Using Motif Databases to Help Improve Multiple Sequence Alignment

Guangyi Wang

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

Auguest 2006

Your file Votre référence
ISBN: 978-0-494-20783-3
Our file Notre référence
ISBN: 978-0-494-20783-3

Abstract

Using Motif Databases to Help Improve Multiple Sequence Alignment

Guangyi Wang

Current progress in genome research projects has generated huge amount of data. As a result, the analysis of these data is now a bottleneck in bioinformatics. Multiple sequence alignment is an important step in this kind of analysis. It compares unknown sequences with well studied ones, and thus infers functional and structural information of the unknown sequences.

However, due to the NP-completeness nature of the multiple sequence alignment, exhaustive searching method is unrealistic. Current algorithms use heuristic approach to get a nearly global optimal result. As a consequence, any specific program may encounter certain cases that it is not good at.

In this work, we use protein motif databases to improve the alignment. The basic idea is to detect possible occurrences of motifs on the sequences, and force those parts to be aligned together. Unlike existing programs, this method uses biological information instead of treating it as purely an optimization problem. It also reduces the searching space. Experiments show that using motif databases could generate good results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multiple sequence alignment is an important tool in bioinformatics research. Better alignment can lead to more accurate automatic annotations. In this paper we present an approach to use functional or structural databases to improve the quality of alignment results.

1.1 The Multiple Sequence Alignment Problem

In current genome projects, scientists have successfully sequenced the whole genome of many different organisms. This has generated large amount of genomic data. These nucleic acid data can be translated into protein sequences. Proteins play essential roles in biochemical reactions, which build the foundation of nearly all living activities. Consequently, the main object of bioinformatics is to use all tools at its disposal to deduce the possible function of the coded protein.

Bioinformatics, as defined here, is the science of using computational techniques to analyze biological data. One of the widely used techniques is multiple sequence alignment.

Multiple Sequence Alignment(MSA) is an important step in these analysis. Once the genomic sequence data has been obtained from sequencing, the parts which encode the proteins can be identified by various tools. The protein sequence deduced from these nucleic acids is called putative or hypothetical protein. Because of the large quantity of these putative proteins, it is impossible to use laboratory procedures to determine the functions of every sequence. MSA programs will align those hypothetical proteins with those that have already been thoroughly analyzed to check for similarity and to deduce the functions of putative proteins. If the functional part does align together and exposes a high level of similarity, even if the total sequence is very divergent, we may still deduce that these proteins may be involved in similar biological activities.

The MSA problem has attracted much attention since the 1970s. Exact alignment of two sequences in the mathematical sense has been studied by Needleman-Wunsch[14] on global cases (where the whole sequences should be aligned), and by Smith-Waterman[20] on local cases (where only the highly identical parts should be aligned). However, these alignment algorithms, when extended to multi-dimensional cases, have exponential time complexity. Due to the NP-hard nature of the problem[25], it is unlikly to find one ultimate algorithm to perfectly solve the MSA problem. The existing algorithms will sacrifice the optimality for acceptable speed. Instead of finding the mathematical optimal solution, these "heuristic" algorithms will find good solutions without guaranteeing that they are the best in order to complete the computation within a resonable time limit.

1.2 Existing Algorithms

There are several existing algorithms. These algorithms can be mainly divided into two category: progressive and iterative.

ClustalW is a typical progressive multiple sequence alignment algorithm. These algorithms break the multiple sequence alignments into several pairwise alignments. In each step, only one sequence is added into the alignment. When all the sequences are added, a multiple sequence alignment is achieved.

The order of those steps plays an important role. In order to construct a good order, different methods have been used. Neighbour-Joining method is used in ClustalW. Other methods include UPGMA[12], maximum parsimony[7].

Iterative algorithms have an initial alignment, and refine that alignment over and over until some optimal criteria is met. DALIGN[13] is a typical example of this kind of algorithms.

For complete surveys of multiple sequence alignment, see [15] and [23]. Although dozens of algorithms have been devised and several of them have been widely used in practice, the alignment result may still need manual adjustment. This might be caused by the fact that

- the current algorithms are all heuristic in nature due to the NP-Completeness nature of MSA; and
- the pure mathematical optimal goal may not reflect 100% the biological goal.

Thus in this work, we use biological information to solve the limitation of the MSA algorithms.

1.3 Structural and Functional Protein Databases

In order to help scientists to analyze the sequences, different protein structural and functional databases were built. The common functional or structural parts are summarized as patterns.

PROSITE[10] is a protein domain and family database. In that database, proteins are grouped according to their similarities. Similar proteins should have similar structures, thus they are grouped together into one family. Domains are the common feature that one group of proteins share. In PROSITE, these domains have two forms: pattern and profiles.

PROSITE also provides tools to search the domain database. In this thesis, we use ScanProsite[9] tool to check if a certain sequence has domains stored in PROSITE.

1.4 Contribution of the Thesis

In this work we use structural and functional information to imporve the alignments. First we introduce the concept of anchor points. These points will force the alignment to be fixed at certain positions. By searching the databases, we could find motif at each sequences participating in the alignment and the same motif should be aligned together. These motifs can serve as anchor points and produce better alignment result.

To verify the validity of this approach, we have implemented the algorithm using Java. We also have tested the program using the alignment benchmark database: BAliBASE. Results show that the alignment scores of this algorithm are higher than those of ClustalW in all 4 references, which means this approach does generate better alignment results.

1.5 Organization

First we discuss in Chapter 2 the nature of the multiple sequence alignment problem. Then, in Chapter 3, we examine the existing protein structural and functional databases. In Chapter 4 we introduce the concept of anchor points to force certain parts of sequences to be aligned together. Using the anchor points generated by database search, we could make sure that known functional parts are not mis-aligned. At Chapter 5 we benchmark our approach with BAliBASE. After that, in Chapter 6 we discuss the results and possible further works.

Chapter 2

Multiple Sequences Alignment

Multiple sequence alignment(MSA) is a method to present several sequences in such a way that the most similar sections of these sequences are put in the same columns. In order to make such a alignment, gaps are often necessary to be inserted in these sequences.

MSA is an extremely useful tool in comparative biological studies. It can be used to construct the phylogenetic relations of several sequences, to find conserved parts within a group of proteins, or to predict the structure and function of proteins. Thus the finding of high quality MSA algorithms attracted much attentions since the 1970s.

Different algorithms have been proposed. These algorithms generally fall into two categories: Progressive or Iterative. Yet these algorithms cannot always produce perfect results, and manual adjustment may be needed.

2.1 Global and Local Alignment

There are two different ways to define multiple sequence alignments: global and local alignments.

Global alignment tries to align every residue in every sequence. The final result will contain whole sequences.

Local alignment will only generate the alignment of the most conserved region of the sequences. The result usually contain only a portion of the total sequence. The alignment is locally best, in the sense that if residues are added to either end of each sequences, the final score will decrease.

We only study global alignments in this thesis.

2.2 Mathematical Definitions

The problem of MSA can be defined as an optimization problem as follows:

First let A be an alphabet of finite elements. These elements represent the residues of biological sequences. For proteins, the characters in A could be the one letter codes defined by IUPAC (International Union of Pure and Applied Chemistry)[18]. Sequence s is defined as a string on A. We extend A to A' by adding a space character, so that we can use it to represent gaps:

$$A' = A \cup \{'-'\}$$

Now we can define a distance matrix D as follows:

$$D : A' \times A' \rightarrow R$$

D is a function, and it maps pair of characters from A to a real number.

Given a sequence set $Q = \{s_i | i = 1, \ldots, n\}$ of n sequences, an $n \times l$ matrix M, consisting of letters from the alphabet A', is said to be an alignment of Q if each sequence s_i of Q appears exactly once as a row of M, but with an arbitrary number of '-' inserted.

The score of alignment is defined as

$$S(M) = SP(M) - GP(M),$$

I	Y	D	G	G	A	V	-	E	A	L
II	Y	D	G	G	-	-	-	E	A	L
III	F	E	G	G	I	L	V	E	A	L
IV	F	D	-	G	I	L	V	Q	A	L
V	Y	E	G	G	A	V	V	Q	A	L

Table 2.1: An alignment of 5 sequences

where

$$SP(M) = \sum_{1 \le i < j \le n} \sum_{1 \le k \le l} D(M_{ik}, M_{jk})$$

is the sum of pairs, and

$$GP(M)$$

is the gap penalty. This penalty may varies from algorithm to algorithm, but in most cases, it is the sum of an open penalty plus an extending penalty for every gap introduced in M.

