SimDiv: a New Solution for Protein Comparison

Hassan Sayyadi[1], Sara Salehi[2], and Mohammad Ghodsi[3]

[1] Computer Engineering Department, Sharif University of Technology, sayyadi@ce.sharif.edu
[2] Computer Engineering Department, Azad University Tehran-South branch, Iran. sarasalehi@gmail.com
[3] Computer Engineering Department, Sharif University of Technology, Iran. IPM School of Computer Science, Tehran, Iran. ghodsi@sharif.edu *

1 Abstract

Proteins are the main players in the game of life. Good understanding of their structures, functions, and behaviors leads to good understanding of drugs, diseases, and thus our health. So, much effort has been done to study and categorize proteins. Nowadays, tens of thousands of proteins have been found. Moreover, the problem of comparing the proteins is hard. Therefore, efficient methods are needed to deal with this problem. In this paper, we use an important computational geometric concept and graph matching algorithm, namely, "Delaunay Tetrahedralization" and "Similarity Flooding", and propose a new idea to extract similar parts of proteins. Furthermore, we used protein fragmentation to reduce the time and storage complexity of the model for larger proteins.

2 Introduction

The number of known proteins is increasing every day; tens of thousands have been studied and categorized by now.

To understand the functions and behaviors of a newly found protein, one should find well studied proteins with similar structure. In fact, the behavior of a protein is related to its sequence of amino acids and its 3D structure. So the comparison of proteins is a key technique not only in finding similarities in the structures of proteins but also to categorize them and define families and super-families among the proteins. Like many comparison problems, this

* This author's work has been partly supported by IPM School of Computer Science (contract: CS1385-2-01)

problem is hard because there is neither an exact definition of the likelihood of proteins structures nor an efficient algorithm exits for it. Although there exist optimal dynamic programming algorithms for comparing the sequences of amino-acids, the result is highly related to the definition of the relations of the sequences, which has not been uniquely defined [1]. The problem of comparing the 3D-structures of proteins becomes even harder. There is no efficient algorithm which guarantees the optimality of the answer. In fact, this problem is NP-hard. When the proteins become more complicated, the relationship models are more varied than the models of sequence relatedness.

In this paper, we will propose a model for protein matching or extracting similar parts of two given proteins. We focus on the computational geometric approach and the graph matching method that are used to model and compare the sequence and 3D-structure of proteins.

The remainder of this paper is organized as follows: We first have a glance at the related works. There are two major methods used in the literature: "Delaunay Tetrahedralization" and "Similarity Flooding". We will explain the required information in the next section as the background knowledge, and then propose a new idea in section 5 which can improve the current methods. We will then present experimental results of the implemented method which shows its effectiveness.

3 Related works

Delaunay triangulation and Delaunay tessellation are common computational geometric methods used in the bioinformatics. For example in [2] the Delaunay tessellation the α-carbons of the protein molecule is used to study the HIV-1 protease. This is because this model provides objective and robust definition of four nearest-neighbor amino acid residues as well as a four-body statistical potential function. The other usages are studied in the fields of packing analysis [2, 3], fold recognition [4], virtual mutagenesis [5], and structure comparison.

Authors of [6] consider the Delaunay tetrahedralization determined by the alpha carbon positions of some particular protein. Starting at the amino-terminal residue, the edges of the tetrahedralization that connect to a residue that has already been encountered are recorded as a relative residue difference. For example, if there is an edge between the 5th alpha carbon and the 3rd one, this edge is represented as 2. When the edge of a particular residue is exhausted a 0 is recorded to indicate a new residue. This linear representation will contain each edge in the tetrahedralization exactly once. Furthermore, secondary structural components will be indicated by particular subsequences. Two one-dimensional representations are then compared by a dynamic programming scheme adapted from protein sequence analysis, thus reducing protein structural similarity to sequence similarity of the appropriate structure strings. In [7] the Euclidean metric for identifying natural nearest

neighboring residues via the Delaunay tessellation in Cartesian space and the distance between residues in sequence space. In addition, authors of [8] find recurring amino-acid residue packing patterns, or spatial motifs, that are characteristic of protein structural families, by applying a novel frequent sub-graph mining algorithm to graph representations of protein three-dimensional structure. Graph nodes represent amino acids, and edges are chosen in one of three ways: first, using a threshold for contact distance between residues; second, using Delaunay tessellation; and third, using the recently developed almost-Delaunay edges. Furthermore, [9] presents a solution that reduced the computation of PSIMAP, a protein interaction map derived from protein structures in the protein databank PDB and the structural classification of proteins SCOP [10] to know about interaction of two proteins. The original PSIMAP algorithm computes all pair-wise atom/residue distances for each domain pair of each multi-domain PDB entry. But they developed an effective new algorithm, which substantially prune the search space. The basic idea of their novel algorithm is to prune the search space by applying a bounding shape to the domains. Interacting atoms of two domains can only be found in the intersection of the bounding shapes of the two domains. Generally, the proposed algorithm steps are:

1. A convex hull for each of the two domains is computed.
2. Both convex hulls are swelled by the required contact distance threshold.
3. The intersection of the two transformed convex hulls is computed.
4. All residue/atom pairs outside the intersection are discarded and for the remaining residues/atoms the number of residue/atom pairs within the distance threshold is computed. If this number exceeds the number threshold the two domains are said to interact.

