The Dissertation Committee for Shu Wang Certifies that this is the approved version of the following dissertation:

On Multiple Sequence Alignment

Committee:

Daniel P. Miranker, Supervisor

Anthony P. Ambler, Co-Supervisor

Joydeep Ghosh

Robin Gutell

Mircea D. Driga

On Multiple Sequence Alignment

by

Shu Wang, B.E.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December, 2007

Dedication

Dedicated to my family

Acknowledgements

I would like to especially thank my supervisor, Dr. Daniel Miranker, for his generous help, patience and support. Throughout my doctoral work, he encouraged me to develop independent thinking and research skills. I am deeply grateful to him for holding me to a high research standard and teaching me how to do research. He has been always there to listen and give advice. I would also like to thank my co-supervisor, Dr. Tony Ambler. I am deeply grateful to him for his continuous encouragement, advice, guidance and kindness. I am also very grateful for having an exceptional doctoral committee and wish to thank Dr. Robin Gutell, Dr. Mircea Driga, and Dr. Joydeep Ghosh for the assistance, generosity, and advice I received from them. I would like to especially thank Dr. Robin Gutell for helping me understand the biological background of my work.

I extend many thanks to my colleagues, Weijia Xu, Rui Mao and Smriti Rajan Ramakrishnan.

Finally, I would like to express my gratitude to my parents, my sister and brother-in-law for love and support during my doctoral study.

Shu Wang

University of Texas at Austin

Austin, TX

On Multiple Sequence Alignment

Publication No._____________

Shu Wang, Ph. D

The University of Texas at Austin, 2007

Supervisors: Daniel Miranker & Tony Ambler

The tremendous increase in biological sequence data presents us with an opportunity to understand the molecular and cellular basis for cellular life. Comparative studies of these sequences have the potential, when applied with sufficient rigor, to decipher the structure, function, and evolution of cellular components. The accuracy and detail of these studies are directly proportional to the quality of these sequences alignments. Given the large number of sequences per family of interest, and the increasing number of families to study, improving the speed, accuracy and scalability of MSA is becoming an increasingly important task. In the past, much of interest has been on Global MSA. In recent years, the focus for MSA has shifted from global MSA to local MSA. Local MSA is being needed to align variable sequences from different families/species. In this dissertation, we developed two new algorithms for fast and scalable local MSA, a three-way-consistency-based MSA and a biclustering -based MSA.

The first MSA algorithm is a three-way-Consistency-Based MSA (CBMSA). CBMSA applies alignment consistency heuristics in the form of a new three-way alignment to MSA. While three-way consistency approach is able to maintain the same time complexity as the traditional pairwise consistency approach, it provides more reliable consistency information and better alignment quality. We quantify the benefit of using three-way consistency as compared to pairwise consistency. We have also compared CBMSA to a suite of leading MSA programs and CBMSA consistently performs favorably.

We also developed another new MSA algorithm, a biclustering-based MSA. Biclustering is a clustering method that simultaneously clusters both the domain and range of a relation. A challenge in MSA is that the alignment of sequences is often intended to reveal groups of conserved functional subsequences. Simultaneously, the grouping of the sequences can impact the alignment; precisely the kind of dual situation biclustering algorithms are intended to address. We define a representation of the MSA problem enabling the application of biclustering algorithms. We develop a computer program for local MSA, BlockMSA, that combines biclustering with divide-and-conquer. BlockMSA simultaneously finds groups of similar sequences and locally aligns subsequences within them. Further alignment is accomplished by dividing both the set of sequences and their contents. The net result is both a multiple sequence alignment and a hierarchical clustering of the sequences. BlockMSA was compared with a suite of leading MSA programs. With respect to quantitative measures of MSA, BlockMSA scores comparable to or better than the other leading MSA programs. With respect to biological validation of MSA, the other leading MSA programs lag BlockMSA in their ability to identify the most highly conserved regions.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Biological Preliminaries

1.1.1 Biological Sequences

Two kinds of biopolymers are represented as sequences in biological databases: protein and nucleic acids.

1.1.1.1 Nucleic Acid Sequences

Nucleic acid sequences are better known as DNA and RNA. DNA carries the primary genetic information of a living organism. It is a long chain, made from two complementary strands that stick together in a double helix form. The two strands are connected by base pairs like the rungs in a ladder. Each base A, C, G, T can only pair with only one base. A pairs with T, and G pairs with C (with rare exceptions). Since the paired strands can be deduced from each other, a DNA sequence is reported as a linear string of bases on one single strand. By way of analogy, it is similar to information stored on a computer's hard drive. It is the "hard-copy" of the genetic material.

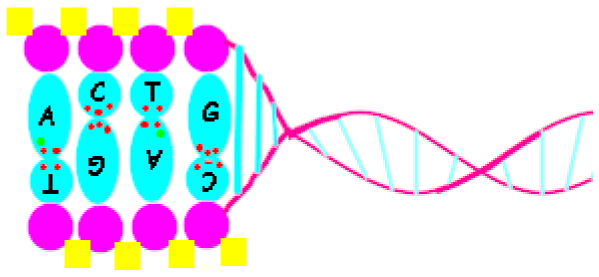


Figure 1.1 DNA Structure (cited from [43])

RNA serves as a "messenger", which delivers the genetic information of DNA to the place where proteins are made. RNA could be compared to information stored in a computer's cache in that the lifetime of RNA is much shorter than that of either DNA or the average protein. RNA is very similar to DNA. The main differences are: it uses base U instead of T (U pairs with A) and it is single-stranded.

1.1.1.2 Protein Sequences

Proteins are building blocks of cells and involved in many life-essential functions. By way of analogy, they could be viewed as the "result of execution" of the cell. They are the physical representation of the abstract information contained within the genome.

All proteins are polymers made from an assortment of 20 essential amino acid "residues". They are labeled A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y and V. Each protein links its amino acids in a chain and self-folds the chain into a specific 3-D structure [44, 45]. There are three structural levels in the conformation of proteins. A fourth level may be present when a protein consists of more than one chain. The primary structure is the first level of a protein structure [44, 45]. It is a linear chain of amino acids linked together by peptide bonds. The sequence we usually refer to means the primary structure of a protein. In the second level, called "secondary structure", the sequence of amino acids is further linked by hydrogen binding to form highly patterned sub-structures in space. There are two common types of secondary structures, alpha helix, and beta pleated sheet. All residues which cannot be classified into one of these two classes are usually referred to as random coil. In the third level, called "tertiary structure", alpha helices and beta sheets fold themselves further and cross-link with one another via their side chains to form a unique 3-D structure. Many proteins are actually assemblies of more than one chain, which in the context of the larger assemblage are known as protein subunits. In addition to the tertiary structure of the subunits, multiple-subunit proteins possess the fourth level, quaternary structure, which is the arrangement into which the subunits assemble.

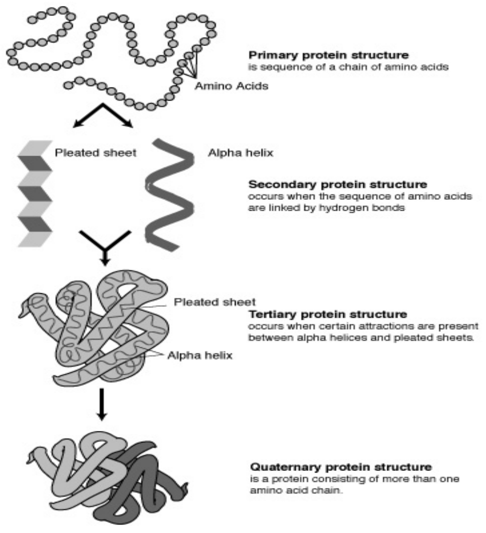

Primary protein structure
is sequence of a chain of amino acids
Amino Acids
Pleated sheet
Alpha helix
Secondary protein structure
occurs when the sequence of amino acids are linked by hydrogen bonds
Pleated sheet
Tertiary protein structure
occurs when certain attractions are present between alpha helices and pleated sheets.
Alpha helix
Quaternary protein structure
is a protein consisting of more than one amino acid chain.

Figure 1.2 Protein Structure (cited from [44])

1.1.2 Central dogma of molecular biology

The central dogma of molecular biology was first stated by Francis Crick in 1958 [114]. The dogma is a framework for understanding the transfer of sequence information between DNA, RNA, and protein. The dogma classifies the transfers into 3 types: general transfers (believed to occur normally in most cells), special transfers (known to occur, but only under abnormal conditions), and unknown transfers (believed to never occur).

The general transfers describe the normal flow of biological information: DNA can be copied to DNA (DNA replication), DNA information can be copied into mRNA, (transcription), and proteins can be synthesized using the information in mRNA as a template (translation) [115].

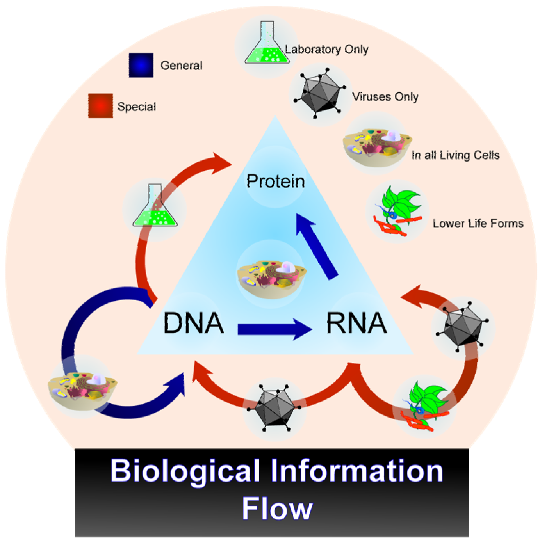


Figure 1.3 Central Dogma of Molecular Biology

[Cited from http://upload.wikimedia.org/wikipedia/en/4/46/CDMB2.png]

1.1.2.1 Gene

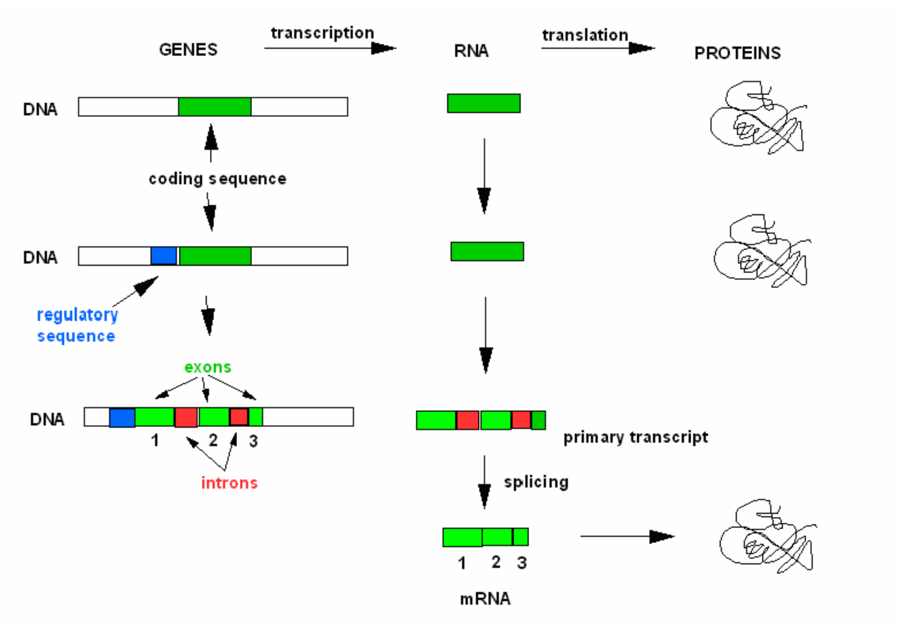


Figure 1.4 Transcription and Translation

A gene is a length of DNA which codes for a particular protein, or in certain cases, a functional or structural RNA molecule [116]. A gene carries biological information in a form that must be copied and transmitted from each cell to all its progeny. Genes can be as short as 1000 base pairs or as long as several hundred thousand base pairs. A gene can even be carried by more than one chromosome.

1.1.2.2 Transcription

The goal of transcription is to make an RNA copy of a gene. Specifically, transcription is the process by which the gene information is transferred to a newly assembled piece of messenger RNA (mRNA). This RNA can direct the formation of a protein or be used directly in the cell. For a given gene, only one strand of the DNA serves as the template for transcription. An example is shown below. The bottom (blue) strand in this example is the template strand. The enzyme RNA polymerase synthesizes an mRNA in the 5' to 3' direction complementary to this template strand.

```
5'  T G A C C T T C G A A C G G G A T G G A A A G G  3'
3'  A C T G G A A G C T T G C C C T A C C T T T C C  5'
5'  U G A C C U U C G A A C G G G A U G G A A A G G  3'
```

Figure 1.5 Transcription Example

(Cited from http://employees.csbsju.edu/hjakubowski/classes/ch331/dna/oldnacentdogma.html)

Transcription and translation differ in eukaryotes and prokaryotes. Specifically, eukaryotes have intervening sequences of DNA (introns) within a given gene that separating coding fragments of DNA (exons). In this case, splicing is needed: the introns are sliced out and exons joined in a contiguous stretch to form messenger RNA (mRNA).

1.1.2.3 Translation

Translation is the process that converts an mRNA sequence into a chain of amino acids that form a protein [117]. In prokaryotic cells, which have no nuclear compartment, the process of transcription and translation may be linked together. In eukaryotic cells, the site of transcription (the cell nucleus) is usually separated from the site of translation (the cytoplasm), so the mRNA must be transported out of the nucleus into the cytoplasm, where it can be bound by ribosomes. The ribosome is a multi-subunit structure containing rRNA and proteins. It is the "factory" where amino acids are assembled into proteins. The mRNA is "read" by the translational machinery, ribosome, in a sequence of nucleotide triplets called "codons". Each of those triplets codes for a specific amino acid. tRNAs are small non-coding RNA chains that transport amino acids to the ribosome. This translation process continues until a "stop" codon appears in the mRNA sequence.

1.1.2.4 DNA duplication

As the final step in the Central Dogma, to transmit the genetic information between parents and progeny, the DNA must be replicated faithfully so the cycle can repeat DNA → RNA → protein in a new generation of cells or organisms. DNA replication is the process of copying a double-stranded DNA molecule. It requires that the two strands of the DNA double helix be unwound and separated; each strand is then used as a template to produce two new daughter strands from the original parental duplex DNA.

1.2 The Multiple Sequence Alignment (MSA) Problem

All living organisms are related to each other through evolution. This means: a pair of organism, no matter how different, has a common ancestor sometime in the past, from which they evolved.

Given a set of related sequences, MSA tries to "reverses" the course of evolution, reveals similarity among the sequences, and recovers the mutations that took place. Quoting from Athur lesk [118]: "One or two homologous sequences whisper... A full multiple alignment shouts out loudly". Gaps may be inserted into sequences to shift residues so that identical or similar residues are aligned in the same column. The discovered commonality/similarity of sequences is used in many areas: characterizing sequence families, inferring the evolutionary history of sequences, searching for homologues in sequence databases, and identifying functionally or structurally important sites, such as phylogenetic footprints [1, 2, 3].

MSA covers two closely related problems: global MSA and local MSA. Global MSA aligns sequences over their whole length, while local MSA aligns over parts of sequences, and tries to locate short conserved regions among them (Figure 1.1) [1,2,4].

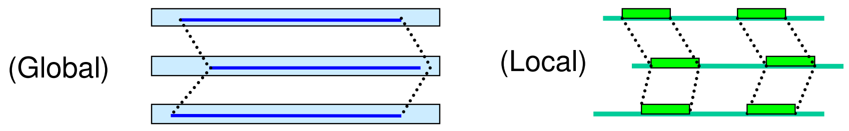


Figure 1.6 Comparison of Global MSA and Local MSA (Schematic)

Global MSA is used when sequences are similar over their whole length. It is very effective when aligning sequences from the same family to deduce their evolutionary history [1]. For example, Prosaposin is an injury-repair protein that exists in various tissues and blood fluids [5]. The Prosaposin sequences of different vertebrates (human, mouse, rat, chicken and zebrafish) evolved from a common ancestor and are known to be similar over their entire length. Based on the global multiple alignment of their Prosaposin sequences, a phylogeny tree of human, chicken, mouse, rat and zebrafish has been successfully reconstructed (Figure 1.7). Figure 1.7 (a) shows the global alignment of

Prosaposin protein sequences [6]. In the alignment, four Saposin domains of Prosaposin sequences are correctly identified: each domain has approximately 80 amino acids and contains six cysteines as homologous sites. Based on the above alignment, a maximum-likelihood phylogenetic tree of human, chick mouse, rat and zebrafish has also been reconstructed (Figure 1.7(b)). It shows the evolutionary path taken to get to the current diversity of vertebrates.

```
Chicken      MARRLLTLLGLLAAAVAACFVAEGLC-KGSEVWCQSLRTASQCGAVKHCQQNVWSKPAVN
Human        -MYALFLLASLLGAALAGPVLGLKECTRGSAVWCQNVKTASDCGAVKHCLQTVWNKPTVK
Mouse        -MYALALFASLLATALTSPVQDPKTCSGGSAVLCRDVKTAVDCGAVKHCQQMVWSKPTAK
Rat          -MYALALLASLLVTALTSPVQDPKICSGGSAVVCRDVKTAVDCRAVKHCQQMVWSKPTAK
Zebrafish    ---MMLLTLLLVTTAVASPLLGTEQCARGPPYWCQNVKTASLCGAVQHCQQNVWNKPQMK

                                   Saposin A
Chicken      SIPCDLCKELVTVVGKVLKDNGTEDEIRSYLEKTCEFLPDQGLASECKEIVDSYLPVIMD
Human        SLPCDICKDVVTAAGDMLKDNATEEEILVYLEKTCDWLPKPNMSASCKEIVDSYLPVILD
Mouse        SLPCDICKTVVTEAGNLLKDNATQEEILHYLEKTCEWIHDSSLSASCKEVVDSYLPVILD
Rat          SLPCDICKTVVTEAGNLLKDNATEEEILHYLEKTCAWIHDSSLSASCKEVVDSYLPVILD
Zebrafish    TVPCDLCKEVLVVVEQLLKDNVTESELLGYLEKACQLIPDEGLANQCKEIVTTTSQFSWA

Chicken      MIKEEFDKPEVVCSALSLCQSLQKHLAAMKLQKQLQSNKIPELDFSELTSPFMANVPLLL
Human        IIKGEMSRPGEVCSALNLCESLQKHLAELNHQKQLESNKIPELDMTEVVAPFMANIPLLL
Mouse        MIKGEMSNPGEVCSALNLCQSLQEYLAEQN-QKQLESNKIPEVDMARVVAPFMSNIPLLL
Rat          MIKGEMSNPGEVCSALNLCQSLQEYLAEQN-QRQLESNKIPEVDLARVVAPFMSNIPLLL
Zebrafish    SSKGELDDPGVVCGALGLCVSQQAALAKA----QLTSNEIPQVDLNQRVSPFLLNIPQLL

                                           Saposin B
Chicken      YPQDKPKQKS--KATEDVCQDCIRLVTDVQEAVRTNATFVKSLVAHAKEECDRLGPGMSD
Human        YPQDGPRSKPQPKDNGDVCQDCIQMVTDIQTAVRTNSTFVQALVEHVKEECDRLGPGMAD
Mouse        YPQDHPRSQPQPKANEDVCQDCMKLVSDVQTAVKTNSSFIQGFVDHVKEDCDRLGPGVSD
Rat          YPQDRPRSQPQPKANEDVCQDCMKLVTDIQTAVRTNSSFVQGLVDHVKEDCDRLGPGVSD
Zebrafish    YP-EEKRETP--KQKGDVCQDCVTFISDTQDEARVNSSFINTLIAQVENQCELLGPGMSD

Chicken      MCKSYISEYSDLAIQMMMHMQPKDICAMVGFCPSVK-SVPLQTLVPAQVVHEVKMETVEK
Human        ICKNYISQYSEIAIQMMMHMQPKEICALVGFCDEVK-EMPMQTLVPAKVASKNVIPALEL
Mouse        ICKNYVDQYSEVCVQMLMHMQPKEICVLAGFCNEVK-RVPMKTLVPATETIKNILPALEM
Rat          ICKNYVDQYSEVAVQMMMHMQPKEICVMVGFCDEVK-RVPMRTLVPATEAIKNILPALEL
Zebrafish    MCKEYISQYGPLVFQQLMSMQPKDICARAGFCPTKQKSVPMEKLLPAKSIPAVKMFPAVK

                                              Saposin C
Chicken      AT-----------VQEKTFSVCEICETMVKEVTGLLESNKTEEEIVHEMEVVCYLLPASV
Human        VEP-----IKKHEVPAKSDVYCEVCEFLVKEVTKLIDNNKTEKEILDAFDKMCSKLPKSL
Mouse        MDP-----YEQNLVQAHNVILCQTCQFVMNKFSELIVNNATEELLVKGLSNACALLPDPA
Rat          TDP-----YEQDVIQAQNVIFCQVCQLVMRKLSELIINNATEELLIKGLSKACSLLPAPA
Zebrafish    VEKPVATMPAKNLVRVRDSPQCAICEYVMKEIENMIQDQTSEAEIVQAVEKVCNILPSTL

Chicken      KDQCKDFIEVYGQALIDMLLEATNPEAVCVMLKCCAAN----------------------
```

```
Human        SEECQEVVDTYGSSILSILLEEVSPELVCSMLHLCSG-----------------------
Mouse        RTKCQEVVGTFGPSLLDIFIHEVNPSSLCGVIGLCAARPELVEALEQPAPAIVSALLKEP
Rat          STKCQEVLVTFGPSLLDVLMHEVNPNFLCGVISLCSANPNLVGTLEQPAAAIVSALPKEP
Zebrafish    TAQCKDLIETYGQAIIDLLVQEADPKTVCSFLALCSG-----------------------

                                     <------------  Saposin D  ------------
Chicken      ---------KPPQQPVVVKP--AGGFCDICKMIVAYADKELEKNATTTEIEALLEKVCHF
Human        --------TRLPALTVHVTQPKDGGFCEVCKKLVGYLDRNLEKNSTKQEILAALEKGCSF
Mouse        TPPKQPAQPKQSALPAHVPPQKNGGFCEVCKKPVLYLEHNLEKNSTKEEILAALEKGCSF
Rat          APPKQPEEPKQSALRAHVPPQKNGGFCEVCKKLVIYLEHNLEKNSTKEEILAALEKGCSF
Zebrafish    ---------VSHVPVMDKQHFAAGGFCDVCKMAVRYVDGILEQNATQSEIEEAVLKVCSF

             ------------------------------------------------->
Chicken      LPESVSDQCVQFVEQYEPVVVQLLAEMMDPTFVCTKLGVCGAAKKPLLGDDACVWGPGYW
Human        LPDPYQKQCDQFVAEYEPVLIEILVEVMDPSFVCLKIGACPSAHKPLLGTEKCIWGPSYW
Mouse        LPDPYQKQCDDFVAEYEPLLLEILVEVMDPGFVCSKIGVCPSAYKLLLGTEKCVWGPSYW
Rat          LPDPYQKQCDEFVAEYEPLLLEILVEVMDPSFVCSKIGVCPSAYKLLLGTEKCVWGPGYW
Zebrafish    LPYAVKDECNQLIEQYEPLLVQLLLQTLDPDFVCMKLGACPEAVQRLLGLNQCSWGPAYW

Chicken      CKNMETAAQCNAVDHCRRHVWN
Human        CQNTETAAQCNAVEHCKRHVWN
Mouse        CQNMETAARCNAVDHCKRHVWN
Rat          CQNSETAARCNAVDHCKRHVWN
Zebrafish    CKNVQTAARCNALNHCRRHVWS
```

Figure 1.7 (a) A Global Multiple Alignment of Five *Prosaposins* Protein Sequences from Human, Chicken, Mouse, Rat and Zebrafish. [cited from [6]]

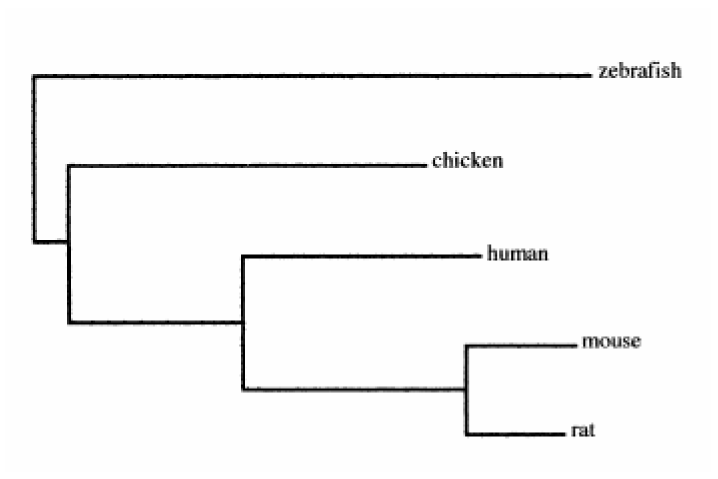


Figure 1.7(b) The Maximum-likelihood Phylogeny Tree of Five Prosaposin Protein Sequences [cited from [6])

However, in many biological applications, local MSA is more meaningful than global MSA [1, 2]. For DNA, only 10% represents coding genes. The sequences outside of coding regions generally tend to be not well reserved among organisms, except those functional sites involved in the regulation of gene expressions (e.g. transcription factor binding sites in promoters) [7,8]. When global MSA is applied, it is likely that the noise of the diverged nonfunctional background will overshadow the short conserved sites and make them undetectable. In this case, local MSA is needed to identify those conserved regions. For protein sequences, local MSA is also very important [1]. Over the evolutionary history, the amino acids in biologically critical parts are often well reserved, while the other parts undergo lots of mutations. Hence it is very often the case that related protein sequences differ significantly except in certain shared conserved regions. Local MSA is a natural response to identify those conserved regions. The identified conserved regions contain very valuable functional and structural information, which usually corresponds to [1,3]:

- Active sites of an enzyme
- Sequence family signatures
- Sites involved in the binding of the protein to its substrate or to another protein
- Other structurally and functionally important sites (e.g. domain)

Below is an example of the local MSA of the ß-lactamase fold super-family. The enzyme super-family has a distinct fold with two domains each supported by a separate ß-sheet and a α-helix lying to the exterior of the ß-sheet (Figure 1.8). Outside their two conserved domains, the super-family has low overall similarity.

