Efficient Processing of 3D Protein Structure Similarity Queries

PhD Confirmation Report

Zi Helen Huang

Supervisors: Prof. Xiaofang Zhou and Prof. Peter Bruza (DSTC)

School of Information Technology and Electrical Engineering
University of Queensland

May 16, 2005
5.2 Future Work ......................................................... 36
  5.2.1 Developing More Effective Filtering Algorithms ........ 36
  5.2.2 Large Scale Experiments to Determine an Optimal Size \( k \)
of Seed Motifs ................................................... 37
  5.2.3 Application and Optimization of High-Dimensional In-
dexing Methods and Dimensionality Reduction Techniques 37
  5.2.4 More Effective Evaluation System .......................... 37
  5.3 Timetable and Milestones ....................................... 37
  5.4 Required Resource .............................................. 38

Acknowledgements .................................................. 38
References ............................................................. 38
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Structure of amino acid</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Protein structure[46]. The polypeptide chains of proteins have a backbone of constant structure and sidechains that vary in sequence. Here $R_{i-1}, R_i$ and $R_{i+1}$ represent sidechains. Those repeating units in protein are called residues.</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Hierarchical structure of proteins[45].</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Cartesian coordinate[11].</td>
<td>7</td>
</tr>
<tr>
<td>2.5</td>
<td>Protein structure file in mmCIF format downloaded from PDB.</td>
<td>8</td>
</tr>
<tr>
<td>2.6</td>
<td>Representation of secondary structure elements as vectors[43].</td>
<td>9</td>
</tr>
<tr>
<td>2.7</td>
<td>Structural environment scheme.</td>
<td>9</td>
</tr>
<tr>
<td>2.8</td>
<td>Product graph $G$ for illustrating problem transformation.</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Protein structure as 3D vectors.</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Spatial relationship between two vectors. Four internal distances are denoted as $d_{\alpha\alpha}, d_{\beta\beta}, d_{\alpha\beta}, \text{and} \ d_{\beta\alpha}$.</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>An example of motif. Each vector represents a $C_\alpha$-$C_\beta$ bond of a residue. The diameter is $\varepsilon$. The dashed line shows the $\alpha-\alpha$ distance ($d_{\alpha\alpha}$) between two vectors.</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Construction of motif signature.</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>The framework of protein structure matching.</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>An example of re-ordering the vectors in a sub-motif.</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Algorithm1: Seed-Motif creation.</td>
<td>26</td>
</tr>
<tr>
<td>4.4</td>
<td>An example of candidate derivation.</td>
<td>29</td>
</tr>
<tr>
<td>4.5</td>
<td>Algorithm2: Matching expansion.</td>
<td>30</td>
</tr>
<tr>
<td>4.6</td>
<td>Number of k-size sub-motif.</td>
<td>31</td>
</tr>
</tbody>
</table>
List of Tables

3.1 A Table of Notations. ........................................... 21
4.1 Statistics of test data ............................................. 30
4.2 Average number of k-sized sub-motifs in a query protein .... 31
4.3 CPU time(seconds) for computing matching k-sized sub-motifs using pairwise comparison ......................................... 32
4.4 CPU time(seconds) for computing matching k-sized sub-motifs using Rule 1 .................................................. 33
4.5 CPU time(seconds) for computing matching k-sized sub-motifs using Rule 1+Rule 3 ............................................. 33
4.6 Recall of Rule 1 ..................................................... 33
4.7 CPU time (seconds) for matching larger sized sub-motifs .... 33
5.1 Timetable for future research. .................................... 38
Abstract

The goal of this PhD work is to develop a solution to efficient 3D protein structure database searching in order to find matching sub-structures (motifs) for a query protein against proteins in the database. We present a vector representation for protein structures, where a motif is formed by the vectors within a region. We propose a structure representation model called motif signature. For a $k$-sized motif, its motif signature consists of $7k - 10$ distances which can determine the motif’s spatial structure. A similarity measure of two motifs is then defined based on their motif signatures. To achieve fast motif matching in large protein databases, a match-and-expand strategy is proposed. Given a query motif, a set of small $k$-sized matching motifs, called candidate motifs, is generated in the match stage. The candidate motifs are further filtered by enlarging $k$ in the expand stage. Our initial experimental results demonstrate encouraging performance. The future work for the remeding two years of my research will include further enhancement of the current searching strategy via developing more effective filtering algorithms; larger scale experiments to determine appropriate size of $k$ for the match stage; design of effective evaluation system; test and optimization some existing high-dimensional indexing and dimensionality reduction approaches. A timetable to complete these future work in the next two years is proposed.
Chapter 1

Introduction

Information science has been applied to computational biology, resulting in a new field called Bioinformatics, which investigates “the collection, archiving, organization and interpretation of biological data” [52]. Over the last two decades, the major research efforts in bioinformatics have evolved from generating biological databases at the beginning to finding evolutionary and functional relationships via initially comparing of nucleotide and amino acid sequences, and more recently via comparing of three-dimensional structures.

Many examples show that, the evolutionary relationship may be detectable by sequence methods alone when the sequence identity (i.e., the degree of similarity between sequences) is between 20% to 30%. However, when the sequence identity is below 20%, only structure analysis can reveal the evolutionary relationship between two proteins, which is yet hidden at the sequence level [12]. “Significant structural similarity is common, even among proteins that do not share any sequence similarity or evolutionary relationship”[49]. Thus, the three-dimensional structures of proteins are more conservative than sequences in evolutionary relationship.

It has been shown that “Protein properties are a direct consequence of the protein’s unique three-dimensional structure” [48]. Certain structural regions of a protein often perform some specific functions. Having one or more matching (similar) regions in three-dimensional structures has been considered as an essential condition for the existence of potential interaction between two proteins (e.g., for a designed drug to be effective on a protein). Protein structure analysis therefore is becoming the main stream and lies in the heart of modern bioinformatics.

As three-dimensional structures for proteins are large and complex, protein structure matching is always computationally very expensive. According to the
keynote speech given by Professor Haruki Nakamura in the Third Asia-Pacific Bioinformatic Conference, searching a database of 1400 proteins to find all the proteins which have a similar surface with a query protein took several days on average on a supercomputer. Note that, PDB (Protein Data Bank), a highly used worldwide archive of structural data of biological macromolecules, now contains nearly 30,000 proteins.

The goal of our research is to address efficiency issues in three-dimensional protein structural similarity query processing from data modeling, database indexing and query processing strategy aspects.

In our work, the protein database \( P_{ALL} \) is built up based on PDB, which arranges a protein on an imaginary Cartesian coordinate frame and assigns \((x,y,z)\) coordinate to each atom. Not all the atoms are used in our protein representation. We simply represent a protein structure as a collection of vectors which are \( C_\alpha-C_\beta \) bonds of amino acids in the protein. The size of a protein structure is defined as the number of vectors in the collection. Given a query protein \( P_Q \), the general problem we investigate is to find all the proteins from \( P_{ALL} \) such that the resultant proteins have one or more similar sub-structures of size \( l \) with \( P_Q \), where \( l \) is a parameter specified by the biologist. Two sub-structures are considered similar to each other, if they can be “fitted” together very well when they are placed on top of each other, allowing rotation and translation of the structures as a whole[38]. Our approach is to decompose proteins to \( k \)-sized sub-structures named motifs. \( k \) is an experimental parameter, which may vary from 3 to 20. The protein similarity searching is then performed on motif level.

To solve this research problem, solutions to the following issues must be addressed:

1) compact representation of motifs;
2) fast searching strategy on the compact representations;

To address the first issue, we propose a novel motif representation model called motif signature. Given a \( k \)-sized motif, its motif signature characterizes the spatial relationship among its constituting vectors via \((7k-10)\) internal inter-atom distances. The similarity between two motifs can then be measured by comparing their motif signatures.

To fast search similar motifs of size \( l \) in a large database (issue 2), a match-and-expand strategy is proposed to operationalize a scalable model. In the match

\[ ^1 C_\alpha, C_\beta \text{ are two kinds of carbon atoms. Each amino acid contains a } C_\alpha-C_\beta \text{ bond, except amino acid GLY.} \]
stage, the matching sub-motifs of a smaller size $k$ ($k < l$) are identified as candidates. A rule-based filtering approach is used in this stage to quickly prune away unmatched results. The real matching is done only on a very small portion of data.

The expand stage extends the $k$-sized matches to larger sizes until reaching $l$.

