for each version of PPG1.
   (b) Identify those edges in path graph which allows the given solvent molecule to pass through.

Dataset	Solvent size(Å)	PPG1	PPG2	PPG3	PPG4
1GRM	0.5	889	889	928	946
1GRM	1.0	741	741	785	799
1GRM	1.5	582	582	621	641
1GRM	2.0	454	454	496	506
1J4N	2.0	1682	1682	1932	2024
1J4N	4.0	809	809	916	1013
1CQW	2.0	1444	1444	1699	1804
1CQW	4.0	626	626	749	842
1BL8	2.0	2082	2082	2395	2534
1BL8	4.0	926	926	1075	1169
3EYX	2.0	2531	2531	2901	3087
3EYX	4.0	1041	1041	1198	1332
1K4C	2.0	3558	3558	4122	4400
1K4C	4.0	1527	1527	1766	1938
2R9R	2.0	11788	11788	13233	13861
2R9R	4.0	5980	5980	6629	7104
1OAU	2.0	12456	12456	14259	14959
1OAU	4.0	4960	4960	5659	6115
1OAU	6.0	2454	2454	2772	3201
2BG9	2.0	13837	13837	15755	16518
2BG9	4.0	4793	4793	5500	5883
2BG9	6.0	2472	2472	2852	3146

Table 6.1: Results of different versions of algorithm PPG1 as number of tetrahedra.

(c) Compute and print the number of tetrahedra in regular triangulation corresponding to identified path edges in above step.

Test 1 is run on the following datasets Gramicidin(1GRM PDB), Nicotinic Acetylcholine Receptor(2BG9 PDB) and PDB 1OAU. Table 6.1 shows the results.

2. **Test 2:** This test computes the number of intersections found while running different versions of algorithm PPG1. Test 2 is run on the following datasets Gramicidin(1GRM PDB), Nicotinic Acetylcholine Receptor(2BG9 PDB) and PDB 1OAU. Results are shown in Table 6.2.

Dataset	PPG1	PPG2	PPG3	PPG4
1GRM	17084	17066	8814	0
1J4N	283118	282788	131093	0
1CQW	360597	360276	166049	0
1BL8	452244	451889	206358	0
3EYX	492573	492046	227843	0
1K4C	639357	638721	290105	0
2R9R	1778613	1776556	806995	0
1OAU	2189753	2186723	986485	0
2BG9	2381394	2379514	1072230	0

Table 6.2: Number of intersections found in different versions of algorithm PPG1. Number of intersections in PPG4 is zero because it doesn't check for any intersection.

Dataset	Number of atoms	PPG1	PPG2	PPG3
1GRM	184	0.55	0.52	0.11
1J4N	2241	8.91	8.53	1.46
1CQW	2806	11.39	10.76	1.89
1BL8	3489	14.01	13.44	2.27
3EYX	3845	15.67	15.03	2.53
1K4C	5003	20.17	19.37	3.28
2R9R	13659	56.47	54.56	9.1
1OAU	16647	69.38	65.14	10.91
2BG9	17923	74.94	71.73	11.81

Table 6.3: Running time of different versions of algorithm PPG1 in seconds: On Intel Xeon CPU @2.00GHz.

3. **Test 3:** This test computes the running time of different versions of algorithm PPG1. Figure 6.1 shows the running time of PPG1.

6.1.5 Conclusion

From Test1 and Test2 we can see that PPG1 and PPG2 compute approximately same results. But Test3 shows that there is not much difference in there running time hence we will take PPG1 as our final proposed algorithm.