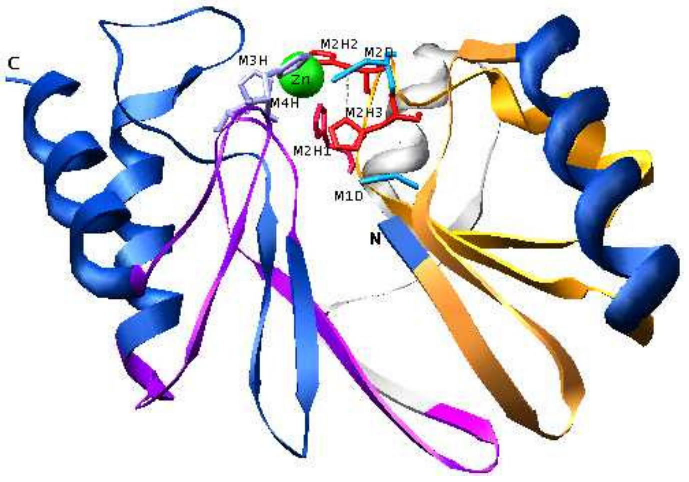


Figure 1.8 A Structural Model of the Core ß-lactamase Fold
(cited from [9])

Figure 1.9 shows the local multiple alignments of ß-lactamase fold super-family [9]. In the alignment, four conserved regions have been identified. It is found that they correspond to functionally important sites: the domains. Conserved regions 1 and 2 are mapped to domain 1 (shown in yellow), and conserved regions 3 and 4 are mapped to domain 2 (shown in purple). It is also found that the second conserved region corresponds to active site and zinc coordination sites: the 2nd conserved region has the HxHxDH signature (x represents any amino acid), the first two H residues participate in zinc coordination while the conserved D residue projects close to the active site and participates in the hydrolysis reaction.

```
                           Conserved Region 1      Conserved Region 2
GLO2_Hi_1073883        15 YQRENLPLIIVDLPET 10 NATIEAVLLTHEHDDHTQGVSAFKKRYPTVPI  69 \
GLO2_Ec_1786406        16 LNDEAGRCLIVDPGDA 10 NWQPEAIFLTHHHHDHVGGVKELVEKFPQIVV  80  |
GLO2_Sc_2494851        21 SDSKNKKSWLIDPAEP 11 KISVEAIVNTHHHYDHADGNADILKYLKEKNP  45  |
GLO4_Sc_2494852        31 STEDRRNSWLIDPAEP 11 KKSIDAIVNTHHHYDHSGGNLALYSILCQENS 116  |
PksB_Bs_2634080        24 EDIVSRSAIVVDPSWE 11 EAELKAVALTHSHYDHVNLVDPLTKMFNAQVY  38  | 1
YqgX_Bs_1731031        16 FLISDDQCLIFDPGGE 11 GLTPLAILLTHAHFDHIGALDEVREKWDIPVY  15  |
YybB_Bs_586832         28 FIKDGDDVYIVDTGIE 12 IGNPKAILLTHGHSDHIGGASKWLERFDIPIF  14  |
YflN_Bs_2443236        37 LPSEPHSFVLIDAGMP 16 GFQLKAIILTHGHFDHIGAIEEILEHWDVPVY  17 /
MJ0534_Mj_2826294       39 YLILDKNNVIIDTTRI 14 NLKLDYIISNHISPDHNECIEKLIELTEAKIV 165 \
MJ0732_Mj_2496115      38 YLIKDKKNVLIDTAKD 15 PKDLDYIIVNHVEKDHSGCVDKLVEISNATII 164  |
MJ0748_Mj_2496117      41 LVFGDEKVALIDNTYP 19 EFKIDVIVQNHVEKDHSGALPEIHKKFPDAPI 128  |
HP1430_Hp_2829454     157 VIETPKSAIVIDAGMS 22 KDKIAGIIITHAHEDHIGATPYLFKELQFPLY 349  ]
MG139_Mg_1723112       32 GIEYDDEIIIIDCGIK 22 QSKVKALFITHGHEDHIGGVPYLLKQVDIPVI 186  |2
MTV002.17c_Mtu_2624274  2 VFEHLGRLLIIDCGVL 22 LDDIEALVLTHGHEDHIGAIPFLLKLRPDIPV 177  |
MJ0861_Mj_2128073      19 AVNVDGEIIILDMGIR 35 EGEVKAIVLSHGHLDHIGAVPKLAHRYNAPII 195  |
YkqC_Bs_2833392        25 AVQFQDEIVLIDAGIK 22 EDKIKGLFITHGHEDHIGGIPYLLRQVNIPVY 181  |
YmfA_Bs_2634050        25 VIEIDSDIFVVDAGLM 22 ADRVKAIFLTHGHDENIGGVFYLLNKLSVPVY 181 /
CPSF_Bt_1707412        28 ILEFKGRKIMLDCGIH 16 PAEIDLLLISHFHLDHCGALPWFLQKTSFKGR 282 \
YLR277c_Sc_1077401     25 ILQYKGKTVMLDAGIH 16 LSKVDILLISHFHLDHAASLPYVMQRTNFQGR 365  |
F09G2.4_Ce_2435621     21 LLQVDGDYILLDCGWD 14 IPKISAVLISHPDPLHLGGLPYLVSKCGLTAP 373  | 3
MJ0047_Mj_2495763      24 EIKTDKSKILLDCGVK 13 IRDVDKVFISHAHLDHSGALPVLFHRKMDVPV  48  |
MJ0162_Mj_2495836      18 EVETQKGRVLLDCGMS 10 DKAVDAVIVSHAHLDHCGAIPFYKFKKIYCTH  49 /
SNM1_Sc_267010        219 IKFNNGHEIVVDGFNY  2 SETISQYFLSHFHSDHYIGLKKSWNNPDENPI  43 \  4
F39H2.1_Ce_1667320    258 QQIKIGEDISVDYFLK    KSGSRYNFLTHAHSDHYRGLDKKWTRSVYCSP  42 /
TREP3H1               24 LLRREGELFLFDAGEG 11 WKKISAIFISHTHADHITGLPGLLMLSSQVAR  55 \
ELAC_Ec_1788603        32 QHPTQSGLWLFDCGEG 11 PGKLDKIFISHLHGDHLFGLPGLLCSRSMSGI  51  |
ARS_Aca_114226         52 AMLVNGNTYVVDAGDG 11 IKNVDAVFLSHLHFDHTGGLPAILSLRWQTSA  55  | 5
CH3H1                 25 LLRWNGEGLLFDPGEG 11 PTVVSRIFISHFHGDHCLGLGSMLMRLNLDRV  61  |
BB0755_Bb_2688688      24 LIEYDGDNFLFDCGEG 11 WQKIKMICITHLHADHITGLLGIVMLMSQSGE  62  |
YqjK_Bs_1731073        25 LLEERRSVWLFDCGEA 11 PRKIEKIFITHMHGDHVYGLPGLLGSRSFQGG  54 /
YK59_Sc_549637        496 DGNTINRNIMLDAGEN 17 FQDLKMIYLSHLHADHHLGIISVLNEWYKYNK  95 \  6
YATA_Sp_1723232       531 DSAISMKNILLDCGEG 16 IASLRWIYISHMHADHHAGVIGVLKAWTKYSD  78 /
CH3H2                 43 WVQSQGKNFIIDTGPD  9 VPRLDGVFLTHPHYDHIGGIDDLRSWYITHLE  43 \  7
PHNP_Ec_1790530        42 VVKFNDAITLIDAGLH  7 PGSFQQFLLTHYHMDHVQGLFPLRWGVGDPIP  21 /
BLA1_Xm_1705478        62 LVQTPDGAVLLDGGMP 16 PRDLRLILLSHAHADHAGPVAELKRRTGAKVA  38 \  8
BLA2_Bsp_115023        74 VLNTSKGLVLVDSSWD 15 QKRVTDVIITHAHADRIGGITALKERGIKAHS  11 /
Deh_Scoel_282554       28 IVVGGDGALVVDTLST 15 AGPGRTVVNTHFHGDHAFGNQVFAPGTRIIAH  87 \  9
AF0090_Af_2650558      28 LIVGEEFSVVVDSVCN 15 AKDFRILINTHGHPDHVWTNHLFDAVAVAHEM  84 /
romA_Eclo_227029      113 YLQLAGKRILIDPVLG 23 MPEIDLLIISHDHYDHLDYATIRALLPKVKRV  46 >  10
AF1264_Af_2649231      10 FLLEGSMKVLIDPFLT 10 EVKADYILVTHGHGDHLGDAVEIAKRNNAPII  34 \  11
c04023_Sso_1707795     12 LLTFGNKNVIIDPMIK 12 KNNLDIIIVTHDHYDHLGDTVELLRMNPKAKL  30 /
YycJ_Bs_1064810        17 YLETEDHAFLVDAGLS 14 LDDVDGIFVTHEHSDHIKGLGVVARKYKLPIY  69 >  12
MJ0448_Mj_2495991      22 LIEINNKRILFDAGQN 13 KEGFDYIVLSHGHYDHCDGLKYVIENDLINGK   1 >  13
C03F11.2_Ce_1049467     8 MVYDGGHYIVVDSPSA 19 PGEIQYVVTTHGHPDHFGQGNFFPNARHFFGS  13 >  14
GumP_Xcam_2226283      39 IRHPQRGALLYDTGYA 39 LEDIGWCLISHFHADHVGGLRDLPTARFVCLH   5 >  15
YddR_Bs_1881317        11 VVKYANKKFLIDPMLA 30 LDGVDAVIVTHLHLDHFDDVAKNVLPKNIKMF  32 \
MJ0888_Mj_2842578      14 YLIIGKKNILIDPGTS 14 IKDIDLIINTHCHFDHTSADYLIEEYFNCPTI  10  |
MJ0301_Mj_2495901      39 ESNGIKKRILFDTATY 14 PKSIDMIILSHNHFDHTGGLFGIMKEINKEIP  23  |
YhfI_Bs_2226242        22 LFQSGDYSLLVDCGSA 10 AEKLDAVVLSHYHHDHIADIGPLQFAKQVGSF  49  | 16
YobT_Bs_2619046        24 LVEEENEVTLIDAALP 13 GKPLQHILLTHAHGDHVGSLDTLAQTFPHAKV  15  |
YqjP_Bs_1731077        22 YLVKGDALTLIDAGPN 18 LSDIEQVVLTHHHADHAGLLDVFSDEIEVIGH  82  |
YtnP_Bs_2293208        29 LIQKDGLNIIIDAGIG 30 VADIDVIAMTHLHFDHACGLTEYEGERLVSVF  18  |
AF2386_Af_2650713      39 LLCPNDGAILFDTGGD 14 KSKIKACFISHRHADHTGGLEWLDDKTEVFFP  22 /
MTH751_Mta_2621840     19 IDGIAGMNIHIDPGPG 11 PRKLDAVMVSHSHTDHYTDAEVLIEAMTRGMT  47 >  17
AF0504_Af_2650122      32 FVKTADVSILIDPGVS 28 MKKAEVAIITHYHYDHHDPNEVEIFSGKKLLL  46 >  18
AF1497_Af_2649063      20 CNVVVVGDVVIDAGAG  5 KVNASLLVLSHLHPDHSSGAWLFKDVLAPAEG  83 >  19
sepA_Lm_1044888       109 IMEGDTGLVITDTLLS 16 KKPIKAIIYTHSHADHYGGVAGLISKEDVASG  57 \
EPA_Shesp_2529414     130 LIRSDNGWIAYDVLLT 19 DLPVVAMIYSHSHADHFGGARGVQEMFPDVKV  | 20
ALKSP_Psp_77793         0 MIEAPEGLIIVDTGES 15 DKPIKAIVYTHFHPDHINGVKAFVSEEQVKSG 309  |
YOL164w_Sc_2132027    119 IIEGNTSLIIIDTLFT 16 QKPVRTVIYTHSHSDHYGGVKGIVKEADVKSG 302 /
ComEC_Bs_1303798      514 GAPHQRGRVLIDTGGT 30 IKQLDALILTHADQDHIGEAEILLKHHKVKRL  77 \
REC2_Hi_1172877       548 LIVKNGKGILYDTGSS 18 GIVLEKLILSHDDNDHAGGASTILKAYPNVEL  82  | 21
COMA_Ng_1705994       459 LVRTANRHLLFDTGTV 15 VRRLDKLVLSHHDSDHDGGFQAVGKIPNGGIY  78 /
CNA1_Vfis_476787       46 KSEADSNFVMLDAGSV 31 KDRIKGYFISHAHLDHVAGLIISSPDDSKKPI  65 \
CNA1_Sp_544049         46 SDGAFQEIISLDGGSH 41 EQRIKTFLITHCHLDHIYGAVINSAMFGPQNP  70  | 22
CNA1_Sc_1705954        27 ARTEDPELIAVDGGAG 74 FQGITDYYITHPHLDHISGLVVNSPSIYEQEN  69 /
YrkH_Bs_1731127        54 MVISNGEAAIIDATRM 11 GATITHVFDTHLHADHISGGRVIAEKTKATYW  75 >  23
CNAMo_Mm_1363107      147 DLKLGDKRMVFDPWLI 21 LCKADLIYISHMHSDHLSYPTLKQLSQRRPDI 205 \  24
MJ0296_Mj_2495897      43 LIITDNNNIIVDTSTK 16 PNDIDVVINTHLHYDHIENNPIFKNATFYASP   4 /
Consensus 75%             h.......hllDsu..    ..plc.lhloH.H.DHhsuh..h.........
```

```
                                   Conserved Region 3       Conserved Region 4

GLO2_Hi_1073883        19 ILTANYQIDVIPTGGHTKQHV  36 LNTLPDETIVCPAHEYTLGN  69 \
GLO2_Ec_1786406        20 AFVLGHEFSVIATPGHTLGHI  36 LSALPDDTLVCCAHEYTLSN  80  |
GLO2_Sc_2494851        25 LHLGDLEITCIRTPCHTRDSI  48 GRQNWSKTRVYPGHEYTSDN  45  |
GLO4_Sc_2494852        25 YHLGNLRVTCIRTPCHTKDSI  48 GETNWNKVKIYPGHEYTKGN 116  |
PksB_Bs_2634080        24 ISIGNTRAQCLLTPGHTAGGM  39 KSEVSPHVRVYPGHSFGKSP  38  | 1
YqgX_Bs_1731031        39 LNIGPFHLETLFTPGHSPGSV  41 LLTLPEHTLVLSGHGPETDV  15  |
YybB_Bs_586832        39 HLPLKYYLTPGHSPGHVVYYH  38 IIDQIKPTLICSSHGEEILY  14  |
YflN_Bs_2443236        56 LDEWMWIATPGHTPGHISLFR  49 KLAGLEPEALLTGHGIPMTG  17 /
MJ0534_Mj_2826294      50 CVEDKILFSNDLFSQHVVYKE  33 ILKDLDLEYICPSHGVIWHI 165 \
MJ0732_Mj_2496115      53 CKEEKILFSNDAFGQHIASSE  34 AVKNLDIELICPSHGVIWKE 164  |
MJ0748_Mj_2496117      61 LFSNDAFGQHLCFPAHKRFDK  78 VYDTMHYSTQKMAHAFAEGL 128  |
HP1430_Hp_2829454      32 PISVGEFIIEWIHITHSIIDS 207 VAYQEFDNIHVSGHAAQEEQ 349  |
MG139_Mg_1723112       32 EFQTKHFKIDFYRVNHSIPDA 208 YENSSQLKLHASGHATQQEL 186  |2
MTV002.17c_Mtu_2624274  30 STRHGVFECEYFAVNHSTPDA 208 VVTNAQARVHVSGHAYAGEL 177  |
MJ0861_Mj_2128073      33 IDLTPNITLEFIRITHSIPDS 211 LGVRIFKGAHVSGHAAKEDH 195  |
YkqC_Bs_2833392        31 IVKFRKTAVSFFRTTHSIPDS 207 VIHGPLNDIHTSGHGGQEEQ 181  |
YmfA_Bs_2634050        31 VITFQSTKVSFFRTIHSIPDS 207 QVIFAQKRVHVSGHGSQEEL 181 /
CPSF_Bt_1707412        50 VKEVAGIKFWCYHAGHVLGAA 219 PLKMSVDYISFSAHTDYQQT 282 \
YLR277c_Sc_1077401      58 TVDVNGIKFTAFHAGHVLGAA 226 PRRCQVEEISFAAHVDFQEN 365  |
F09G2.4_Ce_2435621      53 LKGDSGVHFTALPAGHMLGGS 293 KDFRSFDGSENDAHTFDIMA 373  | 3
MJ0047_Mj_2495763      49 KKYYKDFSYELFSAGHIPGSA 213 KPNLEVCMYNFSCHAGMDEL  48  |
MJ0162_Mj_2495836      40 RQITENIKFKFYNAGHILGSA 215 PIRGKVVKIEFSAHGDYNSL  49 /
SNM1_Sc_267010        33 FWITDTISVVTLDANHCPGAI 274 YNKFQVFNVPYSEHSSFNDL  43 \  4
F39H2.1_Ce_1667320      30 PHKFDSFQVTLVNANHCPGAV 168 NDDEGIIRIPYSDHSSRSEI  42 /
TREP3H1              41 VYRGKDFQVRCFCLDHTKPCM  88 FEKGMEKDAAEKKHMTCVQA  55 \
ELAC_Ec_1788603        40 ILDDGLRKVTAYPLEHPLECY  88 LDITMEAKANSRGHSSTRQA  51  |
ARS_Aca_114226        14 TVDGIFEYMTYGTLGHYGVPG 107 GKFIGIHKHLSKHHLSPKQV  55  | 5
CH3H1                41 VEDFGNFRIESRQLDHLVDTL  77 ILLCESTYLEEHSHLAKSHY  61  |
BB0755_Bb_2688688      46 IYEDKTKKIEYTKLKHSIECV  87 FKNELKKEADKKLHLTAGGA  62  |
YqjK_Bs_1731073        40 VFEDDQFIVTAVSVIHGVEAF  88 FAKEDRKLAYDYYHSTTEQA  54 /
YK59_Sc_549637        93 YEDLSIEYFQTCRAIHCDWAY  48 LENQLLEDAVKKKHCTINEA  95 \  6
YATA_Sp_1723232        56 FKEFDLVSFRTVPAIHCPYSY  39 LEDSMHEIAIKKQHSTYSEA  78 /
CH3H2                28 NSLAASLRYTILNEKCGEQEF  54 GVLPKAFGSRTPSHLTLEQA  43 \  7
PHNP_Ec_1790530        30 VFDLQGLQVTPLPLNHSKLTF  63 VIRSPRVILTHISHQFDAWL  21 /
BLA1_Xm_1705478        39 ITVGGIVFTAHFMAGHTPGST  46 TVRALPCDVLLTPHPGASNW  38 \  8
BLA2_Bsp_115023        26 KFGNTKVETFYPGKGHTEDNI  42 LKRYRNINLVVPGHGKVGDK  11 /
Deh_Scoel_282554       41 HVGERQVELICVGPAHTDHDV  37 RLAELEPEVVVGGHGPVAGP  87 \  9
AF0090_Af_2650558      39 LYNDAEMQIIHPGVAHTRGDC  38 ELLNLDAKIYVPGHGGLAGE  84 /
romA_Eclo_227029       28 VHISDALTVHLLPARHFSGRG  68 ASVDLNAKAVVPGHNGRFVL  46 >  10
AF1264_Af_2649231      23 TARTGSIAVTMVPAWHSADLE  65 ALELVKPKVAIPMHYNTFPL  34 \  11
c04023_Sso_1707795      28 FVEVDGIKLALTKAVHSSTHS  56 VELIKPKKGAIPIHYNTWDL  30 /
YycJ_Bs_1064810        28 VKSFGGLDVESFGVSHDAAEP  51 PWSIKRRILSDVGHVSNEDA  69 >  12
MJ0448_Mj_2495991      79 DMFLIAKGILITGCSHSGIIN  52 LTKLSQLNNFVYGHVGKIIG   1 >  13
C03F11.2_Ce_1049467     17 IMQLTKNVQLWNTPGHTAQDV 163 QPQNEAEISKMMPHLKKWQT  13 >  14
GumP_Xcam_2226283      54 DLFADGSVMAVALPGHVPGQM  54 LVQAHPELAILPSHCQPSLD   5 >  15
YddR_Bs_1881317        23 DTVFEGIQLVKTKGEHGRGEE  69 VHKAAPHAKIISVHMEAVNH  32 \
MJ0888_Mj_2842578      38 EELKSYGLEIIRTPGHTYGSI  41 IANERNIDKLYPGHGEIGDR  10  |
MJ0301_Mj_2495901      93 AIVTEKGLIIVSGCSHPGIVS  36 ALKKLGVKKICTGHCTGFKA  23  |
YhfI_Bs_2226242        38 PLTAGPFTITFLKTIHPVTCY  36 CNFYADQDGTSAGHMNSLEA  49  | 16
YobT_Bs_2619046        42 GGETIGSLLAIPTPGHTPGSM  50 LLADKAPSCLAVGHGKFLRS  15  |
YqjP_Bs_1731077        61 GIDGLEGWSVLEMPGHAESHI  47 RLSQLDPTIVFPGHGEPITS  82  |
YtnP_Bs_2293208        47 TEGITMHHTGGHSDGHSVLIC  43 AFAAEKDAWFIFYHDAEYRA  18  |
AF2386_Af_2650713      20 EQALIYKKIMLIGCSHPGIVR  33 MELRKFTDKIAPCHCTGEKG  22 /
MTH751_Mta_2621840      40 TVDIGDLEVTGTGTVHGDPTG  34 SSVIRPGDEHIRGHMCTDDF  47 >  17
AF0504_Af_2650122      36 YEFGNTVVELSKPVFHGADSR  70 LIAEDVKTLVLDHHLTRDLR  46 >  18
AF1497_Af_2649063      41 VVVKEPEIVAVPVKGHTMDHH  38 KLLDIDFEIFVSAHSKPVFG  83 >  19
sepA_Lm_1044888       100 QFKLLNIAEDAVHNLHNILTL  20 AFGDKYEVCIGQHHWPTWGN  57 \
EPA_Shesp_2529414      98 SKKALWTAELTYQGMHNIYTL  87 KTWHTNGYHGTYSHNAKAVY 238  | 20
ALKSP_Psp_77793       100 LISAEVTQGPTLPNVHTLRGT  13 KLRAFQADVMVPLHGQPVSG 309  |
YOL164w_Sc_2132027     100 QQRVLNMAEDVTHHMHNLYAL  20 AFGSKTDVLIAQHHWPTTGQ 302 /
ComEC_Bs_1303798       27 EVKRGDVLQIKDLQFHVLSPE  39 VFPNIKADVLKVGHHGSKGS  77 \
REC2_Hi_1172877        19 RDWHWQGLHFQILSPHNVVTR  32 ARTLGKIDVLQVGHHGSKTS  82  | 21
COMA_Ng_1705994         3       PEFYEGARHCAEQR  54 YGGNLYSQVLVLGHHGSNTS  78 /
```

```
CNA1_Vfis_476787       46 PVAETTMSVVSLPLSHSGGQS  53 SFTNETPDKSLFGHLTPNWL  65 \
CNA1_Sp_544049        42 TSLTTTLSILPFPVNHGSSFG  58 CSTPDIPDTLLFGHFCPRHL  70  | 22
CNA1_Sc_1705954       48 KCTIFPWDVIPFKVHHGIGVK  62 SCPLSSKPEQLYGHLSPIYL  69 /
YrkH_Bs_1731127       23 IGNTTIKIQPIYSPGHTIGST  43 YKALSKDLIVLPAHFMIIDE  75 >  23
CNAMo_Mm_1363107       48 VHPEMDTCIIVEYKGHKILNT  67 RIYCPFAGYFVESHPSDKYI 205 \  24
MJ0296_Mj_2495897      12 KKFKDKEIEIIETPGHTYGSI  39 KKIRKLRKNVITGHEGIVYK   4 /
Consensus 75%            ...............Hs....   ............sH......
```

Figure 1.9 An Alignment of Selected Representatives of the ß-lactamase Fold Superfamily (cited from [9])

(The residues were shaded according to a 75% consensus prepared using the CONSENSUS script of Nigel Brown (http://www. bork. embl-heidelberg. de/Alignment/consensus. html). The number before each motif is the insert length between blocks. The proteins are grouped together according to their families which are indicated by numbers to the extreme right of the alignment. These families are 1- the glyoxalase family, 2- the FD domain family, 3- the MG139 family, 4- CPSF family, 5- SNM1 family, 6- ElaC family, 7- YK59 family, 8-PHNP family, 9- ß-lactamase, 10- "Dehydrase" family, 11- RomA family, 12- MJ1163 family, 13-YycJ family, 14- MJ0448 family, 15-CE family, 16- GumP family, 17- ungrouped core cluster members, 18-MJ1374 family, 19-MJ1629 family, 20-AF1497 family, 21-alkyl sulfatase family, 22- Rec2 family, 23- phosphodiesterase family, 24- HAL family, and 25-unclustered members.)

1.3 Motivation

In the past, much of interest has been on Global MSA, which is exemplified by numerous existing global MSA methods, such as ClustalW [14], DCA [15], PRALIGN [19,20], T-Coffee [17], IterAlign [21], COFFEE [18] and SAGA [16]. Given the recent data explosion in sequence databases, the focus for MSA has shifted from global MSA to local MSA.

A fast local MSA tool is needed for sensitive database searching to detect remote homology. Homology refers to the similarity between sequences that results from inheritance of traits from a common ancestor [119]. It is assumed that when sequences are homologous, they tend to exhibit similarity in their structures and functions. Based on this assumption, the function of a new sequence can be predicted from the known, characterized functions of its homologous sequences. In bioinformatics, homology is often concluded on the basis of sequence similarity. Traditionally, pair-wise local alignment has been the routine procedure for inferring homology between a newly determined sequence and the known sequences in a database [2]. The pair-wise comparison approaches, such as BLAST [22], FASTA [24, 25] and BLAT [23], have led to identify the biological functions of many protein sequences. However, in many situations sequences have diverged to the extent that their weak similarities of pair-wise comparison are indistinguishable from chance similarities. Moreover, the noise levels of expanding sequence databases are increasing, making it even harder to detect weak similarity. For all these reasons, the function of many sequences in sequence databases remains unknown and a more powerful tool is needed for remote homology detection. One solution is to apply a local-MSA-based search for remote homology detection: multiple-alignment of a query's family/super-family members is used to search against a sequence database. The information afforded by the multiple-alignment represents the common features of the family, allowing an algorithm to identify homology with an evolutionarily remote sequence based on its family signatures, even if its similarity to each of the individual aligned sequences may be insignificant [3]. Today, PSI-BLAST [26] is the most popular MSA-based tool because of its high speed. However, PSI-BLAST does not really implement an MSA. The MSA it uses is only a collection of pairwise alignments between the query and its other family members. The poor multiple alignment quality of PSI-Blast limits its ability to identify highly divergent sequences. Probe [29] and SAM-T99 [27, 28] are the two well-known database searching tools which implement MSA within their tools. However both of them suffer from speed bottleneck. So the key challenge in the local-MSA based searching is how to improve the speed and quality of local MSA in order to support rapid database searching.