The rest of the report is organized as follows. Chapter 2 gives a brief introduction to protein structure analysis and some related work. We present the problem formulation including a model of motif signature and the similarity measure in Chapter 3. A match-and-expand searching strategy is proposed and detailed in Chapter 4. The initial experimental results are also reported in the same chapter. Chapter 5 summarizes a list of crucial research issues involved in the problem and gives my future research plan and timetable.
Chapter 2

Literature Review

2.1 Preliminaries - Introduction to Protein Structure

Biochemically, proteins play variable roles in life processes. The functions of proteins include structure, movement, catalysis, transport, regulation of cellular processes, and response to stimuli[13].

Chemically, a protein is a large molecule composed of one or more chains of amino acids in specific orders. Twenty types of basic amino acids have been identified. They all have similar formations but differ in chemical structures of the sidechains (denoted as $R$). Each amino acid contains a central carbon atom $C_\alpha$ to which a sidechain, an $N-H$ group, a $C'-O$ group and a hydrogen atom $H$ are attached. The sidechain is connected to $C_\alpha$ via atom $C_\beta$ inside the sidechain\(^1\). The mainchain (or backbone) atoms of an amino acid contain a atom $C_\alpha$, an $N-H$ group and $C'-O$ group[13]. This unit is also called residue. A polypeptide chain is formed by residues which linked by peptide bonds. A protein is constructed by one or more polypeptide chains(Fig2.1, Fig2.2).

Protein structures have been determined by the physical method via X-ray Crystallography or 2D NMR. They are described on four different levels(Fig2.3):

1. Primary Structure
   The primary structure is represented as the amino acid sequence which records the succession of sidechains[46]. It is the sequence of the sidechains

\(^1\)Amino acid Gly is an exception. Its sidechain only contains a hydrogen atom.
2. Secondary Structure
The secondary structure of a protein can be thought of as the local conformation of the polypeptide chain, independent of the rest of the protein. The secondary structure have two important elements \( \alpha \)-helix and \( \beta \)-strand[59] which are constructed by a series of residues. The structures of both elements are rigid and stable.

3. Tertiary Structure
As secondary structure is a local three-dimensional structure, the tertiary structure can be thought of as a global three-dimensional structure. It is defined as assembly and interactions of the helices and strands. Thus, at the level of tertiary structure, the side chains play a much more active role in creating the final structure[59].
4. Quaternary Structure
Since many proteins are multi-polypeptide chains, quaternary structure shows the associations of multiple polypeptide chains while tertiary structure describes the structural organization of a single polypeptide chain.

2.1.1 3D Protein Structures Representations
The first step of all methods to do structure analysis is to represent the proteins. Four approaches will be discussed in the following.

Atoms Collection
Protein are composed by thousands of atoms. A simple way to represent 3D protein structure is to arrange protein on an imaginary Cartesian coordinate frame (Fig 2.4) and assign \((x, y, z)\) coordinates to each atom. PDB(Protein Data Bank) uses this method (Fig 2.5).

Distance Matrix
Another representation is distance matrix of all pairwise distances between atoms in a protein. A typical one is \(C_\alpha C_\alpha\) distances matrix, which is used in in program DALI[34][36]. All the internal distances between \(C_\alpha\) atoms of a protein are assigned to this matrix. It considers that if two \(C_\alpha\) atoms are in contact in one protein, the \(C_\alpha\) atoms aligned with these two in a related protein are also likely to be
in contact. In another word, if the distance between two \( C_\alpha \) atoms \((A_i \text{ and } A_j)\) of protein A is similar to the distance between two \( C_\alpha \) atoms \((B_i \text{ and } B_j)\) of protein B, atoms \( A_i \) and \( A_j \) could be mapped to the atoms \( B_i \) and \( B_j \). Algorithm CE(The Combinatorial Extension)[62] is another good example for using this approach.

**Secondary Structure**

The atoms of each secondary structural element in each protein can be replaced by a vector of position, length, and direction determined by the position of the \( C_\alpha \) atoms along the element(Fig2.6).

VAST[28] and SARF[5] are the examples of using this method. The VAST program represents each element of secondary structure as a vector. The length and direction of one vector is determined by the length and direction of the corresponding secondary structure. The information of secondary structure’s type and the number of residues, etc is also encoded. This method makes it easy to compare protein structures on the secondary structure level. The assumption is that if two vectors representing two secondary structures are similar, the internal structure within secondary structures are similar. The comparison of structures is then computationally simplified.

**Structural Environment Scheme**

The 3D structure of protein can also be represented as a structural environment or view for each residue which is the set of vectors from the \( C_\beta \) atom to \( C_\beta \) atoms of all other residues in the protein[53](Fig2.7).

Vectors give more information on relative positions in a view than simple dis-
Figure 2.5: Protein structure file in mmCIF format downloaded from PDB.

stances. Also, using \( C_\beta \) atoms gives more information than \( C_\alpha \) atoms. While conformation of the backbone are changing from time to time, \( C_\beta \)-views are rotationally invariant. It makes their comparison insensitive to the displacement of substructures[53].

This method is used in program SSAP[53][54].

Summary

There are some other methods such as Torsion (dihedral) Angles [16]. However, all the protein structure representations mentioned above are based on either main-chains (via \( C_\alpha \)) or sidechains (via \( C_\beta \)) alone, thus they are insufficient to model the orientation of sidechains. A different way of representing a protein’s structure as vectors of \( \overrightarrow{C_\alpha C_\beta} \) has been proposed by [23]. A pair of \( C_\alpha, C_\beta \) atoms in the same amino acid constructs a vector from its \( C_\alpha \) end to \( C_\beta \) end. We adopt this approach as a basis of our model.

2.2 Graph Theoretical Approach

Once protein structures are well represented, structure comparison is an important research problem. Most structure comparison problems can be transformed to the clique problem. Some fundamental concepts are introduced as follows (taken
Figure 2.6: Representation of secondary structure elements as vectors[43].

Figure 2.7: Structural environment scheme.

from [17]).
A graph $G$ is defined as a vertex set $V(G)$ and an edge set $E(G)$.

Two graphs $G_1$ and $G_2$ are isomorphic if there is a one-to-one function $\phi$ from $V(G_1)$ onto $V(G_2)$ such that $uv \in E(G_1)$ if and only if $\phi(u)\phi(v) \in E(G_2)$. The function $\phi$ is called an isomorphism.

A clique in a graph $G$ is a maximal complete subgraph, that is, a clique is a complete subgraph that is not a proper subgraph of any other complete subgraph of $G$. The maximal order of a clique is the clique number of $G$ and is denoted by $\omega(G)$. A clique detection problem is to find the cliques in a graph $G$.

VAST program[28] is a good example to explain all these concepts. VAST performs the comparison on the secondary structure’s level. The goal of the program is to find maximal subset of secondary structures of a protein that are in the same relative positions in compared protein structures.

The graph theoretical approach is used in VAST to find the maximal common substructure. A graph $G$ (Fig.2.8) is constructed as the nodes are the pairs such as $(a_i, b_j)$ where $a_i$ is a secondary structure in protein A and $b_j$ is a secondary structure in protein B. Consider two nodes, for example, $(a_1, b_2)$ and $(a_2, b_3)$. The edge between these two nodes is existed if $a_1$ and $a_2$ are in the same relative position as $b_2$ and $b_3$. Now, the problem of finding maximal common secondary substructure is transformed to find maximal completed subgraphs (cliques) in graph $G$. It is the clique detection problem. Two cliques are found in $G$, $(a_1, b_2)$, $(a_2, b_3)$, $(a_3, b_4)$ and $(a_1, b_3)$, $(a_3, b_2)$, $(a_4, b_4)$, $(a_2, b_1)$. It means that the secondary substructure $a_1$, $a_2$ and $a_3$ of protein A is common to the secondary substructure $b_2$, $b_3$ and $b_4$ of protein B with the isomorphism $\phi(a_1) = b_2$, $\phi(a_2) = b_3$, and $\phi(a_3) = b_4$. The isomorphism of another case is $\phi(a_1) = b_3$, $\phi(a_3) = b_2$, $\phi(a_4) = b_4$, and $\phi(a_2) = b_1$ for another clique.

Unfortunately, the clique detection problem is NP-Complete. Many heuristic algorithms are developed. For program VAST, Grindley et al.[30] proposed to detect cliques using a maximal common subgraph isomorphism algorithm borrowed from graph theory [15].

The most existing heuristic algorithms for the clique problem are partially enumerative and branch-and-bound based [27]. However, they are insufficient to handle large scale data. Therefore, we do not use the clique detection algorithms...
in our work. Instead, we aim to develop a scalable database solution featured by a highly efficient query processing strategy.