4 Background Knowledge

4.1 Delaunay Tetrahedralization

Delaunay tetrahedralization is a special type of tetrahedralization which is defined based on the Voronoi diagram through the principle of duality[11]. A Voronoi box is formed through the intersection of planes and is therefore a general irregular polyhedron (Fig. 4.1). The facets of the Voronoi boxes correspond in the dual graph to the Delaunay edges which connect the points of P.

- *Voronoi*: Let $P = \{p_1, \ldots, p_k\}$ be a finite set of points in the n-dimensional space R^n and their location vectors $\mathbf{x}_i \neq \mathbf{x}_j \ \forall\, i \neq j$. The region given by

$$V(p_i) = \{\mathbf{x} \mid |\mathbf{x} - \mathbf{x}_i| \leq |\mathbf{x} - \mathbf{x}_j| \ \forall\, j \neq i\}$$

  is called the Voronoi region (Voronoi box) associated with p_i and $\mathcal{V}(P) = \bigcup_{i=1}^{k} V(p_i)$ is said to be the Voronoi diagram of P.

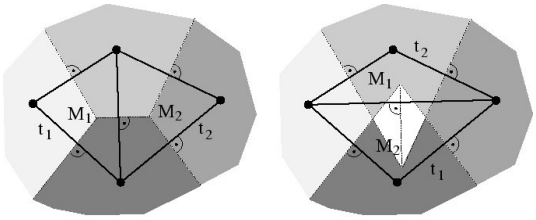

Fig. 1. Each Voronoi box associated with a point is differently shaded. Two triangles t_1 and t_2 with their circumcenters M_1 and M_2 which are the vertices of the Voronoi boxes are depicted for the correct Delaunay case and for the non-Delaunay case. Incorrect Voronoi boxes which are derived from non-Delaunay triangles overlap.[11]

A Voronoi box is formed through the intersection of planes and is therefore a general irregular polyhedron. The facets of the Voronoi boxes correspond in the dual graph to the Delaunay edges which connect the points of P.

- *Delaunay Edge*: Let P be a finite set of points in a sub-domain Ω^n of the n-dimensional space R^n. Two points p_i and p_j are connected by a Delaunay edge e if and only if there exists a location $x \in \Omega^n$ which is equally close to p_i and p_j and closer to p_i, p_j than to any other $p_k \in P$. The location x is the center of an n-dimensional sphere which passes through the points p_i, p_j and which contains no other points p_k of P.

$$\begin{aligned} e_{Delaunay}(p_i, p_j) \Leftrightarrow \exists x: \ & x \in \Omega^n \\ \wedge \ & |x - p_i| = |x - p_j| \\ \wedge \ & \forall k \neq i, j: \ |x - p_i| < |x - p_k| \end{aligned}$$

Combining this criterion for the three edges of a triangle (Fig. 2) and furthermore for the four triangles of a tetrahedron leads to the following criterion for Delaunay tetrahedron.

- *Delaunay Triangle*: Let P be a finite set of points in a sub-domain Ω^n of the n-dimensional space R^n. Three non-collinear points p_i, p_j and p_k form a Delaunay triangle t if and only if there exists a location $x \in \Omega^n$ which is equally close to p_i, p_j and p_k and closer to p_i, p_j, p_k than to any other $p_m \in P$. The location x is the center of an n-dimensional sphere which passes through the points p_i, p_j, p_k and which contains no other points p_m of P. For $n = 2$ only one such sphere exists which is the circumcircle of t.

$$\begin{aligned} t_{Delaunay}(p_i, p_j, p_k) \Leftrightarrow \exists x: \ & x \in \Omega^n \\ \wedge \ & |x - p_i| = |x - p_j| = |x - p_k| \\ \wedge \ & \forall m \neq i, j, k: |x - p_i| < |x - p_m| \end{aligned}$$

implies that an empty circumcircle is necessary but not sufficient for Delaunay surface triangles in three dimensions. This is the reason why a two-dimensional Delaunay Triangulation code is of limited use to construct a three-dimensional Delaunay surface triangulation. The Delaunay edge and Delaunay triangle criteria are depicted in Fig. 2. A Delaunay tetrahedron corresponds to a point in the Voronoi diagram, which is the vertex of four incident Voronoi boxes.