Faster and reliable local MSA tools are also needed for automatic construction of sequence family databases. With the exponential growth of sequenced data, sequence family databases are playing a more and more important role in sequence function annotations. Sequence family databases consist of collections of conserved regions, which are characteristic of sequence families and usually extracted from local multiple alignments [30]. Depending on different databases, the conserved regions can be represented as pattern, motif, profile, or fingerprint (or domains for larger regions) [30]. Family databases have the following advantages: (1) they allow sensitive database searching for remote homology detection, (2) they provide a higher-level of annotation, a family-based view of sequence's function function/structure, and (3) they reduce the redundancy of current sequence databases by organizing sequence data in a systematic way. Because of these characteristics, sequence family databases are becoming more and more important in post genomics. However, the development of family databases is lagging behind the pace of Genome sequencing. One of the key challenges is how to generalize the family signature for a superfamily. The sequences of a superfamily are usually distantly related and only share short conserved regions, which are easily overshadowed by the divergent background-sequences. Based on this, a reliable and accurate local MSA is needed. Currently many family databases, such as Prosite and Prints, are still based on manual construction of the conserved regions of sequence families. This labor-intensive process causes the coverage of family databases to be very limited, and for this reason is not widely used.

Recently, a new analytic method called "phylogenetic footprinting" is becoming more and more frequently used. It is a computational method in which MSA is applied to uncover regulatory elements in a set of non-coding regulatory sequences from multiple species [7, 8]. In biology, non-coding regulatory sequence describes the sequence that has not been identified as coding for mRNA transcription for protein translation, but is instead responsible for regulatory functions. Phylogenetic footprinting was first proposed in 1988. However, it was not applicable at that time, since non-coding sequences from a large number of species were not available yet. With the advances in Genome sequencing techniques, many sequences from different organisms have been identified and provide good sequence candidates for such analysis, making phylogenetic footprinting realizable. The standard method for phylogenetic footprinting used to be global-MSA-based: global MSA is first applied to align the regulatory sequences and then conserved regions are extracted from the alignment. As we have mentioned before, non-coding regulatory sequences generally tend to be not well-reserved among organisms, except those short regions involved in the regulation of gene expressions. In this case, local MSA is a more approximate tool. Cliften *et. al.* [31] and McCue *et al*. [31] have applied local MSA, and Gibbs Sampling, for phylogenetic foot-printing.

Local MSA is also needed for large-scale phylogeny reconstruction. With the availability of massive new sequences, large-scale phylogeny reconstruction from sequences has become possible. Scientists are collaborating to reconstruct the phylogeny, Tree of Life, to infer the evolutionary history of all living organisms [33]. Since MSA is the starting point in phylogeny reconstruction, this has generated great demands for MSA to meet the Tree of Life requirements: reliable local MSA is needed to align divergent sequences from diverse species. It is also desirable that MSA be able to scale up to align large datasets.

Hence, finding ways to improve the speed, accuracy and scalability of local MSA is becoming an increasingly important task. However, current existing local MSA methods cannot really achieve these requirements. We can roughly classify the existing popular local MSA methods into three classes: probability-based, progressive-based, and the block-based method. Probability-based methods, such as MEME [41,42] and Gibbs Sampling [37,38,39], start from an initial guess - a random alignment, then iteratively realign the sequences, and converge toward the set of conserved regions, whose probabilistic matrices have maximum likelihood. This approach is easily stuck into local optimum and is very time-consuming. A progressive-based method, such as Dialign2 [34,35], is a greedy heuristic method which works by building a multiple sequence alignment progressively. Its greedy approach makes it difficult to correct early mistakes. The block-based method, BlockMaker [40], uses spaced-triplet or the Gibbs sampling method to identify possible conserved regions, called "blocks". However both of the block-construction methods have their limitations: spaced-triplet can only identify limited types of conserved regions and the Gibbs sampling method is very time consuming.

1.4 CONTRIBUTIONS

The overall goal of this dissertation is to develop fast, scalable and accurate local MSA. Specifically, we developed two new algorithms for local MSA, a three-way-consistency-based MSA and a biclustering-based MSA.

The first local MSA algorithm is three-way-Consistency-Based MSA (CBMSA). CBMSA explicitly searched for locally conserved regions (blocks) and incorporated alignment consistency information into block-finding process. CBMSA uses alignment consistency heuristics in the form of a new three-way alignment method. Three-way alignments provide more reliable consistency information than traditional pairwise-alignments. Typically, the algorithmic cost of pairwise consistency is $O(N^3L^2)$ (e.g. T-Coffee) or $O(N^4L^2)$ (e.g. Dialign 2), where n is the number of sequences to be aligned and L is the average length of sequences. While our three-way consistency approach is still able to maintain the time complexity $O(N^3L^2)$, it provides better alignment quality. We quantify the benefit of using three-way alignments as compared to pairwise alignments. We have also compared our three-way consistency-based MSA to a suite of leading MSA programs, and our program consistently performs favorably.

We have also developed another MSA algorithm, a biclustering-based MSA. Biclustering is a clustering method that simultaneously clusters both the domain and range of a relation. A challenge in multiple sequence alignment (MSA) is that the alignment of sequences is often intended to reveal groups of conserved functional subsequences. Simultaneously, the grouping of the sequences can impact the alignment; precisely the kind of dual situation biclustering is intended to address. We define a representation of the MSA problem enabling the application of biclustering algorithms. We develop a computer program for local MSA, BlockMSA, that combines biclustering with divide-and-conquer. BlockMSA simultaneously finds groups of similar sequences and locally aligns subsequences within them. Further alignment is accomplished by dividing both the set of sequences and their contents. The net result is both a multiple sequence alignment and a hierarchical clustering of the sequences. BlockMSA was tested on the subsets of the BRAliBase 2.1 benchmark suite that display high variability and on an extension to that suite to larger problem sizes. Also, alignments were evaluated of two large data sets of current biological interest, T box sequences and Group IC1 Introns. BlockMSA was compared to a suite of leading MSA programs. Results for the benchmark suite are sensitive to problem size. On problems of 15 or greater sequences, BlockMSA is consistently the best. On the T box sequences, BlockMSA does the most faithful job of reproducing known annotations. MAFFT and PROBCONS do not. On the Intron sequences, BlockMSA, MAFFT and MUSCLE are comparable at identifying conserved regions.

The thesis is organized as follows: Chapter 2 reviews currently-existing multiple sequence alignment methods. Chapter 3 presents a new three-way-consistency-based local MSA method. Chapter 4 presents a new biclustering-based local MSA method. Chapter 5 concludes the work.

Chapter 2 Background and Related Works

2.1 Multiple Sequence Alignment Basics

Multiple sequence alignment (MSA) is the process of aligning a set of related sequences to highlight their commonalities [1, 2]. An alignment is evaluated by a scoring function, which usually rewards matches and penalizes mismatches and gaps. The goal of MSA is to find an optimal alignment to maximize the alignment score for a given set of sequences [1]. In order to evaluate an alignment, two keys issues usually need to be considered: (1) Scoring scheme and (2) an objective function based on the scoring scheme.

2.1.1 Scoring Scheme

There are two ways to quantify the sequence similarity: a similarity measure and a distance measure. A similarity measure is a function that measures the similarity of sequences. The distance measure is a function that measures the difference between sequences. Usually the larger the distance is, the smaller the similarity is, and vice versa. Depending on the different measurement we choose, we will have different scoring schemes: similarity scoring scheme and distance scoring scheme.

2.1.1.1 Similarity Scoring Scheme

An alignment can be regarded as a set of aligned residue-residue and residue-space pairs. Traditionally, residue-residue pairs are scored with a substitution matrix and residue-space pairs are scored with gap penalties.

Similarity Substitution Matrix.

For nucleic acid alignment, an identity matrix, giving fixed scores for matches and mismatches, is adequate. For example, we could give a score of 1 for matches and 0 for mismatches. For protein sequences, a more complicated scheme is needed since protein evolution and structure information needs to be taken into account. Currently the most widely used matrices are log-odds matrices, in which each entry M(i,j) represents the probabiliy of the ith amino acid being transformed into the jth amino acid over time [2].

$$M_{ij} = \log \frac{q_{i,j}}{p_i p_j} \tag{2.1}$$

where $q_{i,j}$ is the probability that amino acids i and j are aligned together, and p_i is the frequency of amino acid i. There are two major types of log-odds matrices, PAM and BLOSUM. PAM matrices [46,47] are derived from alignments of related proteins. Each PAM matrix has a number associated with it. This is a value used to measure the evolutionary divergence between two sequences: if two sequences are 1 PAM unit diverged, this means one sequence can be converted to another with an average of 1 accepted point-mutation per 100 amino acids. The PAM1 matrix is derived directly from the aligned sequences and other PAM matrices are extrapolated from PAM1. Among PAM matrices, PAM 250 is the most widely used. BLOSUM matrices are based on local alignments of divergent sequences [48]. All matrices are directly calculated; no extrapolations are used like in PAM. Different matrices are obtained by varying the minimum percent identity of aligned sequences. For example, BLOSUM 80 matrix was calculated from sequences with 80% identity. Lower numbered BLOSUM matrices are more suitable for divergent sequences, while higher BLOSUM matrices are more suitable for closely related sequences.

Gap Penalty

A gap is defined as "a maximal, consecutive run of spaces in a single string of a given alignment" [1]. How to define a gap penalty function is very important since it has a great influence on the distribution of spaces in an alignment, which subsequently affects the overall alignment.

In biology, many mutation events result in the insertion or deletion of an entire substring to a sequence. Insertion or deletion easily causes big gaps in an alignment. An affine gap penalty serves as a model for this. It encourages the extension of gaps rather than the introduction of new gaps. Currently it is the most widely used gap penalty model. The affine penalty for a gap of length g is $-d - (g-1)e$, where d is the gap open penalty and e is the gap extension penalty. The value of d is greater than e since opening a gap is more expensive than extending an existing gap.

2.1.1.2 Distance Scoring Scheme

A distance-scoring scheme, especially a metric distance-scoring scheme, has become more and more important. With the exponential growth in sequence databases, metric space indexing has emerged as one of the most effective strategies to manage massive sequence data and support fast on-line homology searching. In metric space indexing, sequences (usually divided into fixed length fragments) are treated as points in a metric space, and distance measurements between sequences should satisfy the mathematical axioms of a metric.

Definition Metric Space: A metric space is a set of data objects with a distance function, d, with the following properties [49]:

(i). $d(Ox, Oy) = d(Oy, Ox)$ (symmetry) (2.2)

(ii) $d(Ox, Oy) > 0$, $Ox \neq Oy$, and $d(Ox, Ox) = 0$ (non-negativity) (2.3)

(iii). $d(Ox, Oy) \leq d(Ox, Oz) + d(Oz, Oy)$ (triangle inequality) (2.4)

A key challenge in this approach is how to define a distance function, which satisfies the metric property and reflects the evolutionary information between sequences as well. Hamming distance and simple edit distance are mathematically-simple metric distance functions. However, evolutionary information is greatly lost during the distance calculations. Sequence alignment based on the popular substitution matrices, log-odd metrics, does contain evolutionary information. However, log-odd metrics are similarity-based and also do not satisfy the metric property. Based on this, a new metric substitution matrix, metric PAM250 (mPAM250), has been proposed recently [49]. It reworks the mathematics of PAM to make its values reflect evolution information from a distance perspective. PAM computes log-odds based on the frequency of mutations, but mPAM computes the expected time for a mutation to occur. At the same time, mPAM also satisfies the metric property. Below shows an example of mPAM.

	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	0	2	2	2	3	2	2	2	2	2	2	2	2	3	2	2	2	5	4
R	2	0	2	2	4	2	2	2	2	3	3	2	2	4	2	2	2	3	4
N	2	2	0	2	4	2	2	2	2	3	3	2	2	4	2	2	2	5	4
D	2	2	2	0	4	2	2	2	2	3	3	2	3	4	2	2	2	6	4
C	3	4	4	4	0	4	4	3	4	3	4	4	4	4	3	3	3	8	3
Q	2	2	2	2	4	0	2	2	2	3	3	2	2	4	2	2	2	5	4
E	2	2	2	2	4	2	0	2	2	3	3	2	3	4	2	2	2	6	4
G	2	2	2	2	3	2	2	0	2	2	3	2	2	4	2	2	2	6	4
H	2	2	2	2	4	2	2	2	0	3	3	2	3	3	2	2	2	5	3
I	2	3	3	3	3	3	3	2	3	0	1	3	2	2	2	2	2	5	3
L	2	3	3	3	4	3	3	3	3	1	0	3	1	2	3	3	2	4	2
K	2	2	2	2	4	2	2	2	2	3	3	0	2	4	2	2	2	4	4
M	2	2	2	3	4	2	3	2	3	2	1	2	0	2	2	2	2	4	3
F	3	4	4	4	4	4	4	4	3	2	2	4	2	0	4	3	3	3	1
P	2	2	2	2	3	2	2	2	2	2	3	2	2	4	0	2	2	5	4
S	2	2	2	2	3	2	2	2	2	2	3	2	2	3	2	0	2	5	4
T	2	2	2	2	3	2	2	2	2	2	2	2	2	3	2	2	0	5	3
W	5	3	5	6	8	5	6	6	5	5	4	4	4	3	5	5	5	0	4
Y	4	4	4	4	3	4	4	4	3	3	2	4	3	1	4	4	3	4	0
V	2	3	2	2	3	3	2	2	3	2	1	3	2	2	2	2	2	5	3

Figure 2.1 mPAM250

2.1.2 Objective Function

An objective function is used to evaluate the quality of an alignment. It is an important aspect of MSA. Ideally we would like to have a precise probabilistic model of sequence evolution to score a MSA [2]: given a correct phylogenetic tree T, which represents the MSA evolutionary relationship, the MSA score should be the product of all the evolutionary events necessary to produce the alignment via ancestral intermediate sequences times prior probability of root ancestral sequence. However this desired phylogeny model T is very complex: the probabilities of evolutionary change would depend on the evolutionary times along each branch of the tree, as well as position-specific structural and functional constraints imposed by natural selection. We do not have enough data to parameterize it. Hence simplifying assumptions must be made.

Currently the standard method of scoring MSA is the Sum of Pairs score (SP-score) [2]. In the SP-score, it is assumed that the columns of the alignment matrix are statistically independent, and the phylogenetic tree is ignored. The definition of SP score is:

Given: a substitution matrix that gives the similarity score s(x,y) for aligning two residues x and y; a MSA matrix $M_{N \times L}$, which has N sequences and each sequence's length is L. The SP-score $S(m_i)$ for the i-th column m_i of the MSA-matrix M is calculated as:

$$S(m_i) = \sum_{1 \le j < k \le N} s(m_i^j, m_i^k)$$, where m_{ij} is the j-th entry in the i-th column. (2.5)

The score for the whole matrix M is:

$$S(M) = \sum_{1 \le i \le L} s(m_i) \qquad (2.6)$$

Although the SP-score is easy to use and gives reasonable results, it has no probabilistic justification. That is, evolutionary events are over-counted [2]: Each sequence is treated as if it was directly evolutionarily related to all other N-1 sequences instead of a single ancestor. Altschul, Carroll & Lipman [3] recognized this problem and proposed a weighted SP-score to partially compensate for the defect in the SP-score. The weighted SP-score Sw is defined as:

$$S_w(m_i) = \sum_{1 \le j < k \le N} w(i,j) * s(m_i^j, m_i^k) \quad \text{and} \quad S_w(M) = \sum_{1 \le i \le L} s_w(m_i) \qquad (2.7)$$

Weight w(i,j) changes the importance given to different pairs of sequences and is intended to reflect known evolutionary distance. Thus using the weight, we could induce the multiple alignment to more accurately reflect known evolutionary history. Other variations of the original SP score also exist, including the use of different gap penalty schemes and different pair-wise scores. Another problem with SP-scores is they assume the substitution score of two residues is uniform and time-invariant at all positions in the alignment. This is unrealistic as the variability may range from total invariance at some positions to complete variability at others, depending on the functional or structural constraints of sequences. It is notable that there are still no ideal MSA score functions available. Among all of the existing MSA score functions, no one outperforms the others.

2.2 Exact MSA Algorithm

The exact MSA algorithm uses dynamic programming for multiple sequence alignment. Dynamic programming (DP) aligns all sequences simultaneously and attempts to optimize an overall objective function, SP-score. DP method has been successfully applied to pair-wise alignment. Conceptually it is straightforward to generalize the pair-wise DP algorithm to MSA. However, it is not feasible in practice because of its time and space complexity. Here we will first introduce pair-wise DP alignments and then introduce DP algorithms for multiple sequence alignment. Last, we will analyze the algorithm complexity.

2.2.1 Pairwise Alignment

Global Alignment

Needleman and Wunsch [53] proposed a dynamic programming algorithm to find the optimal global alignment of two sequences. There are three main steps in the approach: initialization, matrix-fill, and trace-back(55).

Step 1. Initialization.

Given two sequences $X_{1..n}$ and $Y_{1..m}$ to be aligned, we first construct a matrix $F_{(n+1)\times(m+1)}$ by placing sequence X vertically along the matrix *F* and *Y* horizontally along matrix F. We then initialize the top row and left column *F(i,0)=-id* respectively, where d is the gap penalty *i* is from *0 to n* and *j* is from 1 *to m.*

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap Penalty: d=-5

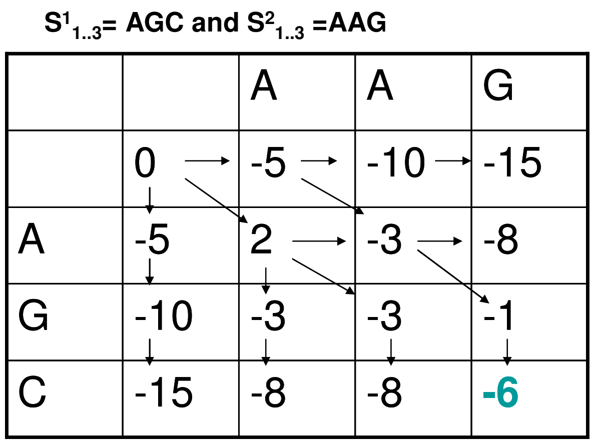
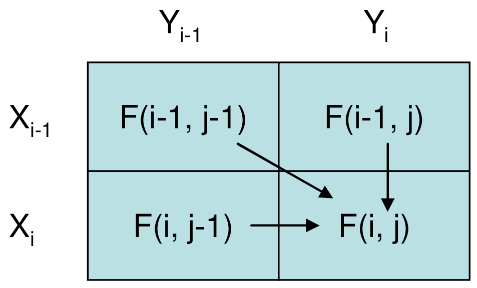


Figure 2.2 A Global Alignment Example (Initialization and Matrix-Fill, cited from [56])

Step 2. Matrix-Fill

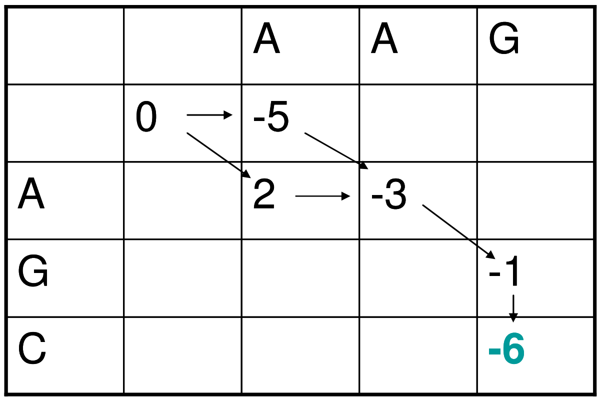
We fill in the matrix progressively from top left corner to bottom right corner. For each *F(i,j),* the value is computed by taking the best move from its three neighbor cells *F(i-1,j-1), F(i-1,j)* and *F(i,j-1).*



$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(a_i, b_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

Figure 2.3 Matrix Cell Calculation

Moving horizontally, *F(i,j)=F(i,j-1)-d* means "Y_j is aligned with a gap"; moving vertically, *F(i,j)=F(i-1,j)-d* means "X_i is aligned with a gap"; and moving diagonally, *F(i,j)=F(i-1,j-1)* means "X_i and Y_j are aligned together". When the matrix is complete, the value in the bottom right corner is the optimal score.



```
Optimal      AAG-   or   AAG-
Alignment:  -AGC         A-GC
```

Figure 2.4 Trace Back Example (cited from [56])

Step 3. Trace-Back

To reconstruct the paths through the matrix that yield the optimal score, we simply need to follow the matrix from the bottom-right cell, all the way back to the start. At each step, we move back from the current cell *(i,j)* to the one of the cells *(i-1,j-1), (i-1,j)* or *(i, j-1)* from which *F(i,j)* is derived. At the same time, we add a pair of symbols onto the front of the alignments: (X_i, Y_j) if the step is to *(i-1,j-1)*, X_i and the gap character '-' if the step is to *(i-1,j)*, or '-" and 'Y_j' if the step is to *(i,j-1)*. At the end, we can reach the start of the matrix, i=j=0. During the trace-back procedure, if at any point two derivations are equal, an arbitrary choice is made between equal options.

The dynamic programming algorithm takes O(nm) time and O(nm) space for aligning two sequences with the lengths of n and m.

Local Alignment

Smith-Waterman [54] proposed a dynamic programming algorithm for local pairwise alignment. It is very similar to global alignment except for two differences

[2,54]. First, for each cell in the matrix, an extra possibility "0" is added to the *F(i,j)* calculation.

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(a_i, b_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \\ 0 \end{cases} \qquad (2.8)$$

The "0" option is added for the following reason: if the best alignment up to some point has a negative score, it is better to start a new one. For the same reason, the top row and the left column are now filled with 0s, instead of *–jd* or *–id*.

The second change is that an alignment can now end anywhere in the matrix, so instead of trace-back from the bottom right *F(n,m)*, we can look for the highest value *F(i,j)* of the matrix *F* and start the trace-back from there. The trace-back ends when we meet a cell with a value of 0.

2.3.2 Multiple Sequence Alignment via Dynamic Programming

It is straightforward to generalize pairwise dynamic programming alignment to the alignment of *N* sequences [2]. Let $\alpha_{i1,\, i2,\, \ldots, iN}$ be the maximum score of an alignment up to the subsequences ending with $X^1_{i1},\, X^2_{i2},\, \ldots, X^N_{iN}$. The recurrence for dynamic programming algorithm is

$$\alpha_{i_1 i_2 \ldots i_N} = \max \begin{cases} \alpha_{i_1-1, i_2-1, \ldots i_N-1} + S(x^1_{i_1}, x^2_{i_2}, \ldots, x^N_{i_N}), \\ \alpha_{i_1, i_2-1, \ldots i_N-1} + S(-, x^2_{i_2}, \ldots, x^N_{i_N}), \\ \alpha_{i_1-1, i_2, i_3-1 \ldots i_N-1} + S(x^1_{i_1}, -, \ldots, x^N_{i_N}), \\ \vdots \\ \alpha_{i_1-1, i_2-1, \ldots i_N} + S(x^1_{i_1}, x^2_{i_2}, \ldots, -), \\ \alpha_{i_1, i_2, \ldots i_N-1} + S(-, -, \ldots, x^N_{i_N}), \\ \vdots \\ \alpha_{i_1, i_2-1, \ldots i_{N-1}-1, i_N} + S(-, x^2_{i_2}, \ldots, -), \\ \vdots \end{cases} \tag{2.9}$$

In the above formula, all combinations of gaps appear except the one where all residues are replaced by gaps. There are 2^N-1 such combinations. Initialization, termination, and trace-back steps for the algorithm follow analogously from the pair-wise dynamic programming algorithm.

2.2.3 Algorithm Complexity Analysis

The algorithm requires the computation of the whole dynamic programming matrix with $L_1L_2..L_N$ entries, where L_i is the length of sequence X^i. To calculate each entry we need to maximize over all 2^N-1 combinations of gaps in a column. Assuming the sequences are of roughly the same length L, the memory complexity of the multi-dimensional dynamic programming algorithm is of the order $O(L^N)$ and the time complexity is of order $O(2^NL^N)$. Because the time and space complexity of MSA grows exponentially with the number of sequences to be aligned, the MSA DP method can only align less than 10 sequences and is not feasible in practice.

2.3 HEURISTIC MSA ALGORITHMS

2.3.1 Local Heuristic MSA Algorithm

The combinatorial complexity of local MSA makes rigorous solutions impractical for large applications. All of the existing popular local MSA methods are heuristic-based. They can be classified into three classes: progressive, block-based and probabilistic-based methods. Every method has its own advantages and disadvantages. In the following sections, I will give detailed reviews for each method. And I will mainly concentrate on reviewing the most commonly used or new local MSA methods.

Probability-based Method

Probability-based methods aim to identify a set of un-gapped and short conserved regions. It uses a probabilistic matrix to represent conserved regions, and evaluate its alignment quality by its likelihood. It starts from an initial guess – a random alignment, then iteratively realigns the sequences, and converges toward the set of conserved regions, whose probabilistic matrices have maximum likelihood. Its implementation depends on how optimization strategy is used. Expectation maximization (EM) [41,42] and Gibbs sampling [37,38,39] are two representative optimization strategies.

1) Multiple EM for the Motif Elicitation (MEME) System

Bailey and Elkan [41,42] implemented the EM algorithm to identify locally conserved regions of sequences (A conserved region is called "motif" here.) It divides sequences into k-long words $X_1, \ldots, X_n$, and treats them as a mixture of two models, the motif and background (non-motif) models. The goal is to identify the likeliest background and motif models and classify each word into either motif or background based on the models.