For a given set of sequences, the MSA problem will try to find an optimum alignment. The global multiple sequence alignment problem could be then defined as: For a given sequence set Q, find the matrix M, such that $S(M)$ is maximized.

Table 2.1 is an example of alignment: There are 5 sequences in this alignment. In order to align similar or identical residues in the same columns, gaps are added to the first, the second and the fourth sequence.

2.3 Dynamic Programming

When the number of sequences is 2, MSA is reduced to a pairwise alignment problem. Needleman-Wunsch and Smith-Waterman have solved such case. They use dynamic programming to find a placement of the residues, such that a global(Needleman-Wunsch) or local (Smith-Waterman) optima is found.

The main idea of dynamic programming is to use previously found partial optima to find the final solution for the entire problem. In this particular problem of pairwise alignment, dynamic programming will construct a matrix, where the columns and rows are indexed by the residue of the two sequences. Figure 2.1 illustrates the alignment of 2 artificial sequences using dynamic programming. Here we use a simple distance matrix: for each identical pair, the score is 1, and for other pairs, the score is -1. The gap penalty is -2.

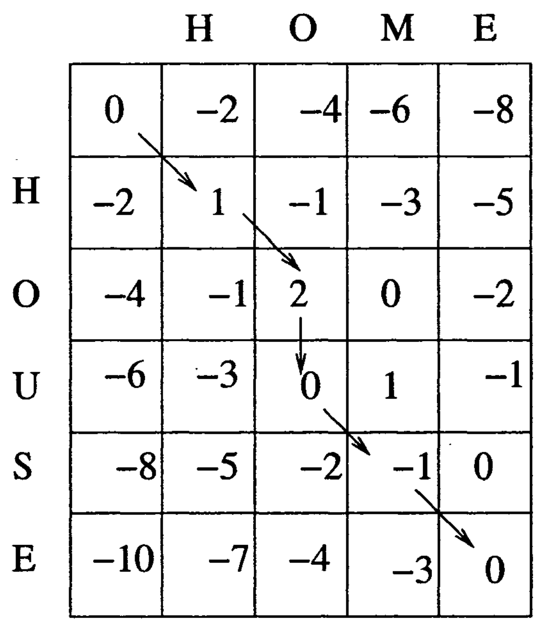
For each cell, there are 3 choices: align the two corresponding residues indexed by the column and row, or add a gap either vertically or horizontally. These 3 choices will generate 3 scores, and the maximum value will be selected as the score of this cell. Figure 2.2 shows the 3 choices of one cell on Figure 2.1. For cell D, its value could be calculated by the following way:

- aligning the "O" in "HOME" with the "U" in "HOUSE", represented by the diagonal arrow in Figure2.2;
- aligning a gap with "U" in "HOUSE", as indicated by the vertical arrow;
- or aligning a gap with the "O" in the sequence "HOME", as indicated by the horizontal arrow.

Because each gap costs -2, and the matching of "O" and "U" costs -1, the value of cell D is decided as follows:

$$V(D) = MAX\{V(A) - 1, V(B) - 2, V(C) - 2\} = MAX\{-2, 0, -5\} = 0$$

This means the second choice is the optimal. Trace information will also be kept for backtracking. In the case of Figure2.2, a pointer will be added, indicating that the



		H	O	M	E
	0	−2	−4	−6	−8
H	−2	1	−1	−3	−5
O	−4	−1	2	0	−2
U	−6	−3	0	1	−1
S	−8	−5	−2	−1	0
E	−10	−7	−4	−3	0

Figure 2.1: Using dynamic programming to align two sequences

value of cell D is derived from cell B. After each cell has been filled, one path will be found from the upper-left to lower-right corner. From this path, we could get the best alignment. In the case of Figure 2.1, the final alignment will be:

$$\begin{array}{ccccc} H & O & - & M & E \\ H & O & U & S & E \end{array}$$

For cases of $n > 2$, theoretically, dynamic programming in n-dimensions can also be used. Yet the complexity of computing is prohibitively huge.

2.4 Heuristic Methods

The dynamic programming method of aligning n sequences of length l has the complexity of $O(l^n)$. It is extremely time consuming to use this method on practical biological sequences analysis. Thus different heuristic solutions has been proposed.

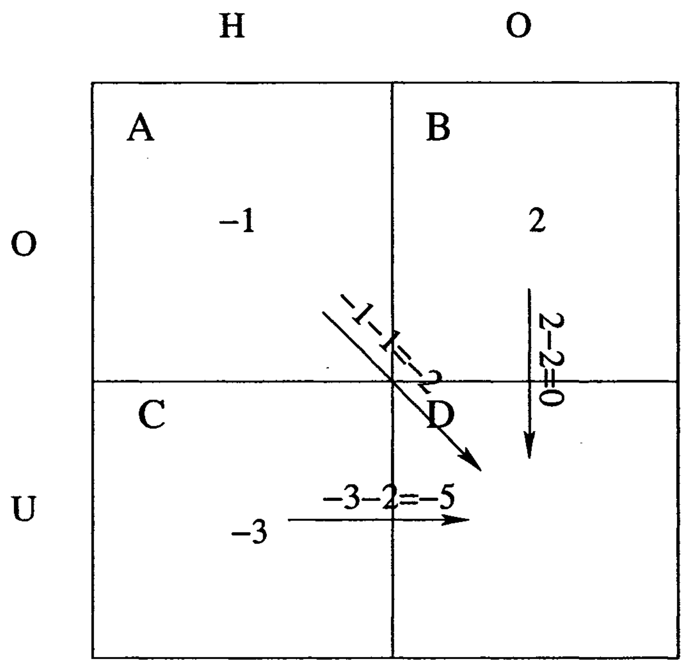


Figure 2.2: Using dynamic programming to align two sequences(partial)

2.4.1 Progressive Alignment

The most popular heuristic method is the progressive method. Its origin can be traced to Feng and Doolittle[8]. The basic idea is to use a step-by-step pairwise alignment to construct the final alignment. The procedure can be briefly described as follows: First, it aligns two sequences from the sequence set, and at each iteration, one new sequence is added to the alignment, without changing the relative position of those sequences that are already in the alignment. When all the sequences have been added, the algorithms will end with a result.

As we can see, the order of alignment could be very critical. If two sequences or two sets of sequences has been aligned at an early stage with an inaccurate result, the latter stage will not correct them and, possibly, even worsen the alignment. Thus, in progressive alignment, choosing a good order to align the sequences is as important as aligning the sequences.

ClustalW [21] is a typical progressive program. To decide the order of aligning n sequences, ClustalW first conduct $n(n-1)/2$ pairwise alignments to compute the similarity. Then, according to these distances of sequences, the program use the Neighbor-Joining method[19] to construct a guide tree. This method starts with a star tree, where each leaf is one sequence. The lengths of the branches reflect the distances between these sequences. The exact vlaue of each branch is not known at the beginning, but the sum of these branches are determined only by the distances value and the number of sequences. This star tree will evolve through $N-2$ steps, where N is the number of sequences. At each step, one pair of the leaves will be selected and combined as one new unit in the new tree. In the next step, this unit will be regared as a single leaf, but inside this unit, the two original leaves can be regared as two children of this unit. The selection of sequence pair will ensure that in the new tree, the sum of all branch lengths is minimal. Then, the distances of this unit to other leaves will be recalculated. The internal branches of the two original leaves will also be produced. After that a new round of selection will begin. The procedure will repeat until only 2 leaves are left, and we will then have a result tree. The original Neighbor-Joining result is a root-less tree, ClustalW then selects a node as the root. The root will make the guide tree as much balanced as possible in the sense that the difference of the two depths of the branches is minimal.

The final alignment is conducted by doing the alignment according to the guide tree. In each step, only one sequence is added to the existing alignment. After $n-1$ steps, the final alignment result of the n sequences is produced.

Other interesting techniques are also used in ClustalW. First, it uses self ad-

Amino Acid	Gap penalty coefficent	Amino Acid	Gap penalty coefficent
A	1.13	M	1.29
C	1.13	N	0.63
D	0.96	P	0.74
E	1.31	Q	1.07
F	1.20	R	0.72
G	0.61	S	0.76
H	1.00	T	0.89
I	1.32	V	1.25
K	0.96	Y	1.00
L	1.21	W	1.23

Table 2.2: Gap coefficent for 20 amino acids

justable matrices during the aligning work. When it first compares each pairs to construct the tree, it uses a general scoring matrix, such as BLOSUM62. When the program is doing the actual alignment, it will choose different matrices according to the identity of the sequences (or alignments). Second, it uses a position specific gap penalty. The penalty of opening a gap is different depending on the amino acid that is just before the gap. This may help increase the accuracy of the alignment. Table 2.4.1 shows the different coefficents of opening gap penalties after each aminio acid. The values are nomalized to 1. Smaller values will decrease the gap penalties, which means gaps are more likely to appear nearby these amino acids than others. The greater values indicates the reverse.

