ALGORITHMS FOR LOCAL STRUCTURAL ALIGNMENT AND STRUCTURAL MOTIF IDENTIFICATION

Sanguthevar Rajasekaran
Computer Science and Engineering Department, University of Connecticut, Storrs, CT, U.S.A

Vamsi K. Kundeti
Computer Science and Engineering Department, University of Connecticut, Storrs, CT, U.S.A

Martin R. Schiller
School of Life Sciences, University of University of Nevada, Las Vegas, NV, U.S.A

A JOHN WILEY & SONS, INC., PUBLICATION

CHAPTER 1

ALGORITHMS FOR LOCAL STRUCTURAL ALIGNMENT AND STRUCTURAL MOTIF IDENTIFICATION

1.1 INTRODUCTION

A protein is characterized by both the amino-acid sequence and the 3-D structure of the underlying atoms. Although it is a common practice of the biologists to use sequence similarity among different proteins to identify any conserved regions during the evolution, it has been proven that the 3-D structures of the proteins are conserved more fundamentally than the sequence during the evolution. Even though two given proteins may not exhibit much of a sequence homology the structural similarity between them might account for similar properties. Proteins with a similar structure might have similar properties [15]. This is the motivation behind the study of the structural alignment problem in a manner similar to that of the sequence alignment problem [5].

Structural Alignment problem has received immense attention in the past few decades especially with the increasing number of tertiary structures available in the Protein Data Bank (PDB) [1]. Given two proteins P_1 and P_2, the problem of structural alignment is to find a highly similar substructure S_{sub} between P_1 and P_2. The number of known protein structures has drastically increased from 10,000 in year 1999 to 45,000 in year 2007. This makes manual structural alignment almost impos-

Algorithms for Local Structural Alignment. By S. Rajasekaran et. al.

sible and hence we need algorithms that can yield almost similar accuracy as manual alignment and are very fast.

Almost all of the existing algorithms perform structural alignment based on the backbone of the protein. For any two given proteins these algorithms try to find the correspondence between the $C\alpha$ atoms on the backbone along with the transformation matrices R (rotational) and T (translational) that will transform one protein to the other minimizing the inter-atomic distance between the corresponding $C\alpha$ atoms (see e.g., [3], [9], [7], [10], and [8]). All these algorithms share a common flavor which consists of two major steps. The first step consists of identifying small structurally similar regions between the two protein back bones. These are known as *Alignment Fragment Pairs* (AFPs). In the next step a subset of these AFPs is identified which essentially forms the alignment between the two structures. In all these algorithms the AFPs are identified by sliding a window of *constant* size along the backbone of the protein. However we feel that using a constant size window in identifying the AFPs is too restrictive especially when we want to improve the accuracy of structural classification. In this work we add an extra degree of freedom in the form of *Variable Length Alignment Fragment Pairs* (VLAFPs) and present a generalized algorithmic framework for structural alignment. Our framework is independent of the scoring schemes used to score the AFP's. Another important fact is that all the existing algorithms only consider the global structural alignment between the two proteins P_1 and P_2 rather than the local alignment. Local structural alignment can be very effective in the identification of structural motifs.

Our contributions are three fold. Firstly we introduce a new idea of using VLAFPs in structural alignment; secondly we provide new scoring schemes based on *Center of Gravity* to identify structurally similar AFPs; and finally we address the problem of identifying *structural motifs* with our algorithm.

The organization of the chapter is as follows. In Section 1.2 we define the Local structural alignment problem. In Section 1.3 we introduce our VLAFP framework. In Section 1.4 we show how we can use Center of Gravity based scores to identify highly structurally similar AFPs. Section 1.5 describes how the VLAFP framework can aid in the identification of structural motifs. Section 1.6 describes how we can classify the proteins based on the VLAFP framework and the Center of Gravity scoring scheme.

1.2 PROBLEM DEFINITION OF LOCAL STRUCTURAL ALIGNMENT

INPUT: Input are two protein structures $P_1 = (a_{1,1}, a_{1,2}, a_{1,3}, \ldots)$ and $P_2 = (b_{1,1}, b_{1,2}, b_{1,3}, \ldots)$ where $a_{i,j}$ represents the j^{th} atom of the i^{th} residue of P_1 and $b_{p,q}$ represents the q^{th} atom in the p^{th} residue of P_2. In fact, $a_{i,j}$ and $b_{p,q}$ have information about the location of the corresponding atoms. For instance, $a_{i,j} = (X_j, Y_j, Z_j)$ and $b_{p,q} = (X_q, Y_q, Z_q)$.

OUTPUT: Define the correspondence between P_1 and P_2 as $C_{1,2} = ((a_{p,q}, b_{r,s}), (a_{m,n}, b_{k,l}), \ldots)$, i.e., specify which atom of P_1 corresponds to which atom of P_2.

The local structural alignment problem is to find a correspondence ($C_{1,2}$) between P_1 and P_2 along with a rotation matrix R and a translation matrix T such that when we apply R and T to one set of coordinates ($a_{p,q}, a_{m,n}, \ldots$) we end up with the other set ($b_{r,s}, b_{k,l}, \ldots$). The optimization version of this problem is to find a correspondence $C_{1,2}$ such that $|C_{1,2}|$ is maximal.