In the EM algorithm, a motif is represented by a matrix *M*, defining the residue frequencies in each position of the motif. A "background" model is defined as a vector *B*, defining the background frequencies. To build the overall system, two other parameters are also defined: a mixed parameter λ (a constant) and a matrix *Z*, defining where the motif starts in each sequence. The objective of EM is to find values for two models $\theta=(B,M)$ and λ by maximizing the likelihood $P(x_1, x_2,..,x_n, Z/\theta,\lambda)$. This can be realized by an iterative process of expectation-step (E-step) and maximization-step (M-step):

1. Make an initial guess as to the motif, including its location and the expected length of the motif. Derive B and M based on the guessed motif.

2. E-step: calculate expected value Z based on B and M.

3. M-step: Using Z to re-compute the background model B and motif model M to maximize defined likelihood, $P(x_1, x_2,.....,x_n, Z/\theta,\lambda)$.

4. If convergence conditions are satisfied, go to next step, otherwise go back to E-step (2).

5. Return *Z* and *M*

The EM algorithm is a hill-climbing algorithm, in which optimization continually improves the realignment. The EM results strongly depend on initial condition (initial guess), and easily get stuck into local optimum. The EM algorithm is also time-consuming, and requires the user to specify the length of the conserved region in advance.

2) Gibbs Sampler

Gibbs sampling [37, 38, 39] is a stochastic variant of EM. It realigns the sequences, but not always for the better: it realigns the sequences stochastically (according to the probability of the realignment quality). The idea in Gibbs sampling is to determine the motif(s) by sliding them back and forth until the ratio of the motif probability to the background probability is at maximum. The algorithm is as follows:

1. Make an initial guess as to the motif, including its length L and random starting positions of the motif within all of the sequences except the one left-out sequence.
2. Calculate the residue frequencies in each position of the motif, and the background frequencies based on the motif. This is similar to the EM algorithm.
3. For the left-out sequence, calculate probabilities for all its sub-sequences of length *L*. This is realized by assigning a probability P/Q to each of its subsequences with P = probability of generating this subsequence, and Q = background probabilities.
4. Draw a random sample of the sub-sequences with the probabilities calculated as above.
5. If convergence is reached, stop. Otherwise, choose another left-out sequence and go back to step 2.

The chosen sample of the Gibbs sampler is based on the probability rather than taking the maximum probability like EM. This makes Gibbs sampling less dependent on initial parameters than EM. However, it is still a hill-climbing method and not able to detect conserved regions with a very bad initial guess. It is also time-consuming, and needs the user to specify the length of the conserved region and the number of expected conserved regions before running the algorithm.

Progressive-base Method

Progressive-based method is a greedy heuristic method. It is a very fast and efficient algorithm and gives reasonably accurate results in practice. Its shortcomings are

(1) the errors made in early stages are propagated and accumulated into later stages. (2) It doesn't globally optimize any objective function.

Dialign 2 [34, 35] is the most recognized method of progressive local MSA. Dialign2 is fragment-based and uses consistency information to guide alignment. It aligns each pair of sequences and obtains their optimal matching fragments, called "diagonals". It then re-weights each diagonal based on its consistency information: given two diagonals, if each of them have one and only one fragment from the same sequence and the fragments on the same sequence are overlapped, each diagonal is re-weighted by an added overlapping weight. After all the diagonals are re-weighted, diagonals are sorted according to their weights. The diagonal with the highest score is selected first for the alignment. Then the next diagonal from the sorted weight list is added to the alignment if consistent. The process is repeated until all the diagonals have been processed.

Besides the inherent limitations of progressive methods, Dialign 2 has another limitation: it needs a better consistency calculation scheme. In Dialign 2, it is assumed that the more overlapping diagonals a diagonal has, the more alike it is in a conserved region. Given a diagonal, some of its overlapping diagonals may be from the same conserved region with it, and some are not. If a diagonal has a lot of overlapping diagonals not from the same conserved region, and the diagonal is still re-weighted with them, this diagonal's weight will contain lots of noise and misguide the alignment. Hence a diagonal should be re-weighted only by the diagonals that are possible from its same conserved region. However, Dialign 2 is pairwise-alignment-based and has no ability to determine if two diagonals are possible from the same conserved region. So a better consistency scheme is needed.

Block-Based Method

Block-based method has two steps. It first uses a heuristic method to generate a set of blocks, corresponding to possible conserved regions. It then chains the constructed blocks, finds an optimal set of blocks with the maximum sum of similarity scores, and returns them as the identified conserved regions. The block-based method is a global optimization approach: given a set of blocks, the chaining method optimizes the Sum-of-Pairs objective function to find a globally optimal solution. Many block-based methods have been proposed and have failed to be used in practice, because their searching of blocks was very time-consuming. So the key issue in the block-based method is how to efficiently construct the best possible set of block candidates. Currently Block-Maker is the most widely used block-based method.

Block-Maker [40] was used to construct the well-known BLOCK database, from which BLOSUM substitution matrices are derived. Block-maker uses two methods to construct blocks for chaining. One is based on space-triplet and the other is based on Gibbs sampling. "Space-triplet method exhaustively checks all spaced triplets out to a maximum distance for their presence in at least a subset of sequences" [40], it then picks 50 top-scoring constructed blocks for chaining. The space-triplet method has an inherent limitation: it assumes all the blocks should have a three-triplet pattern. This is an overly strong assumption, causing blocks without a three-triplet pattern to be missed. The strategy of picking 50 top-scoring blocks is also too rough. This may easily cut off some blocks that are in the optimal solution. The Gibbs sampling approach constructs blocks iteratively. Different block lengths are used in each iteration to create different-length blocks. Since Gibbs sampling is very time-consuming, only a limited number of block lengths can be tried in the iteration processes. This will miss some blocks with some other lengths. The Gibbs sampling method may be stuck into a local optimum with a bad initial alignment guess. This method also always creates at least one block, even when the sequences don't share a conserved region. Block-maker also has some problems in chaining: no gap function is used. In block chaining, it is expected to incorporate spacing into its objective function to make sure the spaces in each row should not be wildly different from the spaces in other rows, otherwise it is considered suspect in biological applications. In order to achieve this, a gap function should be added to the objective function of chaining. This is very important when the aligned sequences have a wide range of lengths. So based on the above analysis, a better block-based method is needed.

2.3.2 Global MSA Algorithm Review

Numerous global MSA methods have been developed. Global MSA algorithms are much more mature than local MSA algorithms. Since most global MSA techniques can be applied to local MSA with small changes, here we are going to review the most commonly used or new global MSA algorithms. We can roughly divide them into three categories: exact, progressive and iterative algorithms. An exact algorithm aligns the sequences simultaneously to do the alignment. It is very useful when analyzing sets of sequences that are very divergent and whose pair-wise alignments are likely to be incorrect. The progressive algorithm tries to align the sequences by adding one at a time. The key idea is only two sequences or profiles or sequence/profiles are aligned at one time. This is the most space and time efficient technique. An iterative algorithm first produces an initial alignment, and refines it through a series of iteration. In the following sections, I will give detailed reviews for each method. And I will mainly concentrate on reviewing the most commonly used or new global MSA methods.

2.3.2.1 Speed-Up Strategies For The Exact Algorithm

Because the time and space complexity of MSA grows exponentially with the number of sequences to be aligned, it is not feasible in practice. Given this problem, speedup strategies are needed to reduce the space and time complexity of the DP method. Over the years, different advanced methods have been proposed. Here I classify them into three categories: “Carrillo-Lipman” algorithm, “Divide-And-Conquer” algorithm and the “A* algorithm” and review them respectively.

Carrillo-Lipman Algorithm

The main idea in the Carrillo-Lipman algorithm is that not every node in the dynamic programming matrix needs to be visited; many of them can be pruned by using the “Carrillo-Lipman bound” [58].

Carrillo-Lipman Bound: Let A* be an optimal alignment of the K strings $S_1,..,S_k$, L be the alignment lower bound and U be the alignment upper bound. Then

$$c(A^*_{i,j})<=c(A^*(S_i,s_j))+U-L, \qquad (2.10)$$

where $c(A^*_{i,i})$ is the cost of the projection of A* to sequences S_i and $S_{j;}$ $c(A^*(s_i,s_i))$ is the cost of the optimal alignment of Si and Sj.

An optimal alignment path cannot pass node r if for any pair i,j holds:

$$c(A^*_{i,j}) -c(A^*(S_i,S_j)) +L >U \qquad (2.11)$$

Let $CL_{i,j}(r)= c(A^*_{i,j}) -c(A^*(S_i,S_j)) +L$ and call a node a CL-valid node if $CL_{i,j}(r)< =U$ for all pairs i,j. This Carrillo-Lipman bound can help to restrict the alignments to CL-valid nodes.

In 1989, the Carrillo-Lipman bound was implemented in the Multiple Sequence Alignment program (MSA) by Lipman etc. [59]. The upper bound is estimated by fast heuristic multiple sequence alignment, and the lower bound is the sum of the costs of all optimal pairwise alignments. In the Lipman’ program, the set of CL-valid nodes are first pre-computed; a dynamic programming (DP) algorithm is then applied to the set of CL valid nodes. In addition to the Carrillo-Lipman pruning, another pruning – "return-cost pruning", is also applied during the DP calculation process. The return-cost pruning eliminates a node when the length of a shortest path from the source to the node, plus the lower bound on the length of a shortest path from the node to the sink, is greater than the upper bound U.

Actually it has been proved that return-cost pruning is stronger than Carrillo-Lipman pruning. The set of nodes obtained from return-cost pruning is always a subset of the set of nodes obtained from Carrillo-Lipman pruning. However the returning cost is space intensive. It was found that MSA runs faster without the return-cost pruning due to reducing the space usage. So it makes sense to combine the two pruning algorithms.

Lipman's MSA program is still extremely slow and memory intensive. It can only align up to five to seven sequences of reasonable length (200-300 residues).

DCA algorithm

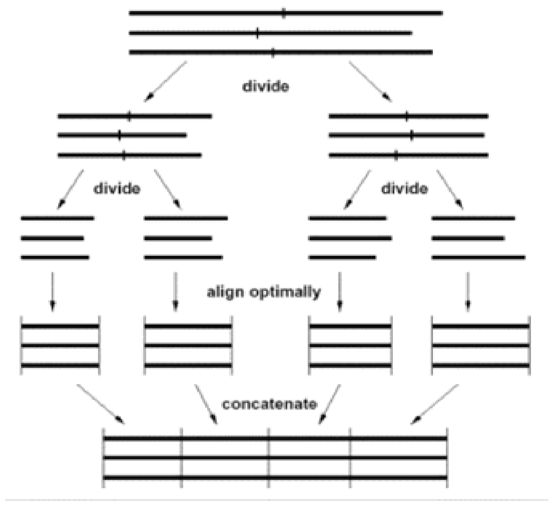


Figure 2.5 DCA Algorithm

In 1997, Stoye etc. [15] proposed a divide and conquer algorithm (DCA) that sits on Lipman's MSA to extend it. The main idea in DCA is to cut the sequences into subsets of segments that are small enough to be fed to MSA. A sketch of the DCA method is shown in Figure 2.5.

The sequences are cut at a certain position near to their centers. This divides the problem of aligning K sequences into two problems of aligning the K prefix and K suffix sequences. If the prefix or suffix sequences are still too long for the MSA program to align, the procedure is applied recursively until the sequences are of a length short enough. The resulting alignments are then concatenated. The choice of the cutting point for the DCA algorithm is realized by heuristic method.

The testing results show DCA can align long sequences that MSA cannot align. It is also a much faster algorithm than the original Lipman's MSA program. However, DCA only gives a modest improvement to MSA; it is still very slow and can only handle 20-30 sequences.

A* algorithm

Based on the above analysis, a more effective pruning algorithm was identified. That is the A* search algorithm was introduced to the MSA DP method.

A* algorithm is an admissible heuristic search algorithm [61]. A search algorithm is admissible if it never overestimates the cost of the optimal path from any node to the goal node. A* algorithm uses an evaluation function to decide which is the next best node to search. The evaluation function is defined as $f(n) = g(n) + h(n)$, where $g(n)$ is the actual cost of the current path from start to n, and $h(n)$ is the heuristic estimate of the cost from n to the goal. Thus $f(n)$ is the estimate of the total cost of the path from the start, through n to the goal and is used to guide the search.

In A* algorithm, there are two lists to be saved, the open list and the closed list. The open list keeps track of the current frontiers of the search and the states are ordered according their *f* values on the open list. The closed list keeps a record of nodes already visited. The A* algorithm is as follows:

```
Open:=[start]
    Closed:=[]
    While open!=[] do{
        Pop the highest ranked node from the list, call it n
        If n=goal, return the path from start to n
        Generate the children of n
        For each child of n do{
            If the child is not on open or closed {
                Calculate its f value and add the child to open;
            }

            Else If the child is already on open {
                if the child was reached by a shorter path,
                then give the state of the shorter path on open list
            }

            Else if the child is already on closed{
                If the child was reached   by a shorter path {
                Then remove the child from closed;
                Add the child to open;
            }
         }
    Put n on closed;
    Reorder states on open by f
    }
```

Figure 2.6 A* algorithm

In the A* algorithm, it is important to choose a good estimation function. A bad estimation can really slow down A* or make it produce bad paths. If you want A* to give you "perfect" paths, the heuristic function should be an underestimate of the actual cost of getting from one spot to the goal. A heuristic that produces such underestimates is said to be admissible.

1) A* algorithm for MSA

Lipman's MSA program is equivalent to a dynamic programming implementation of Dijkstra's shortest path algorithm. The A* algorithm for MSA is similar to Lipman's MSA program, but makes a revision to Dijkstra's algorithm: edge weight is redefined. The costs of all edges ($u \to v$) in the dynamic programming graph are redefined as:

$$C'(u \to v) := C(u \to v) - L(u) + L(v),$$

where L(u) is a lower bound for the cost of a shortest path from node *u* to the terminal/sink node t.

In Lipman's MSA program, the Dijkstra algorithm maintains a priority queue, which stores the current frontier nodes of the search. Each frontier node *n* has a value $C(s \to n)$, the short path from source s to node n. Every time the node with the lowest cost $C(s \to n)$ will be expanded.

In the A* algorithm's MSA, the open list corresponds to the priority queue. Each search frontier node *n* has a redefined value $C'(s \to n)$. Every time, the frontier node with the lowest cost $C'(s \to n)$ will be expanded. Here

$$\begin{aligned} C'(s \to n) &:= C(s \to n) - L(s \to t) + L(n \to t). \\ &:= f(n) - L(s->t), \qquad (2.12) \end{aligned}$$

where $f(n) = C(s \to n) + L(n \to t)$. Since L(s->t) is a constant, which does n ot affect the ranking, we can ignore it. It can be seen that we actually rank all the nodes by A* evaluation function, f(n)=c(s->n)+L(n->t). By using the A* heuristic function, the search is guided towards the goal effectively and the search space can be reduced considerably. In contrast, the Dijkstra algorithm has no such information, and searches in all directions.

In 1999, Lermen and Reinert [62] implemented the above A* algorithm for exact multiple sequence alignments and compared it with Lipman's MSA. Results show A* algorithm considerably speeds up the computation time. However, the space usage is not reduced. This is reasonable because A* algorithm has a big overhead: it needs to maintain large open/closed lists and do the heuristic estimation each time.

In order to improve the A* algorithm performance, variations of A* algorithm have been proposed. The most effective approaches are “Partial Expansion A*” and “Memory-efficient A* Heuristic Function”.

2) Partial Expansion A*

A partial expansion algorithm is used to reduce the size of open and closed lists. It was proposed by Ishida in 2000 [63]. Instead of inserting all child nodes into the open list (an expensive operation), this algorithm only generates the most promising nodes. If a node has unpromising children, it is reinserted back into the open list with the value of the best unpromising child node. Since nothing is lost (the node can be re-expanded again later), correctness is maintained. Since unpromising nodes are not generated, the memory they would otherwise require is saved.

3) Memory-Efficient Heuristic Function

A better heuristic function can result in a substantial reduction to the size of A* open list. In 2002, McNaughton etc. proposed a memory-efficient A* heuristic function [64]. The heuristic function is estimated by including a three-way alignment. An octree is used to realize the three-way alignment. It is a tree structure, representing a 3D alignment using rectangle blocks. Each node in the octree corresponds to a rectangle block and the node of the octree is only calculated on demand. The three-way alignment produces a better heuristic function than pairwise alignment. The better heuristic function, in turn, greatly reduces the size of the A* open list.

4) OMA: A* algorithm for DCA

In 2000, Reinert, Stoye and Will integrated the A* algorithm with DCA [65]. The subsets of segments cut by DCA are fed to an MSA program. The MSA program then uses an A* algorithm to do the multiple alignment.

Since the DCA cutting algorithm is heuristic, it will probably result in errors in some cases. An iterative method is used to improve the result. In the iterative method, the DCA-plus-A* algorithm is applied repeatedly, with different sub-segment dividing lengths.

After comparing OMA with MSA and DCA, it is shown that OMA consistently gave the best performance. The disadvantage for OMA is its slower speed because of its iterative procedures.

2.3.2.2 Progressive Sequence Alignment

Progressive sequence alignment is a greedy heuristic method by giving up optimizing the SP score. It builds a multiple sequence alignment progressively by a series of pair-wise alignments, following the branching order in a phylogenetic tree. It is a very fast and space-efficient algorithm and gives reasonably accurate results in practice. The shortcomings for this approach are: (1) early error propagation: once a sequence has been aligned, that alignment will be “frozen” even if it conflicts with the sequences added later in the process, and (2) its accuracy depends on alignment parameters: weight matrix and gap penalty [2].

There exist a number of progressive sequence alignment programs. Among them, ClustalW is the one widely used to do progressive multiple alignment.

<u>ClustalW</u>

ClustalW [14] is a profile-based progressive multiple alignment. In the early days, multiple aligned blocks of sequences used to be represented by a single consensus sequence. However, most present day methods use profiles to represent sequence blocks. A profile has the advantage of using position-specific and consistency information from the group's multiple alignment. The ClustalW algorithm is as follows:

1. Construct a distance matrix of all N(N-1)/2 pairs of sequences by pairwise sequence alignment. Then convert the similarity scores to evolutionary distances using a specific model of evolution proposed by Kimura in 1983 [66].
2. Construct a guide-tree from this matrix using a clustering method called neighbor-joining proposed by Saitou and Nei in 1987 [67].
3. Progressively align nodes of the tree in order of decreasing similarity using sequences vs. sequences, sequences vs. profile and profile vs. profile alignments.

Various heuristics have been added to get good accuracy. These include: (i) local gap penalties, (ii) automatic selection of the amino acid substitution matrix, (iii) automatic gap penalty adjustment, and (iv) a mechanism to delay alignment of sequences that appear to be distant at the time they are considered.

ClustalW tunes alignment parameters carefully to improve alignment performance. But it still doesn't solve the primary shortcoming of progressive alignment, early error propagation. Recently, different strategies have been proposed to solve the early error problem. They are: profile-preprocessing, induced secondary structure, globalised local alignment and matrix extension.

1) Profile Preprocessing

Profile preprocessing was proposed by Jap Heringa in 1999 [19]. The main idea in profile processing is to construct a profile for each sequence and then use it to replace the individual sequence for alignment.

Profile pre-processing first pair-wise aligns *n* initial sequences, it then finds its close relatives for each sequence. The close relatives of a sequence *n* are the sequences whose pair-wise alignment score with *n* is higher than a specified cutoff value. A profile is created for each sequence by including itself and its close relatives. After the sequence profiles constructed, each sequence can now be replaced with its profile and used for progressive alignment.

Each profile contains the consistency information of the sequence with the other sequences. When we use it to do the alignment, consistency information has been effectively integrated into the alignment process and guides the alignment. In this way, it can effectively reduce early error.

2) Secondary Structure-Induced Alignment

Knowledge about secondary structure is an important aid for multiple sequence alignment since secondary structure elements of related proteins often correspond structurally and can be superposed by a structural comparison technique [20]. By using the secondary structure information to guide progressive alignment, early error can also be reduced.

PRALIGN program implements this technique [20]. It uses SSPRED to predict secondary sequence structure. At the beginning, an initial alignment is generated without guidance from the secondary structure. The SSPRED then predicts the secondary structure for the obtained multiple sequence alignment. The obtained secondary structure in turn guides the multiple sequence alignment. This process is iterated until converge is reached.

With the secondary structure's aid, the alignment errors in the early stages can be greatly reduced and this is shown in [19,20].

3) Globalised Local Alignment

The main idea is to guide the global alignment towards matching the local motif by integrating the local alignment information into the global alignment process. Local alignment information is very valuable since it usually contains motif and consistency information.

For each pair of sequences, local alignments are performed in forward and backward directions of the sequences [20]. Then the values of the resulting two DP search matrices are added for each cell with subtraction of the local score $s[a_i,b_j]$ to avoid the double counting of the local substitution value. After this procedure, each cell in the resulting matrix will have the score of its best corresponding local alignments.

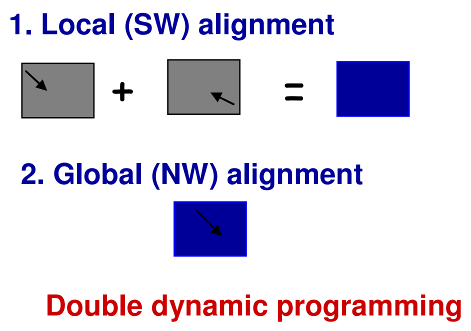


Figure 2.7. Globalised Local Alignment

The thus obtained local aligned scores are then used for a second round of DP, to find the global alignment. A number of methods can be proposed to apply local alignment scores to global scores. For example, convert the local alignment scores to logarithmic values and then add their normalized weights to the residue substitution score corresponding to the search matrix cell.

This method has been proven to be very effective in aligning sequences with local similarity [21]. The above three methods can be used in combination.

4) Consistency-based Approach

The main idea in this approach is to replace the traditional substitution matrix with a consistency-based and position-specific matrix.

Notredame developed the new substitution matrix in 2000 and implemented it in the T-Coffee MSA program [17]. The substitution score is calculated as follows (see Figure 2.8):

1. For each sequence pair, a single global alignment and 10 top-scoring non-intersecting local alignments are generated, respectively by the programs ClustalW [14] and SIM [19].
2. Assign a weight, "sequence identity", to each pair of aligned residues
3. The global and local alignment scores are added. If any pair is duplicated between local and global alignment, it is merged into a single entry that has a weight equal to the sum of the two weights. Otherwise, a new entry is created for the pair being considered. This results in a library of weights for each non-redundant residue pair.
4. The information in the library is then further enhanced by library extension. This is done using a triplet approach aimed at calculating the contribution of third sequences *C* onto the direct alignment of sequence *A* and *B*. It is based on the notion that a triplet alignment *A*–*C*–*B* effectively provides an alternative alignment of *A* and *B*. Each extended score *W'* is calculated as $W'(A(x), B(y))=W(A(x), B(y))+\sum_{L\neq A,B} \mathrm{Min}(W(A(x), C(z)), W(C(z), B(y)))$, where *x*, *y* and *z* are sequence positions in sequences *A*, *B* and the intermediate sequence *C*, respectively, and summation is done over all third sequences *C* other than *A* or *B*.
5. Weights will be zero for all residue pairs that never occur.

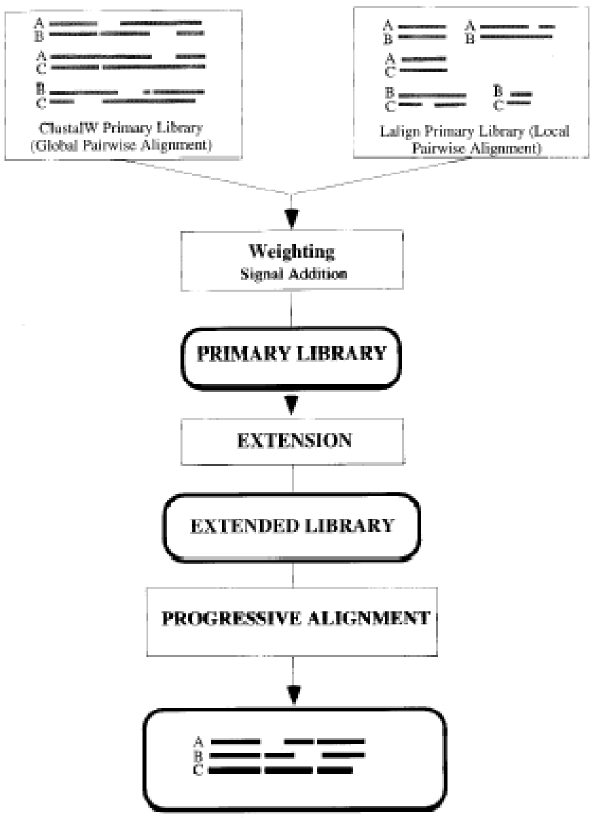

ClustalW Primary Library (Global Pairwise Alignment)
Lalign Primary Library (Local Pairwise Alignment)
Weighting
Signal Addition
PRIMARY LIBRARY
EXTENSION
EXTENDED LIBRARY
PROGRESSIVE ALIGNMENT

Figure 2.8. T-Coffee Strategy

The extended library weights *W'* can be used as a new substitution matrix for progressive alignment. The new substitution matrix combines local and global alignment info. It also contains the consistency info of each pair of residues with other pairs. Test results show that T-Coffee generates much improved alignments as compared to ClustalW.

2.3.2.3 ITERATIVE METHOD

The idea in iterative alignment is to find the optimal alignment by refining the alignment iteratively. It was first proposed by Barton in 1987 [60]. His algorithm is as follows: an initial alignment is generated, then one sequence is taken out and realigned to the remaining sequences; the process is repeated until all sequences have been realigned.

Over the years, different and better iterative methods have been proposed. They can be classified into two categories: stochastic iterative algorithm, and non-stochastic algorithm. Here I will review some representative algorithms in each category.

Stochastic Iterative Algorithm

Stochastic algorithm is a class of algorithm that involves non-deterministic steps. It has an inherent random nature, which makes it very appealing in aligning multiple sequences. In multiple sequence alignment, differences among the sequences arise from mutations during evolution, which we can regard as random processes. Stochastic MSA algorithms attempt to model such behaviour to guide the alignment of the sequences. In Stochastic MSA algorithms, Simulated Annealing (SA) and genetic algorithm (GA) are the representative methods. SA for MSA runs very slow and makes it not feasible in practise. Compared with SA, GA for MSA is much faster.