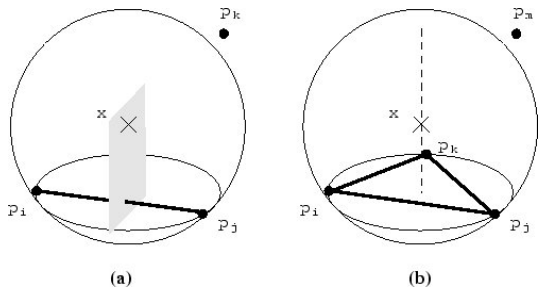

Fig. 2. a) Delaunay Edge b) Delaunay Triangle criteria.[11]

- *Delaunay Tetrahedron*: Let P be a finite set of points in a sub-domain Ω^n of the n-dimensional space R^n, where $n \geq 3$. Four non-coplanar points p_i, p_j, p_k and p_l form a Delaunay tetrahedron T if and only if there exists a location $x \in \Omega^n$ which is equally close to p_i, p_j, p_k and p_l and closer to p_i, p_j, p_k, p_l than to any other $p_m \in P$. The location x is the center of an n-dimensional sphere which passes through the points p_i, p_j, p_k, p_l and which contains no other points p_m of P. For $n = 3$ only one such sphere exists which is the circumsphere of T.

$$\begin{aligned} T_{Delaunay}(p_i, p_j, p_k, p_l) \Leftrightarrow \exists x : \; & x \in \Omega^n \\ & \wedge \; |x - p_i| = |x - p_j| = |x - p_k| = |x - p_l| \\ & \wedge \; \forall m \neq i, j, k, l \; : \; |x - p_i| < |x - p_m| \end{aligned}$$

A Delaunay tetrahedron must consist of Delaunay edges and Delaunay triangles. The edge and triangle criteria are implicit, because the existence of the n-dimensional sphere in *Delaunay Edge* criterion and in *Delaunay Triangle* criterion is guaranteed by the sphere in *Delaunay Tetrahedron* criterion.

4.2 Similarity Flooding

Matching or finding similar elements of two data schemas or two data instances plays a key role in data warehousing, e-business, or even biochemical applications. Authors of [12] present a matching algorithm named "Similarity Flooding" based on a fixpoint computation that is usable across different

scenarios. As the example illustrating the Similarity Flooding Algorithm is shown in Fig. 3, the algorithm takes two graphs (schemas, catalogs, or other data structures) as input, and produces as output a mapping between corresponding nodes of the graphs.

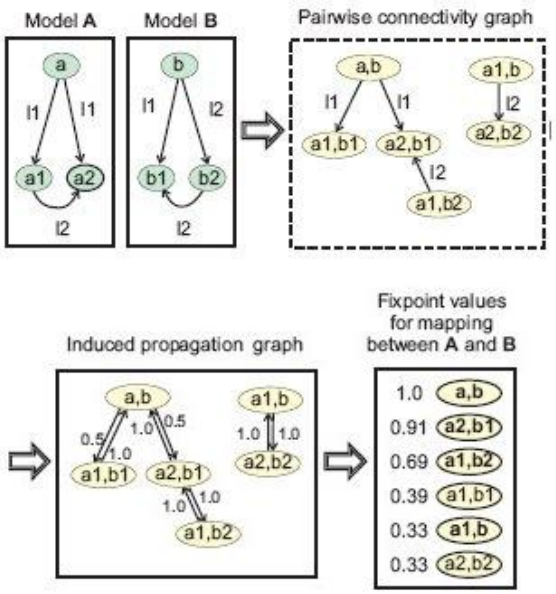

Fig. 3. Example illustrating the Similarity Flooding Algorithm[12]

As a first step, the schemas should be translated from their native format into graphs G_1 and G_2. Next, the pair-wise connectivity graph (PCG) should be made that is an auxiliary data structure derived from G_1 and G_2. If N_1 represent the set of all nodes in G_1 and respectively N_2, each node in the connectivity graph is an element from $N_1 \times N_2$ and is called "map-pair". Furthermore, edges in connectivity graph defined as follow:

$$((x_1, y_1), P, (x_2, y_2)) \in PCG(G_2, G_2)$$
$$\Leftrightarrow$$
$$(x_1, P, x_2) \in G_1 \ \ and \ \ (y_1, P, y_2) \in G_2$$

Each map-pair contains one node from each graph and the similarity score between them. The initial similarity for each map-pair is obtained using a simple string matcher that compares common prefixes and suffixes of literals in each node. Finally, computing the similarities relies on the intuition that

elements of two distinct models are similar when their adjacent elements are similar. In other words, a part of the similarity of two elements propagates to their respective neighbors as follow:

$$\begin{aligned}\sigma^{k+1}(x,y) = \sigma^{k}(x,y) \\ + \sum_{(a_i,x)\in G_1,(b_i,y)\in G_2} \sigma^{k}(a_i,b_i).W((a_i,b_i),(x,y)) \\ + \sum_{(x,a_i)\in G_1,(y,b_i)\in G_2} \sigma^{k}(a_i,b_i).W((x,y),(a_i,b_i))\end{aligned}$$

where $\sigma^k(x,y)$ shows the similarity between x and y after iteration k and $W((a_i,b_i),(x,y))$ is the propagation weight of the similarity between a_i and b_i to the similarity between x and y. The above computation is performed iteratively until the Euclidean length of the residual vector $\triangle(\sigma^n,\sigma^{n-1})$ becomes less than ϵ for some $n > 0$. If the computation does not converge, it will be terminated after some maximal number of iterations.