1.3 VARIABLE LENGTH ALIGNMENT FRAGMENT PAIR (VLAFP) ALGORITHM

The existing algorithms for Structural Alignment share a common flavor which consists of two major steps. The first step consists of identifying small structurally similar regions between the two protein back bones. These are known as *Alignment Fragment Pairs (AFPs)* and in the next step a subset of these AFP's is identified. This subset forms the alignment between the two structures. The following sections will give a brief overview of these steps and the details about our *Variable Length Alignment Fragment Pair* Algorithm.

1.3.1 Alignment Fragment Pairs

AFPs in the first step of the existing algorithms are identified by sliding a window W of *constant* size along the protein backbones. Let $B_1 = c^1_{\alpha 1} c^1_{\alpha 2} \ldots c^1_{\alpha n}$ be the backbone of protein P_1 and similarly let $B_2 = c^2_{\alpha 1} c^2_{\alpha 2} \ldots c^2_{\alpha m}$ be the backbone of protein P_2. The backbones B_1 and B_2 are now transformed into two sequences $W_1 = w^1_1 w^1_2 \ldots w^1_{n-k+1}$ and $W_2 = w^2_1 w^2_2 \ldots w^2_{m-k+1}$ where k is the size of the window W and $w^1_i = c^1_{\alpha i} c^1_{\alpha(i+1)} \cdots c^1_{\alpha(i+k-1)}$, $w^2_j = c^2_{\alpha j} c^2_{\alpha(j+1)} \cdots c^2_{\alpha(j+k-1)}$. The *Alignment Fragment Pairs* are defined as $AFP_{(i,j)} = (w^1_i, w^2_j)$ and each of these AFPs is associated with a normalized cost function $COST_{(i,j)} \in [0, 1]$. If $COST_{(p,q)} \leq \epsilon$ (for some appropriate threshold value ϵ) then it indicates that the structure of the $c - \alpha$ atoms in windows w^1_p and w^2_q have very similar structures. In contrast if $COST_{(p,q)} > \epsilon$ then the AFP at (p, q) is not structurally similar. A careful analysis of the algorithms CE [10], DALI [9], TM-Align [22] and PSIST [3] reveals that these algorithms only differ on the cost functions associated with the AFPs. For example DALI and CE use a pairwise $c - \alpha$ distance matrix to compute $COST_{(i,j)}$. CE also combines some extra statistical information into the cost function. PSIST uses the bond angle information among the $c - \alpha$ atoms within each window. TM-Align uses TM-Score [21] as its cost function.

All the existing algorithms work with *constant* size AFPs. Using constant size AFPs is too restrictive in the identification of good *local alignments* among the backbones of the proteins especially in the presence of noise in estimating the coordinates of $c-\alpha$ atoms during X-Ray crystallography. For example, consider an AFP at (i, j) of constant size k. Let $COST_{(p,q)} > \epsilon$. The cost of the same AFP at (i, j) with a different size $k + \gamma$ may be under the threshold of ϵ. Another good example for the need of VLAFPs is the presence of variable length secondary structure elements which consists of *helices* and *sheets*. An important fact to note is that these sec-

ondary structure elements are not always of the same size (in terms of the residues). It is possible that a *helix* structure may consist of 8 residues in one structure and may consist of 12 residues in other structure. Therefore, the use of constant size AFPs may not yield a good alignment. For example if we assume that the size of any AFP is fixed to be 8 then a *helix* structure of 12 residues could be matched only partially either at the start of the helix or at the 4^{th} position thus making the local alignment only partial. However if we allow the AFPs to take variable length such that $2 \leq |W| \leq 8$ then we can clearly produce an alignment of size $4 + 8$ and could match the 12 residue helix exactly. To address such drawbacks with constant size AFPs, we present a much general idea of *Variable Length Alignment Fragment Pairs (VLAFPs)*. The extra degree of freedom is added in the form of an extra variable into the VLAFP cost function which is defined as follows.

$$VCOST(i, j, q) = \begin{cases} \text{Cost of aligning a fragment of size 'q'} \\ \text{at position 'i' in } P_1 \text{ and at} \\ \text{position 'j' in } P_2 \end{cases}$$

$$\begin{array}{ll} VCOST(i, j, q) \in [0, 1] & \text{Normalized VLAFP Cost} \\ k_1 \leq q \leq k_2 & \text{Range of the VLAFP variable 'q'} \end{array}$$

Our core non-iterative dynamic programming framework is independent of any $VCOST$ function. In the later sections we introduce a new $VCOST$ function based on Center of Gravity.

1.3.2 Finding the optimal local alignments based on the VLAFP Cost function

With the definition of the VLAFP cost function in the previous section we now describe our dynamic programming framework for finding the local structural alignments among the structures. The aim of this dynamic programming formulation is to find the *longest contiguous sequence of VLAFPs* such that the cost of each VLAFP is under the threshold ϵ. Details of the dynamic programming formulation follow. We define the dynamic programming subproblem in the form of $VLCS$. Variables i and j refer to the indices of the residues in the protein backbones of corresponding proteins.

$$VLCS(i, j) = \begin{cases} \text{Longest contiguous sequence of VLAFP's} \\ \text{in the backbones of } P_1 \text{ and } P_2 \text{ ending at} \\ \text{the } i^{th} \text{ and } j^{th} \text{ residues, respectively} \end{cases}$$