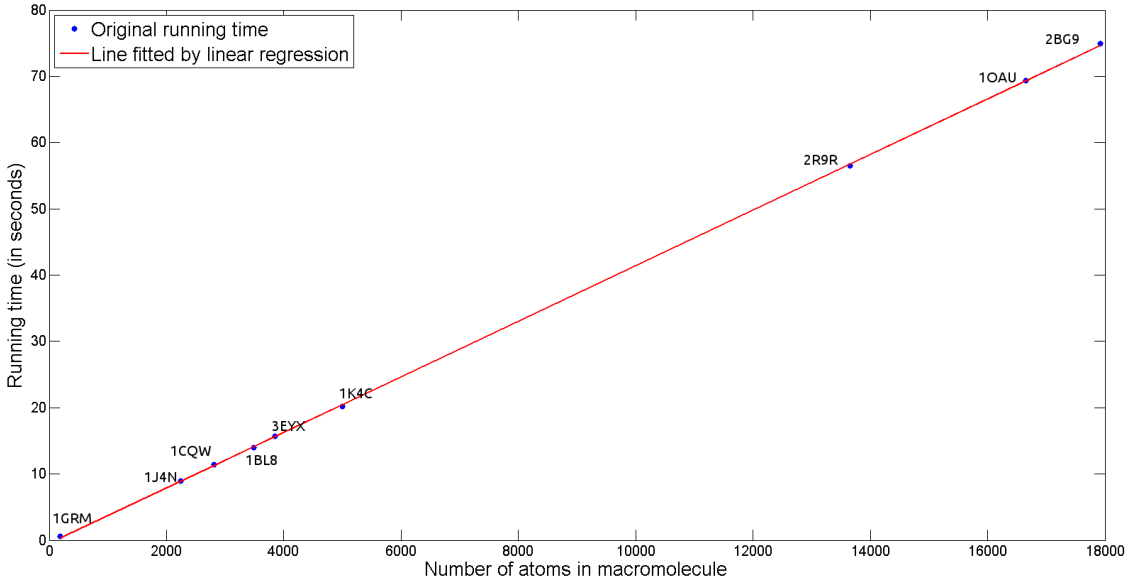


Figure 6.1: Plot of running times of PPG1: On Intel Xeon CPU @2.00GHz.

Dataset	Number of atoms	Solvent size (Å)	Running time (in seconds)
1GRM	184	0.5	0.03
1GRM	184	1.0	0.02
1GRM	184	1.4	0.02
1J4N	2241	1.4	0.1
1CQW	2806	1.4	0.03
1BL8	3489	1.4	0.09
3EYX	3845	1.4	0.06
2OAU	16647	1.4	0.14
2BG9	17923	1.4	0.17

Table 6.4: Run time of algorithm to find significant paths in a given macromolecule as described in Section 5.4 for different protein molecules: On Intel Xeon CPU @2.00GHz

PPG1 has nearly linear running time as shown in Figure 6.1. It is know that in the worst case the number of tetrahedra of regular triangulation $Del_W(S)$ is $O(n^2)$, where $n = |S|$ [3]. However Shewchuk et al.[18] proved that if the vertices in regular triangulation are uniformly or nearly uniformly distributed , the expected number of tetrahedra is nearly $O(n)$. In case of macromolecules like proteins the atoms are nearly uniformly distributed, hence number of edges in the regular triangulation are also nearly $O(n)$. PPG1 finds and processes 2-ring neighboring vertices of each edge in regular triangulation using a breadth first search method. Since there are nearly constant 2-ring neighboring vertices for each edge in regular triangulation therefore running time of PPG1 is nearly linear.

6.2 Analysis of algorithm to find significant paths in a given macromolecule

In this section we analyze significant paths generated by algorithm discussed in Section 5.4, which uses path graph generated from algorithm described in Section 5.3.

6.2.1 Intuition

Weighted alpha shapes have been used efficiently to compute volume, surface area, etc. of macromolecule which mostly involves intersection among atoms. To solve problem related to Euclidean proximity of atoms (like finding molecular channels) using weighted alpha shapes, radii of atoms in macromolecule have to be appropriately changed so that the problem transforms to one involving intersections among changed atoms.

From above mentioned intuition we can accurately find all accessible paths in a given macromolecule for given solvent size as follows:

1. Increament radius of atoms by given solvent radius

2. Get alpha complex at alpha equal to zero.

Dataset	Solvent size(Å)	Number of tetrahedra identified using PPG1	Accurate number of tetrahedra found using alpha complex
1GRM	2.0	442	581
1CQW	2.0	1462	2333
1J4N	2.0	1694	2566
1BL8	2.0	2099	3210
3EYX	2.0	2525	3764
2OAU	2.0	12341	18321
2BG9	2.0	13835	20055

Table 6.5: Number of tetrahedra found using PPG1 and alpha complex.