The main advantage of using progressive alignment is its speed. The time complexity of ClustalW, for example, is $O(n^2l^2)$, where n is the number of sequences and l is the average length of the sequences.

2.4.2 Iterative Alignment

Iterative methods use an initial alignment as the beginning and refine it at each iteration. After a certain number of iterations, the local optimal is reached and the program stops. Unlike the progressive methods, iterative methods use many different ways to build the initial alignment and to adjust the alignment at each iteration. DIALIGN is the typical iterative alignment algorithms. It first builds a set of well aligned pairwise fragments, or diagonals, among all input sequences. All these diagonals are saved in a set M. These diagonals may cause conflicts and could not all be used to generate the final alignment, so the second step is to try build a non-conflicting set M_2. At each iteration, some diagonals are added into M_2. This goes on until none could be added anymore.

DIALIGN also have several interesting features. First, it does not use residues as alignment blocks. Instead, it uses fragments as unit to assemble the result. Second, it does not have gap penalties. Such feature is especially good for local alignments.

DIALIGN is not as fast as ClustalW but the speed is acceptable for practical usage. It also performs better on local alignment than global alignment.

Other iterative methods may use genetic or statistics algorithms to do the refining work. However, experience shows that pure genetic algorithms are too slow for practical usage. Thus some of the algorithms use a hybrid of other approaches and genetic algorithms.

2.4.3 Other Approaches

Several different approaches have been proposed as well.

T-Coffee [16] is a progressive algorithm that combine local and global methods.

During the pairwise stage of the alignment, the program not only evaluates the similarity scores of the sequences pairs, but also store the partial global alignments in a library. Furthermore, a local alignment fragment library is also constructed during this stage. The actual alignment phase will try to select the maximum non-conflicting alignment fragments from these two libraries. This approach will increase the accuracy of some alignments that involve divergent sequences.

POA[11] is a program to do MSA using Partial Ordered Graph. The central idea is to change the data structure from a two-dimensional matrix to a "nearly" one-dimensional partial order graph. If two sequence have common fragments, that part will be merged, and form a chain. This chain diverges into two branches when the content of two sequences are different, and will merge again if common fragments are found again.

There are other approaches in MSA such as Hidden Markov Model (HMM) [6], genetic algorithms[17] and artificial intelligence. As the MSA problem is a CPU-intensive task, people have also tried parallel or distributed algorithms. However, it will not change the exponential complexity nature of the problem.

Chapter 3

Protein Structure Database

As time passes, scientists have accumulated more and more genomic sequences in various sequencing projects. The task of examining the biochemical function and structure of each encoded protein in those data is huge. Biologists have to use "in silico" methods, which means using computer software, before applying "wet lab" techniques to examine the detail feature of the proteins. This approach is possible because of the following principle: The amino acids sequence determines the 3D structure, and thus decides the function of the protein.

3.1 Protein and Its Structures

Proteins play very important roles in life activities. It not only constructs most of the organs, but is also involved in nearly all reactions in the cell. For example, enzymes are one special group of proteins. They act as catalysts that make most metabolism and signal transduction functions possible.

The central dogma states that DNA in the nucleus of the cell are transcribed to RNA, and then translated to proteins at the ribosomes. In this theory, every 3 nucleotides will determine an amino acid in the protein. This means that protein, like DNA, has a linear feature. Because of this linear feature, string is a suitable data structure to represent a protein.

Unlike DNA, which has the double helix form, protein does not have one universal structure. Different proteins may fold in different ways. The folding of one particular protein is ultimately decided by its amino acid sequence. That being said, to correctly predict the folding of a particular protein, people have to consider different interactions of nearby amino acids. These interactions can be very complex. The protein folding problem is still an open problem in bioinformatics now. One less demanding task is to categorize proteins according to their structural or functional characteristics. Biologists have summarized patterns in different levels. The following is a brief description of protein structures.

3.1.1 Primary Structure

The building blocks of protein are amino acids. These amino acid units are linked together by peptide bonds. Together they form a chain structure. This linear structure of protein is called its primary structure.

3.1.2 Secondary Structure

Most proteins contain one or more specific patterns. Those groups of amino acids are called secondary structures. The two most common kinds of structures are alpha helix and beta sheet.

In an alpha helix, the interaction bonds of amino acid groups make the polypeptide chain fold into a coil structure, resembling a telephone cord. This kind of structures sometimes signify the presence of functional domains in the protein, such as DNA binding sites or zinc fingers.

Beta sheet is another major type of secondary structure. In a beta sheet, frag-

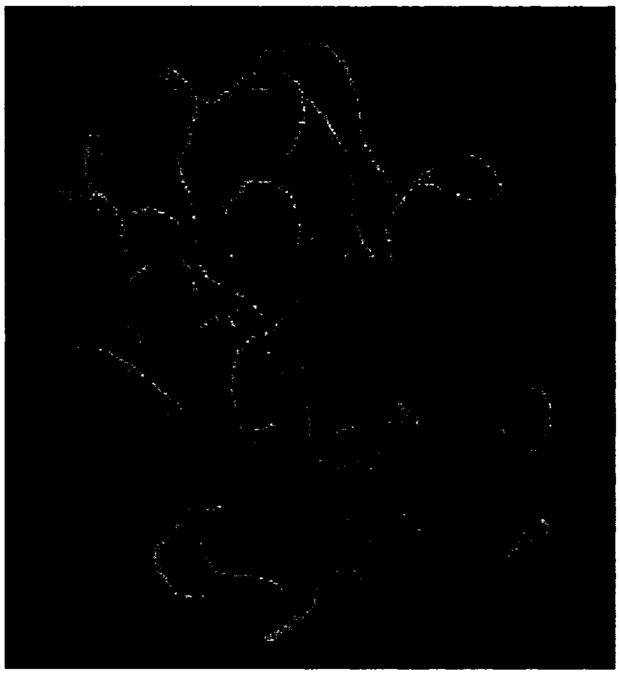


Figure 3.1: Structure of Flavodoxin: Image from PDB[4] database.

ments of polypeptide chain will be fully stretched. Several such strands will run in the same or opposite directions. They form a structure called parallel or anti-parallel sheets.

Figure 3.1 shows one protein(flavodoxin) that contains both the alpha helix and the beta sheet. The dark-colored coils are helices and the light-colored arrows are beta sheets.

3.1.3 Tertiary Structure

Tertiary structure refers to the actual 3-D structure of one polypeptide chain. As we have stated before, the linear line-up of amino acids decides the 3-D folding of the peptide chain. This 3-D structure is the main factor that decides the function of a protein. Genetic mutation may change the amino acids in a protein sequence, and thus changes the final folding of the protein. The resulting protein may be denatured and fail to participate in the biochemical reactions, or, in some case, the change of the structure is beneficial and enhance the function of the protein.

3.1.4 Quaternary Structure

Two or more polypeptide chains together can compose a higher level of structure. In a quaternary structure, one molecule is a subunit. The whole protein group is called a protein complex. One example of quaternary structure is hemoglobin. Hemoglobin is responsible of oxygen transportation in animals. One typical form of hemoglobin contains 4 subunits, while each subunit is a globular protein with oxygen-binding group.

3.2 Secondary Structure Databases

Motifs are functional or structural blocks in protein sequences. The exact occurrences of one motif may vary from sequence to sequence, but a certain pattern is conserved among all the occurrences.

3.2.1 PROSITE Database

PROSITE is a protein structure database. The basic idea is to categorize all proteins into different families. Each family may contain a certain number of proteins that are functionally or evolutionarily related. These relations are reflected in the similarities of their sequences. Certain regions of sequences are more conserved than others. Usually these regions are the evidences of relations among the sequences in one family. The PROSITE database is based on these conserved regions and summarize the regions into signatures, or fingerprints. If a new protein sequence contains a fingerprint of one family, we can then infer that this protein may be a member of this family. This new sequence may potentially share the common functional or structural feature of that family.

There are two forms of fingerprints in the PROSITE database: patterns or profiles.

Pattern Entry

Pattern entry uses regular expression to describe the signature. In this form, fingerprint contains one list of elements. Each element represents one amino acid, or one amino acid groups, or wild card characters of various length. The elements may be in the following form:

- the IUPAC one letter code for amino acid;
- the wild card letter 'x', which can match any amino acids;
- codes in square brackets, indicating an acceptable set of amino acids;
- codes in curly brackets, indicating that those amino acids can not appear at this position.