The genetic algorithm [68] is a stochastic method based on Darwin's evolution theory. It works on chromosome-like data, which encode possible solutions of the problems, and applies crossover and mutation operators to generate new chromosomes. Based on the principle of survival-of-the-fittest, chromosomes with good performance are selected through a selection operator. The performance of chromosomes is evaluated by a fitness function.

At the beginning, GA randomly creates *n* solutions. GA then applies selection, crossover, and mutation operators to create *n* new solutions. The *n* old solution and *n* new solutions compete for survival to the next generation, in which only the n best solutions will be included. This process is repeated until an optimal solution is found.

SAGA is a MSA program, which implements the classic GA algorithm [16]. It defines 19 extra operators for alignment specific requirements, such as gap insertion and block shuffling. The fitness function was originally SP-score. In 1998, a new fitness function, “coffee objective function”, was applied. Coffee objective function calculates the correlation between a multiple sequence alignment and a previously defined library of pair-wise alignments. It greatly improves SAGA’s performance.

SAGA cannot do well in all cases, but it always gives a good estimation of the accuracy of the multiple alignment when high quality pair-wise alignments, such as 3D structural superpositions are available.

The advantages for SAGA are: (1) simulate the evolution processes from a biological perspective; (2) the fitness function is independent from other operators, easily upgraded and extended; and (3) by considering multiple solutions simultaneously, consistency information is integrated and used as a guide toward better solutions. The disadvantage of GA is its running speed. Although it gives promising results, it runs very slowly for more than 20 sequences. Many AI researchers have been working on this to improve its running speed.

Non-stochastic Methods

Of the non-stochastic methods, I will review the most popular method - IterAlign. IterAlign [69] uses segment-to-segment comparison to find local pairwise alignments. However segments pair pairs are not directly included in a multiple sequence alignment, they are used to construct consensus sequence instead. The IterAlign algorithm is as follows:

a. Align each sequence *r* with the others
b. Find the consensus sequence r^c for each sequence r. The consensus sequence is found via iteration process:

1) From the alignment segments of r with the others, an “ameliorated” $r_a^{(1)}$ sequence is extracted, which reflects the aggregate similarities of r with the other sequences.

2) Align the “ameliorated” sequence $r_a^{(1)}$ with the other sequences except r and extract a new “ameliorated” sequence $r_a^{(2)}$

3) Repeat this process, until “ameliorated” sequence converges

c. Replace all the sequences with its consensus sequences

d. Repeat (a) - (b), until the set of consensus sequences converge

Find the core blocks from consensus sequences and chain them to obtain local alignments.

The advantages for IterAlign are: (1) the program performs alignment in a symmetric fashion, calculating the consensus for each sequence. This makes the alignment not greatly affected by its initial alignment; and (2) consensus sequence could effectively guide the alignment toward conserved regions and make it good for distantly related sequences. The disadvantage of IterAlign is: large computational time and space usage, caused by the process of finding the consensus sequences.

Chapter 3 A Three-Way-Consistency-Based Local MSA

3.1. INTRODUCTION

As the amount of sequence data increases, local multiple sequence alignment (MSA) is becoming an increasingly important tool in computational biology. However, local MSA still remains an open problem for protein sequences. Protein sequences are modular and highly variable. The common presence of large N/C terminals or insertion makes its alignment even harder. Although the existing local MSA tools perform well in finding blocks (or motifs) with strong conservation, they often fail to detect the conserved regions in those difficult cases. Hence additional algorithms are needed.

Based on these, we developed a new consistency-based local MSA. It first uses the alignment-consistency of three-way alignments to represent sequence conservation information, then directly incorporated alignment consistency information into block-finding process. The major features of this new approach are: (1) A new three-way-alignment-based method for the accurate extraction of alignment consistency. Typically, the algorithmic cost of pairwise consistency is $O(N^3L^2)$ (e.g. T-Coffee) or $O(N^4L^2)$ (e.g. Dialign 2), where N is the number of sequences to be aligned and L is the average length of the sequences. While CBMSA is still able to maintain the time complexity $O(N^3L^2)$, it provides better alignment quality, and (2) Alignment-consistency-based fast block construction across multiple sequences. Our proposed consistency-based block construction is an intermediate between BlockMSA and Dialign 2. Instead of constructing blocks via a statistic-based or pattern-based method, it uses the inherent conservation information, alignment-consistency, to construct blocks. No pattern needs to be assumed. There is no iteration process and nor risk of combinatorial enumerations for block constructions. So multiple blocks can be constructed very efficiently. Compared with Dialign 2, CBMSA constructs blocks across multiple sequences, while Dialign 2 only looks for blocks shared by two sequences only.

We compared the performance of three-way-consistency-based MSA and pair-wise-consistency-based MSA. We found the proposed three-way-consistency approach was able to provide more reliable alignment consistency information and eventually gave higher alignment accuracy.

We evaluated our program by applying it to discover subtle motifs in classical motif-finding test sets and Prints protein families, and compared our program with other leading protein local MSA programs. Our program consistently gave a favorable performance.

3.2. ALGORITHM

Our algorithm consists of three main steps: generation of alignment consistency model, construction of block candidates, and block assembling. We iteratively apply the above steps to the remaining unaligned regions until they are too small or no more conserved regions could be found.

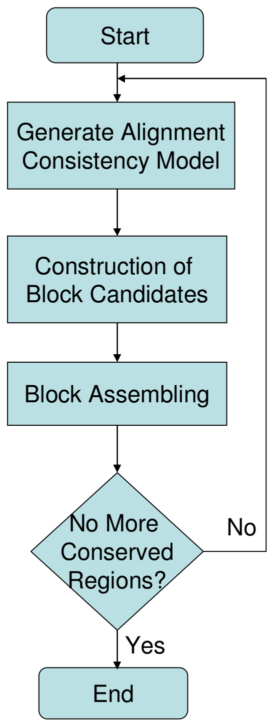

Start
Generate Alignment Consistency Model
Construction of Block Candidates
Block Assembling
No More Conserved Regions?
No
Yes
End

Figure 3.1 Algorithm Flowchart

We first provide key definitions and then expand on each of the three steps.

Definition 1. A k-block (conserved region) represents an un-gapped alignment region in *k* sequences, consisting of equal-length fragments from each of *k* sequences respectively, $B=\{f_1, f_2,\ldots, f_k\}$, where f_i represents a fragment from sequence *i*.

Definition 2. A 2-block is a special case of *k*-block with *k=2*, which represents an un-gapped alignment region in a pairwise alignment.

Definition 3. Block Similarity Score. Given a k-block $B=\{ f_1, f_2,\ldots ,f_k\}$, the score of *B* is the sum of all the similarity scores of the $\binom{k}{2}$ pairs of fragments(2-blocks) within B:

$$S(B) = \sum_{1\le i<j\le k} S(f_i, f_j) \tag{3.1}$$

where $S(f_i, f_j)$ is the similarity score of fragments f_i and f_j.

Definition 4. Fragment Order. Given two fragments a and b on a sequence, $a=(a_1,...,a_{k1})$ and $b=(b_1,...,b_{k2})$, where a_i and b_i represent two residues from the same sequence, a is said to be less than b (written as $a<b$) if and only if the ending position of a is less than the beginning position of b.

Definition 5. Block Order. Given two k-blocks on k sequences, $F=(f_1, f_2,..,f_k)$ and $G=(g_1,g_2,...,g_k)$, where f_i and g_i represent two fragments from the same sequence i, Block F is less than Block G (written as $F < G$) if and only if $\forall i \in [1, k], \quad f_i<g_i.$

Definition 6. Non-overlapping Blocks. Given two k-blocks F and G on k sequences, they are non-overlapping if and only if F is less than G or G is less than F.

Definition 7. Chain. A set of k-blocks $B=\{b_1,b_2,..,b_n\}$ on k sequences is called a chain if all the blocks in B are non-overlapped. The score of a chain is the sum of the scores of all its blocks minus the gap costs between them.

$$Score_{chain}(B) = [\sum_{1\le i<n} S(b_i) - Gap(b_i, b_{i+1})] + S(b_n) \tag{3.2}$$

Definition 8. An Optimal Set of Blocks. Given a set of blocks B, an optimal set of blocks is the chain with maximum score over all the chains of B.

Problem Definition: Given k input sequences, our goal is to identify an optimal set of k-blocks in the given sequences.

3.2.1 Generation of Alignment Consistency Model (Step 1)

In order to derive the alignment-consistency model, we first generate a library of lower-order alignments of all the sequences. The choice of the order k of lower order alignments is flexible. The higher the value of k used, the more accurate information the library represents. If the order k of the lower order alignments in the library is chosen as two, the library consists of all the possible pair-wise alignments of the given sequences. Similarly, if the order k is three, the library consists of all the possible three-way-alignments. After obtaining the library of lower order alignments, we derive the alignment consistency model by representing them as *2*-blocks.

We have tested the cases for the order *k*=2 (pairwise alignment) and *k*=3 (three-way-alignment) respectively. The three-way-alignment-based consistency model provides more accurate information than the pairwise-alignment-based model. At the same time, it is also able to maintain the same time complexity as a pairwise-alignment-based approach.

3.2.1.1 Pairwise-Alignment-Based Consistency Model (k =2)

When the alignment order *k* is 2, the lower order alignments consist of a set of pairwise alignments. For each pairwise alignment, a window of length *w* is slid through it and each position of the window produces a *w*-length sub-region. We only keep those sub-regions that are *2*-block (ungapped) with a similarity greater than a pre-specified threshold. In this way, we can convert all the pairwise alignments into *2*-blocks.

Since all the *2*-blocks are not equally important (some may come from a real conserved region, and some may occur only by chance), we calculate the similarity score for each *2*-block. We encourage those 2-blocks that share transitive alignment consistency with the other *2*-blocks. Here we use the same consistency concept as in Dialign 2.

Definition 9. Transitive Alignment Consistency. Given two 2-blocks $B_1(a,b)$ and $B_2(b,c)$, if they share a common fragment *b*, we define them as overlapped 2-blocks with transitive alignment consistency.

Definition 10. Alignment Consistency Weight. Given a block $B_1(a,b)$, if it is overlapped with another block $B_2(b,c)$, then its alignment consistency weight with B_2 is the similarity score $S(a,c)$ of their non-overlapped fragments *a* and *c*. If $s(a,c)$ is less than 0, it is set as zero.

For each 2-block in the library, we will check its alignment consistency with all the other 2-blocks and add the accumulated weight to its weight. The more a *2*-block shares transitive alignment consistency with the other 2-blocks, the higher its weight is, which means that it is more likely in a conserved region.

3.2.1.2 Three-Way-Alignment-Based Consistency Model (k =3)

When the alignment order *k* is three, the lower order alignment library consists of a set of three-sequence alignments. We first still use a *w*-length window to slide through each three-sequence-alignment and obtain a set of three-fragment regions. The difference is, in order to obtain *2*-blocks, we project each three-fragment-region into two-fragment regions and only keep those regions that are un-gapped, “2-blocks”.

When the alignment order *k* in the library is 3, the alignment consistency between 2-blocks is much richer than pairwise alignments and easy to extract.

First, the high-order alignments directly contain the transitive-alignment-consistency within each 3-sequence alignment itself. For example, given a 2-block *F* within a three-sequence-alignment, we can directly determine *F*’s overlapping 2-block by checking its vertically-corresponding *2*-block *G* in the alignment, which has the same start and end positions with *F* in the alignment. Three-sequence-alignment is a much faster approach to extract the transitive alignment consistency information. It avoids comparing each 2-block with all the other 2-blocks for alignment consistency extraction.

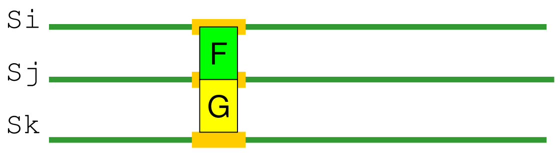


FIGURE 3.2 THREE-WAY CONSISTENCY EXAMPLE

Secondly, in the library of three-sequence alignments, a pair of sequences will appear in different 3-sequence alignments and may have different alignments. If a *2*-block on a pair of sequences appears multiple times in different 3-sequence alignments, we merge the same *2*-blocks generated from different alignments as a single entry and weight it with its similarity weight multiplied by the number of times it is in different three-way alignments. We call the repeated occurrence of a *2*-block in different three-way alignments a "stability consistency". One should note that the "stability consistency" is only unique in higher-order (k>2) alignments.

3.2.1.3 Time complexity Comparison of the Pairwise-Alignment-based Consistency Model and Three-Way-Alignment-Based Consistency Model.

In order to derive an alignment consistency model, two steps are needed: lower order alignments and alignment consistency extraction.

When we use the pairwise-alignment-based approach, the complexity of the lower order alignment is $O(N^2L^2)$ and the alignment consistency extraction is $O(N^3L^2)$, where N is the number of input sequences and L is the average sequence length. So the total time for the pairwise-alignment-based consistency model is $O(N^3L^2)$. For the three-sequence-alignment-based approach, the number of three-sequence alignments is $O(N^3L^2)$. However, the time for its alignment consistency extraction time is $O(N^3L)$. By balancing the tradeoff of lower order alignment and consistency weight calculations, three-sequence alignment consistency can achieve the same time complexity $O(N^3L^2)$ as a pairwise alignment-based approach. At the same time, it also provides more accurate consistency information.

3.2.1.4 Tools for Generating the Library of Lower Order Alignments

In principle, any existing MSA tool can be used here to generate the library of lower order alignments. However, in practice we have to consider the time (or space)-vs.-

quality tradeoff. Although simultaneous alignment gives the best alignment, it is memory-intensive and very slow. Based on this, we chose progressive MSA programs to do our lower order alignment here. We have used Partial Order Alignment (POA) and ClustalW to generate the lower order alignments respectively. Experimental results will be shown in a later section.

3.2.2. Construction of Blocks (Step 2)

After we generate our alignment consistency model, we can use it to construct block candidates. We transformed our block candidate construction to a greedy clustering-based method. We collect the fragments on all the *2*-blocks as a set of fragments *F*. We can think of each fragment as a point and an *n*-block as an *n*-point cluster. To determine an *n*-point cluster, we start the initial cluster with a seed point (fragment) and gradually add one fragment from each sequence to the cluster. Hence, an *n*-point cluster will consist of *n* fragments from *n* sequences and form an *n*-block. To add a fragment to the cluster, we pick the fragment which has the maximum sum of alignment consistency weight scores with the fragments in the cluster. By using the clustering method, we directly integrate alignment consistency information into block construction process. For seed fragments, we choose them from the pair of sequences with the maximum alignment consistency weights.

After we construct all the possible block candidates, we refine the block candidates, extend blocks and merge blocks.

3.2.3. Block Assembly (Step 3)

With all the possible candidate blocks, we want to obtain an optimal set of blocks. The identification of an optimal set of blocks is actually a classic chaining problem, and can be interpreted as a maximum weight path problem in a directed acyclic graph [1]. We have implemented the single source DAG shortest path algorithm to find the optimal chain. The time complexity is $O(n^2)$ time for a given set of n blocks.

3.2.4. Iteration

Since some regions may be missed out in the 1st round, we can iteratively apply this method to the remaining unaligned regions and identify more conserved regions. The process will stop when the unaligned regions are small enough or no more new blocks are found. In order to be time-efficient, we will not do the lower order alignments again. Instead, for each unaligned region, we will find its corresponding alignments in the original lower-order alignments and use them to derive a new alignment consistency model.

3.3. TIME COMPLEXITY

The time for CBMSA includes three major steps: (1) Generation of the Alignment Consistency Model, (2) Construction of blocks candidates, and (3) Block Assembly. Given N sequences of average length L, step (1) takes $O(N^3L^2)$. Step (2) takes $O(NF_sF)$, if we assume there are, on average, F fragments on each sequence and F_s seed fragments. Since Fs is from two sequences, it is at most $2L$. F is at most L, so step (2) takes $O(NL^2)$. F usually is much smaller than L since we only consider the fragments on a sequence, which have 2-blocks in the alignment consistency model. After all the block candidates are constructed, we identify the optimal set of blocks. It takes $O(B^2)$ time for B constructed blocks. By summing up all the time taken in the above steps, the entire alignment process takes $O(N^3L^2) + O(NL^2) + O(B^2)$ time. Since B is at most $O(NL)$, our algorithm actually takes $O(N^3L^2)$. Compared with Dialign 2, which takes $O(N^4L^2)$ and only uses pairwise-consistency, CBMSA has better time complexity and also is able to use three-way-consistency at the same time. In addition, CBMSA construct blocks across multiple sequences, while Dialign 2 constructs blocks for two sequences only.

This alignment process can be recursively applied to smaller unaligned regions, which is much faster than the process of aligning the whole dataset. We also use the original lower order alignments for the construction of a new region's alignment consistency model. This makes this process more efficient.

3.4 IMPLEMENTATION

This CBMSA program is implemented in Java and has also been integrated with Mesquite. It can be run as an independent program or as a module of Mesquite. The lower-order alignments of CBMSA can be obtained from any external MSA program. By default, we use the fast progressive MSA program, "Partial Order Alignment (POA)" to derive the lower order alignments.

POA provides different configuration options, such as progressive, local, global, and non-progressive. In general, the local POA is suitable for sequences with large length differences; and the global POA is suitable for sequences with similar lengths. The progressive option can make POA alignment more accurate, but also makes it slower at the same time.

We allow the users to specify the configurations of CBMSA. By default, we use an adaptive system to set the configurations of CBMSA. For a set of small sequences (<20 sequences) with large length differences, local progressive POA is used and the fragment length of CBMSA is set to be 3; for a small set of sequences with similar lengths, global progressive POA is used and the fragment length of CBMSA is set to be 7; for a larger set of sequences (>=20 sequences) with large length differences, local POA is used and the fragment length of CBMSA is set to be 14; for a large set of sequences with similar lengths, global POA is used and the fragment length of CBMSA is set to be 14.

3.5 EVALUATION

3.5.1. Comparison of Pairwise and Three-Way-Consistency-Based MSA

In order to compare the performance of pairwise and three-way-consistency-based MSA, we applied both pairwise and three-way consistency based CBMSA to the Balibase's reference sets 4, 5, and twilight sequence set Ref1 (<25% identity). Those datasets show high variability and suitable for local MSA. In the Balibase benchmark, two alignment scores, Column Score and SP Score, are defined. The SP Score (SPS) is defined as "the fraction of residue pairs that are aligned the same way as in the reference alignments" [Lassmann, 2002]. The Column Score (CS) is defined as the number of columns that are consistent between reference and test alignments [Lassmann, 2002]. Hence we used these two scores to evaluate our alignment quality.

Table 3.1. Comparison of Three-Way-Consistency-based MSA and Pairwise-Consistency-based MSA

Data Sets		**Average SPS**		**Average CS**	
		Three-Way Consistency	Pairwise Consistency	Three-Way Consistency	Pairwise Consistency
Ref1 (twilight sequences)	Short length	0.696	0.567	0.500	0.327
	Medium Length	0.743	0.683	0.573	0.538
	Long length	0.647	0.594	0.494	0.387
Ref4 (large N/C terminal extension)		0.957	0.874	0.874	0.719
Ref5 (large internal insertions)		0.893	0.654	0.659	0.654

The test results are shown in Table 3.1. For each of the reference datasets, both the average SPS and CS of three-sequence-way-consistency-based MSA are higher than pairwise-consistency-based MSA. These results showed the proposed three-sequence-alignment approach is able to provide more reliable alignment consistency information and eventually better alignment accuracy than pairwise-consistency-based MSA.

3.5.2. Comparison to other leading MSA programs

The purpose of our experimental study is to measure the alignment quality and biology effectiveness of CBMSA. Three datasets, representative of current biology interest are used. Two of them are the classic motif-finding testing sets: HTH and Lipocalins. Another dataset, MAM domain, is selected from Prints database.

For each dataset, we apply CBMSA to it and compare its performance with the other leading MSA programs, BlockMaker, ClustalW, Gibbs Sampling, MEME, PROBCONS, and POA. All the MSA programs are run with default parameters. For Gibbs Sampling, which needs to pre-specify the number of motifs in the given sequences, we provide the correct number of motifs to it. For each of the three biology datasets, the motifs in it are known. So we did biology validation on each known motif of the dataset. We assess each individual motif using SPS and CS.

Single Motif Case: HTH

This dataset is the set of helix–turn–helix (HTH) proteins containing the HTH motif for DNA-binding involved in gene regulation. It consists of 30 highly variable sequences with the average percentage identity of 19.2%. The lengths of these sequences vary from 61 and 524, with the average being 239. All the sequences contain a common 18-residue motif. The correct locations of occurrence of the motifs are known from X-ray and nuclear magnetic resonance structures. Table 3.2 shows the test results.

Table 3.2. HTH Motif Test Results

Program Name	**SPS**	**CS**
CBMSA	1	1
BlockMaker	0.69	0
Gibbs Sampling	1	1
MEME	0.933	0
Dialign 2	0.681	0
PROBCONS	0.357	0
POA	0.121	0
ClustalW	0.091	0

From the test results, we can see that CBMSA and Gibbs Sampling perform the best. They identify the HTH motif in the dataset correctly. In contrast, BlockMaker, Dialign 2, PROBCONS, POA and ClustalW gave CS of 0 and SPS less than 0.7. This is because they are not explicitly optimized to search for conserved regions vertically. When the conserved region is scattered among divergent and various-length sequences, they may miss it.

Lipocalins-Double Motifs

Most protein sequence families contain multiple colinear elements separated by gaps of variable length. One example, which was once regarded as 'one of the most difficult' is the set of five divergent lipocalins containing two weak motifs of width 16 recognized from structural comparisons. The average length of these sequences is 182.

Table 3.3. Lipocalins Motif Test Results

Program Name	Motif 1		Motif 2	
	SPS	CS	SPS	CS
CBMSA	1	1	1	1
BlockMaker	1	1	1	1
Gibbs Sampling	1	1	1	1
MEME	1	1	0	0
Dialign2	1	1	0.963	0.938
PROBCONS	1	1	0.925	0.8125
POA	0.925	0.875	0.7875	0.6875
ClustalW	1	1	0.85	0.6875

Table 3.3 shows the test results of CBMSA and other leading MSA programs.. From the above results, we can see motif 1 is easier to identify. All the MSA programs except POA identify it correctly. However, for motif 2, it is more challenging. Only CBMSA, Block Maker and Gibbs Sampling are able to identify all of its columns correctly.

<u>MAM Domain Signature-Five Motifs</u>

The PRINTS database contains protein family fingerprints, which are groups of motifs that occur in every family member and thus are characteristic of a family. The identification of the fingerprints within the PRINTS database has been made combining manual alignments and database scanning algorithms from sequence analysis tools.

We selected a challenging dataset, the MAM Domain family, from PRINTS. The MAM dataset we chose consists of 6 highly divergent sequences with the average of percentage identity 0.229. Their length varies from 704 to 1457. MAM Domain has a 5-element fingerprint signature, which spans the full alignment length. The motif lengths are 19,17,12,15, and 14 respectively. Table 3.4 and 3.5 show the test results of CBMSA and other leading MSA programs

Table 3.4 MAM Domain SPS Test Results

Program Name	**Motif1**	**Motif2**	**Motif3**	**Motif4**	**Motif5**
CBMSA	1	1	0.4	0.622	1
Block-Maker	0	0	0	0	0
Gibbs Sampling	0	0.235	0.022	0	0
MEME	1	0.047	0.2	0	1
Dialign2	0.639	0.408	0.267	0.453	0.629
PROBCONS	0.267	0.263	0.267	0.267	0.267
POA	0.937	0.906	0.85	0.636	0.962
ClustalW	0.26	0.267	0.267	0.267	0.257

Table 3.5 MAM Domain Column Score Test Results

Program Name	**Motif1**	**Motif2**	**Motif3**	**Motif4**	**Motif5**
CBMSA	1	1	0	0.467	1
Block-Maker	0	0	0	0	0
Gibbs Sampling	0	0	0	0	0
MEME	1	0	0	0	1
Dialign2	0	0	0	0	0
PROBCONS	0	0	0	0	0
POA	0.895	0.824	0.75	0.467	0.929
ClustalW	0	0	0	0	0

From the test results, we can see among the tested programs, CBMSA and POA perform the best. CBMSA gave the best SP and CS scores for three motifs and POA give the best SP and CS scores for two other motifs. MEME identifies two motifs, motif 1 and 5 correctly, and has 0 column scores for all the others. BlockMaker and Gibbs Sampling gave the worst performance, having the column scores of 0 and very low SPSs (less than 0.27). BlockMaker, Gibbs Sampling and MEME are very sensitive to initial parameter settings. When we have five motifs with different lengths, this makes the ideal parameter settings very difficult, and causes them to be stuck in local optimum when bad initial parameter settings are selected.

From the above three examples, we can see that CBMSA consistently performs favorably for single and multiple subtle motifs.

3.6. DISCUSSION

We propose a new consistency-based local MSA, which uses alignment consistency for block (motif) finding. The major features of this new approach are: (1) we apply three-way alignment consistency for subtle block construction, and (2) we introduce a new alignment-consistency-based clustering method for the fast construction of blocks cross multiple sequences.

Currently, all the existing consistency-based approaches are pairwise-alignment-based. The three-way-alignment-based approach is able to provide more reliable alignment consistency information, while maintaining the same time complex as the pairwise-alignment-based approach. It suggests a new direction for consistency-based MSA methods.

Alignment consistency has been proved to be an effective approach for representing conservation information in traditional progressive MSA. Here we apply it to build blocks in block (motif)-based MSA. By using alignment consistency, the block construction only depends on the inherent conservation features of the sequences and no patterns need to be assumed. Thus, subtle/complex blocks can be represented. In addition, the conservation information derived from alignment consistency of lower-order alignments contains the conservation information over the whole set of sequences. It provides a global model of sequence conservations, avoiding the pitfalls of easily being stuck in local optimum like statistic-based local MSA, which samples only partial regions of the sequences. In order to efficiently use alignment consistency to build blocks, we have transformed the problem of block construction into a fragment-based clustering problem. Based on the obtained alignment consistency, we use a clustering-based method to link the fragments, weighted by alignment-consistency, to form block candidates. No iteration processes and large amount of enumeration are needed in conserved region construction. So multiple block candidates can be constructed very efficiently.

We compared the performance of three-way-consistency-based MSA and pair-wise-consistency-based MSA. We found the proposed three-way-consistency approach was able to provide more reliable alignment consistency information and eventually gave higher alignment accuracy.