In our algorithm we need a two step initialization. Since the minimum length of the VLAFP is k_1, pairs of the kind (i, j) with $i < k_1$ and $j < k_1$ are not of interest.

$$\begin{array}{ll} VLCS(i, j) & = \quad 0 \\ 1 \leq (i, j) \leq k_1 - 1 & \end{array}$$

In the second initialization step we consider the first k_1 residues from protein P_1 and check if we can align these residues to any part of the protein P_2 based on the cost

function $VCOST$ as follows. This initialization is similar to the standard sequence alignment initialization.

$$\begin{array}{ll} VLCS(k_1, j) & = \left\{ \begin{array}{ll} k_1 & \textbf{IF } VCOST(k_1, j) \leq \epsilon \\ 0 & \textbf{ELSE} \end{array} \right. \\ 1 \leq j \leq |P_2| & \end{array}$$

The core dynamic programming computation is based on the following equations.

$$\begin{array}{l} QLCS(i, j, q) = \left\{ \begin{array}{l} q + VLCS(i - q, j - q) \\ \quad \textbf{IF } VCOST(i, j, q) \leq \epsilon \\ \\ 0 \\ \textbf{ELSE} \end{array} \right. \\ k_1 + 1 \leq i \leq |P_1|, \\ k_1 + 1 \leq j \leq |P_2| \end{array}$$

$$\begin{array}{l} VLCS(i, j) = \left\{ \begin{array}{l} \textbf{max} \{QLCS(i, j, q)\} \\ k_1 \leq q \leq k_2 \end{array} \right. \\ k_1 + 1 \leq i \leq |P_1|, \\ k_1 \leq j \leq |P_2| \end{array}$$

$$\text{Final Answer Required} = \left\{ \begin{array}{l} \textbf{max} \{VLCS(i, j)\} \\ 1 \leq i \leq |P_1|, \\ 1 \leq j \leq |P_2| \end{array} \right.$$

After the end of the computation we end up with the length of the longest contiguous sequence of VLAFPs such that the cost of each VLAFP is within a threshold ϵ. Along with this we can also compute the exact position in P_1 and P_2 where this sequence starts. So our VLAFP framework has two major steps to compute the local structural alignment as follows.

- Compute the $VCOST(i, j, q)$ function on the backbones of the proteins.
- Compute local structural alignment, which is equivalent to finding a contiguous sequence of VLAFPs in P_1 and P_2 such that the cost of each VLAFP is under a threshold ϵ.

A pseudocode of the core dynamic programming frame work is illustrated in Algorithm 1. Clearly, $VCOST(i, j, q)$ should be such that it takes a value close to 0 for highly structurally similar AFPs and a value close to 1 for structurally dissimilar AFPs. In the next sections we introduce a new $VCOST$ function based on the Center of Gravity that has these desired properties.

1.4 STRUCTURAL ALIGNMENT BASED ON CENTER OF GRAVITY: SACG

One of the ideas that we propose in this work is that of using the sorted distances from the Center of Gravity to identify AFPs. One of the advantages of using the

Center of Gravity is that we can perform structural alignment not only at the $c-\alpha$ level but also including the side chains. Our main goal is to use the algorithms in this section to identify highly structurally similar AFPs and build a $VCOST$ function and then apply the VLAFP algorithm to compute the local structural alignment. Before presenting the details, we provide a summary of how exactly the structure of any protein is described in the PDB file format [1].

1.4.1 Description of protein structure in PDB format

The PDB file for a protein structure is a text description of the 3D-coordinates of the atoms/residues in the protein. The file consists of a linear list L_{pdb} of atoms that are a part of the protein and the corresponding 3-D coordinates of each atom. $L_{pdb} = (a_{1,1}, a_{1,2}, a_{1,3}, \ldots, a_{i,j} \ldots)$, where $a_{i,j}$ is the j^{th} atom in residue i. It is noteworthy that the list L_{pdb} is partially ordered with respect to the residue numbers, i.e., $a_{p,q} < a_{m,n} \iff (p < m)$. Although there is an ordering among the residues, the atoms within a reside may not follow any order. If ${L^1}_{pdb}$ and ${L^2}_{pdb}$ are two PDB structure instances of the same protein, the ordering of the atoms within each residue may be different (though the residues themselves will be in the same order). As an example the atoms in the 1^{st} residue of ${L^1}_{pdb}$ may be ordered as $(a_{1,2}, a_{1,1}, a_{1,5}, a_{1,4}, a_{1,3}, \ldots)$ but the atoms in the same residue of ${L^2}_{pdb}$ may be ordered as $(a_{1,5}, a_{1,1}, a_{1,2}, a_{1,4}, a_{1,3}, \ldots)$. This variation is mainly due to different frames of reference during X-ray crystallography. The variation of the ordering of the atoms within the same residue makes structural alignment algorithms which consider the sidechain conformations non-trivial. In the next sections we will see how our algorithms overcome this ordering issue when side-chain conformations are considered.