2.3 High-dimensional Indexing and Dimensionality Reduction Approaches

Our research problem involves searching a protein structure database. We will show in the next chapter that a motif of protein structure can be represented as a point in a high dimensional space. Therefore, application and optimization of high-dimensional indexing and dimensionality reduction on the motif data will be one of our research tasks. This section gives a brief overview on the existing indexing and dimensionality reduction techniques.

2.3.1 Indexing Approaches

The most commonly used one-dimensional indexing approaches in the database literature are *hashing* and the B$^+$-tree.

*Hash* The hash based method basically uses a hashing function to map search key values into a range of bucket numbers. However, the hashing method does not support range queries.

*B$^+$-tree* The B$^+$-tree maintains a dynamic index structure, which is a balanced
tree where search is directed by its internal nodes (index entries) and data entries are stored in its leaf nodes. An advantage of $B^+$-tree indexing is that it provides efficient support to the range queries without decreasing the efficiency of equality selections.

To index multi-dimensional data objects using one-dimensional indexing method, a simple though extremely expensive way is to create multiple one-dimensional indexes separately on different dimensions. Multiple searches then need to be conducted. The final result will be the intersection of the different sets of intermediate results.

High-dimensional indexing approaches can be applied to directly deal with the multi-dimensional data. The most popular multidimensional (spatial) access method is R-tree indexing, which has been provided in most commercial database management systems such as Oracle.

**R-tree** R-tree is a height-balanced data structure like $B^+$-tree. It is based on the approximation of a complex spatial object (or a group of spatial objects) with the minimum bounding rectangle (MBR) that encloses the geometry. The sides of a MBR are parallel to the axes of the data space. An R-tree consists of a hierarchical index on the MBRs of the geometries. For illustration, Gaede’s paper[26] shows an R-tree for a working example. Because R-tree indexing is fast and works directly on geodetic data, it has been widely used for working with spatial data.

### 2.3.2 Dimensionality Reduction

A possible way of improving efficiency of protein motif matching is to reduce the dimensionality of motif data while maximally preserving the matching function between two motifs in the lower-dimensional space.

The goal of dimensionality reduction is to find an “intrinsic” subspace, which is an approximation of the original space but with a lower dimensionality. It has been demonstrated that there does exist an intrinsic semantic sub-space where the dimensions with lower eigenvalues carry redundant information and therefore can be truncated [22].

A representative dimensionality reduction approach is Singular Value Decomposition (SVD) [21][42].

Singular value decomposition (SVD) is a powerful technique from linear al-
gebra. Given a $m \times n$ data matrix $X$ with rank $r$, $X$ can be decomposed:

$$X = U\Sigma V^T$$

(2.1)

where $U$ and $V$ are orthogonal $m \times r$ and $n \times r$ matrices respectively and $\Sigma$ is an $r \times r$ diagonal matrix whose values are monotonically increasing non-zero singular values of $X$. The columns of $U$ and $V$ are the eigenvectors of $XX^T$ and $X^TX$ respectively.

Dimensional reduction is performed by taking only the first $p$ eigen vectors and singular values to form. The $j_{th}$ vector $x_j$ can be projected to a $p$-dimensional vector on the feature space of span $V_p^T$. The projected vector is actually recorded as the $j - th$ row of $U_p$.

More recently, dimensionality reduction via Locality Preserving Projection (LPP) [31] has been introduced. While SVD aims to preserve inner product in an Euclidean space, LPP aims to preserve the intrinsic geometric structure in term of local neighborhood information of the data on a manifold. For a detailed description of the LPP algorithm, refer to [32][31].
Chapter 3

Problem Formulation

Our work essentially deals with the identification of matching structural regions, called motifs, between two proteins. This problem is important in bioinformatics as matching motifs identify regions of interest. For example, motifs may indicate regions on a protein where a drug can dock. The previous chapter has shown that proteins can be represented as geometric objects. The structure of the geometric space has a direct influence on the motif matching problem. In this chapter, a vector representation of proteins is introduced. The structural representation model called motif signature is then proposed, followed by its similarity measure.

3.1 Vector Representation of Proteins

All the protein structure representations mentioned in the previous chapter are based on either mainchains (via $C_\alpha$) or sidechains (via $C_\beta$) alone, thus they are insufficient to model the orientation of sidechains. A different way of representing a protein’s structure as vectors of $\overrightarrow{C_\alpha C_\beta}$ has been proposed by [23]. A pair of $C_\alpha, C_\beta$ atoms in the same amino acid constructs a vector from its $C_\alpha$ end to $C_\beta$ end. We adopt this approach as a basis of our model, which is formulated as follows.

A protein can be defined as a set $P$ of three dimensional vectors (Fig.3.1):

$$P = \{v_i| 1 \leq i \leq n\}$$  \hspace{1cm} (3.1)

where $n = |P|$.

Each $v_i$ denotes a vector of $\overrightarrow{C_\alpha C_\beta}$ for residue $i$. The number of vectors in a protein is typically from 100 to 500. The length of a vector (i.e., the distance
between its $\alpha$-end and $\beta$-end) is fixed at 1.5 Å (angstrom).

### 3.2 Motif Signature

Since the PDB (Protein Data Bank) supplies the coordinate of each atom of proteins in a three-dimensional space, it is easy to build a $C_\alpha C_\beta$ vector space and calculate Euclidean distances between atoms. In our work, the notion of *characterization of spatial relationship* refers to constraints which tie the vectors so that they have a fixed spatial relationship. That is, they can only rotate or translate globally as a whole without any internal change of positions. As the distances between atoms play a significant role in protein structure analysis, here we consider a distance-based characterization of spatial relationships between vectors.

To introduce the model for representing the spatial relationship, we start with the definition of bowtie.

**Definition 1. Bowtie**

Given the protein $P (n > 1)$. The characterization $B_{v_i, v_j}$ of spatial relationship between vectors $v_i$ and $v_j$ ($i \neq j$) is the set $\{d_{\alpha \alpha}^{i,j}, d_{\beta \beta}^{i,j}, d_{\alpha \beta}^{i,j}, d_{\beta \alpha}^{i,j}\}$ (as shown in Fig3.2) where

- $d_{\alpha \alpha}^{i,j}$: the distance between $\alpha$-end of $v_i$ and $\alpha$-end of $v_j$
- $d_{\alpha \beta}^{i,j}$: the distance between $\alpha$-end of $v_i$ and $\beta$-end of $v_j$
- $d_{\beta \alpha}^{i,j}$: the distance between $\beta$-end of $v_i$ and $\alpha$-end of $v_j$
- $d_{\beta \beta}^{i,j}$: the distance between $\beta$-end of $v_i$ and $\beta$-end of $v_j$

This characterization is called a “bowtie”, formally denoted as:

$$B_{v_i, v_j} (d_{\alpha \alpha}^{i,j}, d_{\beta \beta}^{i,j}, d_{\alpha \beta}^{i,j}, d_{\beta \alpha}^{i,j})$$
in short $B_{v_i,v_j}$.

Note that the vector lengths (i.e., $d_{\alpha \beta}^{i,i}$ and $d_{\alpha \beta}^{j,j}$) will not be considered as part of the characterization because they are invariant (always a constant of 1.5 Å) and therefore not useful to discriminate different combinations of vectors.

In the following, we give two propositions based on internal distance method.

**Proposition 1.** Given a protein $P$ ($n > 1$) and two vectors $v_i, v_j \in P$. Then the bowtie $B_{v_i,v_j}$ is sufficient and necessary to characterize the spatial relationship between $v_i$ and $v_j$.

**Proof.**

1. **Sufficiency:** An axiom used as a basis in this proof is that a triangular pyramid is a rigid structure in 3D space. Points $\alpha_i$, $\beta_i$, $\alpha_j$ and $\beta_j$ construct a triangular pyramid with edges of fixed lengths (i.e. $d_{\alpha \beta}^{i,i}$, $d_{\alpha \beta}^{i,j}$, $d_{\alpha \beta}^{j,i}$, $d_{\alpha \beta}^{j,j}$, 1.5 Å and 1.5 Å). So, $v_i$ and $v_j$ are constrained in this pyramid with unchanged internal distances namely spatial relationship.

2. **Necessity:** We have to show that the spatial relationship is not stable by dropping any one of the four internal distances. Consider the case of only using $d_{\alpha \beta}^{i,i}$, $d_{\alpha \beta}^{i,j}$, and $d_{\alpha \beta}^{j,i}$ to represent the spatial relationship between $v_i$ and $v_j$. In this case, the two vectors have 360 degree of freedom of rotating around the edge $\langle \alpha_i, \alpha_j \rangle$ to separately different directions. Therefore, the spatial relationship between points $\beta_i$ and $\beta_j$ can not be fixed. The other cases can be proved similarly. Thus, the necessity of the stated condition is proved.