Each element can be followed by a number or a range of number in one pair of parenthesis. This means the repetition of the previous element. Elements are separated by hyphens '-'. Table 3.1 shows a sample entry in the PROSITE database. The entry describes that the first residue of the pattern should be a tryptophan. The second element, x(9,11) means that the pattern could have 9 to 11 amino acids following the tryptophan, and the exact types of these amino acids are not important. The following elements could be interpreted in a similar way, until we reach {R}, which means the third last position can be any amino acids except arginine. The second

W-x(9,11)-[VFY]-[FYW]-x(6,7)-[GSTNE]-[GSTQCR]-[FYW]-{R}-{SA}-P

Table 3.1: Sample Pattern in PROSITE

last position uses the same notation to indicate that neither alanine nor serine can appear.

Patterns usually describe small conserved regions. If the pattern is very short, it may generate a high number of false positive hits.

Profile Entry

A profile, which is a matrix form of fingerprints, uses a table containing position-specific amino acid weights and gap penalties. A cut-off value is also defined in the entry. Suppose we have one such profile. In order to decide if a specific sequence contains that profile, the sequence is aligned with the profile. This alignment is exactly the same as a pairwise alignment of two sequences. With this alignment, we can get a score. If the alignment score is higher than the cut-off value, a positive hit is reported.

A profile usually contains large regions of sequences. It may even contain a whole domain. A profile is considered to be more accurate and robust than pattern entry, because it contains more information. Instead of having only two states of rejection and acception, the profile gives a number value to reflect the likelihood of the sequence having the pattern. This will reduce the number of false negative results.

Figure 3.2 shows a sample profile entry in PROSITE. Some matrix description lines are omitted, but the structure of the file is clear. The lines that begin with "MA" all describe the matrix. Among these lines, the first few lines contain meta-information such as name and cut-off value. The main part is composed of lines with

```
MA   /GENERAL_SPEC: ALPHABET='ACDEFGHIKLMNPQRSTVWY';
MA   /DISJOINT: DEFINITION=PROTECT; N1=1; N2=53;
MA   /NORMALIZATION: MODE=1; FUNCTION=GLE_ZSCORE; R1=44.55; R2=-0.0035;
MA      R3=0.7386; R4=1.001; R5=0.208; TEXT='ZScore';
MA   /NORMALIZATION: MODE=2; FUNCTION=LINEAR; R1=0.0; R2=0.1;
MA      TEXT='OrigScore';
MA   /CUT_OFF: LEVEL=0; SCORE=90; N_SCORE=7.0; MODE=1;
MA   /DEFAULT: MI=-26; I=-3; IM=0; MD=-26; D=-3; DM=0;
MA   /M: SY='F';M=-2,-3,-3,-4,2,-3,-2,1,-2,0,-1,-2,-3,-3,-4,-2,-1,0,-5,2;
MA   /M: SY='I';M=-1,-5,-2,-3,-2,-3,0,1,1,-1,1,-1,-2,-1,1,-1,0,1,-4,-4;
MA   /M: SY='A';M=2,-3,1,0,-5,2,-2,-1,-1,-3,-2,1,1,0,-2,2,2,0,-8,-5;
... Some lines omitted..
MA   /M: SY='V';M=0,-4,-3,-4,-1,-3,-3,5,-3,3,3,-2,-2,-2,-3,-2,0,5,-8,-4;
MA   /M: SY='L';M=-1,-6,-3,-3,-1,-3,-2,2,-3,3,2,-2,-2,-2,-3,-2,-1,2,-5,-3;
MA   /M: SY='D';M=0,-6,3,3,-6,0,1,-3,2,-5,-2,2,-1,2,1,0,0,-4,-7,-5;
MA   /M: SY='K';M=-1,-6,0,0,-2,-1,0,-3,3,-4,-1,1,-1,0,1,0,0,-3,-6,-4;
MA   /M: SY='N';M=1,-4,1,1,-5,0,0,-2,0,-3,-2,1,1,0,-1,1,1,-1,-7,-5;
MA      /I: MI=0; I=-1; MD=0; /M: SY='X'; M=0; D=-1;
MA   /M: SY='G';M=1,-5,0,0,-5,1,-2,-1,-2,-3,-2,0,0,-1,-2,0,0,-1,-8,-6;
MA   /M: SY='G';M=1,-6,3,3,-7,3,0,-4,-1,-5,-4,2,-1,1,-2,1,0,-3,-10,-6;
MA   /M: SY='W';M=-9,-12,-9,-11,1,-11,-4,-8,-5,-3,-6,-6,-8,-7,3,-4,-8,-9,26,0;
MA   /M: SY='W';M=-7,-9,-9,-9,0,-9,-4,-5,-5,-1,-4,-6,-7,-6,2,-3,-6,-6,18,-1;
MA   /M: SY='K';M=-1,-7,0,0,-3,-2,0,-2,2,-3,-1,1,-1,1,2,0,-1,-3,-5,-5;
MA   /M: SY='G';M=2,-3,0,-1,-6,3,-3,-2,-3,-4,-3,0,0,-2,-3,1,0,0,-10,-6;
MA   /M: SY='Q';M=-2,-6,0,0,-3,-3,1,-2,0,-2,-1,0,-2,1,1,-1,-1,-3,-5,-3;
MA      /I: MI=0; I=-2; MD=0; /M: SY='X'; M=0; D=-2;
MA   /M: SY='T';M=0,-4,-1,-1,-4,0,-2,0,-1,-2,0,0,-1,-1,-1,0,1,0,-7,-5;
MA   /M: SY='T';M=0,-5,0,0,-3,-1,-1,-1,1,-3,-1,1,-1,0,0,1,1,-1,-6,-4;
...Some lines omitted..
MA   /M: SY='Y';M=-5,-1,-7,-7,10,-8,-1,-1,-5,-1,-3,-3,-7,-6,-6,-4,-4,-5,0,13;
MA   /M: SY='V';M=0,-3,-3,-5,-2,-2,-3,5,-3,2,2,-2,-2,-3,-4,-1,0,5,-8,-5;
MA   /M: SY='E';M=1,-6,2,3,-6,0,0,-2,1,-4,-2,1,0,2,0,0,0,-3,-8,-6;
MA   /M: SY='P';M=0,-5,-1,-1,-2,-2,-1,-2,-1,-3,-2,0,1,-1,-2,0,-1,-2,-6,-3;
```

Figure 3.2: Profile entry in PROSITE

keywords "/M" or "/I", which indicate insertion position and matching position. Take the ninth line of Figure 3.2 for example, The "SY='F'" means we can use one character "F" to indicate this position. The following "M=-2,-3,...-5,2;" are the values that each amino acid will get at this position: an alanine(A) will get -2, a cysteine(C) will get -3 and so on. The "/I" keyword is used for insertion. This part indicates the expense of opening a gap, or finishing an existing gap etc. Reference [5] gives a detailed explanation of the syntax.

Both pattern entry and profile entry describe widespread structural or functional patterns among protein sequences. Sometimes, these patterns are also called motifs. Other protein structure databases, such as Pfam[3] and PRINTS[1], also store structural or functional motifs. In the next chapter, we will discuss how to use motifs from these databases to mark the corresponding regions of the sequences, force these regions to be aligned together, and thus create more accurate alignment results. We will only use PROSITE in the next chapter, however, the same principle can be applied to all of these databases.

Chapter 4

Use Motifs to Improve Alignment

The previous algorithms are all based on modeling the alignment problem as a mathematical optimization question. Yet not every solution to this optimization question is the best alignment in the biological sense. As a result, manual adjustment is often necessary to refine the alignments produced by those algorithms. During this refinment, an experienced biologist may decide that certain parts of the sequences are more important than others, especially if some functional sites are involved in these parts. Thus the alignment of these regions are much more important than the alignment of those regions that do not contribute to the function of the proteins.

In this chapter, we will explain how to simulate this manual adjustment work automatically. The goal is to eliminate, or at least reduce the manual editing to refine the alignment. To achieve this goal, it is not enough to rely only on the mathematical optimization model. Biological knowledge should also be used in this automated procedure as well. We will search the protein motif database to gain such information. After that, we use this information as constrains to do the actual alignment.

```
> 1aboA
sitkGEKLRVLgynh
> 1ycsB
mkeGDCMTIIhrededei
```

Table 4.1: Sequences with motifs Capitalized

```
> 1aboA
5,11
> 1ycsB
4,10
```

Table 4.2: The anchor points file

4.1 Anchor Points

First we have to find a way to override the dynamic programming procedure, because the domain information should take precedence over the scoring scheme during the alignment. We propose the concept of anchor points.