1.4.2 Related work

The problem of checking if two point sets (in 2D or 3D) are rigidly transformable from one to the other is a well studied problem in Computational Geometry. This problem is known as *Geometric Congruence*. Several algorithms for exact geometric congruence were given in [16], [17], [18], and [20]. All of these algorithms solve the exact geometric congruence in $O(n \log n)$ time. There is also a much general version of the geometric congruence known as the ϵ-congruence. In this version, we are required to determine if two given point sets of the same cardinality are rigidly transformable from one to the other within a tolerance of ϵ. The ϵ-congruence problem can be solved in time $O(n^8)$ deterministically (see e.g., [18]). The problem of ϵ-congruence is closely related to the substructure identification problem but a run time of $O(n^8)$ may not be practical. In the literature of structural alignment of proteins several *iterative* dynamic programming based algorithms have been proposed (see e.g., [8] and [19]). However there are several issues on the convergence of these algorithms. In these algorithms the correspondence between the atoms is changed in every iteration and hence it is possible for these algorithms to never converge to an optimal solution. In our algorithm we first find the sub structures that are highly sim-

ilar and we will not change this correspondence throughout the algorithm and finally use the VLAFP framework in Section 1.3 to find the longest common substructure among the protein structures.

Algorithm 1: Core VLAFP Algorithm to compute local structural alignments

```
INPUT  : VCOST,|P1|,|P2|
OUTPUT: Length of optimal local alignment and its location
Initialize VLCS
MaxLen = 0
for i = k1 to |P1| do
    for j = k1 to |P2| do
        CurrentMax = 0
        for q = k1 to k2 do
            if VCOST(i, j, q) ≤ ε then
                if VLCS(i − q, j − q) + q > CurrentMax then
                    CurrentMax = VLCS(i − q, j − q) + q
                end
            end
        end
        VCLS(i, j) = CurrentMax
        if CurrentMax > MaxLen then
            MaxLen = CurrentMax
            StartPosition1 = i − CurrentMax + 1
            StartPosition2 = j − CurrentMax + 1
        end
    end
end
return (MaxLen,StartPosition1,StartPosition2)
```

1.4.3 Center of Gravity based algorithm

If P_1 and P_2 are two given proteins with n residues each, a simple algorithm to find the correspondence between P_1 and P_2 will take $O(n!)$ time. The key idea behind our structural alignment algorithm based on Center of Gravity is based on the following theorem.

Theorem 1 *Given two 3-D pointsets S_1 and S_2 each of size n, with $S_1 = \{({x^1}_1, {y^1}_1, {z^1}_1)$, $({x^1}_2, {y^1}_2, {z^1}_2), \ldots\}$ and $S_2 = \{({x^2}_1, {y^2}_1, {z^2}_1), ({x^2}_2, {y^2}_2, {z^2}_2), \ldots\}$, we can check if S_1 is a rigid transformation of S_2 in $O(n \log n)$ time and $O(n)$ space.*

This directly follows from the Atkinson's algorithm (exact geometric congruence) (see [16]). The proof is based on a very simple fact that the relative position (from any of the points in the point set) of the center of gravity of a set of 3-D points remains

unchanged when these 3-D points are transformed by any rigid transformation. The Center of Gravity (CG) for a 3-D point set is defined as follows.

$$S_1 = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\};$$
$$X_{CG} = \frac{\sum_{i=1}^{n} x_i}{n}; Y_{CG} = \frac{\sum_{i=1}^{n} y_i}{n}; Z_{CG} = \frac{\sum_{i=1}^{n} z_i}{n}.$$

If the relative position of the CG with respect to any of the points in the point set changes due to a transformation then the transformation is not rigid. We use this fact and compute the Euclidean distance of each point from (X_{CG}, Y_{CG}, Z_{CG}). Let this distance for the ith point be ${d_i}^{cg}$.

$${d_i}^{cg} = \sqrt{(X_{CG} - x_i)^2 + (Y_{CG} - y_i)^2 + (Z_{CG} - z_i)^2}.$$

Once we compute ${d_i}^{cg}$ we sort these distances and create a distance vector ${V_1}^{cg}$ for the point set S_1. Similarly we create a vector ${V_2}^{cg}$ for S_2 and compare if ${V_1}^{cg}$ and ${V_2}^{cg}$ are the same. If the distance vectors are the same we find the convex hulls of the point sets and check if the hulls are the same. This can be done in $O(n \log n)$ time and hence the entire algorithm runs in $O(n \log n)$ time.

Theorem1 readily yields an algorithm for structural alignment. Although in Theorem1 we mentioned that we also need to find the convex hulls and check if the hull are same, in practice just using the sorted distance vectors from the center of gravity seems to be sufficient (see Algorithm2).

Algorithm 2: algorithm to check if pointsets S_1 and S_2 are rigidly transformable

INPUT : Pointsets S_1 , S_2
OUTPUT: True if S_1 can be transformed (rigidly) to S_2
(X^1, Y^1, Z^1) = COMPUTE_CG(S_1);
(X^2, Y^2, Z^2) = COMPUTE_CG(S_2);
for $i \leftarrow 1$ **to** n **do**
  $V^1[i] = \sqrt{(X^1 - {x^1}_i)^2 + (Y^1 - {y^1}_i)^2 + (Z^1 - {z^1}_i)^2}$;
  $V^2[i] = \sqrt{(X^2 - {x^2}_i)^2 + (Y^2 - {y^2}_i)^2 + (Z^2 - {z^2}_i)^2}$;
end
SORT(V^1);
SORT(V^2);
if V^1==V^2 **then**
  return true;
else
  return false;
end