The structural regions on a protein can be described as *motifs* which are subsets of vectors in the protein structure within a certain distance cutoff.

**Definition 2. Motif**

A motif is defined as a spherical region of protein $P$, whose diameter is $\varepsilon$ (i.e. a

---

1Note that $B_{v_i,v_i}$ is the mirror image of $B_{v_i,v_j}$ in the 3D space.
Figure 3.3: An example of motif. Each vector represents a $C_\alpha-C_\beta$ bond of a residue. The diameter is $\varepsilon$. The dashed line shows the $\alpha-\alpha$ distance ($d_{\alpha\alpha}$) between two vectors.

distance cut-off (Fig3.3). More formally, $M = \{v_1, v_2, ..., v_m\} \subseteq P$ ($m > 2$) is a motif if and only if ($\forall v_i, v_j \in M, d_{\alpha\alpha} \leq \varepsilon$) $\land$ ($\forall v_k \in M, \forall v_l \notin M, d_{\alpha\alpha} > \varepsilon$).

We can observe from the above definition that a motif actually represents a maximal (un-extendable) structural region with respect to a distance cutoff $\varepsilon$ (8Å in this report). It is a set of vectors with particular constraints on spatial arrangement. Therefore, it is necessary to generalize the notion of bowtie for motif.

**Proposition 2.** For a $k$-sized motif ($k > 2$), $7k - 10$ internal distances are sufficient to characterize the spatial relationship among the vectors.

A way of introducing the internal distances is illustrated by Fig.3.4 (the dashed lines as internal distances). The first two vectors $v_1$ and $v_2$ form a bowtie (Fig.3.4(a)). When the $k^{th}$ ($k > 2$) vector $v_k$ comes in, it always brings in four additional distances $d_{\alpha\alpha}^{1,k}, d_{\beta\beta}^{1,k}, d_{\alpha\beta}^{1,k}$ and $d_{\beta\alpha}^{1,k}$ via forming a new bowtie between $v_1$ and $v_k$, and other three distances $d_{\alpha\alpha}^{2,k}, d_{\beta\beta}^{2,k}$ and $d_{\alpha\beta}^{2,k}$. Therefore, the total number of internal distances is $4 + (k - 2) \times 4 + (k - 2) \times 3 = 7k - 10$.

**Proof.** As we have proved in Proposition1, four internal distances are applied for fixing the spacial relationship of two vectors. When a new vector $v_k$ comes in, consider one point $\beta_k$ (Fig.3.4(b)) of the $v_k$. Three internal distances $d_{\alpha\beta}^{2,k}, d_{\beta\beta}^{2,k}$ and $d_{\alpha\beta}^{1,k}$ are used to make a stable structure - triangular pyramid $\alpha_1\beta_1\alpha_2\beta_k$ (Fig.3.4(b)). However, a mirror image of this pyramid $\alpha_1\beta_1\alpha_2\beta'_k$ (Fig.3.4(c)) exists. The point $\beta'_k$ is the mirror image of the point $\beta_k$. The internal distances from point $\beta_3$ to points $\alpha_1$, $\beta_1$ and $\alpha_2$ are equal to those gotten from $\beta_3$. Thus, the position of the point $\beta_k$ cannot be identified by just using three internal distances. One more internal distance $d_{\beta\beta}^{2,k}$ (Fig.3.4(d)) is introduced to solve this problem. It is the similar way to fix another point $\alpha_k$ of $v_k$. Three internal distances $d_{\alpha\alpha}^{1,k}, d_{\beta\alpha}^{1,k}$ and $d_{\alpha\beta}^{2,k}$ (Fig.3.4(e)) are used together with the length of vector itself to fix the spacial
relationship between $\alpha_k$ and the other vectors. Now, $v_k$ is a rigid part of the whole structure.

So the total number of internal distances required is $4 + (k - 2) \times 4 + (k - 2) \times 3 = 7k - 10$. \hfill \square

For a $k$-sized motif $M$ ($k > 2$), the set of $7k - 10$ internal distances is called its motif signature, which identifies the spatial relationship of vectors in the motif. It can be formally defined as:

$$Sig_M = (d_{1a_a}^{1,2}, d_{1a_a}^{1,3}, ..., d_{1a_a}^{1,k}, d_{a_a}^{2,3}, ..., d_{a_a}^{2,k}, d_{\beta\beta}^{1,2}, d_{\beta\beta}^{1,3}, ..., d_{\beta\beta}^{1,k}, d_{\beta\beta}^{2,3}, ..., d_{\beta\beta}^{2,k}, d_{\alpha\beta}^{1,2}, d_{\alpha\beta}^{1,3}, ..., d_{\alpha\beta}^{1,k}, d_{\alpha\beta}^{2,3}, ..., d_{\alpha\beta}^{2,k}, d_{\beta\alpha}^{1,2}, d_{\beta\alpha}^{1,3}, ..., d_{\beta\alpha}^{1,k}, d_{\beta\alpha}^{2,3}, ..., d_{\beta\alpha}^{2,k})$$

(3.2)

For convenience of use, we denote it in short as $Sig_M = (d_1, ..., d_z)$, where $z = 7k - 10$.

This proposition is of significance because the motif matching algorithms presented later are based on internal distances. The less distances that need to be dealt with, the faster in motif comparison. In other words, a motif signature is represented as a vector of $7k - 10$ elements. The problem of motif similarity match is mapped to find the nearest neighbors in the $(7k - 10)$-dimensional space. Clearly,
the lower dimensions we use, the better are those indexing and processing steps later.

Our motif signature model is similar to the graph rigidity problem [41], where $3n-6$ distances (as edges) are required to form a rigid graph of $n$ points (as nodes) in 3D space. They both aim to construct a stable structure. However, the motif signature model is stricter. This can be illustrated by the following example. Fig. 3.4(a) is a rigid graph of four points. When a new point $\beta_k$ is added, three additional distances $d_{\alpha\beta}^k$, $d_{\alpha\beta}^{2k}$ and $d_{\beta\beta}^1$ are used to fix $\beta_k$ and form a rigid graph of five points (Fig. 3.4(b)). Another rigid graph can also be constructed (Fig. 3.4(c)) with the same distances constraints, i.e., $d_{\alpha\beta}^{1k} = d_{\alpha\beta}^{1k'}$, $d_{\alpha\beta}^{2k} = d_{\alpha\beta}^{2k'}$ and $d_{\beta\beta}^1 = d_{\beta\beta}^{1k'}$. $\beta_k'$ is a mirror image of the point $\beta_k$. Hence, $3n-6$ distances may not construct a unique rigid graph. On the other hand, the motif signature model does not allow the reflection.

Since a protein can be represented as a set of three dimensional vectors, we propose to perform 3D protein structure comparison by comparing the spatial relationship among vectors between two proteins. In other words, if two protein structures are similar, the spatial relationship among vectors of one structure must be similar to that of the other. The motif signature serves the core in computing the similarity of two proteins.

### 3.3 Similarity Measures

Similarity between proteins are derived from the similarities of their motifs. We first define the similarity measure between two $k$-sized sub-motifs. Since they are unordered collections of $k$ vectors, they both have $k!$ representations depending on different ordering of vectors. For each representation, there is a corresponding motif signature. Each $k$-sized sub-motif $S$ has a set of $k!$ motif signatures, denoted as $\text{SIG}_S$.

**Definition 3.** Similarity measure between two $k$-sized sub-motifs

Given two $k$-sized sub-motifs $S$ and $S'$, and their motif signatures sets $\text{SIG}_S$ and $\text{SIG}_{S'}$. $S$ is similar to $S'$ (denoted as $S \approx_\delta S'$ or in short $S \approx S'$) if and only if there exists a $\text{Sig}_S = (d_1, ..., d_z) \in \text{SIG}_S$ and a $\text{Sig}_{S'} = (d'_1, ..., d'_z) \in \text{SIG}_{S'}$, such that $\forall i \in 1..z$, $d_i \approx_\delta d'_i$.

Note that, the relationship “$\approx_\delta$” means “equal to” with a tolerance $\delta$.

**Definition 4.** Similarity measure between two Motifs

Two motifs $M$ and $M'$ are similar (denoted as $M \approx_{\delta,1} M'$ or in short $M \approx M'$)
if there exist sub-motif $S \subseteq M$, sub-motif $S' \subseteq M'$ and $S$ is similar to $S'$ under condition of $|S'| = |S'| \geq l$, where $l$ is a parameter determining the minimum size of sub-motifs. Normally it is set to be within the range of 5-20.