We have also compared CBMSA to a suite of leading MSA programs. Our program is very effective in discovering subtle motifs in real protein families. In comparing the motif finding results to other leading MSAs, our program consistently gave the best performance.

Chapter 4 A Biclustering-based Local MSA

4.1 INTRODUCTION

Biclustering is a clustering method that simultaneously clusters both the domain and range of a relation. A challenge in multiple sequence alignment (MSA) is that the alignment of sequences is often intended to reveal groups of conserved functional subsequences. Simultaneously, the grouping of the sequences can impact the alignment; precisely the kind of dual situation biclustering algorithms are intended to address.

Based on this, we propose a biclustering-based local MSA. We define a representation of the MSA problem enabling the application of biclustering algorithms. We develop a computer program for local MSA, BlockMSA, that combines biclustering with divide-and-conquer. BlockMSA simultaneously finds groups of similar sequences and locally aligns subsequences within them. Further alignment is accomplished by dividing both the set of sequences and their contents. The net result is both a multiple sequence alignment and a hierarchical clustering of the sequences.

We applied BlockMSA to highly variable ncRNA for local MSA. ncRNA genes produce some of the cell's most important products, including transfer RNA (tRNA) and ribosomal RNA (rRNA) [104, 110]. However, the role of ncRNA has been underestimated for a long time. These RNAs are now being implicated in various regulatory functions, in addition to their roles in protein synthesis [107]. As a result of this recent change, the number of ncRNAs and our understanding of their importance in cellular metabolism will increase dramatically in the next few years. Large scale MSA is especially needed for highly variable ncRNA families to reveal their functionally conserved regions. Experience is showing that the larger the set of sequences considered, the more biological details reliably emerge. For example, comparative analysis of large-scale multiple RNA sequence alignments has revealed new types of base pairings (e.g. U:U and C:C instead of the canonical A:U, G:C, and G:U) and new structural motifs [111]. Current global MSA methods, such as ClustalW, are useful for closely related RNAs [14]. They become less effective when the number of sequences increases and the sequences are more variable. Current local MSA methods also become less effective for large and variable sequence alignment problems [34].

We tested BlockMSA on three sources of ncRNA problems and compared its results with alignments computed by ClustalW [14], MAFFT [112], MUSCLE [109] and PROBCONS [106]. Since the intention is to support the analysis of large sets of highly variable ncRNAs, first we determined the subsets of BRAliBase 2.1 [113] that display high variability. Second, since the maximum problem size in BRAliBase 2.1 is 15 sequences, we increased test set size and formed problem instances as large as 80 sequences. Third, two large data sets of current biological interest, from the Comparative RNA Web (CRW) Site, are evaluated, T box sequences[103] and Group IC1 Introns [104].

We found on the large-scale benchmark based testing that the alignment programs score in a consistent fashion. BlocksMSA consistently scores best for larger problems, 15 sequences or greater. Otherwise, BlockMSAs performance scores competitively with the best scores produced by the other alignment programs.

Results on the two large biological tests demonstrate that BlocksMSA is the most effective. This is only to be expected as BlocksMSA was explicitly developed for large-scale problems. What was unexpected is that the formal measure of alignment quality, SPS, does not always bear correlation with biological effectiveness.

4.2 BICLUSTERING METHOD

4.2.1 Definition

Clustering is an unsupervised process of grouping together similar data items. In clustering, input data is usually arranged in a data matrix, where the rows correspond to data objects and the columns correspond to their features/attributes. Based on how we analyze the data matrix, we can classify clustering methods into one-way clustering and biclustering [73]. One-way clustering only clusters rows or columns while biclustering clusters both rows and columns simultaneously. In one-way clustering, the similarity between data objects is calculated by using all the features presented in the data matrix. It assumes the data items in a cluster behave similarly over all their features and derives a global model. It works well for a data set with a small number of features. In high-dimensional data, however, not every feature may be relevant to a cluster. Some objects may be correlated over only a few features. Some objects or features may be noisy, irrelevant to any cluster. Biclustering is used in this case. It uses the duality (the "causal" relationship) of the rows and columns to simultaneously cluster them. In biclustering, each object in a cluster is selected using only a subset of features and each feature in a cluster is selected using only a subset of objects. In this way, it discovers local signals/coherences in a data matrix, and derives local clusters within it. It is also able to deal with noisy data by allowing erratic objects or features to belong to no cluster.

Recently, Biclustering has been applied to various areas, such as gene expression data analysis, collaborative filtering, recommendation systems, and text mining [73]. In text mining, biclustering is used to perform simultaneous clustering of documents and words [74]. In this approach, a clustering matrix is represented as a word-by-document matrix, where the rows correspond to words and the columns to documents and each element in the matrix indicates the existence of a word in a matrix. Biclustering is then applied to the matrix to find subsets of documents and subsets of words that are correlated. By using this approach, documents are clustered based on the words they contain and words are clustered based on the documents where they co-occur. This approach has been shown as effective in practical examples, especially for sparse and high-dimensional data matrices. In gene expression data analysis, biclustering has been used to identify groups of genes that show similar activity patterns under a specific subset of experimental conditions [74,75]. Biclustering has also been used in collaborative filtering to identify subgroups of customers with similar preferences towards a subset of products [73].

4.2.2 Biclustering Classification

The exact solution for biclustering is NP-hard. All existing biclustering methods are heuristic-based. They can be roughly divided into seven classes according to the structures of their identified biclusters [73]. They are:

1) One bicluster
2) Exclusive row and column biclusters
3) Non-overlapping biclusters with checkerboard structure
4) Exclusive-rows biclusters
5) Exclusive-columns biclusters
6) biclusters with hierarchical structure
7) Arbitrarily positioned overlapping biclusters

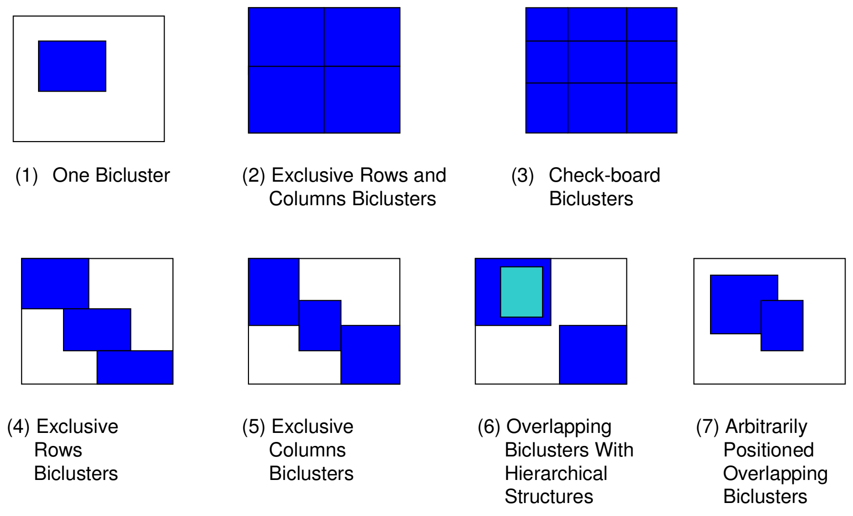

(1) One Bicluster
(2) Exclusive Rows and Columns Biclusters
(3) Check-board Biclusters
(4) Exclusive Rows Biclusters
(5) Exclusive Columns Biclusters
(6) Overlapping Biclusters With Hierarchical Structures
(7) Arbitrarily Positioned Overlapping Biclusters

Figure 4.1 Bicluster Classes (cited from [73])

Class 1 assumes only one bicluster in a data matrix and tries to find the best one among them [76,77]. Classes 2-7 assume there are *k* biclusters in the data matrix. In class 2, every row and every column of the data matrix belongs exclusively to one of the biclusters [74]. Class 3 is a checkerboard structure, where each row belongs to exactly K biclusters and each column belongs to exactly *k* biclusters [78,79]. Class 4 assumes exclusive-rows biclusters, in which a row can only belong to one cluster, while a column can belong to several clusters [80,81]. This algorithms for class 4 can also produce "class 5" when they are used to cluster the transpositions of a data matrix. Class 5 creates exclusive-columns biclusters, where the columns of the data matrix can only belong to one bicluster, while the rows can belong to more than one bicluster. Classes 2-5 assume that the biclusters are exhaustive, that is, every row and every column of the data matrix belongs at least to one bicluster. Classes 6 and 7 are non-exhaustive structures, in which some rows and columns do not belong to any bicluster. Class 6 requires that either the biclusters are disjoint or one includes the other [82,83,84], and Class 7 is a more general bicluster structure, which allows the existence of possibly overlapping biclusters.

4.2.3 Biclustering Examples

Many biclustering algorithms [75, 85, 86, 87, 88, 89, 90, 91, 92, 95, 96] have been developed to allow the more general structure of class 7. Here we review three biclustering algorithms of class 7. They are Plaid Model [75], Flexible Overlapping Clustering (FLOC) [91, 92] and BiMax [95, 96].

4.2.3.1 Plaid Model

Lazzeroni and Owen [75] propose a statistical-model, called a "plaid model", to identify *k* overlapping biclusters. The plaid model describes the input matrix as a linear function of variables corresponding to its biclusters. Specifically, in a plaid model, each element p_{ij} models the interactions between biclusters for the matrix position *(i,j)*:

$$p_{ij} = \sum_{k=0..K} \theta_{ijk} \rho_{ik} k_{jk} \text{ , where} \qquad (4.1)$$

- *K*: *K* biclusters
- θ_{ijk} : contribution of bicluster *k*
- ρ_{ik} : if bicluster k contains row *i*, 1, otherwise 0
- κ_{jk}: if bicluster k contains column *j*, 1, otherwise 0
- $\theta_{ijk}=\mu_k+\alpha_{ik}+\beta_{jk}$, where μ_k is the background level of layer/bicluster k and α_{ik} and β_{jk} are row and column effects of layer/bicluster k respectively.

The key in the plaid model is to identify the correct values of θ_{ijk} , ρ_{ik} and κ_{jk}. Lazzeroni and Owen apply an iterative approach to estimate their values and find one cluster at a time. Assume *k-1* biclusters have been identified, the parameters of the *k*th bicluster are identified by minimizing the sum of squared errors *Q*.

$$Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (Z_{ij} - \theta'_{ijk} \rho'_{ik} k'_{jk})^2 \qquad (4.2)$$

$$Z_{ij} = p_{ij} - \theta_{ij0} - \sum_{k=1}^{K-1} \theta'_{ijk} \rho'_{ik} k'_{jk} \qquad (4.3)$$

Let $\theta^{(s)}$, $\rho^{(s)}$ and $k^{(s)}$ denote all the θ_{ijk} values, all the ρ_{ik} values and all the ρ_{jk} at iteration *s* respectively. The whole plaid algorithm is as follows:

```
Initialize membership $\kappa^{(0)}$ and $\rho^{(0)}$ and specify the number of iterations $S$
            For (i=0;i<S;i++)
                    Determine $\theta^{(i+1)}$ from $\kappa^{(i)}$ and $\rho^{(i)}$
                    Determine $\rho^{(i+1)}$ from $\theta^{(i+1)}$ and $\rho^{(i})$
                    Determine $\kappa^{(i+1)}$ from $\theta^{(i+1)}$ and $\kappa^{(i)}$
                    $\theta^{(i+1)}$,$\rho^{(i+1)}$ and $\kappa^{(i+1)}$ are obtained by lagrange multiplier
```

Figure 4.2. Plaid Algorithm

The advantages of the plaid model are: (1) This model is very flexible. It can identify either class 2 or class 7, depending on different constraints. It allows a cluster to be defined with respect to only a subset of features, not necessarily all of them. (2) There is no need to preset the number of biclusters. During the process, the number of specified clusters can be justified. The disadvantage of the plaid model is it is a local optimal approach, depending on the parameter initializations. With bad initializations, it may be stuck in local optimum.

4.2.3.2. FLOC-Flexible Overlapping Clustering

Flexible Overlapping Clustering (FLOC) [91,92] discovers a set of k overlapping biclusters simultaneously via an iterative greedy approach. It has been successfully applied to analyze gene expression data. In FLOC, gene expression data is represented as a matrix: the rows correspond to genes, the columns correspond to samples and each element of the matrix corresponds to the expression level of a gene in a specific sample. The clustering starts from an initial guess of *k* biclusters in the matrix, and then iteratively refines them until convergence is reached. FLOC converts each element d_{ij} of a given matrix *D* to a residue r_{ij} , which is defined as:

$$r_{ij} = d_{ij} - d_{iJ} - d_{Ij} + d_{IJ} \tag{4.4}$$

where $$d_{IJ} = \frac{\sum_{i \in I, j \in J} d_{ij}}{v_{IJ}} \tag{4.5}$$

$$d_{I,j} = \frac{\sum_{i \in I'} d_{ij}}{|I'_j|} \tag{4.6}$$

$$d_{i,J} = \frac{\sum_{j \in J'} d_{ij}}{|J'_i|} \tag{4.7}$$

and

- $|I|$ and $|J|$ are the number of rows and columns of matrix D
- $|V_{IJ}|$ are the number of specified elements in the matrix D
- $|I'_i|$ and $|J'_i|$ are the number of specified elements on row i and column j in the matrix D

Then for each bicluster B_{IJ}, a "mean squared residue" is calculated based on their residues:

$$H_{ij} = \frac{\sum_{i \in I, j \in J} | r_{ij} |^2}{V_{IJ}} \tag{4.8}$$

Where r_{ij} is the residue of the entry d_{ij} in B_{IJ} and V_{IJ} is the number of specified elements of the bicluster $B_{IJ.}$

Given k biclusters, the goal of FLOC is to reduce their overall mean squared residues. FLOC has two phases. In the first phase, k initial biclusters are generated via a random switch scheme. In the second phase, it uses an iterative approach to improve the quality of the biclusters. During each iteration, the best action for each row and column is determined, and the set of best actions for all the rows and columns are then performed according to a random weighted order. An action of a row/column r with respect to a cluster *c* is defined as: if a row/column *r* already belongs to *c*, the action can be "remove it from cluster *c*"; if it is not in the cluster yet, the action can be "add it to cluster *c*". Given *k* clusters, there are then *k* actions associated with each row/column. Among the *k* actions, the best action for a row/column is the one with the maximum gain, which is defined as a function of the relative reduction of the bicluster residue and the relative enlargement of the bicluster volume. The iteration process continues until no further improvements in biclusters' quality can be made.

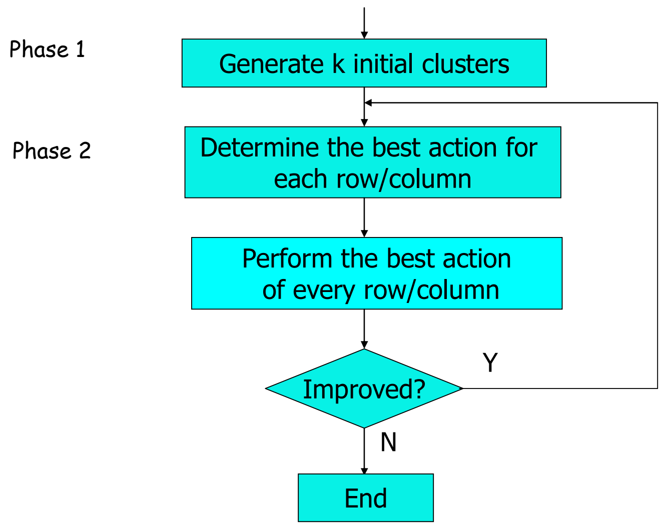

Phase 1
Generate k initial clusters
Phase 2
Determine the best action for each row/column
Perform the best action of every row/column
Improved?
Y
N
End

Figure 4.3. FLOC Algorithm

The advantage of FLOC is it can identify possibly overlapping biclusters simultaneously. The disadvantages are: (1) it is an iterative greedy approach, and probably will be stuck in local minimum, and (2) It needs to preset the number of clusters.

4.2.3.3 BiMax

BiMax [95, 96] applies a divide-and-conquer algorithm to find the biclusters in a matrix. It has been successfully applied to analyze gene expression data.

Bimax is based on a binary matrix model. The model assumes two possible expression levels per gene: no change and change with respect to a control experiment. Accordingly, a set of m microarray experiments for *n* genes can be represented by a binary matrix *E*, where a cell e_{ij} is 1 whenever gene *i* responds in the condition *j* and otherwise it is 0. A bicluster *(G, C)* corresponds to a subset of genes $G \subseteq \{1,\ldots,n\}$ that jointly respond across a subset of samples $C \subseteq \{1,\ldots,m\}$. In other words, the pair *(G, C)* defines a sub-matrix of *E* for which all elements equal to 1. Note that, by definition, every cell e_{ij} having value 1 represents a bicluster by itself. However, such a pattern is not interesting; Bimax hence tries to find all biclusters that are inclusion-maximal, i.e. that are not entirely contained in any other bicluster.

Definition. The pair $(G, C) \in 2^{\{1,\ldots,n\}} \times 2^{\{1,\ldots,m\}}$ is called an inclusion-maximal bicluster if and only if (1) $\forall\ i \in G, j \in C : e_{ij} = 1$ and (2) $(G', C') \in 2^{\{1,\ldots,n\}} \times 2^{\{1,\ldots,m\}}$ with (a) $\forall i' \in G',\ j' \in C': e_{i'j'} = 1$ and (b) $G \subseteq G' \wedge C \subseteq C' \wedge (G', C') \neq (G, C)$.

Since the size of the search space is exponential, an enumerative approach is infeasible to determine the set of inclusion-maximal biclusters. Bimax proposed to use a divide-and-conquer approach to find the binary inclusion-maximal biclusters in the matrix.

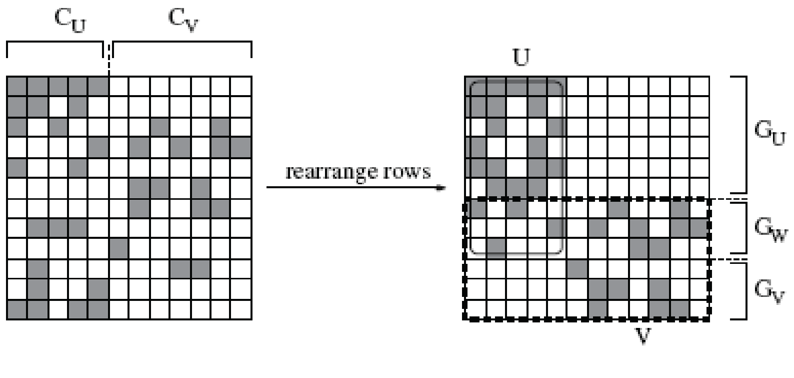


Figure 4.4 Illustration of the Bimax algorithm.

(To divide the input matrix into two smaller, possibly overlapping submatrices U and V, first the set of columns is divided into two subsets C_U and C_V, here by taking the first row as a template. Afterwards, the rows of E are resorted: first come all genes that respond only to conditions given by C_U, then those genes that respond to conditions in C_U and in C_V and finally the genes that respond to conditions in C_V only. The corresponding sets of genes G_U, G_W and G_V then define in combination with C_U and C_V the resulting submatrices U and V which are decomposed recursively.)

It tries to identify areas of E that contain only 0s and therefore can be excluded from further inspection. More specifically, the idea behind the Bimax algorithm, which is illustrated in Figure 4.4, is to partition E into three submatrices, one of which contains only 0-cells and therefore can be disregarded from further consideration. The algorithm is then recursively applied to the remaining two submatrices U and V; the recursion ends if the current matrix represents a bicluster, i.e. contains only 1s. If U and V do not share any rows and columns of E, G_W is empty and the two matrices can be processed independently from each other. However, if U and V have a set G_W of rows in common as shown in Figure 4.4, special care is necessary to only generate those biclusters in V that share at least one common column with C_V.

The advantages of BiMax are: (1) there is no need to preset the number of biclusters, (2) it provides a filtering function, which can be used to filter results into non-overlapping biclusters if needed, and (3) the size of the biclusters can be constrained during the search process. The disadvantage of the BiMax is that its performance depends on its parameter settings.

4.3. A BICLUSTERING-BASED LOCAL MSA

In order to apply biclustering to MSA problem, we define the following mapping from MSA to biclustering (Fig.4.5). Specifically, we represent the MSA problem in a biclustering matrix. Given a set of sequences, we first identify candidate "blocks", possible local alignments of multiple subsequences. We then use them to construct a biclustering matrix where each row corresponds to a candidate block and each column corresponds to a sequence. The value in the matrix is “1” if a block is on the sequence, “0”, otherwise. Biclustering is a two-way clustering. Instead of clustering sequences over all blocks, biclustering can cluster sequences with respect to subsets of blocks and vise versa (Fig.4.6). For each identified bicluster matrix, its columns consist of a subset of sequences, corresponding to a sequence group and its rows consist of a subset of blocks, corresponding to the conserved features for the sequence group (Fig.4.7). We recursively apply the biclustering by excluding the aligned blocks from further considerations and continue the MSA in a divide-and-conquer fashion, one sub-problem for each sequence group.

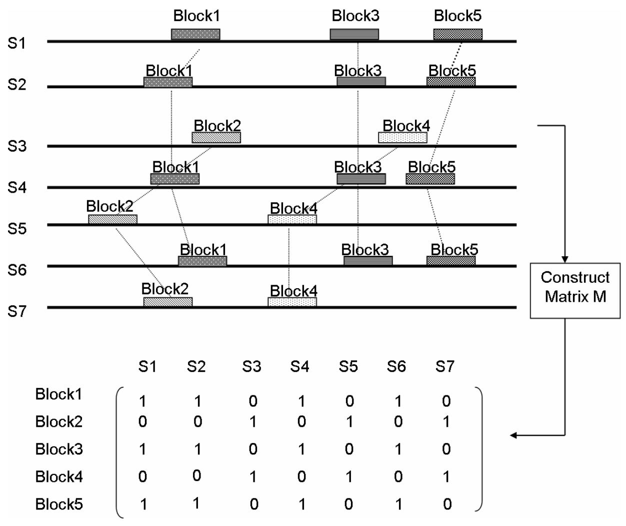


Figure 4.5 MSA to Biclustering Mapping.

(Given a set of sequences, {S1, S2, …, S7}, we first identify the possible “blocks”, local multiple alignments of subsequences. We then represent the MSA problem in a biclustering matrix M, where each row corresponds to a block and each column corresponds to a sequence. The value in the matrix is “1” if a block is on the sequence, “0”, otherwise.)

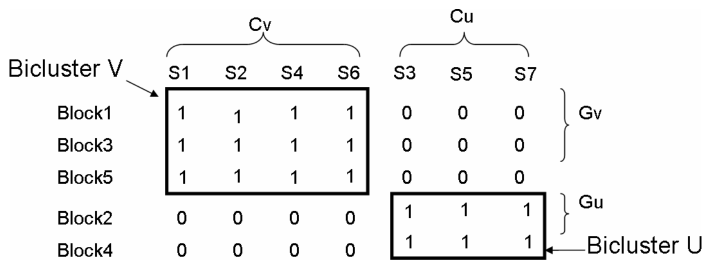


Figure 4.6 Biclustering Matrix *M*

(To bicluster the matrix *M*, we applied the biclustering program, BiMax, to it. BiMax rearranged the rows and columns of Matrix *M* and identified two biclusters *U* and *V*. In this example, a bicluster is defined as a sub-matrix with all it elements equal to 1.)

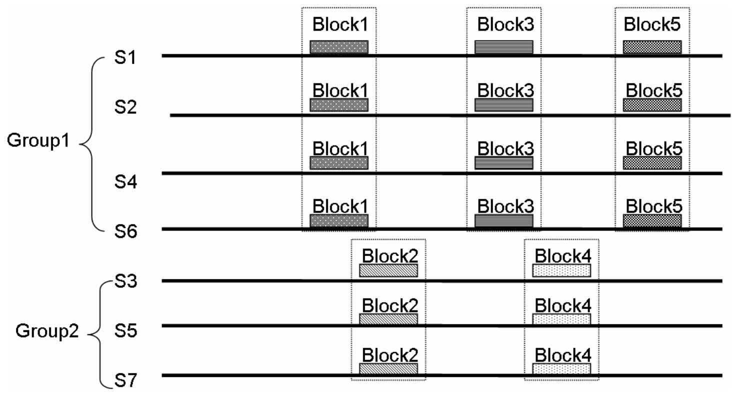


Figure 4.7 Map Biclustering results back to MSA.

(For each bicluster (U or V), the columns (sequences) correspond to a sequence group and the rows (blocks) represent the local alignments in the sequence group. Bicluster V consists of the sequence group {S1, S2, S4, S6 } and its local alignments are {Block1, Block3, and Block5}. Bicluster U consists of the sequences {S3, S5, S7} and its local alignments on them are {Block2, Block4}.)

We applied our algorithm, BlockMSA, to align non-coding RNA (ncRNA) datasets. BlockMSA was tested on the subsets of the BRAliBase 2.1 benchmark suite that display high variability and on an extension to that suite to larger problem sizes. Also, alignments were evaluated of two large data sets of current biological interest, T box sequences and Group IC1 Introns. The results were compared with alignments computed by ClustalW, MAFFT, MUCLE, and PROBCONS alignment programs using Sum of Pairs (SPS), and Consensus Count.

Results for the benchmark suite are sensitive to problem size. On problems of 15 or greater sequences, BlockMSA is consistently the best. On none of the problems in the test suite are there appreciable differences in scores among BlockMSA, MAFFT and PROBCONS. On the T box sequences, BlockMSA does the most faithful job of reproducing known annotations. MAFFT and PROBCONS do not. On the Intron sequences, BlockMSA, MAFFT and MUSCLE are comparable at identifying conserved regions.

4.3.1 Algorithm

The algorithm has three main steps.

1) Identify candidate blocks.
2) Represent MSA as a biclustering problem. Apply biclustering to simultaneously cluster sequences into groups and find the conserved regions within each group.
3) For each sequence group, recursively apply the above two steps, until the sequence group is small enough or no more conserved regions are found within it.

We expand on each of the three steps. For the definitions of Block, Block Similarity Score, Block Order, Chain and an Optimal Set of Blocks, please refer to the definitions given in section 3.2.

4.3.1.1 Identify Candidate Blocks

Typically a given set of RNA sequences has multiple conserved-regions within it. Our goal here is to identify the set of possible conserved regions. Our algorithm is heuristic, consisting of three main steps: (1) dividing the sequence into overlapping k-length fragments, (2) calculation of fragment similarity scores, and (3) construction of candidate blocks. Each of the above steps is detailed as follows:

Step (1) Dividing sequences into fragments

For each sequence, it is divided into overlapping k-length fragments.