1.4.4 Extending Theorem 1 for atomic coordinates in protein structure

Algorithm 2 returns true if there exists an exact rigid transformation (R, T) which when applied to the point set S_1 will give S_2 or vice-versa. But in the context of pro-

tein structures where there is a considerable noise while measuring the coordinates during X-Ray crystallography, **exact** rigid transformations may not be meaningful. We need an algorithm that can take the coordinates of the protein sub-structures and determine if one sub-structure can be approximately transformed into another sub-structure using some rigid transformation. Keeping this in mind we extend the exact version of the algorithm based on Theorem 1. We define weighted distance ($W_{i,j}$) between two sorted vectors V_i and V_j (each of length n) as follows.

$$W_{i,j} = \sum_{k=1}^{n} (n-k) * \sqrt{(V_i[k] - V_j[k])^2}$$

We also define an approximation threshold ϵ, whose value is proportional to n. The typical value of ϵ is 1.8 for $n = 20$. We have determined the value of ϵ from several experimental runs of our program. Algorithm 3 incorporates these definitions and it can detect if two given atomic coordinate sets (from protein structures) P_1 and P_2 can be approximately transformed from one to the other, with an error of ϵ. Algorithm 3 is much faster and simpler than the $O(n^8)$ algorithm of [18]. As our experimental data indicate, the accuracy of Algorithm 3 is very good. Algorithm 3 can be very effective in checking if two sets of atoms have the same structure but our main intention is to compute the local structural alignment among the protein backbones. This is where we seek the help of our VLAFP framework presented in Section 1.3. We use Algorithm 3 to build a cost function $VCOST(i, j, q)$ and apply the VLAFP algorithm on top of this cost function thus obtaining the required local structural alignment. Section 1.4.5 gives more details on building this cost function.

Algorithm 3: algorithm to check if atomic coordinates P_1 and P_2 are approximately transformable

INPUT : Pointsets P_1 and P_2; ϵ
OUTPUT: True if P_1 can be transformed (approx) to P_2
(X^1, Y^1, Z^1) = COMPUTE_CG(P_1);
(X^2, Y^2, Z^2) = COMPUTE_CG(P_2);
for $i \leftarrow 1$ **to** n **do**
  $V^1[i] = \sqrt{(X^1 - {x^1}_i)^2 + (Y^1 - {y^1}_i)^2 + (Z^1 - {z^1}_i)^2}$;
  $V^2[i] = \sqrt{(X^2 - {x^2}_i)^2 + (Y^2 - {y^2}_i)^2 + (Z^2 - {z^2}_i)^2}$;
end
SORT(V^1);
SORT(V^2);
$W_{1,2} = \sum_{k=1}^{n} (n-k) * \sqrt{(V^1[k] - V^2[k])^2}$;
if $W_{1,2} \leq \epsilon$ **then**
  return true;
else
  return false;
end

1.4.5 Building VCOST(i,j,q) function based on Center of Gravity

We define VCOST(i, j, q) for a fragment pair of length q at indices i and j in the protein backbones of P_1 and P_2 as follows. Let $W1_i^q = \{a_i^1, a_{i+1}^1, \ldots, a_{i+q-1}^1\}$ be the atoms in the fragment corresponding to protein P_1 at index i in the backbone. Similarly we can define $W2_j^q$ corresponding to protein P_2. The pair $(W1_i^q, W2_j^q)$ is an AFP of size q. Let $(CG1_x, CG1_y, CG1_z)$ be the center of gravity for the set of atoms in $W1_i^q$ and $(CG2_x, CG2_y, CG2_z)$ be the center of gravity for the set of atoms in $W2_j^q$. The cost function VCOST(i, j, q) is defined as follows.

$$d_k^1 = (x_k^1 - CG1_x)^2 + (y_k^1 - CG1_y)^2 + (z_k^1 - CG1_z)^2, 1 \leq k \leq q$$

$$d_k^2 = (x_k^2 - CG1_x)^2 + (y_k^2 - CG1_y)^2 + (z_k^2 - CG1_z)^2, 1 \leq k \leq q$$

$$V^1 = \text{Sorted distance vector of } d_k^1, 1 \leq k \leq q$$

$$V^2 = \text{Sorted distance vector of } d_k^2, 1 \leq k \leq q$$

$$DAFP(i, j, q) = \sum_{k=1}^{q} (V^1[k] - V^2[k])^2$$

$$VCOST(i, j, q) = \frac{DAFP(i, j, q)}{\sqrt{DAFP(i, j, q)^2 + q^2}}$$

Once we have the normalized cost function VCOST we can then apply the VLAFP dynamic programming framework (see Algorithm 1) to compute the local structural alignment. Figure 1.1 illustrates the outcome of the VLAFP local alignment between 1C2N and 1COT pdb structures based on the Center of Gravity VCOST function. Also Figure 1.2 displays the local alignment between 1HIJ and IITI. The local alignments are marked in red. We refer to the combination of the Center of Gravity based VCOST function with VLAFP framework as Structural Alignment based on Center of Gravity (SACG).