3. Find all tetrahedra in regular triangulation which are not in alpha complex and does not have 2 or more than 2 triangles in alpha complex.

6.2.2 Experimental Setup

We get tetrahedra corresponding to all accessible paths in a macromolecule for a given solvent size as follows:

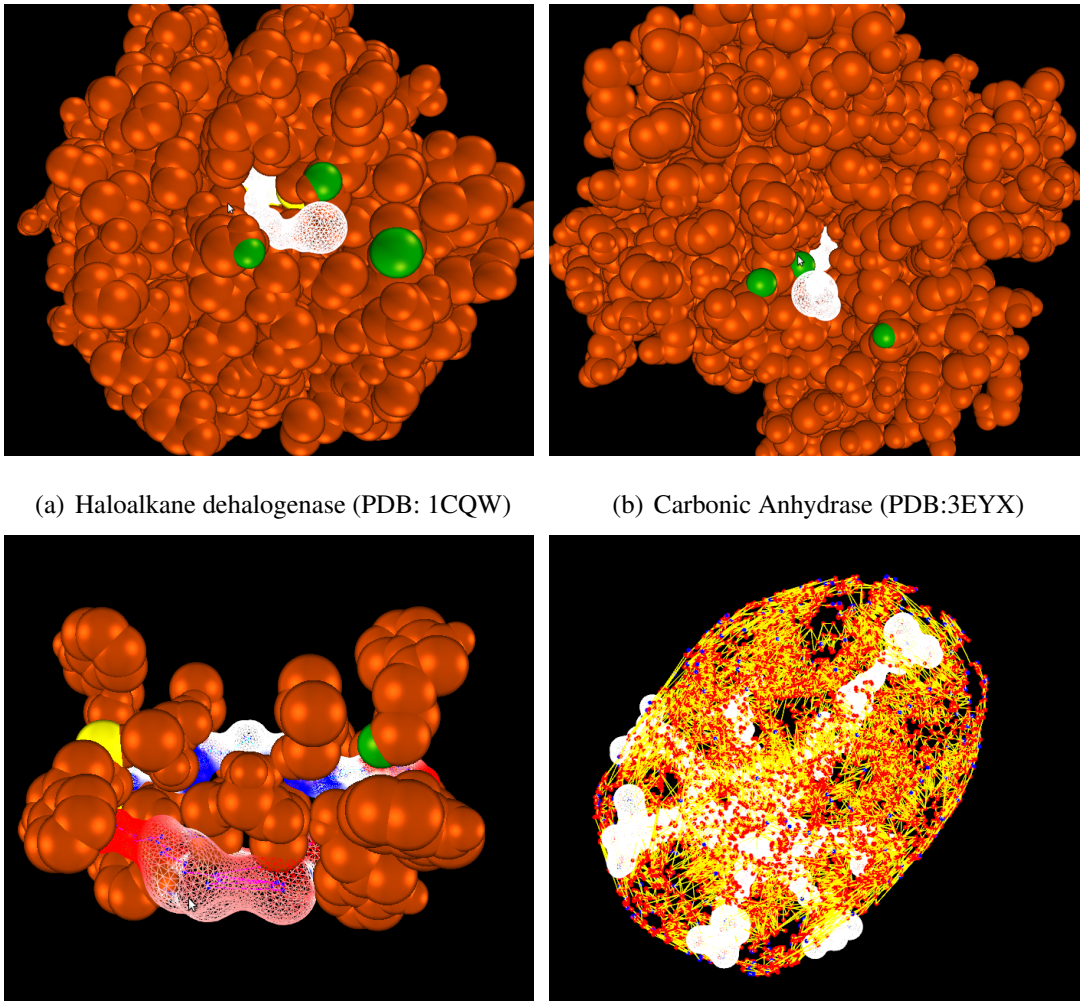
1. Increament radius of atoms by given solvent radius
2. Run PPG1 algorithm to get path graph.
3. Identify all those path edges which allows given solvent to pass through.
4. Find set of all tetrahedra in regular triangulation corresponding to identified path edges.