Step (2) Calculation of fragments' similarity scores

We first generate a library of pairwise alignments for all possible sequence pairs. The pairwise alignments can be obtained from any external sequence alignment program. We used ClustalW to do the pairwise alignment.

For each pairwise alignment, a window of length k is slid through it and each position of the window produces a k-length sub-region. We only keep those un-gapped sub-regions whose similarity scores are greater than a pre-specified threshold. Thus each sub-region is actually a 2-block. Hence, for each 2-block, its un-gapped alignment region provides its fragments' similarity score. For the fragments not appearing in the alignments, their similarity score is set to 0.

Step (3) Construction of candidate blocks

In this step, we construct block candidates from the pool of 2-blocks. To do so, we use a greedy clustering-based method. We can think of each fragment as a point and an n-block as an n-point cluster. To determine an n-point cluster, we start the initial cluster with a seed point (fragment) and gradually add one fragment from each sequence to the cluster. Hence an n-point cluster will consist of n fragments from n sequences and form an n-block. To add a fragment to the cluster, we pick the fragment which has the maximum sum of similarity scores with the fragments in the cluster. The reason behind this is if a fragment has a higher similarity score with a cluster, it means it has more alignment consistency with the fragments in the cluster and shows more vertical conservation consistency. By using the clustering method to construct blocks, we directly integrate consistency calculation into the block construction process and don't need to exhaustively calculate all the consistency possibilities. For seed fragments, we choose them from two closest sequences.

4.3.1.2 Represent MSA as a Biclustering Problem

Our biclustering procedure has two main steps:

1) Convert to Biclustering Matrix

2) Biclustering with respect to the matrix to identify sub- groups of sequences and conserved regions among them.

Convert to Biclustering Matrix

The initial clustering matrix is defined on matrix M. The rows of M represent the set of block candidates $B=\{B_1, B_2,\ldots,B_n\}$, the columns of M represent the set of sequences $S=\{S_1, S_2,\ldots,S_m\}$, and each element M_{ij} of the matrix is set as "1" if the block B_i covers the sequence S_j. Otherwise M_{ij} is set as "0".

Special Case in Biclustering Matrix

Before we apply biclustering to the matrix, we need to first consider a special case: there may exist highly conserved regions (blocks) across all the sequences.

In the biclustering matrix M, each row corresponds to a block's coverage on sequences. If a block spans all the sequences, its values in the matrix are all-1s. So before applying biclustering, we check if there are any all-1's rows, note the corresponding blocks as being conserved across all the sequences. We refine the matrix M by excluding those all-1's rows.

Biclustering With Respect to the Matrix

In this step, the matrix is biclustered. Among many biclustering packages, BiMax (Barkow *et. al.*, 2006) is the most suitable. BiMax doesn't need the users to pre-specify the number of the biclusters. It uses a divide-and-conquer approach to find the inclusion-maximal biclusters. Although it allows overlapped biclusters, it provides a filtering function, which we used to filter results into non-overlapping biclusters.

After applying biclustering, for each obtained bicluster matrix, its columns consist of a subset of sequences, corresponding to a sequence group and its rows consist of a subset of blocks, corresponding to the conserved features of the sequence group. The blocks for each sequence group can be further refined via block assembly and post-processing.

Block Assembly and Post-Processing

After we obtain the blocks for a sequence group, which may be too short, we extend them to both sides until their similarity score falls below a predefined threshold. We also merge two blocks if they are within a relatively short distance. After extending and merging, we can identify an optimal set of non-overlapping blocks. The identification of an optimal set of blocks is actually a classic chaining problem, and can be interpreted as a maximum weight path problem in a directed acyclic graph [1]. We have implemented a single source DAG shortest path algorithm to find the optimal chain. The time complexity is $O(n^2)$ time for a given set of n blocks.

Sometimes a conserved region may not cover all sequences. A block may miss one or two sequences. After we identify all the well-conserved blocks across all the sequences, we can look for those weakly conserved blocks, which may miss a few fragments, and add these blocks back to the optimal chain.

4.3.1.3 Recursion

For each sequence group, BlockMSA recursively applies the above two steps 4.3.1.1 and 4.3.1.2, until each sequence group is small enough or no more conserved regions are found within it.

4.3.1.4 Time Complexity

BlockMSA consists of three main procedures. The time for step (1), the identification of block candidates takes $O(nf_sl)$, where n is the number of sequences, f_s is the number of seed fragments and l is the average sequence length in an unaligned region. In step (2), constructing biclustering matrix takes $O(bn)$, where b is number of block candidates. For biclustering, we used BiMax, which takes $O(bnMin(b,n)\beta)$, where β is the number of all inclusion-maximal biclusters. In the BiMax, BlockMSA allows setting up a threshold to limit the number of biclusters, β, to be identified. When the number of sequences is large, we can set a higher threshold to make β smaller. The total time for step (1) and (2) is $O(nf_sl)+O(bnMin(b,n)\beta)$.

Step (3) is a recursion process, which recursively clusters sequences and identifies the blocks in them. This step should be much faster, since we decompose the problems into smaller and smaller problem. This also reflects the advantage of divide-and-conquer, which makes the following step work on smaller problems. The worst running time of BlockMSA is $O(n^2f_sl)+O(bn^2Min(b,n)\beta)$.

4.3.2. Implementation

The BlockMSA program is implemented in Java. The pairwise alignments can be obtained from any sequence alignment program. We used ClustalW to do the pairwise alignment. BlockMSA has been tested on Linux, Unix and OS-X. BlockMSA is available under open source licensing and is distributed with documents and examples.

4.3.3 Evaluation

4.3.3.1 Alignment Test Sets

Currently, there is no standard benchmark for RNA local MSA. The BRAliBase 2.1 benchmark suite is not specifically designed for local alignment testing. It has the datasets with percentage identity ranging from 20% to 95% and the maximum number of sequences per test set is 15, which is not enough for testing scalability and robustness. Further, the ratio between the average sequence length and the total number of columns in a reference alignment is usually high, which does not represent cases with large insertion/deletions.

Based on the analysis above, we chose three types of test sets: (1) The subsets of BRAliBase which are highly variable and suitable for local MSA; (2) LocalExtR, an extension of BRAliBase 2.1, comprising larger-scale test groups and patterned on BRAliBase 2.1. (3) LSet, a pair of large-scale test sets representative of current biological problems.

The subsets from BRAliBase are selected from the most variable test sets within the suite. They are from the THI, Glycine riboswitch and Yybp-Ykoy RNA families, and contain 232 test datasets. LocalExtR uses the same seed alignments from Rfam that BRAliBase uses and forms larger test groups. The BRAliBase convention is to label a test group *ki*, where *i* is the number of sequences for each test set in the group. We created four new test groups, *k20*, *k40*, *k60* and *k80*, totaling 90 test sets (see table 4.1 and 4.2).

The new test sets maintain the percentage identity to be less than 60%. To model large insertions/deletions, the ratio of the average sequence length to the total number of columns in the reference alignments is as small as 0.36 (See Table 4.2)

Table 4.1. Test Dataset Number of Each Test Group.

(The table lists the test dataset number in each test group. Group k5, k7, k10, and k15 are chosen from the existing BRAliBase 2.1 and Group k20, k40, k60, k80 of LocalExtR are the extension to the test groups of BRAliBase 2.1 and generated from the Rfam database.)

Test Group		THI Family	yybP-ykoY Family	gcvT Family	∑
BRAliBase 2.1 (232 datasets)	k5	69	33	22	124
	k7	32	18	12	62
	k10	17	12	3	32
	k15	5	8	1	14
LocalExtR (90 datasets)	k20	10	10	10	30
	k40	10	5	10	25
	k60	10	0	10	20
	k80	10	0	5	15
∑		163	86	73	322

Table 4.2. Measures of the BRAliBase and LocalExtR Test Groups.

(ki indicates a test group containing i-sequence datasets. Note there are many test datasets for each ki. The table details, the number of test datasets, the average sequence length, and the average of the minimum and maximum sequence length per test sets within a group. Similarly, the ratio of the average sequence length to the reference alignment's length and percentage identity (PI).)

Test Group		Sequence Length, Average Of Each Value Over ki Set			Average Ratio: Avg-Length/ Reference	Avg PI (%)
		Avg.	Min	Max.		
BRAliBase 2.1 Subsets 232 datasets	k5	109	96	125	0.79	51.1
	k7	110	94	131	0.75	49.8
	k10	108	94	129	0.72	49.3
	k15	110	88	137	0.67	49.3
LocalExtR 90 datasets	k20	115	90	172	0.53	48.4
	k40	114	87	180	0.47	48.5
	k60	107	81	189	0.40	50.7
	k80	106	77	204	0.36	54.8

LSet contains a set of 248 T box leader sequences and a set of 90 Group IC1 Introns from the CRW Site. T box leader sequences are located upstream of many aminoacyl-tRNA synthetase (AARS), amino acid biosynthesis and amino acid transporter genes in gram-positive bacteria. Group IC1 Introns represent a family of RNA molecules with a specific higher-order structure and the ability to catalyze their own excision by a common splicing mechanism [105]. In addition to containing many more sequences, both the average sequence length and the differences in sequence lengths are much larger than for the BRAliBase test sets (See Table 4.3).

Table 4.3. LSet.

(The table details the number of sequences in a test dataset, average, minimum and maximum sequence length, the ratio of the average sequence length to the reference alignment's length and the percentage identity (PI))

Test Group	Data-Set Size	Sequence Length			Ratio Avg-Length/ Reference	PI (%)
		Avg.	Min	Max		
T box	248	269	78	365	0.40	40.0
Group IC1 Intron	90	563	347	1910	0.12	38.0

4.3.3.2 Scoring Alignments

We used two independent yet complementary scores to evaluate alignment quality, the Sum-of-Pairs Score (SPS) and Consensus Count.

SPS [50, 51] measures the level of sequence consistency between a test and a reference alignment by comparing all possible pairs per column between both alignments; it ranges from 0 (no agreement) to 1 (complete agreement).

The Consensus Count is a measurement for column conservation. It can be computed without identifying a reference alignment. Given a threshold, a consensus sequence represents each column of an alignment with the majority character in that column. Specifically the major character's percentage in the column should be greater or equal to the given threshold. The consensus sequence represents a column with a gap if the major character's percentage in that column is less than the specified threshold. After we obtain the consensus sequence in an alignment, the Consensus Count is the number of non-gap residues within the consensus sequence. It measures the vertical conservation in an alignment.

4.3.3.3 Biological Validation

In order to test the biological effectiveness of the different alignments we compared the output of the five programs (BlockMSA, ClustalW, MAFFT, MUSCLE, and PROBCONS) to each other and noted their ability to correctly align conserved areas. The T box sequences have been studied enough that we could make a quantifiable assessment with respect to known conserved functional subsequences. First, all the T box gene sequences have evolved from a common ancestor and contain a conserved 14-nucleotide sequence, 5′-AGGGUGGNACCGCG-3′ [103]. In addition, the T box dataset contains sequences from 12 major gene groups. Each gene group shares a common triplet sequence representing a specifier sequence codon for the amino acid matching the amino acid class of the downsteam coding region [103]. For example, all tyrosyl genes contain a UAC tyrosine codon, leucine genes contain a CUC leucine codon. Identification of the specifier sequence in each gene group can provide insights into amino acid specificity.

Here a major gene group in the reference alignment is defined as having more than four sequences and strongly showing a specifier sequence pattern. This is because if a gene group does not have enough representative sequences, the specifier sequence feature may not be apparent. Thus, consistent with the biological goals of a local multiple RNA sequence alignment, the T box test set allows us to check if a program correctly align the T box motif and count the number of specifier sequences successfully identified by a program.

The Intron data set is not yet as well annotated as the T box data set. We simply note the program's relative ability to identify conserved regions.

4.3.3.4. Results

BlockMSA is compared with the leading RNA MSA programs, ClustalW, MAFFT(L-INS-i), MUSCLE and PROBCONS. ClustalW, MAFFT(L-INS-i), MUSCLE and PROBCONS were run using their default parameter settings. BlockMSA was run using block size of 11 per the following.

<u>*Choice of BlocksMSA Block Size*</u>

We evaluated a choice of block size ranging from 3 to 15 on the 322 datasets from BRAliBase and LocaExtR. For each block size, we calculate the mean of SPSs over all the test groups. Block size of 11 gives the best result. (See Table 4.4 and Fig. 4.8).

Table 4.4. Block Size Test Results

Block Size	k5	k7	K10	k15	k20	k40	k60	k80	Mean of SPS
3	0.656	0.655	0.668	0.671	0.654	0.681	0.741	0.739	0.683
4	0.651	0.652	0.667	0.706	0.647	0.684	0.737	0.736	0.685
5	0.654	0.660	0.681	0.693	0.652	0.694	0.737	0.750	0.690
6	0.649	0.664	0.656	0.707	0.656	0.697	0.748	0.771	0.694
7	0.649	0.662	0.658	0.689	0.651	0.709	0.742	0.759	0.690
8	0.656	0.652	0.647	0.727	0.663	0.695	0.733	0.755	0.691
9	0.647	0.660	0.663	0.718	0.653	0.704	0.738	0.755	0.692
10	0.662	0.679	0.678	0.741	0.673	0.718	0.774	0.781	0.713
11	0.663	0.670	0.676	0.748	0.668	0.715	0.780	0.791	0.714
12	0.655	0.665	0.661	0.731	0.658	0.723	0.774	0.763	0.704
13	0.649	0.663	0.682	0.719	0.672	0.708	0.764	0.789	0.706
14	0.634	0.646	0.659	0.722	0.658	0.689	0.747	0.767	0.690
15	0.620	0.629	0.611	0.691	0.650	0.699	0.741	0.723	0.671

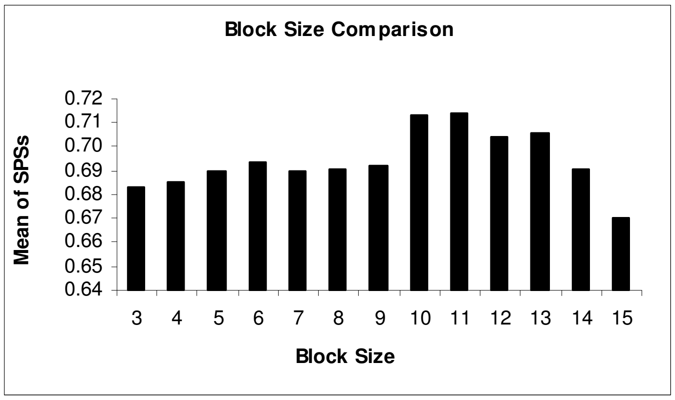


Figure 4.8 Block Size Comparison.
(We test the block size from 3-15 on the test groups from BRAliBase and LocaExtR. For each block size, we calculate the mean of SPSs over all the test groups. Block size of 11 gave the best result.)

SPS Scores

The SPS score of each dataset is calculated by using the compalign program [108] (See Table 4.5 and Fig. 4.9). BlockMSA has the leading performance for the test groups for the benchmark tests containing 15 sequences or larger, i.e. *k15* through *k80* of BRAliBase and LocalExtR. BlockMSA demonstrates a trend of increasingly better performance with larger problem size. This trend is contradicted for the T box test set but not the Intron test set. For the three smallest test sets, *k5, k7* and *k10,* MAFFT has the best scores, followed by PROBCONS and then BlockMSA.

Table 4.5. SPS Test Results

Program Name	k5	k7	k10	k15	k20	k40	k60	k80	Group IC1 Intron	T box
BlockMSA	0.663	0.670	0.676	0.748	0.668	0.715	0.780	0.791	0.510	0.391
ClustalW	0.488	0.454	0.466	0.433	0.403	0.416	0.419	0.421	0.373	0.225
MAFFT	0.676	0.678	0.694	0.686	0.667	0.684	0.748	0.761	0.487	0.384
MUSCLE	0.632	0.624	0.613	0.613	0.579	0.635	0.685	0.689	0.429	0.353
PROBCONS	0.676	0.677	0.671	0.686	0.650	0.670	0.721	0.729	0.499	0.435

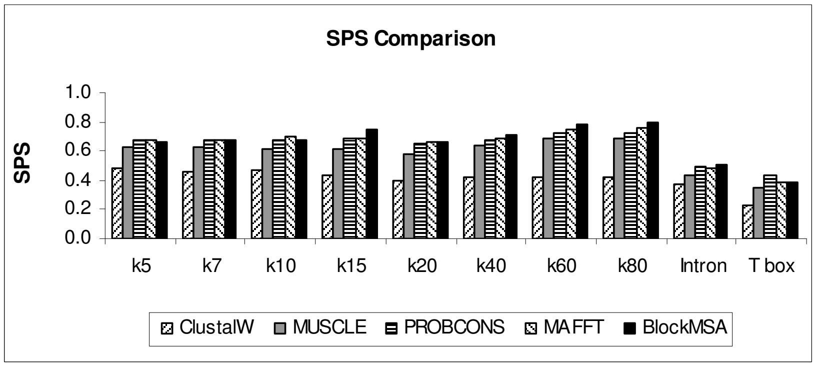


Figure 4.9. SPS Comparison.
(BlockMSA, ClustalW, MAFFT, MUSCLE, and PROBCONS are run on BRAliBase, LocalExtR and LSet. In BRAliBase and LocalExtR, BlockMSA has the leading performance as the input set size increases to 15. For Intron dataset, BlockMSA also leads.)

Consensus Count

Consensus Counts were calculated using a 90% consensus threshold (See Table 4.6 and Fig. 4.10). BlockMSA displays the highest Consensus Count for k15 and ties with MAFFT for the highest consensus count for the benchmark sets larger than 15 sequences. On the smaller problems, PROBCONS has the leading performance. On all the *ki* test groups, except *k5*, all three of these programs attain Consensus Counts no different than 2 from each other, on counts that range from 19 to 25.

Table 4.6 Consensus Count Results

Program Name	k5	k7	k10	k15	k20	k40	k60	k80	Group IC1 Intron	T box
BlockMSA	24	24	24	21	21	19	21	21	55	27
ClustalW	21	18	20	12	13	10	10	9	20	0
MAFFT	25	23	23	19	21	19	21	21	55	19
MUSCLE	24	21	23	17	18	17	18	19	42	16
PROBCONS	28	25	25	20	21	19	19	19	33	20

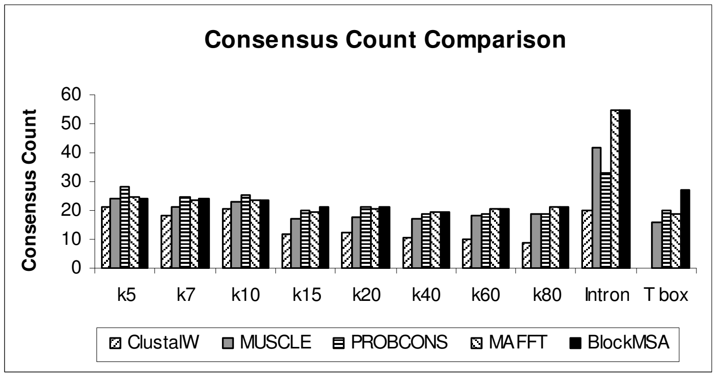


Figure 4.10. Consensus Count Comparison.
(BlockMSA, ClustalW, MAFFT, MUSCLE, and PROBCONS are run on BRAliBase, LocalExtR and LSet. For each of the test group, we calculate their average Consensus Count over all the datasets of the test group with the 90% consensus threshold. BlocMSA has leading performance as the number of sequences increases.)

BlockMSA also displays the highest Consensus Count for the two biological data sets of LSet. On the T box data, BlockMSA achieves a Consensus Count of 27. Both PROBCONS and MAFFT fall behind, scoring 20 and 19. For the Intron data set, BlockMSA and MAFFT tie for the highest score, 55, followed by MUSCLE, 42, and PROBCONS, 33.

Biological Validation

BlockMSA demonstrates the best results on the T box data set. BlockMSA identifies the T box motif as a conserved region. It identifies 10 of 12 specifier sequences in the data set.

Table 4.7. Identification of "Specifier Sequences"

(A T box is defined as identified if a 60% consensus of an alignment produces the motif in its entirety and no gap is inserted within the motif to break its contiguousness. Specifier sequence, (maximum of 12), is defined as identified if all three nucleotides in the codon are present by simple majority consensus)

Number	Gene Group	Specifier Seq. In Gene Group	Identify Specifier Sequence Pattern? 1: Yes and 0: No				
			BlockMSA	ClustalW	PROBCONS	MUSCLE	MAFFT
1	Asn	AAC	1	1	0	1	1
2	Trp	UGG	1	0	0	1	0
3	Pro	CCU	1	1	0	1	1
4	Ile	AUC	1	0	0	1	0
5	Gly	GGC	1	0	0	0	0
6	Ala	GCU	0	0	0	0	0
7	Cys	UGC	1	1	0	1	1
8	Leu	CUC	1	0	0	1	0
9	Phe	UUC	0	1	0	0	0
10	Thr	ACC	1	1	0	1	1
11	Tyr	UAC	1	0	0	1	0
12	Met	AUG	1	1	0	1	1
Total Number of Identified Specifier Sequences			10	6	0	9	5

MUSCLE and ClustalW also identify the T box motif, and 9 and 6 specifier sequences respectively. MAFFT and PROBCONS, which score well using SPS and Consensus

Count does not identify the T box motif. Their ability to identify the specifier sequences, 5 and 0 respectively, is consistent with their results on the T box motif.

The entire set of Intron sequences is not yet annotated, thus we can't quantify the results as we did for the T box. Disappointingly, none of these five programs, separately or together, produce a sufficiently palpable alignment for us to promptly annotate the sequences.

We computed consensus sequences and their Consensus Counts for each of the 5 alignments with 4 consensus thresholds (Table 4.8). By Consensus Count, MAFFT and BlockMSA continue to be very close in performance for consensus thresholds of 80% or greater. At lower consensus threshold, 70%, MAFFT and MUSCLE hold an edge.

Table 4.8. Consensus counts for Group IC1 Intron test set at 4 consensus thresholds for 5 multiple sequence alignment programs

MSA Program	Number of Columns	Consensus Count			
		70% Consensus Threshold	80% Consensus Threshold	90% Consensus Threshold	95% Consensus Threshold
BlockMSA	3301	101	83	55	36
ClustalW	2691	71	48	20	7
MAFFT	3274	115	82	55	39
MUSCLE	1945	108	79	42	26
PROBCONS	4931	100	64	33	7

We manually aligned the 20 consensus sequences. From an inspection of that alignment, we are able to make the following qualitative assessment. The 5 alignment programs largely agreed that the group IC1 Introns contain 8 conserved regions. At comparable Consensus Counts, independent of the consensus threshold needed to achieve that count, the 5 programs largely agreed on the location and contents of the conserved region. Thus, qualitatively, we conclude that BlockMSA, MAFFT, and MUSCLE were better at aligning the Intron sequences than PROBCONS and ClustalW, a conclusion that correlates with Consensus Count.

Special Case: Biclustering the Sequences Grouped A Prior.

In some cases, biologists may already know that some sequences belong to the same functional group. In this case, we can integrate the group information into alignment. We can regard a data set as a collection of sequence groups. A sequence group will be the smallest unit in a dataset and the sequences in the same pre-specified group will always be together. If a sequence's group is not specified, it will be regarded as a sequence group with only one sequence. We can then cluster sequence groups and find conserved regions across sequence groups.

We aligned the T box dataset with prior group information and without group information respectively. The T box with known group information gave better results, but the difference is marginal. For the T Box with prior group information, its SPS is 0.3907. The Sum-of-Pairs score for the alignment without prior group information is 0.3902.

Running-Time Test

We compared the running time of BlockMSA, PROBCONS, MUSCLE, MAFFT, and ClustalW on the test groups from BRAliBase 2.1 and LocalExtR (see Table 4.9 and Figure 4.11). Our program BlockMSA is written in Java. All the other programs are written in C. A program written in C usually runs faster than the program written in Java when they have the same algorithm complexity. So our program speed is affected by Java programming language. Even though BlockMSA is written in Java, as the number of sequences increase to be 40 or greater than 40, BlockMSA runs faster than PROBCONS on the test suite. Our program's time complexity also depends on the time complexity of biclusteirng algorithm. Currently, biclustering is still an active research area. We expect as the biclustering techniques improve and mature, BlockMSA running speed will also improve.

Table 4.9. Running time test results on the test groups of BRAliBase and LocalExtR

Program Name	k5 (sec)	k7 (sec)	k10 (sec)	K15 (sec)	k20 (sec)	k40 (sec)	k60 (sec)	k80 (sec)
BlockMSA	0.419	0.507	0.653	0.855	1.458	4.152	7.359	14.783
ClustalW	0.015	0.026	0.041	0.080	0.133	0.392	0.759	1.282
MAFFT	0.111	0.137	0.177	0.214	0.361	0.877	1.652	2.751
MUSCLE	0.167	0.224	0.290	0.516	0.694	1.699	3.127	5.473
PROBCONS	0.071	0.144	0.254	0.594	1.208	5.183	11.409	21.594

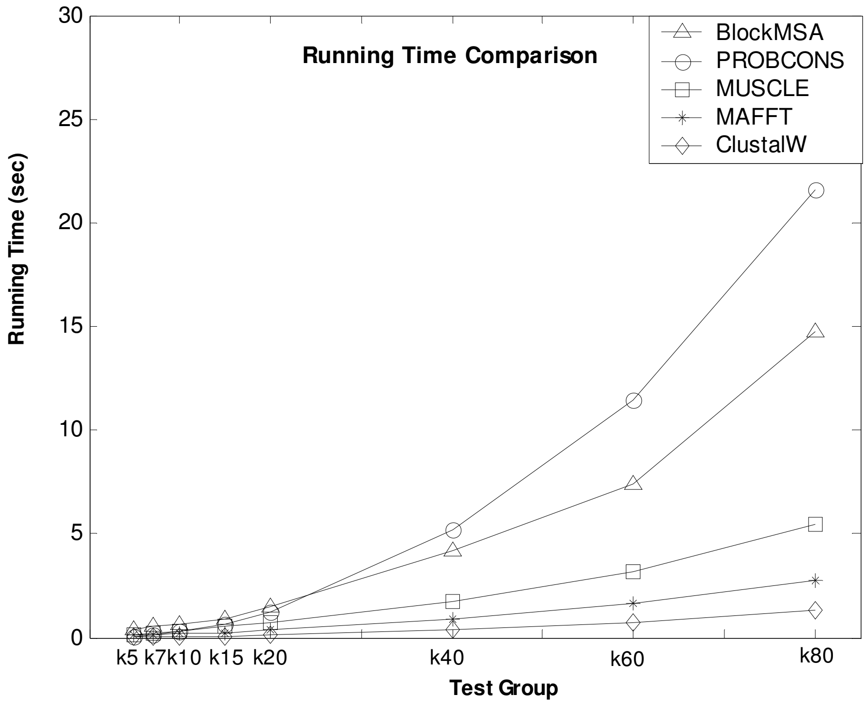


Figure 4.11 Running Time Comparison

4.4. DISCUSSION

Comparative studies of biological sequence data present us with an opportunity to decipher the structure, function and evolution of cellular components. The accuracy and detail of these studies are directly proportional to the quality of the sequence alignments. ncRNAs pose special problems compared to other sequences. Unlike gene sequences, where single nucleotide polymorphism is often significant, a nucleotide substitution in a helix of an ncRNA is easily compensated by a substitution in the corresponding base pair. Consequently, large homologous sets of ncRNAs often display a high of variability in both length and content.