1.5 SEARCHING STRUCTURAL MOTIFS

Finding structural patterns among the protein structures is of immense interest for biologists who often look for structural patterns including the side-chain conformations [14]. None of the existing algorithms addresses this issue of identifying structurally similar patterns including the side-chain conformations. Biologists often want to search for a part of the protein structure (sub-structure) in the existing proteins in the PDB. Finding similar sub-structures including the side chains is a much difficult problem because of the ordering of the atoms on the side chains is not necessarily fixed. The ordering of side chain atoms in the pdb file for the same structure can vary from experiment to experiment.

Our algorithms to identify similar sub-structures can easily address this ordering issue since we use the sorted distances from the center of gravity as a signature to identify the sub-structure. The problem with different ordering of the atoms will not

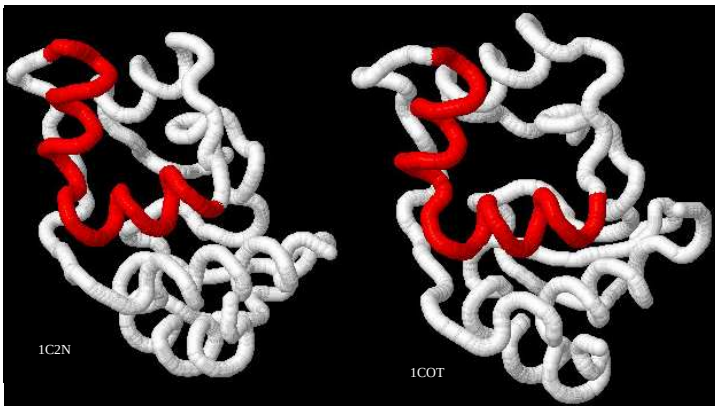

Figure 1.1 Local structural alignment between 1C2N and 1COT using our SACG algorithm

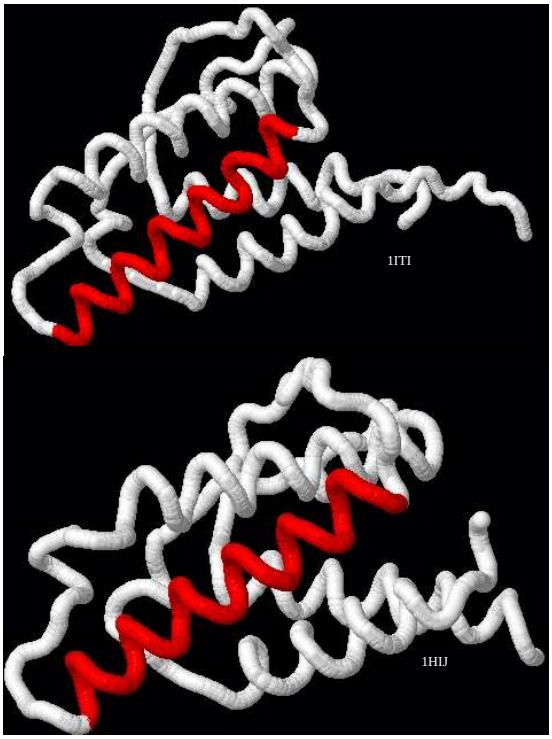

Figure 1.2 Local structural alignment between 1HIJ and IITI using our SACG algorithm

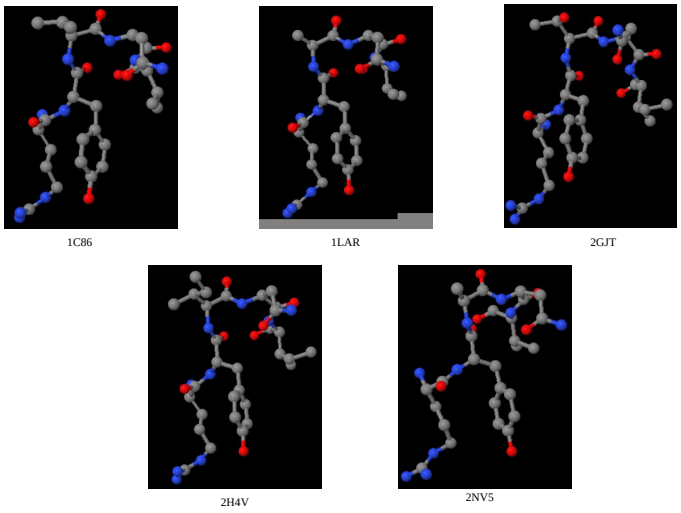

Figure 1.3 Tyrosine phosphorylated substrates (XYXNX motifs) identifed by Algorithm 3 in **1C86, 1LAR, 2H4V, 2GJT, and 2NV5**

affect our algorithm. Algorithm 3 can be readily used to identify structural motifs including the side chains. Biologists can supply the list of 3D co-ordinates of the atoms (in any order) to Algorithm 3. The algorithm then creates a sorted distance vector (d_{sig}) for that set of 3D co-ordinates and search the entire PDB database to identify the regions which have signatures similar to d_{sig}. All the regions that have a signature close to d_{sig} can be potential structural motifs. Figure 1.3 shows the real substructures (Tyrosine phosphorylated substrates [14]) found by Algorithm 3. We have got these regions by taking a subset of atomic coordinates from a known YXN motif and searched the entire PDB database for regions having a signature similar to the atoms in YXN motif and identified the regions in 1C86, 1LAR, 2H4V, 2GJT and 2NV5 as shown in Figure 1.3.