5 Proposed Method

In this section we present the proposed approach. Here we combine sequence similarity which is a simple extension of amino acid or nucleotide similarity and structural similarity which is the residues position similarity. Using both techniques leads to an efficient method for extracting similar parts of proteins.

The different phases of the proposed method may be represented as follow:

1. Protein Tetrahedralization
2. Creating pair-wise graph
3. Similarity propagation
4. Extracting similar components.

We will discuss each phase in the following subsections.

5.1 Protein Tetrahedralization

Each Protein is a sequence of residues in the 3D space in which each two consequent residues are connected by one edge called "chain-edge". Firstly, for each protein, we use Delaunay tetrahedralization algorithm to convert the protein sequence to a tetrahedralized shape (Fig. 4(b)). Since all proteins have 3D shape, using Delaunay algorithm leads to create edges called "tetrahedralization-edge" between atoms which are close to each other in space, regardless of their distance in protein sequence. This closeness has an extremely high influence on structural similarity which will be discussed in proposed method for extracting similar parts of proteins.
Inasmuch as Tetrahedralization algorithm creates convex shape, in order to

have a much more similar shape to the real protein shape, we need to eliminate edges whose length are more than α for tetrahedralization-edges and more than β for chain-edges. Obviously, the value of β is more than α, because of the importance of chain-edges in proteins comparison (Fig. 4(c)).

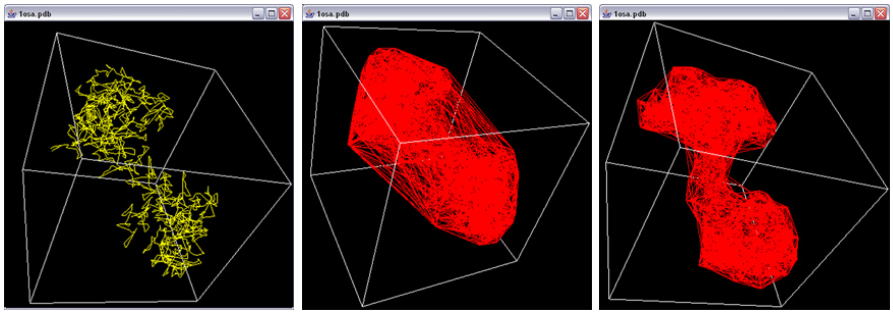

Fig. 4. a) Protein chain b) Tetrahedralized protein c) Tetrahedralized protein after removing worthless edges

We now construct a graph from the tetrahedralized shape. Each node in this graph contains one number which is the amino acid number of corresponding atom in protein and coordinates (x, y, z) expressing the coordination of that atom. This graph has two different types of edges:

1. Edges which belong to the protein chain and also can be the part of the tetrahedrons named chain-edges.
2. Edges obtained from tetrahedralization which do not belong to the protein chain named tetrahedralization-edges.

Consequently, each protein is converted to one graph which not only contains the protein chain but also contains edges connecting atoms near each other in 3D space. These graphs are data structures for similarity flooding algorithm used in protein matching.

5.2 Creating Pair-wise Graph

Pair-wise Connectivity graph (PCG) arises from two protein's graphs P_1 and P_2 which were created through tetrahedralization in the first phase. If N_1 and N_2 show the sets of all nodes in P_1 and P_2, each node in the pair-wise graph is an element from $N_1 \times N_2$. We call such nodes map-pairs. The edges of pair-wise graph are categorized to 3 parts depending on their map-pairs (see Fig. 5):

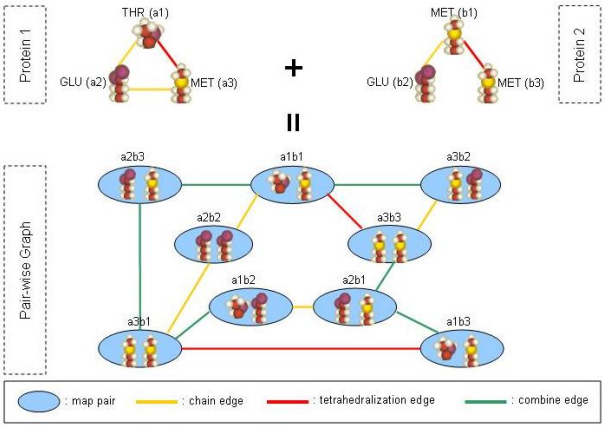

Fig. 5. Pairwise connectivity graph for proteins

1. If a chain-edge exists between the first nodes of two map-pairs and there is a chain-edge between the second nodes of those map-pairs in their proteins, then we will connect these two map-pairs with an edge of type chain.