In our use of biclustering, intuitively, we first perform a large scale search for sets of short, local, multiple alignments, (blocks). Many of these blocks overlap in sequences and suggest different sequence groupings. Rather than simply marking conflicting blocks as mutually exclusive and maximizing the number of consistent blocks, biclustering seeks to maximize the total consistency. Conflicting aspects of the block definitions are simply removed. Blocks, once chosen are not further subdivided. The approach finds and then maintains local areas of agreement.

With respect to quantitative measures of MSA, BlockMSA scores comparable to or better than the other leading MSA programs. A contrast surfaces between BlockMSA and the other leading MSA programs in the alignment of large sets of T box and Intron sequences. The other leading MSA programs lag BlockMSA in their ability to identify the most highly conserved regions, and therefore functionally and structurally important parts of the ncRNA sequences. Our conjecture is that PROBCONS and likely MAFFT, which are iterative optimization methods, are introducing gaps to improve SPS score which then results in the break up and misalignment of contiguous conserved regions. It is plausible that this strategy is effective for proteins and coding genes but not ncRNAs.

In highly variable ncRNAs, critical (conserved) structures are often flanked by less conserved sequences. These flanks, incorrectly, may provide iterative methods with flexibility to improve score. Without a penalty for breaking up contiguous conserved regions, local optimizations may insert gaps in those contiguous areas. In proteins the larger alphabet size may be enough to limit this effect. In coding genes, misaligning reading frames can also be expected to come at a large price.

Our alignment results depend on the performance of the biclustering program. Currently, biclustering is still an active research area [73, 95, 96, 101, 102]. We expect as the biclustering techniques improve and mature, the performance of BlockMSA will improve.

Chapter 5 Conclusion

Recent advances in Genome sequencing projects have led to an explosion of data in sequence databases. Fast and scalable local multiple sequence alignment is being needed for remote homology detection, phylogeny tree reconstruction and function annotations. In this dissertation, we developed two new algorithms for fast and scalable local multiple sequence alignment, a three-way-consistency-based MSA and a biclustering-based MSA.

The first local MSA algorithm, the thee-way-Consistency-Based MSA (CBMSA) has been implemented and evaluated. Our algorithm uses alignment consistency heuristics in the form of a new three-way alignment approach. While our three-way consistency approach is still able to maintain the same time complexity as the traditional pairwise-consistency-based approach, it provides more accurate consistency information and eventually better alignment quality. We directly incorporated alignment-consistency information into block-finding process. By using alignment consistency, the block construction only depends on the inherent conservation features of the sequences and no patterns need to be assumed. Thus, subtle/complex blocks can be represented. In addition, the conservation information derived from alignment consistency of lower-order alignments contains the conservation information over the whole set of sequences. It provides a global model of sequence conservations, avoiding the pitfalls of easily being stuck in local optimum like statistic-based local MSA, which samples only partial regions of the sequences.

We quantify the benefit of using three-way alignment consistency as compared to pairwise alignment consistency. We also compared our CBMSA to a suite of leading MSA programs and CBMSA consistently performs favorably.

We also develop a biclustering-based local MSA program, BlockMSA. Biclustering has been applied to text mining, collaborative filtering and gene expression analysis [73, 95, 96, 100, 101, 102]. Our approach is the first effort to represent a MSA problem as a biclustering problem. A challenge in multiple sequence alignment is that the alignment of sequences is often intended to reveal groups of conserved functional subsequences. Simultaneously, the grouping of the sequences can impact the alignment – precisely the kind of dual situation biclustering is intended to address. Biclustering is a clustering method that simultaneously clusters both the domain and range of a relation. We define a representation of the MSA problem enabling the application of biclustering algorithms. We incorporate this method into a divide-and-conquer for local MSA, such that conserved blocks of subsequences are identified and further alignment is accomplished by solving subproblems, by subdividing both the set of sequences and their content. The net result is both a multiple sequence alignment and a hierarchical clustering of the sequences.

BlockMSA was tested on non-coding RNA datasets. Specifically, it was tested on the subsets of the BRAliBase2.1 benchmark suite that displays high variability and on an extension to that suite to larger problem sizes. Also, alignments were evaluated of two large data sets of current biological interest, T box sequences and Group IC1 Introns. The results were compared with alignments computed by ClustalW, MAFFT, MUCLE, and PROBCONS alignment programs using Sum of Pairs (SPS), and Consensus Count.

Results for the benchmark suite are sensitive to problem size. On problems of 15 or greater sequences, BlockMSA is consistently the best. On the T box sequences, BlocksMSA does the most faithful job of reproducing known annotations. MAFFT and PROBCONS do not. On the Intron sequences, BlocksMSA, MAFFT and MUSCLE alignments are comparable at identifying conserved regions.

We foresee at least two natural extensions of BlockMSA. First, there is considerable flexibility in how candidate blocks are defined. One extension is to introduce secondary structure constraints into the construction of candidate blocks. This approach may improve the quality of the blocks while reducing the size of the biclustering matrix. Second, our use of biclustering has been to use existing clustering packages as black-boxes. A consequence is we can make no interpretation about our hierarchical clustering except as a functional taxonomy. However, our biclustering matrix is not so dissimilar to a phylogenetic matrix. The sequences represent unique taxa. Ones in the matrix represent shared features. Biclustering identifies those features for which there is sufficient heuristic evidence to assert the features as conserved within a group. It seems likely that if the algorithmic structure of BlockMSA were refined with erudite choices of method for the creation of candidate blocks a phylogenetic interpretation of the hierarchical clustering could be inferred.

Reference:

[1] Dan Gusfield. Algorithms On Strings, Trees, and Sequences. Cambridge University Press, Cambridge, UK.(1997).

[2] Durbin R., S. Edy, A. Krogh and G. Mitchison.. Biological Sequence Analysis.Cambridge University Press, Cambridge, UK (1998).

[3] Higgins D., and W. Taylor. Bioinformatics: Sequence, Structure and Databanks. Oxford University Press, New York(2002).

[4] Introduction to Bioinformatics, Lecture 3: Multiple Sequence Alignment. http://kbrin.a-bldg.louisville.edu/~rouchka/CECS694/Lecture3.htm

[5] Einat. H.C., N. Altman, Horowitz, and D. Graur. The Evolutionary History of Prosaposin: Two Successive Tandem-Duplication Events Gave Rise to the Four Saposin Domains in Vertebrates. Journal of Molecular Biology. 54: 30-34 (2002).

[6] Einat. H.C. Supplementary Data: prosaposin alignment. http://kimura.tau.ac.il/~einat/prosaposinalignment.pdf.

[7] Blanchette M., and M. Tompa**.** Discovery of Regulatory Elements by a Computational Method for Phylogenetic Footprinting. Genome Research. 12(5): 739-748(2002).

[8] Phylogenetic Footprinting Note. http://n-c-thoughts.blogspot.com/2003_08_01_n-c-thoughts_archive.html

[9] Aravind L. An Evolutionary Classification of the Metallo-ß-lactamase Fold Proteins. *In Silico* Biology. 1, 0008 (1998).

[14] Thompson J, Higgins D, Gibson T. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22, 4673-4690(1994).

[15] Stoye J, Moulton V, Dress AW: DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. Comput. Appl.Biosci. 13(6), 625-6 (1997).

[16] Notredame C, Higgins DG: SAGA: sequence alignment by genetic algorithm. Nucleic Acids Res. 24, 1515-1524 (1996).

[17] Notredame C, D.G. Higgins and J. Heringa. T-Coffee: A novel algorithm for multiple sequence alignment. J. Mol. Biol. 302, 205-217(2000).

[18] Notredame C, L. Holm, and D.G. Higgins. COFFEE: an objective function for multiple sequence alignments. Bioinformatics 14(5), 407-422 (1998).

[19]. Heringa, J. Two strategies for sequence comparison: profile-preprocessed and secondary structure-induced multiple alignment. Comp. Chem. 23, 341-364 (1999).

[20] Heringa, J. Local weighting schemes for protein multiple sequence alignment. Comput. Chem., 26(5), 459-477(2002).

[21] Brocchieri L & Karlin S. A symmetric-iterated multiple alignment of protein sequences. J. Mol. Biol. 276:249-264. (1998)

[22] Altschul, SF, W. Gish, W. Miller , E.W. Myers, and D.J. Lipman. Basic Local alignment search tool. J. Mol. Biol. 215:403-410 (1990)

[23] Kent , WJ BLAT The BLAST-Like Alignment Tool. Genome Research, 12(4):656-664(2002).

[24] Pearson, WR. Rapid and Sensitive Sequence Comparison with FASTP and FASTA. Methods Enzymol. 183:63-98 (1990).
[25] Lipman, DJ and Pearson WR. Rapid and Sensitive Protein Similarity Searches. Science 227:1435-1441(1985).
[26] Pietrokovski S. Searching databases of conserved sequence regions by aligning protein multiple-alignments. Nucleic Acids Research. 24(19): 3836-3845 (1996)
[26] Altschul, S.F., T.L. Madden, A.A. Schäffer, J.H. Zhang, Z. Zhang, W. Miller, and D.J. Lipman, Gapped BLAST and PSI-BLAST: a new Generation of Protein Database Search Programs, Nucleic Acids Res. 25:3389-3402 (1997).
[27] Hughey, R. and Krogh, A. Hidden Markov Models for Sequence Analysis: Extension and Analysis of the Basic Method. CABIOS, 12, 95–107. (1996)
[28] Hughey, R., Karplus, K. and Krogh, A. SAM: Sequence Alignment and Modeling software system, version 3. Technical Report UCSC-CRL-99-11, University of California, Santa Cruz, Computer Engineering, UC Santa Cruz, CA 95064. (1999)
[29] Neuwald, A.F., J. S. Liu, D.J. Lipman and C.E. Lawrence. Extracting protein alignment models from the sequence database. Nucleic Acids Research, 25(9): 1665-1677(1997)
[30] Biology 326/760: Introduction to Bioinformatics and Genomics.http://www.ist.temple.edu/~vucetic/cis595spring2003/bioinformatics%20and%20genomics%20-%20week%209.htm
[31] Cliften, P., L. Hillier, L. Fulton, T. Graves, T. Miner, W. Gish, R. Waterston, and M. Johnston. Surveying *Saccharomyces* Genomes to Identify Functional Elements By Comparative DNA Sequence Analysis. *Genome Res.* **11:** 1175-1186 (2001)
[32] McCue, L., W. Thompson, C. Carmack, M. Ryan, J. Liu, V. Derbyshire, and C. Lawrence. Phylogenetic Footprinting of Transcription Factor Binding Sites in Proteobacterial Genomes. *Nucleic Acids Res.* **29:** 774-782(2001).
[33] The Tree Of Life Web Project. http://tolweb.org/tree/phylogeny.html
[34] Morgenstern B. DIALIGN2: Improvement of the Segment-To-Segment Approach to Multiple Sequence Alignment. Bioinformatics 15(3), 211-8(1999).
[35] B. Morgenstern, K. Frech, A. Dress, and T. Werner. DIALIGN: Finding local similarities by multiple sequence alignment. Bioinformatics 14, 1998, 290-294.
[36] Lawrence, C. E., S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald and J.C. Wootton. Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy For Multiple Alignment. Science, 262, 208-214. (1993).
[37] Liu, X. D. Brutlag, D. and J. Liu. BioProspector: Discovering Conserved DNA Motifs In Upstream Regulatory Regions of Co-expressed Genes. Pacific Symposium on Biocomputing 6:127-138(2001).
[38] Roth, F.P., J.D. Hughes, P.W. Estep, and G.M. Church G.M. Finding DNA Regulatory Motifs Within Unaligned Noncoding Sequences Clustered By Whole-genome mRNA Quantitation. Nat. Biotechnol. 16, 939-945(1998).
[39] Hughes, J.D., P.W. Estep, S. Tavazoie, and G.M. Church. Computational Identification of Cis-regulatory Elements Associated With Groups of Functionally Related Genes in Saccharomyces Cerevisiae. J. Mol. Biol. 296, 1205-1214(2000).

[40] S. Henikoff, J. G. Henikoff, W. J. Alford & S. Pietrokovski, Automated Construction and Graphical Presentation of Protein Blocks from Unaligned Sequences, Gene-COMBIS, Gene 163:17-26. (1995)
[41]Bailey, T.L. and C. Elkan. Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Biopolymers. Proc. Int. Conf. Intell. Syst. Mol. Biol. 2, 28-36 (1994).
[42] Bailey, T.L. and M. Gribskov. Combining Evidence Using p-values: Application to Sequence Homology Searches. Bioinformatics 14, 48-54 (1998).
[43] DNA-Structure. http://www.eurekascience.com/ICanDoThat/dna_structure.htm
[44] Protein Structure, http://www.accessexcellence.org/AB/GG/protein.html
[45] Introduction to Protein Structure. http://www.paccd.cc.ca.us/instadmn/physcidv/chem_dp/chemweb/protein/index.html
[46] Dayhoff M.O. ,R.M. Schwarz, and B.C. Orcutt. A Model of Evolutionary Change in Proteins. Altas of Protein Sequence and Structure, 5:345-352(1978).
[47] Schwarz R. and M. Dayhoff. Matrices for Detecting Distant Relationships. Altas of Protein Sequence and Structure., Natl. Biomed. Res. Found. 353-358 (1979).
[48] S. Henikoff & J.G. Henikoff, Amino Acid Substitution Matrices From Protein Blocks. Proc Natl. Acad. Sci. USA, 89:10915-10919 (1992).
[49] Xu W.J. and D.P. Miranker. A Metric Model of Amino Acid Substitution. 20(8):1214-1221(2004)
[50] Thompson,J.D., F. Plewniak and O. Poch. BAliBASE: A Benchmark Alignment Database for the Evaluation of Multiple Alignment Programs. Bioinformatics, 15, 87–88 (1999).
[51] Bahr A., J. D. Thompson, J.-C. Thierry and O. Poch. BAliBASE (Benchmark Alignment dataBASE): Enhancements for Repeats, Transmembrane Sequences and Circular Permutations. Nucleic Acids Research, Vol. 29(1):323-326(2001).
[52] Lassmann T. and E. Sonnhammer. Quality Assessment of Multiple Alignment Programs. FEBS Letters, 529:126-130 (2002).
[53] Needleman S.B. and C. D. Wunsch. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. Journal of Molecular Biology, 48:443–53(1970)
[54] Smith T. F. and M. S. Waterman. Identification of Common Molecular Subsequences. Journal of Molecular Biology, 147:195–7(1981)
[55] Advanced Dynamic Programming Tutorial. http://www.sbc.su.se/~per/molbioinfo2001/dynprog/adv_dynamic.html
[56] Genome 373. http://noble.gs.washington.edu/~noble/gs373/lectures/lecture6.ppt
[57] Notrdame C. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics* 3(1):131–144(2002).
[58] Carrillo H, Lipman DJ: The multiple sequence alignment problem in biology. SIAM J. Appl. Math. 48, 1073-1082 (1988).
[59] Lipman DJ, Altschul SF, Kececioglu JD: A tool for multiple sequence alignment. Proc. Natl. Acad. Sci. USA 86, 4412-4415 (1989).
[60] Barton GJ, Sternberg MJE: A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. J. Mol. Biol. 198, 327-337 (1987).

[61] A* Algorithm. http://theory.stanford.edu/~amitp/GameProgramming/
[62] Lenhof H.P., B. Morgenstern, and K. Reinert. An Exact Solution for the Segment-To-Segment Multiple Sequence Alignment Problem. Bioinformatics 15(3), 203-210(1999).
[63] Takayuki Y. and T. Miura and T. Ishida. A* with Partial Expansion for large branching factor problems. AAAI/IAAI: 923-929 (2000)
[64] Matthew McNaughton, Paul Lu, Jonathan Schaeffer, Duane Szafron: Memory-Efficient A* Heuristics for Multiple Sequence Alignment. AAAI/IAAI: 737-743(2002)
[65] Reinert K, Stoye J, Will T: An iterative method for faster sum-of-pair multiple sequence alignment. Bioinformatics 16(9), 808-814 (2000).
[66] Kimura, M. The Neural Theory of Molecular Evolution. Cambridge Press (1983).
[67] Saitou, N. and Nei, M. The neighboring-joining method: a mew method for reconstructing phylogenetic trees. Molecular Biology and Evolution. 4:406-425(1987)
[68] Genetic Algorithm. http://cs.felk.cvut.cz/~xobitko/ga/
[69] Brocchieri L & Karlin S. A symmetric-iterated multiple alignment of protein sequences. J. Mol. Biol. 276:249-264(1998).
[70] Lee C., C. Grasso, and M. Sharlow. Multiple Sequence Alignment Using Partial Order Graphs. Bioinformatics 18:452-464 (2002).
[71] Myers G. and W. Miller. Chaining Multiple-Alignment Fragments In Sub-Quadratic Time. SODA: ACM-SIAM Symposium on Discrete Algorithms (1995)
[72]. Miranker D. The MoBIoS Project, 2003. http://www.cs.utexas.edu/~mobios/Publications/MoBIoS-one-pager2.pdf
[73] Madeira S.C. and A.L. Oliveira. Biclustering Algorithms for Biological Data Analysis: A Survey. IEEE/ACM Transactions On Computational Biology and Bioinformatics. 1-30 (2004)
[74] Dhillon I.S., S. Mallela, and D. S. Modha. Information-theoretical biclustering. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03). 89–98 (2003).
[75] Lazzeroni L. and A. Owen. Plaid models for gene expression data. Technical report, Stanford University, 2000.
[76] Amir B.-D., B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order–preserving submatrix problem. In Proceedings of the 6th International Conference on Computacional Biology (RECOMB'02). 49–57(2002).
[77] Murali T. M. and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In Proceedings of the Pacific Symposium on Biocomputing. 8: 77–88(2003).
[78] Busygin S., G. Jacobsen, and E. Kramer. Double conjugated clustering applied o leukemia microarray data. In Proceedings of the 2nd SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data (2002).
[79] Klugar Y., R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. In Genome Research, volume 13: 703–716(2003).

[80] Sheng Q., Y. Moreau, and B.D. Moor. Biclustering micrarray data by gibbs sampling. In Bioinformatics, volume 19 (Suppl.2). 196–205(2003).
[81] Tang C., L. Zhang, I. Zhang, and M. Ramanathan. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering. 41–48(2001).
[82] Wang H., W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. 394–405(2002).
[83] Tibshirani R., T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of DNA microarray data. Technical report, Department of Health Research and Policy, Department of Genetics and Department of Biochemestry, Stanford University. 1999.
[84] Hartigan J. A. Direct clustering of a data matrix. Journal of the American Statistical Association (JASA), 67(337):123–129, 1972.
[85] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional space. In Proceedings of the 3rd IEEE International Conference on Data Mining, pages 187–194(2003).
[86] Segal E., B. Taskar, A. Gasch, N. Friedman, and D. Koller. Decomposing gene expression into cellular processes. In Proceedings of the Pacific Symposium on Biocomputing, volume 8: 89–100(2003).
[87] Tanay A., R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. Bioinformatics, volume 18 (Suppl. 1):136–S144(2002).
[88] Cheng Y. and G.M. Church. Biclustering of expression data. In Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00), pages 93–103(2000).
[89] Getz G., E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. In Proceedings of the Natural Academy of Sciences USA, pages 12079–12084(2000).
[90] Califano A., G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In Proceedings of the International Conference on Computacional Molecular Biology, pages 75–85(2000).
[91] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. δ-clusters: Capturing subspace correlation in a large data set. In Proceedings of the 18th IEEE International Conference on Data Engineering, pages 517–528(2002).
[92] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. Enhanced biclustering on expression data. In Proceedings of the 3rd IEEE Conference on Bioinformatics and Bioengineering, pages 321–327(2003).
[93] Introduction to Patterns, Profiles, HMMs and PSI-BLAST. http://search.yahoo.com/search?p=an+introduction+to+patterns%2C+profile%2C+HMMS&ei=UTF-8&fr=FP-tab-web-t&n=20&fl=0&x=wrt
[94] Motifs, profiles and hidden Markov models. http://www.ludwig.edu.au/course/course2002/talks/crc02motif_edit/
[95] Barkow, S., Bleuler, S., Prelic, A., Zimmermann, P., and Zitzler, E. BicAT: A Biclustering Analysis Toolbox, *Bioinformatics,* 22(10), 1282-1283 (2006)

[96] Prelic , A., Stefan, B. , Philip, Z., Anja, W., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L. and Zitzler, E. A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data, *Bioinformatics* 22(9), 1122-1129 (2006)

[97] Griffiths-Jones S., M. Simon, M. Mhairi, K. Ajay, R.E. Sean and A. Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res*. 33:D121-D124 (2005).

[98] Rfam 7.0: ftp://ftp.sanger.ac.uk/pub/databases/Rfam/

[99] Carl Woese. Bacterial evolution. *Microbiol Rev*. 51:221-71 (1987)

[100] Grundy F.J. and T.M. Henkin. The T Box and S Box Transcription Termination Control Systems. *Frontiers In Bioscience* 8, d20-31 (2003)

[101] Dhillon I.S.. Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, August 26-29, 2001.

[102] Zhang Ya, Chao-Hsien Chu, Xiang Ji, Hongyuan Zha: Correlating summarization of multi-source news with k-way graph bi-clustering. SIGKDD Explorations 6(2): 34-42 (2004)

[103] Grundy, F.J. and Henkin, T.M. The T Box and S Box Transcription Termination Control Systems. *Frontiers in Bioscience,* 8, 20-31. (2003)

[104] Cannone J.J, Subramanian S., Schnare M.N., Collett J.R., D'Souza L.M., Du Y., Feng B., Lin N., Madabusi L.V., Müller K.M., Pande N., Shang Z., Yu N., and Gutell R.R. The Comparative RNA Web (CRW) Site: An Online Database of Comparative Sequence and Structure Information for Ribosomal, Intron, and Other RNAs. BioMed Central Bioinformatics. 3:2. (2002).

[105] Cech,T.R. Self-splicing of group I introns. *Annu. Rev. Biochem.*, 59, 543–568 (1990)

[106] Do, C.B., Mahabhashyam, M.S.P., Brudno, M., and Batzoglou, S (2005). PROBCONS: Probabilistic Consistency-based Multiple Sequence Alignment. *Genome Research* 15: 330-340. (http://PROBCONS.stanford.edu/download.html)

[107] Eddy S. SQUID – C function library for sequence analysis. (http://selab.wustl.edu/cgi-bin/selab.pl?mode=software#squid).

[108] Economist. Really New Advances.(2007) (http://www.economist.com/science/displaystory.cfm?story_id=9333471)

[109] Edgar, R.C. MUSCLE: multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Research* 32(5), 1792-97 (2004) (MUSCLE Version 3.6. http://www.drive5.com/MUSCLE/download3.6.html).

[110] Gutell R.R., Larsen N., and Woese C.R. Lessons from an Evolving Ribosomal RNA: 16S and 23S rRNA Structure from a Comparative Perspective. Microbiological Reviews 58:10-26. (1994)

[111] Griffiths-Jones, S., Moxon, S., Simon, Marshall, M., Khanna, A., Eddy, S.R., and Bateman A. Rfam: Annotating Non-coding RNAs in Complete Genomes. *Nucleic Acids Res*. 33, 121-124. (2005)

[112]   Katoh K., Kuma K., Toh H. and Miyata T. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.* 33:511-518. (2005)
[113] Wilm, A., Mainz, I. & Steger, G. An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol. Biol.* 1, 19. (2006)
[114] Rick, F.H.C.: On Protein Synthesis. Symp. Soc. Exp. Biol. XII, 139-163.(1958)
[115] Crick, F. Central Dogma of Molecular Biology. Nature 227, 561-563. PMID 4913914 (1970)
[116] Glossary of Common Biotech Terms.http://www.nabda.gov.ng/glossaries.htm
[117] Translation (biology): http://en.wikipedia.org/wiki/Translation_(genetics)
[118] Hubbard T.J.P., A.M. Lesk, and A. Tramontano. Gathering them into the fold. Nature Structural Biology. 4:313. 1996
[119] Freemanand S.   and J.C. Herron . Evolutionary Analysis. Prentice Hall. (1998)

VITA

Shu Wang was born in Yanji, Jilin Province, P. R. China, on March 28th, 1973, the daughter of Junlu Wang and Suming Liu. After completing her study at Fushun No. 1 Middle School, Fushun, Liaoning Province, P.R. China, in 1990, she entered The University of Science and Technology Beijing, Beijing, P.R. China. She received the degree of Bachelor of Engineering from Department of Electrical Engineering in 1994. Then she attended the graduate school of the Chinese Academy of Sciences (CAS), Beijing, P.R. China and obtained the degree of Master of Science in Electrical Engineering, from Institute of Automation of the Chinese Academy of Sciences, P.R. China in 1997. She joined The University of Georgia in 1997 and received the degree of Master of Science in Agricultural Engineering from The University of Georgia in 1999. She attended The University of Texas At Austin in 2000 for her Ph.D. degree and right now also works as a software engineer in Samsung Telecommunications America.

Permanent Address: 1517 County Road 132A
Kingsland, TX, 78639
U.S.A.

This dissertation was typed by the author.