4.1.1 The Concept of Anchor Points

Anchor points are the positions at which all sequences should be forced align together. For example, if we have two sequences as shown in Table 4.1, and we decide that the capitalized part of the sequences should be aligned together, we could define the anchor points as in Table 4.2. In this file, two numbers are associated with each sequence. For 1aboA, these two numbers are 5 and 11, and for 1ycsB, those two are 4 and 10. By assigning these values, we are indicating that the fifth residue of 1aboA (G) should be aligned with the fourth residue of 1ycsB (also a G). The second values of each sequence have similar meaning: align the eleventh residue of 1aboA (L) with the tenth residue of 1ycsB (I).

The final alignment with such anchors is shown in Table 4.3.

s	i	t	k	G	E	K	L	R	V	L	g	y	n	h	-	-	-	-
-	m	k	e	G	D	C	M	T	I	I	h	r	e	d	e	d	e	i

Table 4.3: Alignment after anchors are applied

4.1.2 The Benefits of Using Anchor Points

The function of the anchor points in an alignment is to force certain parts to be aligned together. One sequence may contain several anchor points, and if every sequence in the set has the same number of anchor points, a valid anchor points set is defined and could be used during the alignment.

As we have stated, the main purpose of introducing anchor points is to force the relevent parts to be aligned together. To decide why some parts are more relevent than others, extra informations are needed. These information could be achieved from other auxiliary tools, such as protein motif database search, or protein structure prediction. In other words, these are the informations that could not be simply reflected by scores and penalty functions in the mathematical optimization model.

The anchor points can also reduce the search space. Suppose that we have two sequences, and each sequence has exactly one anchor point defined in it. Let us also suppose that the two anchor points occur at the middle of each sequence. The actual searching would be reduced by half. Because aligning two sequences of length n requires cn^2 computing time. Here c is some constant value. By inserting one anchor point at each sequence, we split the alignment problem to two small problems of size $\frac{n}{2}$. Each problem requires $c(\frac{n}{2})^2$. The total amount of work is thus $c(\frac{n}{2})^2+c(\frac{n}{2})^2 = \frac{c}{2}n^2$, which is half of the original cost. If each sequence has exactly t evenly spaced anchor points, we have to do $t+1$ alignments each at a cost of $c(\frac{n}{t+1})^2$, so the total cost is $\frac{c}{t+1}n^2$. Figure 4.1 illustrates the case of adding one pair of anchor points. The whole

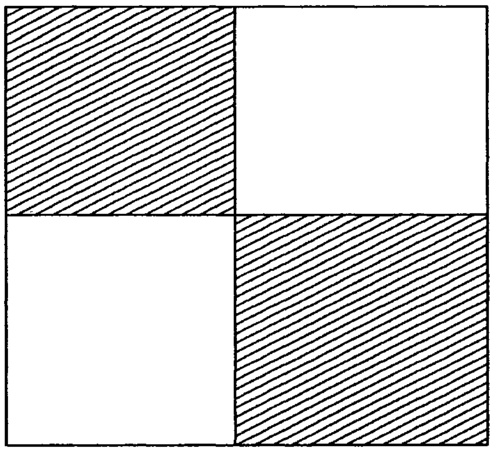


Figure 4.1: Using one anchor point to reduce half searching space

rectangular space represents the search space of the original problem. After adding one anchor point at each sequence, the space is split into 4 parts. Now we only have to consider the two shaded areas.

Please note that the above analysis is based on the best case scenario, where the t anchor points divide the sequence into $t+1$ exactly equal parts. In real alignment, this is rarely the case, and the search space will not be reduced so significantly. However, we will not go much further in this direction in this thesis, because the quality of the alignment, rather than the speed, is our main concern of using anchor points.

4.1.3 The Generation of Anchor Points

We have discussed that introducing anchor points may help produce more acurate alignments. Now the problem is how to generate accurate anchor points. The approach we are using is to detect the motif among each sequence before the alignment. After motifs are found, we try to choose a set that could cover all the sequences without causing a conflict.

It is also possible to use other methods to generate the anchor points. For example, secondary structure prediction could be used to detect the helix or the sheet position in the sequences. Assuming that the conserved region should have the same secondary structure, we could use the start point and end point in the prediction result as anchor points. In this way, we could force the structure to be aligned together.

4.1.4 Using Anchor Points in the Alignment

To apply the anchor points constraints during the alignment, we have to split the optimal search path described in Chapter 2 into several fragments, and, to ensure that the anchor points are correctly aligned, we use them to serve as the split points.

We choose one progressive alignment algorithm, ClustalW, as the starting point of implementation. This is not only because ClustalW is the most widely used algorithm, but also because it is relatively easy to add the concept of anchor point into ClustalW. The original ClustalW uses a divide-and-conquer method. First it will choose the middle point M of one sequence, then the program will use dynamic programming to find the matching point N in the other sequence. N is chosen to maximize the sum of scores of the two paths (from upper-left corner to (M, N), and from $(M+1, N+1)$ to the lower-right corner). By splitting the two sequences at (M, N), the original alignment problem becomes two smaller problems. If we happen to have anchor points in the two sequences, we will just use them as the splitting points. If there is no anchor point in the current aligning fragments, the program will be exactly the same as the original method.

4.2 Using Secondary Databases to Improve Multiple Sequence Alignments

In the following section we will explain in detail the procedure of using a secondary database to help MSA.

The first step is to take each sequence in the alignment file and check the sequence against the PROSITE database. The detected motifs are stored and the matching parts of the sequence are also stored.

The motif detection is done by the ps_scan program provided by PROSITE. Suppose ps_scan has detected that sequence i has domain P, we put the tuple (P_{name}, i, P_a, P_b) into a candidate pool. Here P_a and P_b are the start and end point in the query sequence. P_{name} is the name of the domain detected. After each sequence has been searched, the set will contain all the domains that have been detected.

It is very likely that one sequence will contain several motifs. Thus the second step will select the motif that can cover as many of the sequences as possible. The ideal motif should appear at each sequence. The choice is done by sorting the set of P_{name} according to the number of occurrences in these squences. The most frequently appearing domain will be examined first. If this domain does appear in all the sequences, an anchor point file is generated. Two anchors are defined at each sequence. Those two points are decided by the beginning and the end of the domain detected in each sequence.

In the case where one motif appears on the same sequence more than once, we have to select the one that best align with the other sequences. This is done by comparing the quality of these occurrences. The one that has the highest identity ratio will be selected.

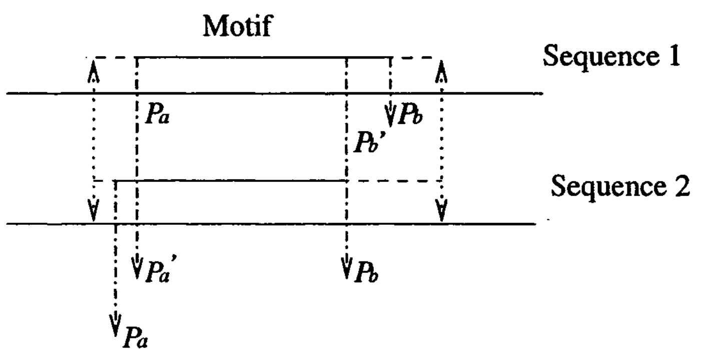


Figure 4.2: Adjustment of motif start and end point

It is also worth noting that the motif occurrence may not necessarily be an exact hit. If the occurrence in the query sequence has several residues missing at the beginning or at the end of a motif, we have to adjust the P_a and P_b value of this motif in the other sequences. This is because we want the anchor points to signify the same relative position of the motif. Figure 4.2 illustrates an example of such adjustment. In this case, both sequences contain the same motif. But the two occurrences of this motif have several residues missing at the beginning and the end. This is represented as the dashed lines. As the missing parts are not the same in two sequences, P_a and P_b of sequence 1 and 2 are not pointing to the same positions of the motif. We have to adjust P_b of sequence 1 and P_a of sequence 2 to P'_a and to P'_b. Same adjustment is applied to the case of more than 3 sequences. The result will ensure that all the motifs in different sequences are precisely aligned.

We have now discussed the theory of the multiple sequence alignment problem and the idea of using a motif database to help the alignment. In next chapter, we will test the validity of this approach by implementing the algorithm and evaluating the program with alignment benchmark data set.

Chapter 5

Implementation and Results

We have implemented the algorithm stated in Chapter 4 and have tested the program using the BAliBASE database.