$$((x,y),CH,(x',y')) \in PCG(P_1,P_2) \Leftrightarrow$$
$$(x,CH,x') \in P_1 \; and \; (y,CH,y') \in P_2$$

   where CH represents the edge of type chain.
2. If a Tetrahedralization-edge exists between the first nodes of two map-pairs and there is a tetrahedralization-edge between the second nodes of those map-pairs in their proteins, then we will connect these two map-pairs tetrahedralization-edge.

$$((x,y),T,(x',y')) \in PCG(P_1,P_2) \Leftrightarrow$$
$$(x,T,x') \in P_1 \; and \; (y,T,y') \in P_2$$

   where T represents the edge of type tetrahedralization.
3. If a tetrahedralization-edge exists between the first nodes of two map-pairs and there is a chain-edge between the second nodes of those map-pairs in their proteins or vice versa, then we will connect these two map-pairs with edge of type combine.

$$((x,y),C,(x',y')) \in PCG(P_1,P_2) \Leftrightarrow$$
$$(x,T,x') \in P_1 \; and \; (y,CH,y') \in P_2$$
$$or$$
$$(x,CH,x') \in P_1 \; and \; (y,T,y') \in P_2$$

   where C represents the edge of type combine.

We categorized these edges to three types, because the influence of their nodes in similarity propagation weights in the proposed method will differ from each other.

5.3 Similarity Propagation

In the created pair-wise graph, the primary similarity of each map-pair depends on the similarity between two nodes of that map-pair which are the atoms of the proteins. This similarity derives from amino acids scoring matrix. The two-dimensional matrix contains all possible pair-wise amino acid scores. Scoring matrices are also called substitution matrices because the scores represent relative rates of evolutionary substitutions. According to the similarity flooding algorithm the similarity of a map-pair increment based on the similarities of its neighbors in the pair-wise graph. Hence, the similarities of neighbors are affective in calculating final similarity between two atoms in each map-pair. It seems more rational for neighbors related to chain-edges to be much more affective in protein matching. Similarly, the weight of neighbors related to combine edges is more than those of tetrahedralization-edges. Thus, over a number of iterations, the initial similarity of any two nodes propagates through the graphs. Similarity propagation in each iteration is computed as follows:

$$\begin{aligned} Sim^{i+1}(x,y) &= a * Sim^{i}(x,y) \\ &\quad + (1-a) * \mathit{NeighborAffect}(x,y) \end{aligned}$$

$$\begin{aligned} \mathit{NeighborAffect}(x,y) &= \mathit{CHF} * \mathit{CHAffect}^{i}/\mathit{NF} \\ &\quad + \mathit{CF} * \mathit{CAffect}^{i}/\mathit{NF} \\ &\quad + \mathit{TF} * \mathit{TAffect}^{i}/\mathit{NF} \end{aligned}$$

In the above equation, $Sim^{i}(x,y)$ defined as the similarity between x and y in each map-pair after i number of iteration(s), and a is the learning rate from the neighbors. Moreover,

- *CHAffect* is the average similarity of neighbors connecting with chain-edge to the respective map-pair and is calculated as bellow:

$$\mathit{CHAffect}^{i}(x,y) = \sum_{\substack{((x,y),CH,(x_i,y_i)) \\ \in PCG(P_1,P_2)}} \frac{Sim^{i}(x_i,y_i)}{\mathit{CHSize}}$$

  and *CHF* is the propagation weight of *CHAffect*.
- *CAffect* is the average similarity of neighbors connecting with combine edge to the respective map-pair and is defined as bellow:

$$CAffect^i(x,y) = \sum_{\substack{((x,y),C,(x_i,y_i)) \\ \in PCG(P_1,P_2)}} \frac{Sim^i(x_i,y_i)}{CSize}$$

and CF is the propagation weight of *CAffect*.

- *TAffect* is the average similarity of neighbors connecting with Tetrahedralization-edge to the respective map-pair and is computed as bellow:

$$TAffect^i(x,y) = \sum_{\substack{((x,y),T,(x_i,y_i)) \\ \in PCG(P_1,P_2)}} \frac{Sim^i(x_i,y_i)}{TSize}$$

and TF is the propagation weight of *TAffect*.

In the above formula, $CHSize$ ($CSize$ and $TSize$) is the number of edges of type chain (combine and tetrahedralization) connected to the map-pair. The sum of the CHF,CF and TF must be equal 1 to restrict *NeighborAffect* between valid rang which will be discussed in experimental results section. Furthermore, NF is normal factor applying to cases in which there is no neighbors related to edges of one type. For example, assume that there isn't any neighbor related to tetrahedralization-edge, but there are edges of chain and combine types. Hence, we should set $NF = CHF + CF$ to normalize *NeighborAffect* into valid rang.