1.6 USING SACG ALGORITHM FOR CLASSIFICATION OF NEW PROTEIN STRUCTURES

An important problem in structural alignment is to accurately predict protein structures from the PDB that are close to a newly discovered protein structure P_{new}. This can be easily addressed by computing all the pairwise local structural alignments between the new protein P_{new} and existing proteins in the PDB database, and ranking all of the alignments according to length of the local structural alignment and the normalized costs. We have used our VLAFP framework to perform the local structural alignments and ranked the proteins based on length (number of residues) and the cost($VCOST$) of the alignment.

1.7 EXPERIMENTAL RESULTS

We provide two sets of experimental data. The first set covers all the experimental data related to the classification accuracy of SACG Algorithm and the second set contains the experimental data related to structural motif search. The algorithm was implemented in C and the entire source code and all datasets/results can be downloaded from `http://trinity.engr.uconn.edu/~vamsik/VAFP_ALGO/`. The program was run on a 1GB (RAM), 1.3GHZ intel processor linux machine.

1.8 ACCURACY RESULTS

Our dataset is the same standard dataset used by PSIST [3] and other algorithms like ProgRESS and geometric hashing. Please see [3] for additional details of the dataset. The dataset consisted of 181 superfamilies and each of the superfamilies has at least 10 protein structures. The proteins are chosen in such a way that there is less than 30% of sequence homology between any two proteins from the same superfamily. The superfamilies are based on SCOP [2] classification. So our database consists of around 2000 proteins. The query sample is a sample of 176 proteins selected randomly from these 2000 proteins. PSIST used the same sample size. Once the sample is selected we run our algorithm (SACG) and PSIST and classify the results based on the most frequently occuring superfamily and class in the top-20 ranked proteins. The results indicate that our algorithm achieves an average accuracy of 84.09% (super family) and 86.93 %(class). See Table 1.1 for additional details. Table 1.2 and Table 1.3 show the results of the top ranked proteins for query proteins 1c2n, 1hsm using our algorithm.

Algorithm	Correct (SF)	Correct (Class)	Top-K	Accuracy (SF)	Accuracy (Class)
PSIST	120	129	K=20	68.18%	73.29%
CG_ALGO	148	153	K=20	84.09%	86.93%

Table 1.1 Accuracy comparison between PSIST and SACG

Now we illustrate practical results in identifying a functional structural motif (Tyrosine phosphorylated substrate) in some of the PDB structures. We started with 1C86 that has a functional motif between atoms (348 and 392) (please refer to the PDB file of protein 1C86). We make the atom list from 348 to 392 in 1C86 as S_1 and apply our Algorithm 3. We found out that 1LAR, 2GJT, 2NV5 and 2H4V (see Table 1.4) have highly similar substructures (Tyrosine phosphorylated substrate) to the one in 1C86 between atoms 348 and 392. Please see the table below for the actual locations of this in the PDB files. Please refer to Figure 1.3 for 3D-Visualization of these sub structures.

Match	Length	Cost	pdb-id	Super Family (**sf**)	Class (**cl**)
c	44	24.13	pdb1mbj-	46689	46456
c	34	21.99	pdb2bby-	46785	46456
c	40	26.39	pdb1jtb-	47699	46456
c	46	31.96	pdb1hsn-	47095	46456
c	36	25.84	pdb1nhm-	47095	46456
c	32	23.05	pdb1mbe-	46689	46456
	37	26.80	pdb2cjo-	54292	53931
c	35	25.40	pdb1uxd-	47413	46456
c	36	26.39	pdb1aab-	47095	46456
c	39	28.69	pdb1mbk-	46689	46456
	40	29.54	pdb1eot-	54117	53931
c	37	27.86	pdb1etd-	46785	46456
c	46	35.43	pdb1nhn-	47095	46456
	47	36.75	pdb1e09-A	55961	53931
c	47	38.05	pdb2new-	48695	46456
	45	36.70	pdb1bt7-	50494	48724
c	50	41.15	pdb1gjt-A	46997	46456
	32	26.90	pdb4ull-	50203	48724
c	37	31.49	pdb1a2i-	48695	46456
c	47	40.16	pdb1wjd-B	46919	46456
c	47	40.57	pdb1wjd-A	46919	46456
+ve **cl** classification (46456) occurs 16 times, -ve **sf** classification (47095) occurs 4 times					

Table 1.2 Top scored proteins for query pdb1c2n **sf**(46626) **cl**(46456) with SACG. '*c*' indicates that the class of query matches the class of the corresponding protein.

1.9 CONCLUSION

In this chapter we have introduced a new idea of using Variable Length Alignment Fragment Pairs in performing local structural alignment among the proteins. We also showed how to use the VLAFP framework to classify proteins and search for