For two proteins $P$ and $P'$, they have a matching patch if there exist a motif $M \subseteq P$ and a motif $M' \subseteq P'$ such that $M \approx M'$.

In summary, given a query protein $Q$, the problem we investigate is to find all the proteins from a protein database such that the resultant proteins have one or more matching motifs with $Q$ and to identify possible locations and sizes of such matching motifs.

### 3.4 Ordered Sub-motifs

It is inefficient to use $k!$ motif signatures to represent a $k$-sized sub-motif. In this section, we will introduce an ordering function to generate less numbered ordered sub-motifs. To generate an ordering of a set of vectors, the first vector in the ordered set, named base vector $v_b$, needs to be selected as a starting point. An ordering function $\phi_b$ which is subject to $v_b$ is defined as follows.

**Definition 5. Ordering Function**

Given a $k$-sized sub-motif $S = \{v_1, v_2, ..., v_k\}$ and a base vector $v_b$ ($b = 1..k$), an ordering function subject to $v_b$ is defined as $\phi_b : q \rightarrow q'$ such that $q, q' = 1..k$ and

1. $\phi_b(b) = 1$
2. if $\phi_b(i) < \phi_b(j)$, then $(d^b_{\alpha\alpha} < d^b_{\alpha\beta}) \lor (d^b_{\alpha\alpha} = d^b_{\alpha\beta} \land d^b_{\alpha\beta} < d^b_{\beta\beta})$ ($i, j = 1..k$)

By selecting any vector $v_b$ from $S$, an ordered $k$-sized sub-motif, denoted as $S^b_{\phi} = (v_{\phi^{-1}(1)}, ..., v_{\phi^{-1}(k)})$, can be generated. Its motif signature is denoted as $\text{Sig}_{S^b_{\phi}}$.

**Definition 6. Similarity measure between two ordered $k$-sized sub-motifs**

Given two ordered $k$-sized sub-motifs, $S_{\phi}$ and $S'_{\phi'}$, their motif signatures are $\text{Sig}_{S_{\phi}} = (d_1, ..., d_z)$ and $\text{Sig}_{S'_{\phi'}} = (d'_1, ..., d'_z)$. They are similar (denoted as $S_{\phi} \approx_{\delta} S'_{\phi'}$ or in short $S_{\phi} \approx S'_{\phi'}$) if and only if $\forall i \in 1..z$, $d_i \approx_{\delta} d'_i$.

To compare two $k$-sized sub-motifs $S$ and $S'$, we can first obtain an ordered $k$-sized sub-motif $S_{\phi} \in S'$ by fixing a base vector. Each ordered $k$-sized sub-motif $S'_{\phi} \in S'$ should be compared with $S_{\phi}$. $S \approx S'$ if $\exists S'_{\phi} \in S'$ such that $S_{\phi} \approx S'_{\phi'}$. 20
Compared to Def.3 which requires $k! \times k!$ comparisons between motif signatures of $k$-sized sub-motifs in the worst case, this approach requires only $k$ comparisons on ordered $k$-sized sub-motifs. Therefore, the efficiency will be largely improved. However, it can not guaranty 100% recall for matching with tolerance. More details about this issue will be discussed in the next chapter.
Chapter 4

A Match-and-Expand Searching Strategy

In this chapter, we introduce a match-and-expand strategy for fast protein structure matching.

If two motifs $M$ and $M'$ have a maximal matching sub-motif of $K$ vectors, they must also have matching sub-motifs of $1, 2, \cdots, K-1$ vectors. The match-and-expand strategy, similar to the philosophy of BLAST [6], matches the sub-motifs of same size, then expand the size of matched sub-motif to further reduce the number of candidates. A set of all motifs of size $k(k \leq K)$ is pre-computed for all proteins in the database. In order to check if $M$ and $M'$ have a matching sub-motif (of size no smaller than $K$), the $k$-sized sub-motifs of $M$ and $M'$ are checked first. If no $k$-sized matching sub-motifs are found, $M$ and $M'$ will not have any matching sub-motifs. Otherwise, $M$ and $M'$ will be further checked, starting from their matching $k$-sized sub-motif, until finding maximal sized matching sub-motifs. The $k$-sized sub-motifs thus serve as seeds, from which larger sized motifs can be generated and compared with much lower computational cost.

The choice of $k$ is important. If it is too small, then the match step may generate too many false hits; if it is too large, then the cost of materializing all $k$-sized motifs can be very high. The experimental result of choosing $k$ will be discussed later.

According to the match-and-expand strategy, the protein structure matching problem can be split into five tasks (Fig4.1), which include $C_{\alpha}C_{\beta}$ vector extracting from PDB, motif detection, $k$-sized seed motif creation, seed motif matching and expansion. The maximum matching motifs of proteins to a query protein will be returned for post-processing of functional analysis. The pre-processing (vector
extraction and motif detection) can be done by using established classical heuristic algorithms. They are not the key points of our research, we now suppose the motifs have been derived and focus on the efficiently protein structure similarity query processing.

4.1 Seed-motif Database Generation

In this section we will investigate the generation of seed-motifs, based on which a match-and-expand strategy is developed for motif matching (i.e., finding maximal matching sub-motifs).

Definition 7. Distance Sequence
A distance sequence $D^b_S$ is defined as:

$$D^b_S = < d_{\alpha \alpha}^{b,b}(1), ..., d_{\alpha \alpha}^{b,b}(k) >.$$  

We use the notation $D^b_S[p]$ for the $p^{th}$ element of $D^b_S$ which is $d_{\alpha \alpha}^{b,b}(p)$. The ordered $k$-sized sub-motif based on $\phi_b$ are denoted as

$$S^b_\phi = (v_{\phi_b}^{-1}(1), ..., v_{\phi_b}^{-1}(k)).$$

Definition 8. Basebowtie
Given a $k$-sized sub-motif $S = \{v_1, v_2, ..., v_k\}$, $B_{v_m,v_n}$ is the basebowtie of $S$, if the following conditions hold:

1. if $B_{v_i,v_j} \neq B_{v_p,v_q}$, then $d_{\alpha \alpha}^{p,q} \leq d_{\alpha \alpha}^{i,j}$
2. if $B_{v_i,v_j} \neq B_{v_p,v_q}$ and $d_{\alpha \alpha}^{p,q} = d_{\alpha \alpha}^{i,j}$, then $d_{\alpha \beta}^{p,q} < d_{\alpha \beta}^{i,j}$ ($i, j = 1..k$)

Given a $k$-sized sub-motif $S = \{v_1, ..., v_k\}$. Suppose the basebowtie of $S$ is $B_{v_b,v_c}$, and $v_b$ is selected as the base vector, then $S^b_\phi = (v_{\phi_b}^{-1}(1), ..., v_{\phi_b}^{-1}(k))$ is called a seed-motif. A set of $k$-sized seed-motifs is denoted as $SEED^k$.

Obviously, the distance sequence is part of the motif signatures. It is used as the constraints in the filter step.

For illustration, an example is shown in Fig4.2.

The seed-motif can be generated by Algorithm 1 (in Fig4.3). Each seed-motif is described by its motif signature. Each seed-motif in $SEED^k$ will then be inserted into seed-motif database with the attributes such as protein entry, ordered vector ids, and motif signature.

After a seed motif database has been built, next we present an efficient query processing strategy for a scalable solution for large motif databases.
Figure 4.1: The framework of protein structure matching.
protein entry: 1A1I

\[ S = \{v_3, v_4, v_5\} \]

The \(d_{\alpha\alpha}\) matrix is:

\[
\begin{array}{c|ccc}
  & v_3 & v_4 & v_5 \\
\hline
  v_3 & 0 & 3.832595 & 5.688909 \\
v_4 & 3.832595 & 0 & 3.804697 \\
v_5 & 5.688909 & 3.804697 & 0 \\
\end{array}
\]

Suppose \(d_{5,4}^{a\beta} < d_{4,5}^{a\beta}\), then \(B_{v_5,v_4}\) is basebowtie of \(S\).

Let \(v_5\) be the basevector.

\[
d_{5,5}^{a\alpha} = 0, \quad d_{5,4}^{a\alpha} = 3.804697, \quad d_{5,3}^{a\alpha} = 5.688909
\]

The ordering function \(\phi_5\) is defined as:

\[
\phi_5(5) = 1, \quad \phi_5(4) = 2, \quad \phi_5(3) = 3
\]

\[
D_5^S = \langle d_{5,5}^{a\alpha}, d_{5,4}^{a\alpha}, d_{5,3}^{a\alpha} \rangle
\]

\[
= \langle 0, 3.804697, 5.688909 \rangle
\]

\[
S_5^S = \langle v_5, v_4, v_3 \rangle \text{ is the seed-motif representing } S.
\]

Figure 4.2: An example of re-ordering the vectors in a sub-motif.