5.1 Implementation

The program, ProMSA, is implemented in Java. The reason to use Java for implementation is due to its object oriented and platform independent features. Object-Oriented programming makes the designing and future extention of the program easier. The biologists may need to work on different computing environment. Java's cross-platform feature eliminates the work of re-compiling and re-installing the program among different machines.

We use ClustalW version 1.83 as our base of implementation. The source code is in C. We have re-designed the whole program using object techniques. Sequences, alignments and score matrices are now encapsulated as classes. However, the core part of the algorithm remains the same, except that a part on anchor points is added. Figure 5.1 shows the 4 phases of ProMSA. In most cases, if no anchor points file is provided, the alignment result is exactly the same as the ClustalW result. In some sequence sets, there is a small difference between the two scores, mainly due to the

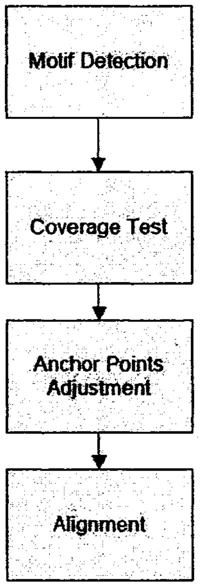


Figure 5.1: Flowchart of ProMSA

different choices of numerical precisions in ClustalW and ProMSA. ProMSA uses doubles to store the scores of the dynamic programming, while ClustalW only keeps two significant digits after the decimal point.

The part on domain detection is done by ps_scan, which is the standalone edition of ScanPROSITE[9]. In order to minimize false hits, we use the '-s' option. This prevents the program from returning very short and frequently appearing motifs. We exclude these motifs because they are very likely only false positive hits.

The result of ps_scan is piped into ProMSA as input. Information such as domain names, starting and ending points goes from ps_scan to ProMSA. This information is used to select the best coverage of sequences. If one motif is selected, a text file containing the anchor points will be generated. Then, ProMSA launches the alignment module. This module will check the presence of an anchor point file. If there is one anchor point file, these points will be read and added as attributes to the sequences. The dynamic programming part will check if any anchor points fall in the range of the current alignment segments. If there is no anchor points, normal dynamic programming is done. If there are anchor points, the fragments are split according to these points. This is a recursive procedure untill all residues are aligned.

5.2 Test Database

Historically, the evaluation of alignment programs is done on data selected by the authors. In order to eliminate the bias of evaluating alignment programs, BAliBASE was introduced.

BAliBASE[24, 2, 22] is a database specially designed to measure the performance of MSA programs. Version 1 of BAliBASE has 142 alignments, over 1000 sequences. Version 2 has 167 alignments, 2100+ sequences. Version 3 has 217 alignments and 6255 sequences. These alignments are divided into several reference sets according to some structural or functional characteristics. These references contain different cases that alignment algorithms should consider. Reference 6-8 in version 2 evaluate special kinds of alignments concerning transmembrane regions, repeats and inverted domains. Version 3 removed these special cases but increased the number of sequences in the first 5 reference set. Checking the performances of one program on different reference sets may reveal the preference of that program, and give a direction on how to improve it. Within one reference set, the length of the sequences and the similarity among the sequences in the alignment also varies. For example, in reference set 1 of BAliBASE 1 and 2, the alignments may be short($<$ 100 residues), medium(200-300 residues) or long ($>$ 400 residues). The similarity of these alignment can also further

Sequence	Domains	Anchor points
DYN3_RAT	PS50003,PS00410	517,614
bmx_human	PS50003	6,104
b21_huma	PS50003	0,108
1btkA	PS50003	4,125
1pls	PS50003	6,94
1awe	PS50003	24,118

Table 5.1: Domains detected in 1dynA

be divided into low($< 25\%$ identity) medium ($20 - 40\%$ identity) and high ($> 35\%$ identity). BAliBASE 3 removes the high identity sequences in reference set 1, because "the comparision of alignment in this level is indecisive"[22].

All the alignments in BAliBASE are manually verified and adjusted. Thus we have an "answer" for a certain alignment question. The package also has a comparison program, named baliscore, to score the computer-generated results with the result in the database. The more the computer generated result agrees with the hand made result, the higher it will score.

In order to evaluate ProMSA, we ran the data set of BAliBASE version 2 and version 3 with both ProMSA and ClustalW. The following is a discussion on the results. The scores in the following tables are evaluated by the baliscore program. Appendix A contains a complete set of scores of running ProMSA and ClustalW on sequences in BAliBASE 3.0.

5.3 Result

5.3.1 A sample sequence set: 1dynA

As an example, we show the alignment generated by ProMSA on sequence set 1dynA of BAliBASE 2.0 reference set 4. This data set contains 6 sequences, with the longest

```
DYN3_RAT    501  qrssqvhkkstignq....virKGWLTVsnigimk...ggskgywfvlta
bmx_human     1  ...............mdtksilEELLLKrsqqkkkmspnnykerlfvltk
b21_human     1  ....................rEGWLLKlgggrvk....twkrrwfiltd
1btkA         1  .................aavilESIFLKrsqqkkktsplnfkkclflltv
1pls          1  ...............mepkrirEGYLVKkgsvf.....ntwkpmwvvlle
1awe          4  iqknidgwegkdiggccnefimEGTLTRvg.........akherhiflfd

DYN3_RAT    597  ...rslelacdsqEDVDSWkasllragvypdksf.....tendengqaen
bmx_human    87  ...gllyvyasneESRSQWlka............................
b21_human    88  gnhtvyrisaptpEEKEEWikcikaaisrdpfye................
1btkA       108  ...gplyvfspteELRKRWihqlknvirynsdlvqkyhpcfwid.gqylc
1pls         77  ...qdhffqaaflEERDAWvrdinkaikcieglehhhhhh..........
1awe        101  ...nsvifsaksaEEKNNWmaalislqyr.stle................
```

Figure 5.2: BAliBASE reference of 1dynA

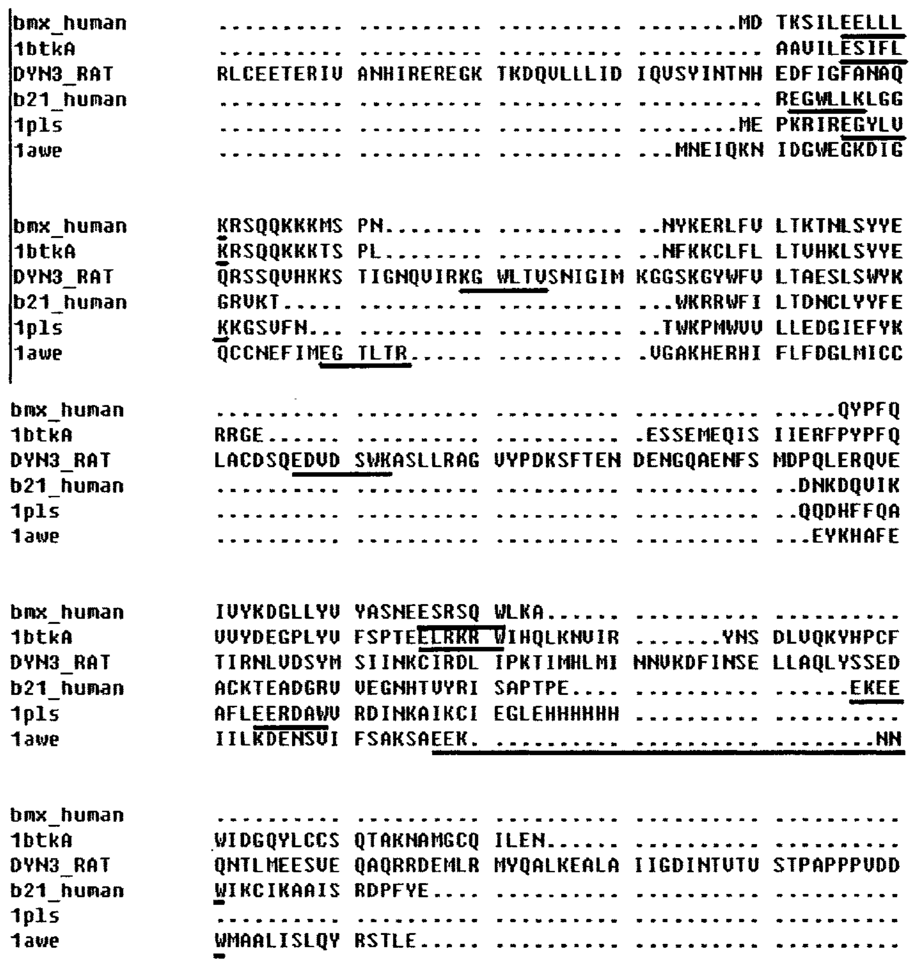
```
bmx_human    .......... .MDTKSILEE LLLKRSQQKK KMSPNNYKER LFVLTKTNLS
1btkA        .......... ...AAVILES IFLKRSQQKK KTSPLNFKKC LFLLTVHKLS
DYN3_RAT     QRSSQVHKKS TIGNQVIRKG WLTVSNIGIM KGGS...KGY WFVLTAESLS
b21_human    .......... .......REG WLLKLGGGRV KTWK....RR WFILTDNCLY
1pls         .......... .MEPKRIREG YLVKKGSVFN TWKP.....M WVVLLEDGIE
1awe         IDGWEGKDIG QCCNEFIMEG TLTRVGAKHE RHIFLFDGLM ICCKSNHGQP

bmx_human    .......... QYPFQIVYKD GLLYVYASNE ESRSQWLKA. ..........
1btkA        MEQISIIERF PYPFQVVYDE GPLYVFSPTE ELRKRWIHQL KNVIRYNSDL
DYN3_RAT     .......... TEQRNVYKDY RSLELACDSQ EDVDSWKASL LRAGVYPDKS
b21_human    DQVIKACKTE ADGRVVEGNH TVYRISAPTP EEKEEWIKCI KAAISR....
1pls         .......... .......TKQ QDHFFQAAFL EERDAWVRDI NKAIKCIEG.
1awe         .......... ........DE NSVIFSAKSA EEKNNWMAAL ISLQYRSTLE
```