6.2.3 Conclusion

Table 6.5 shows results of Experimental setup described in above sections. We compared tetrahedra in both cases and came to know that tetrahedra found using our algorithm (PPG1) are subset of tetrahedra found using alpha complex. This conforms that results from our algorithm are consistant with actual network of paths generated by using alpha complex.

6.3 Results

This section shows known molecular channels found using our algorithm mentioned in Section 3.1. We have mentioned the minimum number of iterations of Steps 3 and 4 in algorithm to find significant paths(described in Section 5.4) containing the known molecular channels.



(a) Haloalkane dehalogenase (PDB: 1CQW)

(b) Carbonic Anhydrase (PDB:3EYX)

(c) Gramicidin (PDB: 1GRM)

(d) Mechanosensitive channel (PDB: 2OAU)

Figure 6.2: Known channels computed by our algorithm (a) Haloalkane dehalogenase (PDB: 1CQW), 1 iteration. (b) Carbonic Anhydrase (PDB: 3EYX), 1 iteration.(c) Gramicidin (PDB: 1GRM), 2 iterations. (d) Mechanosensitive channel (PDB: 2OAU), 8 iterations.

Chapter 7

VISUALIZATION FEATURES

We have added following features to our tool made in collaboration with other students.

1. Visualization of tetrahedra in regular triangulation corresponding to significant paths edges computed by our algorithm. Tetrahedra give better visualization of significant path volume as compared to skin surfaces. As shown in Figure 7.2(a).

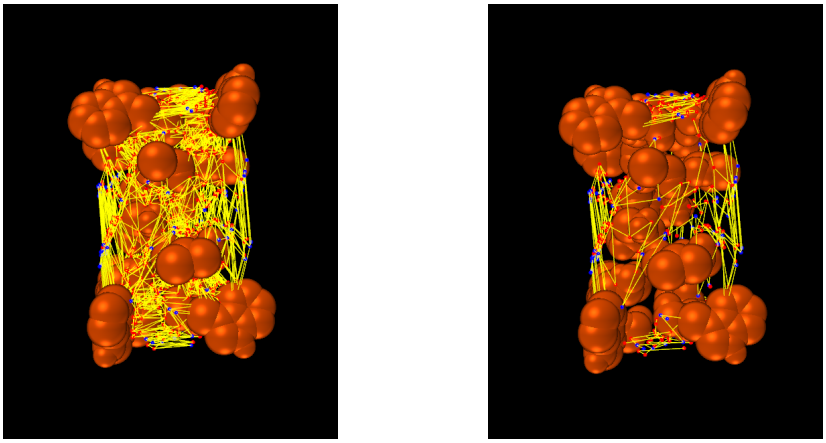
2. Visualization of skin surfaces of significant paths. Skin surfaces describe the inner surface of molecule enclosing the paths. Skin surfaces are constructed by taking each path vertex on a significant path as sphere having radius equal to tangent sphere to atoms in triangle of regular triangulation corresponding to that path vertex. We used implementation based on work of Cheng and Shi[2]. As shown in Figure 7.2(b).

3. Cut section of given macromolecule at arbitrary plane. For better visualization of significant paths. As shown in Figure 7.2(a), 7.2(b) and 7.2(c).

4. Graph to show size of solvent allowed by path edges along a significant path. It helps the user to identify bottlenecks in the significant paths. As shown in Figure 7.2(d).

5. Graph to show electric field along a significant path. It helps the user to analyze the electrostatic properties of the significant path. As shown in Figure 7.2(e).