### 4.2 Match-and-Expand

We can first transform a protein \(P\) of \(n\) vectors into a group \(SEED^k\) of \(k\)-sized seeds. Note that \(|SEED^k| < C^k_n\) as a sub-motif in \(SEED^k\) is not an arbitrary combination of \(k\) vectors in \(P\). Instead, the \(C_\alpha-C_\alpha\) distances between any two vectors in the motif must be within a distance cutoff \(\varepsilon = 8\text{Å}\). Nonetheless, \(|SEED^k|\) can still be a very large number when \(n\) is large.

Given a query protein \(Q\) containing a set of motifs \(\mathbb{M}\). Each motif \(M\) in \(\mathbb{M}\) can be split into \(C^k_{|M|}\) \(k\)-sized sub-motifs.

For each \(k\)-sized query sub-motif, \(S_q = \{v_1, ..., v_k\}\), a set of ordered query sub-motifs is defined as \(S_{aq} = \{S_{q}^0 | q = 1..k\}\). All these ordered sub-motifs are used to search the seed-motifs in the database.

To find all maximal matching motifs between proteins \(P\) and \(Q\). The match-and-expand strategy using \(k\)-sized seed motifs consists of the following two stages:

1. **Match stage**: to find a set of matching \(k\)-sized sub-motifs \(C = \{(S, S') | S \in SEED^k, S' \in S_{aq}, S \approx S'\}\).

2. **Expand stage**: for each \((S, S') \in C\), check if there exist \(S^K \subseteq P\) and \(S'^K \subseteq Q\), such that \(S \subseteq S^K, S' \subseteq S'^K, S^K \approx S'^K\) for \(K > k\).
Algorithm 1 [To determine the k-sized seed-motif of k-sized sub-motif \( S = \{v_{i1}, v_{i2}, ..., v_{ik}\} \).]

/* find the basebowtie and the basevector \( v_b \) */
1. \( \text{mind}_{\alpha\alpha} = \text{maximum} \);
2. \( \text{mind}_{\alpha\beta} = \text{maximum} \);
3. for (p = 1; p <= k; p++)
   4.   for (q = 1; q <= k; q++)
      5.     if (p \neq q) {
         6.       if \( d_{i_p,i_q}^{\alpha\alpha} < \text{mind}_{\alpha\alpha} \) {
            7.         \( \text{mind}_{\alpha\alpha} = d_{i_p,i_q}^{\alpha\alpha} \); \( v_b = i_p \);
            8.         \( \text{mind}_{\alpha\beta} = d_{i_p,i_q}^{\alpha\beta} \); }
      9.     } elseif (\( d_{i_p,i_q}^{\alpha\beta} = \text{mind}_{\alpha\alpha} \) and \( i_p < v_b \)) {
         10.        \( \text{mind}_{\alpha\alpha} = d_{i_p,i_q}^{\alpha\alpha} \); \( v_b = i_p \);
            11.        \( \text{mind}_{\alpha\beta} = d_{i_p,i_q}^{\alpha\beta} \); }
   12.   }
13. for (p = 1; p <= k; p++) \( \phi^{-1}(p) = i_p \)
14. for (q = p+1; q <= k; q++)
15.   if \( d_{i_p,i_q}^{\alpha\alpha} > d_{i_p,i_q}^{b\alpha} \) {
         16.     swap \( d_{i_p,i_q}^{\alpha\alpha}, d_{i_p,i_q}^{b\alpha} \);
         17.     swap \( \phi^{-1}(p), \phi^{-1}(q) \); }
18. Output(\( \phi \));

Figure 4.3: Algorithm1: Seed-Motif creation.

Operationally the expand stage can be accomplished by incrementally expanding k-sized sub-motifs \( S \) and \( S' \) by one vector each time until maximum matching motifs are reached.

As illustrated before, matching two sub-motifs is too time consuming. When they are two ordered sub-motifs, however, we can solve the problem in a heuristic way via a match-and-expand approach.

4.2.1 Match

In the match stage, we apply the filter-and-refine approach to speed up the matching processing for initial sized motifs. Filter step derives a set of candidate match-
ings by applying heuristic rules, based on the orders on $C_\alpha$-$C_\alpha$ distances. The refine step performs matching based on the motif signatures of the candidates. As the seed database is very large, an efficient indexing structure is desirable to facilitate faster searching. This is left as one of our future work.

Filter

Several rules will be introduced in this section to derive the candidate matching functions $\psi_c : v_p \rightarrow u_q$ on two ordered sub-motifs $S_q = (v_1, ..., v_k)$ and $S'_q = (u_1, ..., u_k)$ in a heuristic way. Note that $v_1$ and $u_1$ are base vectors of these two ordered sub-motifs respectively. Candidate deriving can be considered as a filtering strategy of sub-motif matching.

**Proposition 3.** Given two $k$-sized ordered sub-motifs $S_q = (v_1, ..., v_k)$ and $S'_q = (u_1, ..., u_k)$. If $S_q \approx S'_q$ and $\psi(v_1) = u_1$, then $D_S[p] \approx D_{S'}[p]$ (i.e. $d^{q,p}_{\alpha\alpha} \approx d'^{q,p}_{\alpha\alpha}$), for any $p = 2..k$ (the proof is trivial and thus omitted here)

We can know from Proposition 3 that given two $k$-sized ordered sub-motifs $S_q$ and $S'_q$, for any $p = 1..k$, if $D_S[p] \approx D_{S'}[p]$, then there does not exist a matching function $\psi$ with $\psi(v_1) = u_1$ to make $S_q \approx S'_q$. It can be used to prune a large number of unnecessary comparisons. The matching function $\psi$ can be derived from following rules.

**Rule 1.** $\psi(v_p) = u_p$, if $D_S[p] \approx D_{S'}[p]$, $\forall p = 1..k$

Rule 1 conducts a linear comparison between $D_S[p]$ and $D_{S'}[p]$. However, it does not consider possible cross-position mappings between $v_p$ and $u_1$ for $p \neq q$. The next rules address this problem.

**Rule 2.** A new candidate matching function $\psi'$ can be obtained from $\psi$ by partial modification: $\psi(v_f) = u_g$, $\psi(v_q) = u_f$, if $D_S[p] \approx D_{S'}[p]$, $D_S[f] \approx D_{S'}[g]$ and $D_S[g] \approx D_{S'}[f]$, $f, g, p = 2..k$

**Proposition 4.** Given two $k$-sized ordered sub-motifs $S_q = (v_1, ..., v_k)$ and $S'_q = (u_1, ..., u_k)$ satisfying $D_S[p] \approx D_{S'}[p]$, $p = 2..k$. If there exist $2 \leq f, g \leq k$, $f \neq g$ such that $D_S[f] \approx D_{S'}[g]$ and $D_S[g] \approx D_{S'}[f]$, then $|D_S[f] - D_{S'}[g]| \leq 2\delta$

**Proof.** Suppose $x = D_{S'}[g] - D_{S'}[f]$.

$D_S[f] \approx D_{S'}[g]$.
⇒ |DS[g] − DS[g]| ≤ δ
⇒ |DS[f] − (DS'[f] + x)| ≤ δ
⇒ {−δ ≤ DS[f] − (DS'[f] + x) ≤ δ
⇒ {DS[f] − DS'[f] − δ ≤ x
x ≤ DS[f] − DS'[f] + δ (1)

∴ DS[p] ≈ DS'[p], p = 2..k
∴ |DS[f] − DS'[f]| ≤ δ (2)

(1) + (2)
⇒ −2δ ≤ x ≤ 2δ
⇒ |x| ≤ 2δ
⇒ |DS'[g] − DS'[f]| ≤ 2δ

Rule 3. A new candidate matching function ψ' can be obtained from ψ by partial modification: ψ(vf) = ug, ψ(vg) = uf, if DS[p] ≈ DS'[p], and |DS'[f] − DS'[g]| < 2δ, 2 ≤ f, g ≤ k, ∀p = 2..k

Rule 3 is derived from Proposition 4 and Rule 2 as a generalization of Rule 2. The resultant set of candidate matching functions from Rule 3 is a superset of that of Rule 2. As a consequence, Rule 3 may result in more false hits. When searching a database of n sub-motifs against a query sub-motif, on the other hand, Rule 3 requires less computations than Rule 2. C2k × n inter sub-motif cross-position comparisons for DS[g] ≈ DS'[f] are needed in Rule 2, while Rule 3 requires only C2k internal cross-position comparisons once for DS'[g] ≈ DS'[f]. Therefore, we use Rule 3 in our experiments instead of Rule 2. An example is given in Fig4.4 for applying candidate deriving rules.