5.4 Extracting Similar Components

Due to the fact that the similarity degree of each map-pair in pair-wise graph expresses the matching degree of its atoms, we should extract similar components of two proteins by eliminating map-pairs and their related edges in pair-wise graph which have similarity degree less than γ. Consequently, the pair-wise graph transform to forest in which we have several connected components. Each connected component declares one similar part of two proteins, and each map-pair in connected components expresses matched atoms between two proteins. Similarly, each edge in connected components shows conforming edges between two proteins. Hence, by extracting each protein's nodes and edges from the connected component, we obtain two connected sub-graphs, each of which belongs to one of two given proteins. Connected components with the number of nodes less than η are not valuable for the result of matching, therefore they should be removed. For example, assume that you have two pictures, and you want to match them. Obviously if one pixel of them is analogous, you can not assert that these two pictures are the same or you find valuable matching. Hence, the number of nodes of each connected components should be noticed and connected components containing less than η number of atoms should be eliminated.

6 Experimental Results

In this section we will explain SimDive engine which extract similar parts of proteins by using proposed method. We read protein information from PDB files. Then we used Visad[4] package to do Delaunay tetrahedralization on the input protein chain, this tetrahedralization contributes our sequence chain of protein to convert to tetrahedralized shape which is needed in counting structural similarity in proposed protein matching algorithm. We can construct filters that apply $\alpha = 2$ for creating boundary for tetrahedralization-edges length and $\beta = 5$ for removing worthless chain-edges. Hence, edges which are longer than these thresholds were removed. Fig. 6 shows the accuracy obtained after applying above filters in two proteins shape.

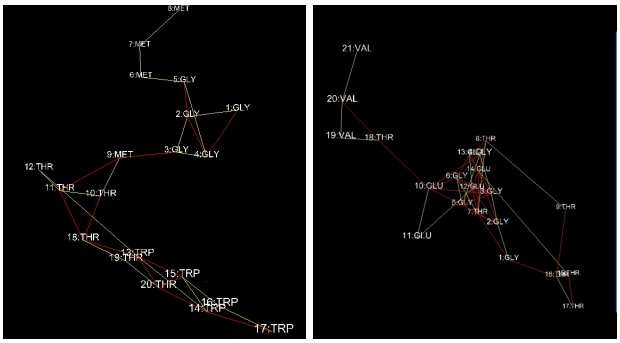

Fig. 6. a) Protein number 1 after tetrahedralization b) Protein number 2 after tetrahedralization

Two tetrahedralized shapes in the form of graph data structure were used to create pair-wise graph. After this creation, we used amino acid scoring matrices presented in Blast book[5] to assign primary similarity to each map-pair. Table 1 shows the Amino Acid Scoring Matrix. Inasmuch as a part of the similarity of two nodes in each map-pair propagates to their respective neighbors, we propagated the similarities of map-pair via proposed equation in which CHAffect,CAffect and TAffect were defined as the affect of neighbors of the map-pairs. The final similarities were obtained after applying filters shown in bellow.

Parameter	a	CHF	CF	TF
Value	0.8	0.5	0.3	02

[4] http://www.ssec.wisc.edu/ billh/visad.html
[5] http://safari.oreilly.com/0596002998/blast-CHP-4-SECT-3

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ALA	1	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
ARG	2	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
ASN	3	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
ASP	4	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
CYS	5	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
GLN	6	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
GLU	7	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
GLY	8	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
HIS	9	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
ILE	10	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
LEU	11	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
LYS	12	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
MET	13	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
PHE	14	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
PRQ	15	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
SER	16	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
THR	17	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
TRP	18	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
TYR	19	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
VAL	20	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Table 1. Amino Acid Similarity matrix

As you see in the above table, owing to the importance of neighbors in chain, we set the CHF factor more than two times of the TF factor, and correspondingly the CF factor more than the TF factor.

The similarity values between two amino acids in the scoring matrix are between -4 and +11, and these similarities between one amino acid with itself are between +4 and +11.

After propagating, the similarities still remained between -4 and +11. Hence, to avoid removing map-pairs which contain two identical nodes or two different nodes with reasonable similitude degree, we set the threshold $\gamma = +3$. Therefore, map-pairs whose similarities are less than this threshold were eliminated. After this removal, components which included more than $\eta = 3$ atoms from each protein were extracted. Extracted similar components from small parts of two proteins which shown in Fig. 6 are represented in Fig. 7 (Owning to have clear figures, we choose very small part of proteins for this figures).