6. Individual path vertex on a significant path as sphere having radius equal to tangent

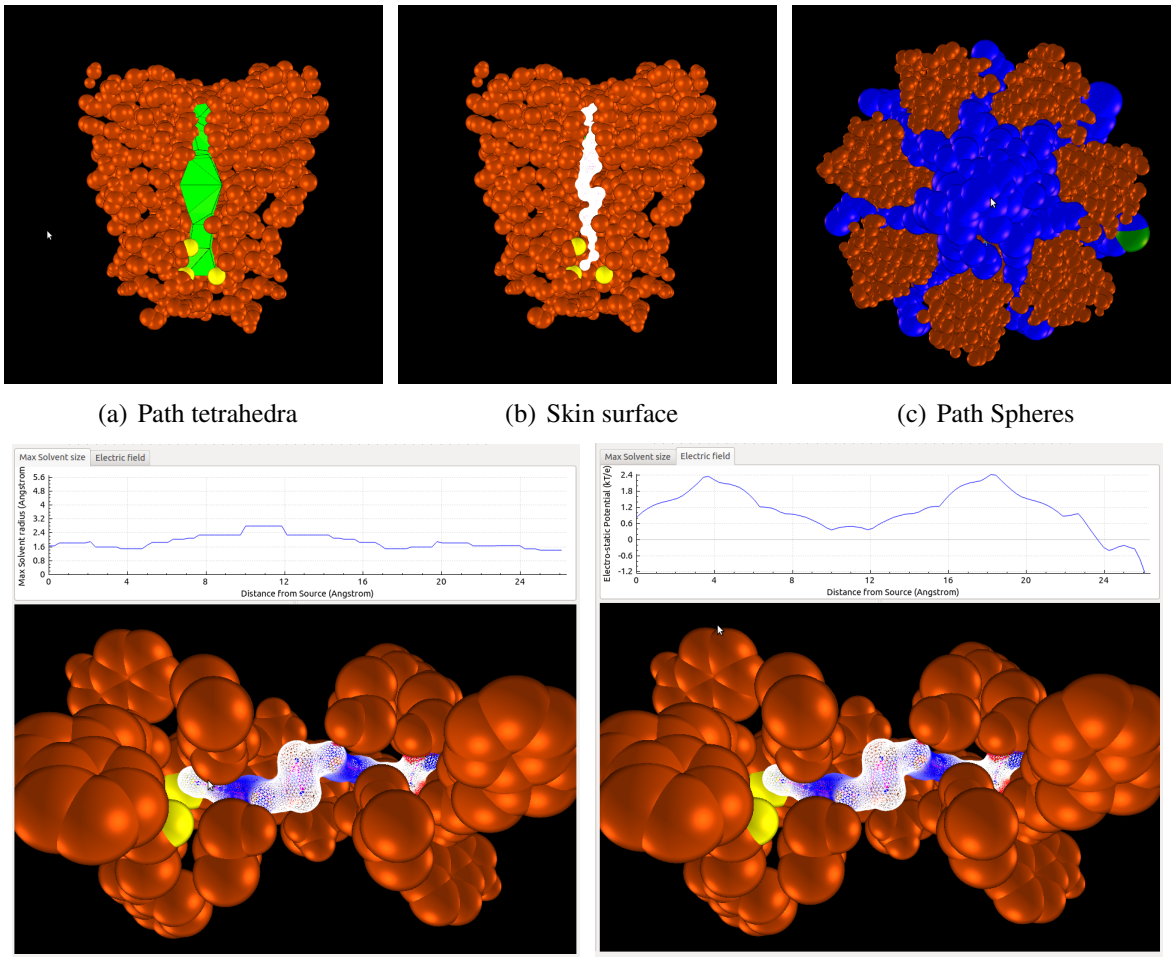


(a) Max solvent radius = 0 (b) Max solvent radius = 3.01

Figure 7.1: (a) Path Graph for solvent of radius atleast 0. (b) Path Graph for solvent of radius atleast 3.01 .

sphere to atoms in triangle of regular triangulation corresponding to that path vertex. As shown in Figure 7.2(c).

7. Filter and visualize the path graph based on the size of the solvent as shown in Figure 7.1. We can change the size of the path graph based on the size of the solvent molecule. This reduces processing time to find significant paths by algorithm described in section 5.4.



(a) Path tetrahedra (b) Skin surface (c) Path Spheres

(d) Max solvent graph (e) Electric field graph

Figure 7.2: (a) Tetrahedra corresponding to significant path edges. Cut section of PDB 1BL8. (b) Skin surface of significant path. Cut section of PDB 1BL8. (d) Graph of solvent size allowed by path edges along a significant path of Gramicidin (PDB 1GRM). (d) Graph of electric field along a significant path of Gramicidin (PDB 1GRM). (c) Tangent sphere corresponding to path vertices of significant paths of PDB 2OAU cut section.