Refine

A collection of candidate k-sized matching functions is then generated through the filter stage. They are further pruned by comparing all corresponding (6k-8) distances between two sub-motifs. After the refine process, a collection of correct k-sized matching functions is obtained.
RULE1:
\[ \Rightarrow \psi[v_2] = u_9, \psi[v_1] = u_8, \psi[v_3] = u_7, \psi[v_4] = u_6 \]

RULE2:
\[ \psi' = \psi \]
\[ D_S'[2] \approx D_S'[3], D_S'[3] \approx D_S'[2] \]
\[ \Rightarrow \psi'[v_2] = u_9, \psi'[v_1] = u_7, \psi'[v_3] = u_8, \psi'[v_4] = u_6 \]

or RULE3:
\[ \psi' = \psi \]
\[ |D_S'[2] - D_S'[3]| = 1.211014 < 3 = 2\delta \]
\[ \Rightarrow \psi'[v_2] = u_9, \psi'[v_1] = u_7, \psi'[v_3] = u_8, \psi'[v_4] = u_6 \]
\[ \psi'' = \psi \]
\[ |D_S'[3] - D_S'[4]| = 1.836778 < 3 = 2\delta \]
\[ \Rightarrow \psi''[v_2] = u_9, \psi''[v_1] = u_8, \psi''[v_3] = u_6, \psi''[v_4] = u_7 \]

Figure 4.4: An example of candidate derivation.

4.2.2 Expand

Given two motifs \( M = \{v_1, ..., v_m\} \) and \( M' = \{u_1, ..., u_{m'}\} \), and two matching \( k \)-sized sub-motifs \( S_q = (v_{i_1}, ..., v_{i_k}) \subseteq M \) and \( S'_q = (u_{j_1}, ..., u_{j_k}) \subseteq M' \) determined by a matching function \( \psi : i_q \rightarrow j_q \mid q = 1..k \), a collection of \((k + 1)\)-sized matching sub-motifs are derived by Algorithm 2 (in Fig4.5). The matching expansion continues until the maximal sized sub-motif is reached.

4.3 Experiments

In this section, we set up the experiments and report the results of an extensive performance study conducted to evaluate the proposed representation model and the match-and-expand strategy on protein data.
**Algorithm 2** [To find the \((k + 1)\)-sized sub-motif matching.]

1. for (cv = 1; cv<=m; cv++)
2. if \((v_{cv} \notin S_d)\) {
3.   for (cu = 1; cu<=m'; cu++)
4.     if \((v_{cu} \notin S_d')\) {
5.       for (q = 1; q<=k; q++)
6.         if \(B_{v_{cv},v_{iq}} \approx B_{v_{cu},v_{jq}}\) {
7.           AddNewMatchedVector(cv,cu);}}}}

![Figure 4.5: Algorithm2: Matching expansion.](image)

<table>
<thead>
<tr>
<th>Total number of proteins</th>
<th>811</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of vectors</td>
<td>190,669</td>
</tr>
<tr>
<td>Average number of vectors per protein</td>
<td>216</td>
</tr>
<tr>
<td>Total number of motifs</td>
<td>254,491</td>
</tr>
<tr>
<td>Average number of motifs per protein</td>
<td>314</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of test data

### 4.3.1 Experimental setup

**Test Data**

A total number of 811 sample proteins are selected for our initial experiments according to the PDB_LIST_20040601 (R-factor<0.2 and Resolution<1.9) in the WHATIF relational database. The PDB structures stored in the WHAT IF relational database are a representative set of sequence-unique (a sequence identity percentage cutoff of 30%) structures generated from the X-ray protein PDB files available at a certain moment[3]. After pre-processing, the data statistics are shown in Table 4.1.

We also investigate the number of different sized sub-motifs for the test data. The distribution of \(k\)-sized sub-motifs extracted from the test collection is shown in Fig4.6 for various \(k\). It can be observed that the number of sub-motifs reaches the peak when \(k\) is increased to 3 and then falls down as \(k\) increases. There are very few \(k\)-sized sub-motifs when \(k\) is larger than 7. This suggests that 3 to 6 sized sub-motifs are most common and there is no much room for reducing candidate space via filtering using 7 (or above)-sized sub-motifs. As a result, we mainly test the effect of \(k\) for values of 3 to 6 in the experiment.
Figure 4.6: Number of k-size sub-motif.

Table 4.2: Average number of k-sized sub-motifs in a query protein

<table>
<thead>
<tr>
<th>k</th>
<th>Average number of k-sized sub-motifs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>480</td>
</tr>
<tr>
<td>4</td>
<td>329</td>
</tr>
<tr>
<td>5</td>
<td>104</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

Query proteins

Ten different sized proteins are selected as queries. All the experimental results reported later will be averaged over the 10 query proteins. The average number of vectors per query is 81. In these queries, the average number of k-sized sub-motifs is listed in Table 4.2.

Parameter settings

There are several parameters need to be set for our model and search method, two of which are fixed in our experiments:

- Distance cutoff $\varepsilon$: 8Å
- Minimal size of sub-motifs to output $l$: 5

Two other parameters are variables. We will test how the different settings of them affect the performance.

- Similarity tolerance $\delta$: 0.1Å, 0.3Å, 0.5Å, 0.7Å and 1Å
• Size of seed motifs $k$: 4, 5, 6

We do not consider the case of $k = 3$ due to the setting of $r = 5$. All the matching $k$-sized sub-motifs can be generated from the $k−1$ matching sub-motifs. It has been shown in Fig4.6 that the number of 3-sized seed-motifs is much larger than that of the 4-sized. Thus, starting the match step at $k = 3$ and expanding to $k = 4$ would be more expensive than performing match step at $k = 4$ directly.

Performance Indicators

Our programs are written in C++ and running on Pentium 4 CPU (2.8GHZ) with 1G RAM. The major performance indicator we used is the CPU time to complete a query.

To choose a baseline for comparison with our match-and-expand method, we perform pairwise comparison of all distances between two motifs for matching. Its performance is listed in Table 4.3. $\delta$ has no effect on the CPU time. Because the number of sub-motifs becomes less and less as $k$ changes from 4 to 6, its time plunges as $k$ increases.

4.3.2 Experimental results

Here we present the performance of the match-and-expand method. In match stage, different candidate deriving rules are combined and compared: Rule 1 and (Rule 1+Rule 3).

Effectiveness of Match Stage

<table>
<thead>
<tr>
<th>size$\delta$</th>
<th>$\delta = 0.1$</th>
<th>$\delta = 0.3$</th>
<th>$\delta = 0.5$</th>
<th>$\delta = 0.7$</th>
<th>$\delta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=4</td>
<td>15800</td>
<td>15800</td>
<td>15800</td>
<td>15800</td>
<td>15800</td>
</tr>
<tr>
<td>k=5</td>
<td>3960</td>
<td>3960</td>
<td>3960</td>
<td>3960</td>
<td>3960</td>
</tr>
<tr>
<td>k=6</td>
<td>121</td>
<td>121</td>
<td>121</td>
<td>121</td>
<td>121</td>
</tr>
</tbody>
</table>

Table 4.3: CPU time(seconds) for computing matching k-sized sub-motifs using pairwise comparison