Match	Length	Cost	pdb-id	Super Family(**sf**)	Class (**cl**)
*c*sf	168	72.65	pdb1hsn-	47095	46456
c	33	19.64	pdb2bby-	46785	46456
	31	19.40	pdb2cjo-	54292	53931
*c*sf	145	95.48	pdb1nhm-	47095	46456
c	38	26.18	pdb1ba5-	46689	46456
c	38	28.57	pdb1edj-	46997	46456
c	39	30.66	pdb1wtu-B	47729	46456
*c*sf	127	102.17	pdb1nhn-	47095	46456
*c*sf	107	86.43	pdb1hmf-	47095	46456
*c*sf	110	89.27	pdb1hme-	47095	46456
	33	27.14	pdb1bc6-	54862	53931
	35	29.19	pdb1grx-	52833	51349
	42	35.98	pdb2cjn-	54292	53931
c	37	31.73	pdb1tnt-	46785	46456
	41	35.81	pdb1mit-	54654	53931
*c*sf	52	46.02	pdb1hma-	47095	46456
c	36	32.18	pdb1bqv-	47769	46456
c	46	41.35	pdb1hue-A	47729	46456
c	42	38.25	pdb1mbg-	46689	46456
c	35	31.92	pdb1bdc-	46997	46456
	33	30.16	pdb1svq-	55753	53931
+ve **cl** classification (46456) occurs 15 times, +ve **sf** classification (47095) occurs 6 times					

Table 1.3 Top scored proteins for query pdb1hsm **sf**(47095) **cl**(46456) using SACG. '*c*sf' indicates that both class and super family of the query matches with the corresponding protein.

Protein(PDB-ID)	Region(start-end)
1LAR	Residue 383 to 425
2GJT	Residue 427 to 472
2NV5	Residue 430 to 470
2H4V	Residue 440 to 486

Table 1.4 Regions having Tyrosine phosphorylated substrate found by Algorithm 3

structural motifs. In addition, we have introduced a new scoring function based on Center of Gravity. Experimental results indicate that using the VLAFP framework can produce better local structural alignments compared to using constant size AFPs.

1.10 ACKNOWLEDGEMENTS

This research has been supported in part by the NIH Grant 1R01GM079689-01A1 and NSF Grant ITR-0326155.

REFERENCES

1. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, The protein data bank, *Nucleic Acids Research* 28, 2000, pp. 235-242.
2. A.G. Murzin, S.E. Brenner, T. Hubbard, C. Chothia, SCOP: a structural classification of proteins database for the investigation of sequences and structures, *J. Mol. Biol.* 247, 1995, 536-540.
3. F. Gao and M.J. Zaki, PSIST: indexing protein structures using suffix trees, *Proc. IEEE Computational Systems Bioinformatics Conference*, 2005.
4. C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton, CATH - a hierarchic classification of protein domain structures, *Structure* 5(8), 1997, pp. 1093-1108.
5. S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, Basic local alignment search tool, *J Mol Biol* 215(3), 1990, pp. 403-410.
6. A.Singh and D.Brutlag, Protein structure alignment: a comparison of methods, http://cmgm.stanford.edu/brutlag/Abstracts/singh00.html
7. G.H. Cohen, ALIGN: A program to superimpose protein coordinates, accounting for insertions and deletions, *J. Appl. Crystallogr.*, 1997.
8. M. Gerstein and M. Levitt, Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures, *Proc. Fourth Int. Conf. on Intell. Sys. for Mol. Biol.*, Menlo Park, CA, AAAI Press, 1996, pp. 59-67.

9. L. Holm and C. Sander, Protein structure comparison by alignment of distance matrices, *J. Mol. Biol.* 233, 1993, pp. 123-138.

10. I.N. Shindyalov and P.E. Bourne, Protein structure alignment by incremental combinatorial extension (CE) of the optimal path, *Protein Engineering* 11(9), 1998, pp. 739-747.

11. L. Pauling, R.B. Corey, and H.R. Branson, The structure of proteins; two hydrogen-bonded helical configurations of the polypeptide chain, *Proc Natl Acad Sci USA* 37(4), 1951, pp. 205-211.

12. T.F. Smith and M.S. Waterman, Identification of Common Molecular Subsequences, *Journal of Molecular Biology* 147, 1981, pp. 195-197.

13. S. Needleman and C. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J. Mol. Biol.* 48(3), 1970, pp. 443-53.

14. K.H. Kirsch, M. Kensinger, and H. Hanafusa, A p130Cas tyrosine phosphorylated substrate domain decoy disrupts v-Crk signaling, *BMC Cell. Biology*, 2002.

15. M.J. Betts, G. Agarwal, and R.B. Russell, Exon structure conservation despite low sequence similarity: a relic of dramatic events in evolution?, *EMBO J.* 20(19), 2001, pp. 5354-5360.

16. M.D. Atkinson, An optimal algorithm for geometric congruence, *J. Algorithms* 8, 1987, pp. 159-172.

17. M.J. Atallah, Checking similarity of planar figures, *Intern. J. Comput. Inform. Sci.* 13, 1984, pp. 279-290.

18. H. Alt, K. Melhorn, H. Wagener, and E. Welzl, Congruence, similarity, and symmetries of geometric objects, *Discrete and Computational Geometry* 3, 1988, pp. 237-256.

19. T. Akutsu, Protein structure alignment using dynamic programming and iterative improvement, *IEICE Trans. Inf. Syst.* E78-D(0), 1996.

20. T. Akutsu, Algorithms for determining the geometrical congruity in tow and three dimensions, *Proc. 3rd. International Symposium on Algorithms and Computation*, Nagoya, Japan, December 1992.

21. Y. Zhang, J. Skolnick, Scoring function for automated assessment of protein structure template quality, *Proteins*, 2004 57: 702-710.

22. Y. Zhang, J. Skolnick, TM-align: A protein structure alignment algorithm based on TM-score , *Nucleic Acids Research*, 2005 33: 2302-2309.