Chapter 8

CONCLUSIONS AND FUTURE WORK

The following section mentions key contributions of our algorithm.

8.1 Key Contributions

1. This algorithm efficiently computes path graph from regular triangulation. Path graph is unique for a given macromolecule. Hence path graph can be stored and used to compute significant paths.

2. This algorithm uses regular triangulation of a macromolecule as input, hence we get a volumetric representation of paths. This gives us a unique advantage of viewing significant path volume as tetrahedra. From tetrahedra it is easy to compute surface area and volume of significant paths.

3. This algorithm can be used as plugin in already existing tools which uses regular triangulation (or its dual weighted Voronoi diagram) of macromolecules for visual analysis like PyMol[4].

8.2 Limitations and Future work

This algorithm computes conservative estimate of significant paths as concluded in Section 6.2.3, so it may miss some paths.

In future this algorithm can be parallelized and implemented as plugin in molecular visualization systems like PyMol[4].

Bibliography

[1] F. Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[2] H. Cheng and X. Shi. Quality mesh generation for molecular skin surfaces using restricted union of balls. *Computational Geometry*, 42(3):196–206, 2009.

[3] M. De Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational geometry*. Springer, 2008.

[4] W. L. DeLano. The pymol molecular graphics system. 2002.

[5] J. Dundas, Z. Ouyang, J. Tseng, A. Binkowski, Y. Turpaz, and J. Liang. Castp: computed atlas of surface topography of proteins with structural and topographical mapping of functionally annotated residues. *Nucleic acids research*, 34(suppl 2):W116–W118, 2006.

[6] H. Edelsbrunner, M. Facello, and J. Liang. On the definition and the construction of pockets in macromolecules. *Discrete Applied Mathematics*, 88(1):83–102, 1998.

[7] H. Edelsbrunner and E. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72, 1994.

[8] B. Ho and F. Gruswitz. Hollow: generating accurate representations of channel and interior surfaces in molecular structures. *BMC structural biology*, 8(1):49, 2008.

[9] G. Kleywegt and T. Jones. Detection, delineation, measurement and display of cavities in macromolecular structures. *Acta Crystallographica Section D: Biological Crystallography*, 50(2):178–185, 1994.

[10] G. Langlet. A new fast direct solution to the problem of the sphere tangent to four spheres. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 35(5):836–837, 1979.

[11] R. Laskowski. Surfnet: a program for visualizing molecular surfaces, cavities, and intermolecular interactions. *Journal of molecular graphics*, 13(5):323–330, 1995.

[12] D. Levitt and L. Banaszak. Pocket: A computer graphies method for identifying and displaying protein cavities and their surrounding amino acids. *Journal of molecular graphics*, 10(4):229–234, 1992.

[13] J. Liang, C. Woodward, and H. Edelsbrunner. Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Science*, 7(9):1884–1897, 2008.

[14] N. Lindow, D. Baum, and H. Hege. Voronoi-based extraction and visualization of molecular paths. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2025–2034, 2011.

[15] J. Munkres. *Elements of algebraic topology*, volume 2. 1984.

[16] M. Petřek, P. Košinová, J. Koča, and M. Otyepka. Mole: a voronoi diagram-based explorer of molecular channels, pores, and tunnels. *Structure*, 15(11):1357–1363, 2007.

[17] M. Petřek, M. Otyepka, P. Banáš, P. Košinová, J. Koča, and J. Damborskỳ. Caver: a new tool to explore routes from protein clefts, pockets and cavities. *BMC bioinformatics*, 7(1):316, 2006.

[18] J. R. Shewchuk. Stabbing delaunay tetrahedralizations. *Discrete & Computational Geometry*, 32(3):339–343, 2004.

[19] O. Smart, J. Neduvelil, X. Wang, B. Wallace, and M. Sansom. Hole: a program for the analysis of the pore dimensions of ion channel structural models. *Journal of molecular graphics*, 14(6):354–360, 1996.

[20] N. Voss and M. Gerstein. 3v: cavity, channel and cleft volume calculator and extractor. *Nucleic acids research*, 38(suppl 2):W555–W562, 2010.