Because of the large number of residues in each protein, the pair-wise graph created from two proteins will contain a large number of map-pairs. Consequently dealing with these large number of map-pairs needs large amount of memory. Moreover, the process of similarity propagation and similar components extraction will become very time consuming jobs. Hence, for managing this problem, we do protein fragmentation for larger proteins to accelerate the whole process of comparison. In this case, we divided each of our proteins to

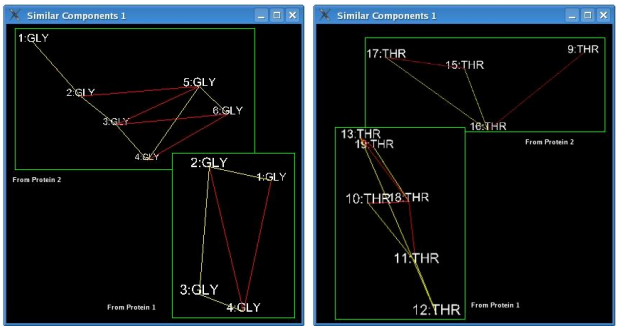

Fig. 7. Extracted similar components

many segments and then each segment of protein will be compared with all segments of the other one. After that we extracted similar parts between each two fragments. Using this method reduces the model time and space complexity. if $f(n)$ presents the running time of our proposed model for comparing two proteins of size n without using fragmentation, the time complexity of the model will be exponential, However by using fragmentation it will be linear. In other words, if m be the size of each fragment the time complexity function $f'(n)$ will calculated as follow:

$$f'(n) = (\frac{n}{m})^2 * f(m)$$

Nevertheless, using fragmentation leads the similar parts of two proteins which are located on fragmentation points to be broken. As you can see in Fig. 8, (A, B) is a similar component of the two given proteins, and you can see that after applying fragmentation the component (A, B) converts to $(A1, B1)$ and $(A2, B2)$ which is an undesirable fracture.

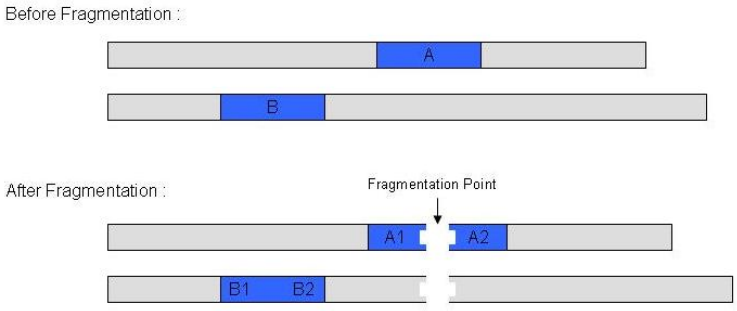

Fig. 8. Protein Fragmentation

To avoid these undesirable fractures, we let our segments to overlap with each other over their boundaries(Fig.9). Hence, if m be the size of each fragment and l be the size of segments overlap, the time complexity function $f''(n)$ will calculated as follow:

$$f''(n) = (\frac{n}{m})^2 * f(m + 2 * l)$$

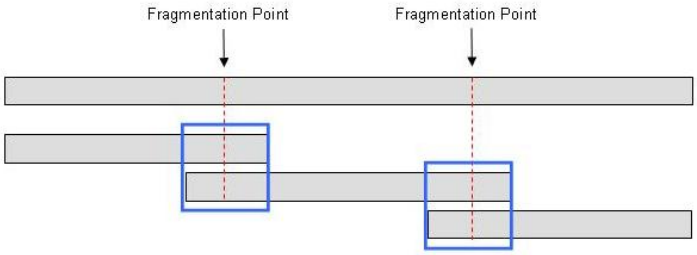

Fig. 9. Overlapping

Fig. 10 shows our experimental results for running the whole algorithm. It is obvious that by using fragmentation we hasten the whole process. This chart related to segments of size 200 and overlapping parameter is equal to 50.

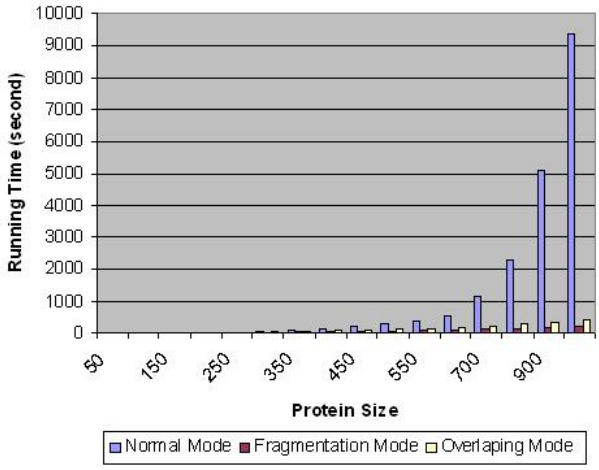

Fig. 10. Running Time

We also calculate the tetrahedralization time with and without using fragmentation. You can see the results in Fig. 11.

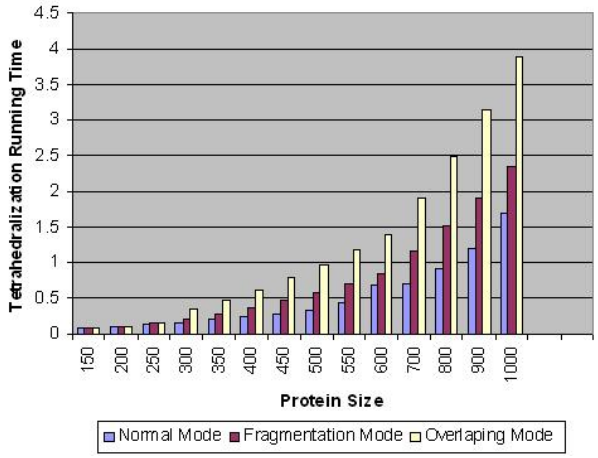

Fig. 11. Triangulation Time

although we mentioned that using fragmentation leads to unwonted fractures, Fig.12 expresses that the number of these unwonted fractures are not worthy.