For Rule 1 and (Rule 1+Rule 3), their CPU time used for k-sized sub-motif matching are summarized in Table 4.4 and Table 4.5 respectively, with respect to different $k$ and $\delta$. As we can see from Table 4.4, for the same $k$, the CPU time goes
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\text{size} \ \& \ \text{\(\delta\)} & \text{\(\delta = 0.1\)} & \text{\(\delta = 0.3\)} & \text{\(\delta = 0.5\)} & \text{\(\delta = 0.7\)} & \text{\(\delta = 1\)} \\
\hline
\text{k=4} & 188 & 296 & 520 & 744 & 1040 \\
\text{k=5} & 24 & 25 & 65 & 73 & 75 \\
\text{k=6} & <1 & <1 & <1 & <1 & <1 \\
\hline
\end{tabular}
\caption{CPU time (seconds) for computing matching k-sized sub-motifs using Rule 1}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\text{size} \ \& \ \text{\(\delta\)} & \text{\(\delta = 0.1\)} & \text{\(\delta = 0.3\)} & \text{\(\delta = 0.5\)} & \text{\(\delta = 0.7\)} & \text{\(\delta = 1\)} \\
\hline
\text{k=4} & 363 & 671 & 984 & 1411 & 9123 \\
\text{k=5} & 25 & 106 & 186 & 376 & 811 \\
\text{k=6} & <1 & 3 & 4 & 17 & 53 \\
\hline
\end{tabular}
\caption{CPU time (seconds) for computing matching k-sized sub-motifs using Rule 1+Rule 3}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\text{size} \ \& \ \text{\(\delta\)} & \text{\(\delta = 0.1\)} & \text{\(\delta = 0.3\)} & \text{\(\delta = 0.5\)} & \text{\(\delta = 0.7\)} & \text{\(\delta = 1\)} \\
\hline
\text{k=4} & 0.7 & 0.72 & 0.76 & 0.79 & 0.82 \\
\text{k=5} & 0.73 & 0.75 & 0.79 & 0.82 & 0.86 \\
\text{k=6} & 0.78 & 0.79 & 0.8 & 0.83 & 0.88 \\
\hline
\end{tabular}
\caption{Recall of Rule 1}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\text{size} \ \& \ \text{\(\delta\)} & \text{\(\delta = 0.1\)} & \text{\(\delta = 0.3\)} & \text{\(\delta = 0.5\)} & \text{\(\delta = 0.7\)} & \text{\(\delta = 1\)} \\
\hline
\text{k=4} & 12 & 37 & 95 & 170 & 310 \\
\text{k=5} & <1 & <1 & <1 & 2 & 5 \\
\text{k=6} & <1 & <1 & <1 & 3 & 7 \\
\hline
\end{tabular}
\caption{CPU time (seconds) for matching larger sized sub-motifs}
\end{table}
up as $\delta$ increases since more sub-motifs are compared. On the contrary, for the same $\delta$, the CPU time drops as $k$ increases since less sub-motifs are larger sized. When Rule 3 is also considered, the constraint is relaxed and more sub-motifs are compared so that all correctly matching sub-motifs are returned. As shown in Table 4.5, the performance of (Rule 1+Rule 3), which returns correct results, is worse than that of Rule 1 only. Though not listed in the table, it is worthy noting that the selectivity of the filtering stage by Rule 1 is about 25% on average, lower than the 53% of (Rule1+Rule3).

Since Rule 1 does not guarantee to include all matching results, its recall, which is defined as the percentage of correctly matching sub-motifs being returned as candidates, should be measured. Table 4.6 indicates its recall for different $k$ and $\delta$. As observed, Rule 1 achieves best performance with satisfactory recall of about 80%. So it is a good choice when CPU time is a critical requirement.

**Effectiveness of Expand Stage**

Next, we test the expanding time based on the candidates returned from matching stage by Rule 1 and Rule 3. Table 4.7 shows the average CPU time for expanding $k$-sized sub-motifs to find maximal matching sub-motifs. We can observe that the expansion process is accomplished much more quickly than the matching stage as shown in Table 4.5. This is because the number of candidate sub-motifs are much small after the match stage.

The total CPU time for a query motif to find the matching motifs consists of both matching and expanding time. However, re-look at Table 4.3, we can see that our match-and-expand method outperforms baseline by more than an order of magnitude.
Chapter 5

Research Issues, Future Research Plan and Timetable

5.1 Research Achievements

During the first year of my PhD study, I have completed:

- Identification and formulation of the research problem
- Development of the motif signature model
- Development of a match-and-expand searching strategy
- Collection and processing of protein structure data for the experiments
- Pre-processing of the data to extract $\overline{C_{\alpha}}C_{\beta}$ vectors
- Experimental analysis of existing database indexing structures for protein structure queries
- Detecting motifs based on an established algorithm [15]
- Designing and running initial experiments to evaluate the match-and-expand strategy

The research activities and achievements I have made during this year are summarized as follows.

Publications:

- Huang, Z., X, Song, D., Bruza, P.D., and Zhou, X., Dimensionality Reduction in Motif-Signature Based Protein Structure Matching, Submitted to SIGIR’05
- Huang, Z., Zhou, X, Song, D., Bruza, P.D., and Shen, H.T., 3D Protein Structure
Matching by Motif Signatures, Submitted to SIGMOD’05


Reports:
- Three-Month Progress Report
- Six-Month Progress Report (Literature Review)

Presentations:
- Oral Presentation in 3rd Asia Pacific Bioinformatics Conference (Jan 2005)
- Poster Presentation in DSTC Research Symposium (Jul 2004)
- Poster Presentation in IMB All Hands Annual Meeting (May 2004)

Training:
- Winter School in Mathematics and Computational Biology, IMB, UQ, (Jul 2004)

5.2 Future Work

In the rest of my PhD work, I will be focusing on investigating how to further improve performance of motif matching. The tasks of my research include:

5.2.1 Developing More Effective Filtering Algorithms

We can observe from the experimental results in the previous chapter that the filter step plays a crucial role in motif matching. The current heuristic rules are all based on the $d_{\alpha\alpha}$ distances only. Enhanced rules incorporating the other three types of distances (i.e., $d_{\alpha\beta}$, $d_{\beta\alpha}$, and $d_{\beta\beta}$) will be developed to feature a more efficient filter step.
5.2.2 Large Scale Experiments to Determine an Optimal Size $k$ of Seed Motifs

In our match-and-expand strategy, it is essential to select an appropriate value for the size $k$ of seed-motifs to start with. If it is too small, then the match step may generate too many false hits; if it is too large, then the cost of materializing all $k$-sized motifs can be very high and a lot of smaller sized matches will be lost. By running more experiments on larger data set, we will be able to more precisely analyze the data distribution of different sized motifs. A method will be developed to determine the setting of $k$ according to the data distribution.

5.2.3 Application and Optimization of High-Dimensional Indexing Methods and Dimensionality Reduction Techniques

Currently, we conducted motif matching without using any indexing structures. However, when we are dealing with large amount of data, it will be impossible to read everything to the main memory. Therefore, we shall investigate suitable indexing method to facilitate more efficient motif searching. As a $k$-sized motif signature can be viewed as a point in $(7k-10)$-dimensional space, we plan to investigate the application and optimization of high-dimensional indexing approaches such as $R$-Tree and its variations, and dimensionality reduction techniques such as Singular Value Decomposition[21] and Locality Preserving Projection[32][31].

5.2.4 More Effective Evaluation System

As we conducted our initial experiments on a smaller data set and have not yet used indexing structures at this stage, all the data were read into the memory before the comparison. As a result, our current evaluation system is mainly based on CPU time and recall. Our future experiments will involve larger scale data and indexing structured, so that additional performance indicators such as matching selectivity, I/O bound, CPU bound, etc. will be introduced to evaluate our framework.

5.3 Timetable and Milestones

The timetable and milestones for my future research are proposed in Table5.3. I will keep targeting to publish papers in top conferences and journals in database,
### Table 5.1: Timetable for future research.

<table>
<thead>
<tr>
<th>Period</th>
<th>Tasks</th>
<th>Scheduled submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb. 2005 - Jul. 2005</td>
<td>Develop enhanced filtering algorithm</td>
<td>ICDE06 CIKM05</td>
</tr>
<tr>
<td>Aug. 2005 - Nov. 2005</td>
<td>Conduct larger scale experiments and determine appropriate $k$</td>
<td>SIGMOD06</td>
</tr>
<tr>
<td>Feb. 2006 - May 2006</td>
<td>Test and optimize existing indexing and dimensionality reduction approaches</td>
<td>VLDB06</td>
</tr>
<tr>
<td>Jun. 2006 - Dec. 2006</td>
<td>Thesis Writing</td>
<td>ICDE07 SIGMOD07</td>
</tr>
</tbody>
</table>

bio-informatics and information retrieval.

### 5.4 Required Resource

My initial experiments with 811 proteins have occupied 20G hard disk. More experiments will be conducted on larger data set involving 20,000 proteins. The machine I am currently using with 1G ram and 80G hard disk is obviously far not enough. Therefore, I request a more powerful computer.

### Acknowledgements

The work reported in this confirmation report has been funded in part by the Australian Research Council (Grant No. DP0344488) and the Co-operative Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government’s CRC Programme (Department of Education, Science and Training). I would like to thank my advisors Prof. Xiaofang Zhou and Prof. Peter Bruza for their kind support and advice. I also want to thank Richard Cole and Sham Prasher for their kind assistance.
Bibliography


42


[57] Teresa Prezytycka. lecture notes of bioinformatics course. Lecture Notes.


43