Figure 5.3: ProMSA alignment of 1dynA

sequence having 627 residues, and the shortest sequence having 105 residues. As we can see in Fig 5.2, the underlined residues indicate the two core blocks that has been marked by the biologists.

When we run ProMSA on 1dynA, in the motif detection phase, one motif (PH domain, PROSITE entry PS50003) is found in each of the sequences and the anchor points is duly placed. Table 5.1 shows the domains and the anchor points of each sequences. In sequence Dyn3_RAT, another motif (PS00410) is also detected. However, this motif only appears in one sequence and is discarded. The actual values of the



```
bmx_human     .......... .......... .......... ........MD TKSILEELLL
1btkA         .......... .......... .......... .......... AAVILESIFL
DYN3_RAT      RLCEETERIV ANHIREREGK TKDQVLLLID IQVSYINTNH EDFIGFANAQ
b21_human     .......... .......... .......... .......... REGWLLKLGG
1pls          .......... .......... .......... ........ME PKRIREGYLV
1awe          .......... .......... .......... ...MNEIQKN IDGWEGKDIG

bmx_human     KRSQQKKKMS PN........ .......... ..NYKERLFV LTKTNLSYYE
1btkA         KRSQQKKKTS PL........ .......... ..NFKKCLFL LTVHKLSYYE
DYN3_RAT      QRSSQVHKKS TIGNQVIRKG WLTVSNIGIM KGGSKGYWFV LTAESLSWYK
b21_human     GRVKT..... .......... .......... ...WKRRWFI LTDNCLYYFE
1pls          KKGSVFN... .......... .......... ..TWKPMWVV LLEDGIEFYK
1awe          QCCNEFIMEG TLTR...... .......... .VGAKHERHI FLFDGLMICC

bmx_human     .......... .......... .......... .......... .....QYPFQ
1btkA         RRGE...... .......... .......... .ESSEMEQIS IIERFPYPFQ
DYN3_RAT      LACDSQEDVD SWKASLLRAG VYPDKSFTEN DENGQAENFS MDPQLERQVE
b21_human     .......... .......... .......... .......... ..DNKDQVIK
1pls          .......... .......... .......... .......... ..QQDHFFQA
1awe          .......... .......... .......... .......... ...EYKHAFE

bmx_human     IVYKDGLLYV YASNEESRSQ WLKA...... .......... ..........
1btkA         VVYDEGPLYV FSPTEELRKR WIHQLKNVIR .......YNS DLVQKYHPCF
DYN3_RAT      TIRNLVDSYM SIINKCIRDL IPKTIMHLMI NNVKDFINSE LLAQLYSSED
b21_human     ACKTEADGRV VEGNHTVYRI SAPTPE.... .......... ......EKEE
1pls          AFLEERDAWV RDINKAIKCI EGLEHHHHHH .......... ..........
1awe          IILKDENSVI FSAKSAEEK. .......... .......... ........NN

bmx_human     .......... .......... .......... .......... ..........
1btkA         WIDGQYLCCS QTAKNAMGCQ ILEN...... .......... ..........
DYN3_RAT      QNTLMEESVE QAQRRDEMLR MYQALKEALA IIGDINTVTV STPAPPVDD
b21_human     WIKCIKAAIS RDPFYE.... .......... .......... ..........
1pls          .......... .......... .......... .......... ..........
1awe          WMAALISLQY RSTLE..... .......... .......... ..........
```

Figure 5.4: ClustalW alignment of 1dynA

anchor points is adjusted, so that the first points in each sequences are all pointing to the same relative position of the motif. Similarly, the second set of points in the anchor file are also adjusted. With the help of these anchors, in the alignment phase, the two core blocks are aligned exactly as the reference given by the BAliBASE, while ClustalW fails to do so. Figure 5.3 and Figure 5.4 show the alignment results of ProMSA ClustalW, respectively.

Name	ProMSA	ClustalW
1ckA	0.814	0.814
1csp	0.592	0.592
1dynA	0.603	0.194
1lkl	0.809	0.773
1mfa	0.442	0.442
1pfc	0.166	0.328
1pysA	0.581	0.581
1vln	0.894	0.881
1ycc	0.834	0.745
2abk	0.457	0.457
kinase1	0.844	0.633
kinase2	0.685	0.658
average	0.643	0.592

Table 5.2: ProMSA and ClustalW score of Ref 4 in BAliBASE 2.0

5.3.2 Results of Ref 4 on BAliBASE 2.0

Table 5.2 shows the scores of the alignments produced by ProMSA and ClustalW using BAliBASE 2.0 reference set 4 as the test set. Baliscore normalizes the scores of each sequence set to 1. These scores reflect the similarity of the subject alignment with the reference alignments.

To get these scores, we have used the same parameters on both programs. The matrices are from the BLOSUM series. We use the same penalty coefficient (Open penalty -10.0, extending penalty -0.2) as well.

From Table 5.2, we could see that the idea of using constraints generated from structural database does improve the quality of sequence alignment. Reference 4 shows a significant improvement, because sequences in this reference are more divergent and the conventional program cannot generate satisfying results. A notable exception is the sequence set 1pfc, which has a lower score than ClustalW result. The reason is that one sequence in this set, 1yuh, has two occurrences of one same motif:

Reference	ProMSA	ClustalW
Ref 1 Part 1	0.359	0.323
Ref 1 Part 2	0.744	0.737
Ref 2	0.731	0.689
Ref 3	0.542	0.538
Ref 4	0.652	0.597

Table 5.3: ProMSA and ClustalW score of BAliBASE 3.0

PS50835. The rest of the sequences in the set have just one PS50835 in each of them. Although ProMSA has correctly selected this motif at the coverage test phase, the program selected the first occurrence to generate the anchor points. This is different with the reference result given by BAliBASE, which align the second occurrence instead. This exception shows the possibility of improvement in future research.

Chapter 6

Summary and Discussion

In this thesis, we have presented the idea of using structural databases to improve the multiple sequence alignments. Unlike other current algorithms, this idea uses not only mathematical means but also information from biological databases to optimize the alignment of protein sequences. This approach increases the accuracy of the alignment result, especially in the cases of divergent sequences. BAliBASE was used as the benchmark to evaluate the validity of this idea and confirmed that this is a promising approach.

6.1 Summary

Using ClustalW as the foundation, ProMSA added several steps before the actual alignment. First, the sequences are examined by the ps_scan program to detect the presence of motifs. Then the program will do a coverage examination. If there is one motif that covers every sequence, an anchor points file is generated. The purpose of this anchor points file is to guarantee that the same motif is correctly aligned. In the actual alignment stage, the program follows the ClustalW algorithm. Except that if anchor points appear in the sequences, then they will be selected as split points. This will force the positions indicated by these anchors to be aligned together.

BAliBASE has been selected as the test set for evaluating this approach. Compared to ClustalW results, scores are generally improved, especially in the cases of divergent sequences. This demonstrated the validity of the approach.

6.2 Possible Future Works

There are still possible future works to increase the accuracy. The current method will find only one motif which appears at every sequence. Yet it is possible that the sequence set contains several motifs and each motif should be aligned as well.