On the other hand, this figures shows that the number of similar component

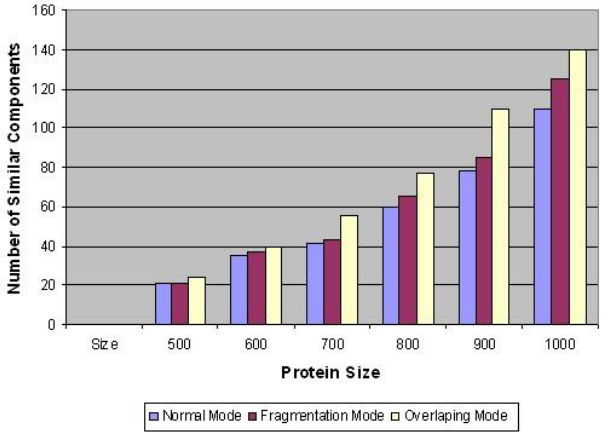

Fig. 12. Number of Similar Components

extracted in Overlapping mode is more than the basic fragmentation mode because in the overlapping mode the similar component in overlapped section may be calculated twice. The important thing is that by overlapping we find both the fractured and not fractured components and just the duplicated components should be ignored.

7 Conclusion and Future Works

In this paper we proposed a novel method used to extract similar parts of proteins based on computational geometry and graph matching methods. Our method used the Delaunay tetrahedralization of the α-carbon atoms in the protein molecules to add some edges in protein structure for short distance nodes in counting structural similarity. The basic method can build a robust model for protein matching but it needed some enhancements. Because of the large number of residues in each protein, the pair-wise graph created from two proteins will contain a large number of map-pairs and therefore large amount of memory was needed. Hence, we applied protein fragmentation and overlapping to reduce the time complexity of our algorithm. We implement SimDiv engine for testing the proposed method and the experimental results show the final verifications and effectiveness of our proposed method.

Furthermore, we need some test collections for optimizing proposed model parameters such as similarity propagation weight of each type in propagation graph and thresholds used in the model. Moreover, this model will be test on more real proteins with real size. In our experiments we use some parts of real proteins instead of complete structure of real protein because of mentioned problems.

Moreover, our proposed method can extracts similar parts of the two given protein precisely. Furthermore, it includes both structural and sequential similarity. Our model has great flexibility in all aspects and by changing different model parameters such as propagation weight of different edge type, we can change the influence of structural and sequence similarity.

References

1. Eidhammer I, Jonasses I, Taylor W (2000) Structure comparison and structure patterns
2. Finney J (1970) Random packing and the structure of simple liquids, the geometry of the random close packing. In: Proc R Soc. 479493
3. Tropsha A, Carter C, Cammer S, Vaisman I (2003) Simplicial neighborhood analysis of protein packing (snapp) a computational geometry approach to studying proteins. In: Methods Enzymol. 509544
4. Cho W Z S, Vaisman I, Tropsha A (1997) A new approach to protein fold recognition based on delaunay tessellation of protein structure. In: Pacific Symposium on Biocomputing, Singapre 487-496

5. Carter C, LeFebvre B, Cammer S, Trosha A, Edgell M (2001) Four-body potentials reveal protein-specific correlations to stability changes caused by hydrophobic core mutations. In: J Mol Biol. 625638
6. Roach J, Sharma S, Kapustina M, Carter C (2005) Structure alignment via delaunay tetrahedralization. In: Proteins: Structure, Function, and Bioinformatics. Volume 60. 66–81
7. Bostick D, Shen M, Vasiman I (2004) A simple topological representation of protein structure: Implications for new, fast, and robust structural classification. In: Proteins: Structure, Function, and Bioinformatics. Volume 56. 487–501
8. Hun J, Bandyopadhyay D, Wang W, Snoeyink J, Prins J, Trosha A (2005) Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. In: Journal of Computational Biology. Volume 12. 657-671
9. Dafas P, Gomoluch A K , Schroeder M (2004) Structural protein interactions: From months to minutes. In: Elsevier B.V, Parallel Computing: Software Technology, Algorithms, Architectures & Applications. 677-684
10. Park J, Lappe M, Teichmann S A (2001) Mapping protein family interactions: Intramolecular and intermolecular protein family interaction repertoires in the pdb and yeast. In: J. Mol. Biol. 92938
11. (http://www.iue.tuwien.ac.at/phd/fleischmann/node43.html)
12. Garcia-Molina S M H, Rahm E (2002) Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proc. 18th Intl. Conf. on Data Engineering(ICDE), San Jose CA