Another possibility is that if a motif can not be found for every sequence, the program could use some strategy to generate a partial alignment including the sequences that do have a motif in common. The other sequences could be added to the alignment at a later stage. Comparing to the current treatment of discarding this motif and running an alignment without any anchor, this might increase the quality of alignment to some degree.

Bibliography

[1] T.K. Attwood, M.E. Beck, A.J. Bleasby, and D.J. Parry-Smith. PRINTS-A database of protein motif fingerprints. *Nucleic Acids Research*, 22:3590–3596, 1994.

[2] A. Bahr, J.D. Thompson, J.-C. Thierry, and O. Poch. BAliBASE: enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Research*, 29(1):323–326, 2001.

[3] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S.R. Eddy, S. Griffiths-Jones, K.L. Howe, M. Marshall, and E.L. Sonnhammer. The PFAM protein families database. *Nucleic Acids Res.*, 30:276–80, 2002.

[4] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliand, T.N. Bhat, H. Wissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[5] P. Bucher and A. Bairoch. A generalised profile syntax for protein and nucleic acid squence motifs. *Proc Int Conf Intell Syst Mol Biol.*, pages 53–61, 1994.

[6] S.R. Eddy. Multiple alignment using hidden markov models. *Proc Int Conf Intell Syst Mol Biol.*, 3:114–120, 1995.

[7] J. Felsenstein. Cases in which parsimony and compatibility methods will be positively misleading. *Syst. Zool.*, 27:401–4100, 1978.

[8] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 60:351–360, 1987.

[9] A. Gattiker, E. Gasteiger, and A. Bairoch. ScanProsite: a reference implemention of a Prosite scanning tool. *Applied Bioinformatics*, 1:107–108, 2002.

[10] N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. DeCastro, P. S. Langendijk-Genevaux, M. Pagni, and C.J.A. Sigrist. The PROSITE database. *Nucleic Acids Res.*, 34:D227–D230, 2006.

[11] C. Lee, C. Grasso, and M. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[12] C. D. Michener and R. R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.

[13] B. Morgenstern, K. Frech, A. Dress, and T. Werner. DIALIGN: Finding local similarities by multiple sequence alignment. *Bioinformatics*, 14:290–294, 1998.

[14] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[15] C. Notredame. Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics*, 3:131–144, 2002.

[16] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, 302:205–217, 2000.

[17] C. Notredame and D.G. Higgins. Saga: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24:1515–1524, 1996.

[18] IUPAC-IUB Joint Commission on Biochemical Nomenclature. Nomenclature and symbolism for amino acids and peptides, recommendations 1983. *Biochem. J.*, 219:345–373, 1984.

[19] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol.*, 4:406–425, 1987.

[20] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147(1):195–197, 1981.

[21] J. D. Thompson, D.G. Higgins, and T. J. Gibson. ClustalW :improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[22] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *PROTEINS:Structure, Function, and Bioinformatics*, 61:127–136, 2005.

[23] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch. A Comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, 27:2682–2690, 1999.

[24] J.D. Thompson, F. Plewniak, and O. Poch. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–88, 1999.

[25] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.

Appendix A

Result of ProMSA on BAliBASE 3.0

A.1 Ref 1

A.1.1 Part 1

Name	ProMSA	ClustalW
1	0.947	0.947
2	0.208	0.208
3	0.563	0.563
4	0.277	0.199
5	0.214	0.212
6	0.097	0.097
7	0.465	0.465
8	0.356	0.472
9	0.669	0.297
10	0.195	0.195
11	0.070	0.078
12	0.818	0.818
13	0.173	0.173
14	0.644	0.624
15	0.579	0.579
16	0.319	0.319
17	0.647	0.433
18	0.222	0.174
19	0.499	0.499
20	0.354	0.415
21	0.294	0.156
22	0.096	0.096
23	0.202	0.319

24	0.246	0.056
25	0.082	0.082
26	0.103	0.031
27	0.260	0.176
28	0.129	0.129
29	0.505	0.336
30	0.296	0.310
31	0.313	0.304
32	0.113	0.113
33	0.475	0.515
34	0.254	0.264
35	0.594	0.498
36	0.363	0.374
37	0.360	0.360
38	0.598	0.402
Average	0.359	0.323

A.1.2 Part 2

Name	ProMSA	ClustalW
1	0.689	0.681
2	0.828	0.828
3	0.894	0.894
4	0.843	0.843
5	0.836	0.836
6	0.864	0.868
7	0.781	0.793
8	0.869	0.869
9	0.906	0.906
10	0.824	0.824
11	0.643	0.643
12	0.489	0.455
13	0.915	0.918
14	1.00	1.00
15	0.878	0.444
16	0.628	0.773
17	0.882	0.881
18	0.897	0.897
19	0.808	0.808
20	0.814	0.801

21	0.841	0.837
22	0.728	0.728
23	0.784	0.784
24	0.836	0.836
25	0.362	0.387
26	0.780	0.776
27	0.815	0.803
28	0.725	0.725
29	0.780	0.779
30	0.823	0.843
31	0.746	0.761
32	0.647	0.647
33	0.646	0.646
34	0.879	0.879
35	0.764	0.749
36	0.886	0.886
39	0.760	0.760
40	0.950	0.952
41	0.581	0.581
42	0.519	0.519
43	0.739	0.743
44	0.854	0.829
Average	0.744	0.737

A.2 Ref 2

1	0.778	0.077
2	0.184	0.184
3	0.894	0.892
4	0.637	0.641
5	0.716	0.726
6	0.896	0.868
7	0.718	0.734
8	0.526	0.002
9	0.917	0.917
10	0.695	0.690
11	0.863	0.653
12	0.586	0.539
13	0.312	0.325

14	0.745	0.748
15	0.490	0.490
16	0.669	0.683
17	0.878	0.876
18	0.870	0.866
19	0.758	0.758
20	0.795	0.791
21	0.819	0.817
22	0.661	0.685
23	0.764	0.779
24	0.457	0.457
25	0.778	0.784
26	0.739	0.744
27	0.718	0.709
28	0.923	0.923
29	0.739	0.739
30	0.861	0.877
31	0.894	0.749
32	0.913	0.915
33	0.907	0.902
34	0.680	0.652
35	0.748	0.751
36	0.782	0.784
37	0.684	0.691
38	0.912	0.875
39	0.886	0.880
40	0.556	0.554
41	0.615	0.603
Average	0.731	0.689

A.3 Ref 3

Name	ProMSA	ClustalW
1	0.672	0.662
2	0.501	0.340
3	0.451	0.426
4	0.826	0.818
5	0.680	0.686
6	0.328	0.576

7	0.829	0.645
8	0.569	0.567
9	0.396	0.400
10	0.720	0.711
11	0.824	0.824
12	0.473	0.365
13	0.266	0.342
14	0.621	0.545
15	0.377	0.372
16	0.011	0.283
17	0.353	0.353
18	0.737	0.698
19	0.543	0.552
20	0.246	0.246
21	0.556	0.528
22	0.793	0.749
23	0.563	0.622
24	0.525	0.461
25	0.365	0.482
26	0.512	0.447
27	0.551	0.570
28	0.539	0.536
29	0.746	0.731
30	0.672	0.697
Average	0.542	0.538

A.4 Ref 4

Name	ProMSA	ClustalW
1	0.877	0.836
2	0.607	0.017
3	0.813	0.789
4	0.851	0.846
5	0.771	0.807
6	0.678	0.671
7	0.626	0.624
8	0.778	0.770
9	0.695	0.676
10	0.853	0.853

11	0	0
12	0.797	0.759
13	0.451	0.438
14	0.712	0.712
15	0.759	0.037
16	0.773	0.773
17	0.773	0.773
18	0.693	0.693
19	0.666	0.814
20	0.806	0.803
21	0.801	0.801
22	0.785	0.718
23	0.526	0.467
24	0.460	0.446
25	0.807	0.790
26	0.415	0.415
27	0.276	0.313
28	0.811	0.804
29	0.747	0.747
30	0.589	0.582
31	0.541	0.541
32	0.831	0.841
33	0.526	0.181
34	0.523	0.531
35	0.587	0.591
36	0.865	0.870
37	0.206	0.168
38	0.560	0.549
39	0.810	0.810
40	0.748	0.774
41	0.274	0.298
42	0.697	0.676
43	0.586	0.618
44	0.792	0.720
45	0.476	0.476
46	0.364	0.362
47	0.747	0.714
48	0.865	0.680
49	0.774	0.071
Average	0.652	0.597